DOMAIN SIMILARITY MEASURES

On the use of distance metrics in natural language processing

Vincent B. Van Asch

Proefschrift voorgelegd tot het behalen van de graad van doctor in de Taalkunde aan de Universiteit Antwerpen

Promotor: Prof. Dr. Walter Daelemans
Antwerpen, 30 mei 2012

Universiteit Antwerpen

Universiteit Antwerpen

Faculteit Letteren en Wijsbegeerte
Departement Taalkunde

DOMAIN SIMILARITY MEASURES
On the use of distance metrics in natural language processing

---

DOMEINGELIJKENISMATEN
Over het gebruik van afstandsmaten in natuurlijketaalverwerking

Proefschrift voorgelegd tot het behalen van de graad van
doctor in de Taalkunde
aan de Universiteit Antwerpen te verdedigen door

Vincent VAN ASCH

Promotor: Prof. Dr. Walter Daelemans

Antwerpen, 2012

Graag wil ik iedereen bedanken die deze thesis geïnspireerd, verbeterd of gewoon gevolgd heeft en me daarmee onvergelijkbaar hielp.

Print: Silhouet, Maldegem

© 2012 Vincent Van Asch

2012 Uitgeverij UPA University Press Antwerp
UPA is an imprint of ASP nv (Academic and Scientific Publishers nv)
Ravensteingalerij 28
B-1000 Brussels
Tel. + 32 (0)2 289 26 50
Fax + 32 (0)2 289 26 59
E-mail: info@aspeditions.be
www.upa-editions.be

ISBN 978 90 5487 843 8
NUR 616
Legal Deposit D/2012/11.161/049

Abstract

A common view in natural language processing is that a language can be understood as a collection of language objects that follow an unknown probability distribution. A sample or *corpus*, drawn from all available data about a language, can be used to estimate these probability distributions. One such distribution is the relative frequency of the tokens in the sample. Apart from extracting a distribution from it, the sample, if annotated, can also be used to develop a natural language processing tool, such as a part-of-speech tagger or a word sense disambiguation system. Probability distributions and natural language processing tools are the two main ingredients of this dissertation.

There are many ways to collect a (text) corpus. The goal of corpus collection can be to represent all written data of, for example, the English language. To create this corpus, one should randomly sample from all English texts, which means that shopping lists, novels, and text messages could all end up together in the sample. The question is not only whether it is possible to obtain a sample representative for a language but also whether a natural language tool can deal with the enormous variance in the sample. To overcome this problem, researchers adopt a different sampling strategy and collect texts that show some resemblance. This resemblance is expressed by saying that the gathered texts all belong to the same *domain.* An example is the well-known Wall Street Journal corpus, which only contains news articles from that paper. This corpus, for which the internal variance is substantially limited when compared to a corpus of all English texts, can effectively be used to develop natural language processing tools.

Problems arise when these tools are tested on texts that discuss another domain (e.g. biomedical texts). The tools become much less efficient because the probability distribution underlying biomedical texts will differ from the distribution underlying the news article corpus. If collecting a new biomedical corpus to mediate the performance loss is not an option, making the natural language processing tool more robust to these domain shifts may be a solution.

This dissertation investigates the relationship between the underlying probability distributions of different corpora and the performance of natural language tools that were developed using these corpora. A better knowledge of this relationship may be a first step in making natural language processing tools more robust. In this dissertation, a range of divergences is evaluated to measure the similarity between corpora from different domains. The divergences can also be called *domain similarity measures* or *distance metrics.*

The domain similarity measures are tested for their correlation with the performance of natural language processing tools while varying several parameters such as the task, the algorithm, the corpora, and the homogeneity of the corpora. We found that, for selected similarity measures, the correlation is linear for the performance of a part-of-speech tagger and the distance between the corpus on which the tagger is trained and the corpus on which the tagger is tested.

The linearity of the correlation is the starting point for the application of the domain similarity measures in three setups: training data selection, feature selection, and self-training.

Training data selection consists of creating a training corpus with only those sentences that decrease the distance between the training corpus and a test corpus from another domain. We showed that it is possible to compress a training corpus without significantly harming performance.

For feature selection, it was investigated whether useful features influence the correlation between distance and the out-of-domain performance of a feature-based machine learner in a different way than superfluous or harmful features. We found that a method based on instance distance can be used to identify harmful features for synthetic data. Interdependencies between features and a varying number of values per feature complicate feature selection for real data.

Self-training is a semi-supervised domain adaptation technique for which unannotated out-of-domain corpora are automatically labeled before adding them to the training data. We show that not all newly added data leads to a performance increase. A performance indicator, based on similarity measures, can be used to select only the best-suited unlabeled corpora.

In view of these applications, it becomes clear that there are two aspects of a similarity measure that come into play: quantifying the (dis)similarity of the corpora, and linking distance to performance. Depending on the usage of the measure, one of those aspects is predominant. If the second aspect is more important, the linear correlation between performance and distance can be used to select the best-suited similarity measure.

To conclude, in this dissertation, an effort has been made to get a clearer view on the strengths and weaknesses of similarity measures in natural language processing. When certain conditions are fulfilled, the best-suited measures can be selected in a reasoned manner and, subsequently, the selected measure can be successfully applied. Follow-up research could focus on more complex uses, further refinement of the ideas about the quantifying qualities of the measures, and use of the measures in unsupervised feature selection.

Samenvatting

In natuurlijketaalverwerking wordt taal vaak beschouwd als een verzameling van talige objecten die verdeeld zijn volgens een onbekende waarschijnlijkheidsverdeling. Een steekproef of *corpus*, genomen uit alle data die beschikbaar is, kan gebruikt worden om de waarschijnlijkheidsverdeling te achterhalen. Een voorbeeld van zo een waarschijnlijkheidsverdeling is de relatieve frequentie van de tokens in de steekproef. Onafhankelijk van de waarschijnlijkheidsverdeling, is het eveneens mogelijk om met de steekproef, indien ze geannoteerd wordt, een systeem voor natuurlijketaalverwerking te ontwikkelen. Voorbeelden van zulke systemen zijn een systeem om woorden te labelen met hun woordsoort of een systeem voor de desambiguering van de betekenis van een woord. Waarschijnlijkheidsverdelingen en natuurlijketaalverwerkingssystemen zijn de twee hoofdingrediënten van deze thesis.

Er bestaan verschillende manieren om een (tekst)corpus te verzamelen. Een doel om een corpus te verzamelen kan zijn om, bijvoorbeeld, alle geschreven data van het Engels te representeren. Om dit corpus te kunnen verzamelen moeten er willekeurige steekproeven genomen worden van alle geschreven data waardoor boodschappenlijstjes, romans en sms-berichten samen terecht kunnen komen in een corpus. De vraag is niet enkel of het mogelijk is om op deze manier een representatieve steekproef te nemen maar bovendien of een natuurlijketaalverwerkingssysteem de enorme variatie in de steekproef aankan. Daarom kunnen onderzoekers hun corpus ook op een andere manier verzamelen; door enkel die teksten te kiezen die een zekere gelijkenis vertonen. De gelijkenis tussen teksten kan dan verwoord worden door te zeggen dat ze tot hetzelfde *domein* behoren. Een voorbeeld van een corpus met teksten uit hetzelfde domein is het veelgebruikte Wall Street Journal corpus, dat enkel krantenartikelen bevat. Dit corpus, met een veel kleinere interne variatie dan een corpus dat alle geschreven Engelse data zou omvatten, kan effectief gebruikt worden om natuurlijketaalverwerkingssystemen te ontwikkelen.

Natuurlijketaalverwerkingssystemen kunnen niet even succesrijk toegepast worden wanneer een ontwikkeld systeem getest wordt op data uit een ander domein (bijvoorbeeld biomedische teksten). De efficiëntie van de systemen vermindert omdat de waarschijnlijkheidsverdeling onderliggend aan biomedische teksten anders zal zijn dan deze voor krantenartikelen. Wanneer het verzamelen van een nieuw aangepast corpus geen optie is dan kan het meer robuust maken van het systeem een oplossing zijn.

In deze dissertatie wordt de relatie onderzocht tussen waarschijnlijkheidsverdelingen van verschillende corpora en de prestatie van natuurlijketaalverwerkingssystemen die ontwikkeld werden met behulp van die corpora. Een betere kennis van deze relatie kan een eerste stap zijn om natuurlijketaalverwerkingssystemen meer robuust te maken. Omwille van deze reden wordt in deze dissertatie een reeks van divergentiematen om de gelijkenis tussen corpora te meten geëvalueerd. Deze divergentiematen kunnen ook *domeingelijkenismaten* of *afstandsmaten* genoemd worden.

De domeingelijkenismaten worden geëvalueerd op basis van de correlatie met de prestaties van een natuurlijketaalverwerkingssysteem terwijl parameters zoals de exact taak, het gebruikte algoritme, de gebruikte corpora en de homogeniteit van de corpora veranderd worden. Voor bepaalde afstandsmaten stelden we een lineaire correlatie vast tussen de prestatie van een woordsoortlabeler en de afstand tussen het corpus waarmee de labeler ontwikkeld werd (*ontwikkelingscorpus*) en het corpus waarmee het geëvalueerd werd (*evaluatiecorpus*).

Deze lineaire correlatie wordt gebruikt als uitgangspunt voor drie toepassingen van domeingelijkenismaten: het selecteren van ontwikkelingsdata, het selecteren van features en het selecteren van een geschikt corpus voor self-training.

Ontwikkelingsdata kan getrieerd worden door van een groot corpus enkel die zinnen bij te houden die de afstand tussen het ontwikkelingscorpus en het evaluatiecorpus uit een ander domein verkleinen. We vonden dat het op deze manier mogelijk is om een ontwikkelingscorpus te comprimeren zonder dat de prestaties significant verslechteren.

In verband met featureselectie werd er gekeken of bruikbare features de correlatie tussen afstand en prestaties op een andere manier beïnvloeden dan nadelige features of features zonder informatie-inhoud. We toonden aan dat, voor synthetisch aangemaakte data, een selectiemethode gebaseerd op een gelijkenismaat gebruikt kan worden om nadelige features te identificeren. Bij reële data zorgen afhankelijkheid tussen de features onderling en een verschillend aantal featurewaardes er echter voor dat deze selectiemethode beduidend moeilijker toe te passen valt.

Self-training is een semi-gecontroleerde domeinadaptatiemethode waarbij ongeannoteerde corpora automatisch gelabeld worden zodat ze bij het ontwikkelingscorpus gevoegd kunnen worden. We stelden vast dat niet elk ongeannoteerd corpus tot betere resultaten leidt wanneer het op deze manier aan een corpus toegevoegd wordt. Een indicator, die gebaseerd is op gelijkenismaten, kan gebruikt worden

om uit een verzameling van corpora enkel die te kiezen die de uiteindelijke resultaten zullen verbeteren.

Deze drie toepassingen van gelijkenismaten kunnen beschouwd worden als een goede illustratie van twee verschillende aspecten van gelijkenismaten: het kwantificeren van de (on)gelijkheid van corpora en het uitdrukken van de relatie tussen afstand en prestaties van een natuurlijketaalverwerkingssysteem. Welk aspect het belangrijkst is hangt af van de toepassing. Als het tweede aspect meer belangrijk is dan kan de kwaliteit van de lineaire correlatie tussen afstand en prestaties gebruikt worden om de geschikte gelijkenismaat te vinden.

Ter afsluiting kunnen we zeggen dat in deze dissertatie een inspanning werd gedaan om een betere kijk te krijgen op de zwaktes en sterktes van gelijkenismaten bij hun gebruik in natuurlijketaalverwerking. Wanneer aan bepaalde voorwaarden voldaan is dan kan de meest geschikte gelijkenismaat voor een bepaalde taak op een beredeneerde wijze geselecteerd worden. Vervolgonderzoek kan zich richten op het gebruik van gelijkenismaten in een meer complexe opstelling, het verder verfijnen van de kennis over het kwantitatief gebruik van de maten en het verdere gebruik van gelijkenismaten in ongecontroleerde featureselectie.

Contents

Chapter 1

Introduction

1.1 Language as a set of distributions

A common view in natural language processing (NLP) is that a language can be understood as a collection of productions that follow an unknown probability distribution. One of the well-known examples of how a collection of words can be associated with probabilities is Zipf's law (Zipf, 1972). Zipf's law states that the probability of a word, $P(w)$, is related to its rank s when the words of a corpus are ordered along their frequency – with C, α as constants:

$$P(w) = \frac{C}{s^{\alpha}} \tag{1.1}$$

It has been argued that other equations can model the occurrences of words frequencies better (Montemurro, 2001) and that a random token generator can also give rise to a Zipfian-like law (Li, 1992), but regarding texts as the outcome of an unknown probabilistic process remains a fruitful paradigm.

Although the probabilistic process underlying the production of words is unknown, it certainly is not random: vocabulary, grammar, purpose and medium are only a small selection of factors that influence a person trying to get a message across. In fact, the presence of those regulating factors can be observed in the Zipfian distribution of words when the language of a random token generator is compared with natural language (Cohen et al., 1997). Cohen et al. (1997)

argue that the main difference between real and artificial texts can be located in the frequencies of the least frequent words and that the difference is detected by the α parameter that is part of the inverse Zipf analysis. Applications developed for natural language processing assume that at least some of those regulating factors remain reasonably constant and that inferences drawn from the language of one corpus can lead to predictions about the language of another corpus. In contrast, differences in the Zipfian distribution, that can be attributed to a difference in regulating factors, have been used to distinguish scientific from belletristic literature (Voloshynovska, 2011). This is an illustration of the weakness of probability-based applications: probability distributions of corpora may differ substantially depending on which regulating factors are constant when a corpus is being gathered.

The differences between the distributions underlying corpora can lead to decreased performance when it comes to natural language processing applications. Over time, researchers in computational linguistics have attempted to create robust applications that can cope with these differences, but nevertheless, building probability models for one corpus based on a different corpus continues to be a source of performance loss. The work presented here, is dedicated to the investigation and exploration of the relation between varying probability distributions in written language and performance loss in a set of natural language processing applications.

The research questions that will be addressed focus on the connection that exists between distance metrics and the performance of a natural language processing tool. To answer these questions, experiments to investigate the nature and the sensitivities of the relation are set up. The insight gained will help interpret the outcome of experiments in which the distance–performance relation is used in an application.

The remainder of this chapter provides definitions of some basic concepts, an overview of how differences in probability distributions have been tackled in previous research, an overview of how the differences in probability distributions have been measured, and a description of the research questions motivating this work.

1.2 Probability distribution differences

Domain adaptation issues are frequently observed in natural language processing applications and the main source of domain-sensitivity can be attributed to the machine learning component in those applications. Since the experiments in this dissertation are machine learning experiments, we will first summarize the basic principles of a machine learning setup.

1.2.1 Basic concepts

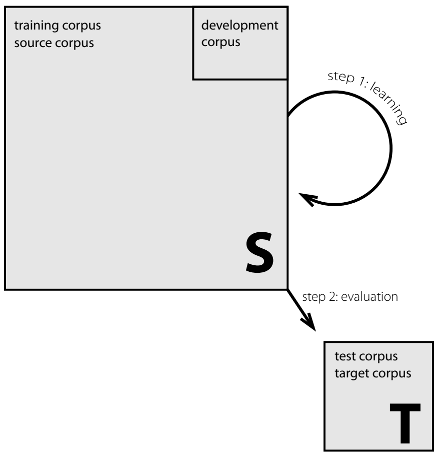


Figure 1.1: Illustration of a general machine learning setup – step 1 is used to train a machine learner on the training or source corpus S, ideally a development corpus is chosen from the training corpus for evaluation during the learning step. In a second step, the trained machine learner is evaluated on the test or target corpus T. Step 2 will provide the final evaluation scores for the trained machine learner.

Supervised machine learning experiments draw upon the idea that an observation X is labeled with a (class) label Y and that it is possible to label unseen observations by modeling the known links between observations and labels in a given corpus. An illustration of the general machine learning setup is given in Figure 1.1.

The corpus that is used for the modeling or training step is called the training corpus S. The corpus of previously unseen observations is called the test corpus T.

In domain adaptation experiments, the domain, for which an NLP tool is developed, is called the *target* domain. The domains that are covered by the training corpus are the *source* domains. In most domain adaptation experiments, training and test corpus come from a single domain and the training corpus may also be referred to as the source (domain) corpus while the test corpus may be referred to as the target (domain) corpus.

During the training step of a machine learner, the test corpus cannot be used to obtain intermediate evaluations, since this can lead to overfitting of the machine learner on the test data. A method to overcome this problem is to take a part of the training corpus as a temporary test corpus. This temporary test corpus is called the development corpus. During the final evaluation on the test corpus, the *development* corpus can be re-incorporated in the training corpus.

In a machine learning setup, the practical implementation of observations X are called *instances*. The probabilities of instances can be very low depending on the nature of the instances. For example, if every token of a text is taken as an instance, it is possible to obtain probabilities that are sufficiently expressive. In contrast, if the context of the token and other information is included in the instance, it may well be the case that each instance occurs only once. Leading to probabilities with almost no information content. To overcome this sparseness problem, machine learners are able to extract information from a sub-instance level (*i.e.* the features) and the same ability should be present in distance metrics that are based on these complicated instances.

The probability $P(X, Y)$ is the relative frequency of an observation linked to a class label and can be estimated by:

$$P(X, Y) = \frac{\text{number of observations } X \text{ that have label } Y \text{ in the corpus}}{\text{total number of observations in the corpus}} \tag{1.2}$$

The joint probabilities $P_s(X, Y)$ from the source corpus, denoted by s, can be used to construct a model, although not every machine learner tries to model the probabilities directly. An example of a machine learning approach that is not based on calculating probabilities, is memory-based learning (Daelemans & van den Bosch, 2005). In this setup, all observations are stored in memory and a

domain	adjective	noun
imaginative	74%	26%
natural science	32%	68%

Table 1.1: Example of difference in joint distributions $P(X,Y)$ for the token *fat* in the *imaginative* and *natural science* domain from the British National Corpus (BNC, 2001). The token is predominantly an adjective in the *imaginative* domain (74% of the occurrences) and a noun in the *natural science* domain (68% of the occurrences).

new observation is labeled according to its similarity to the stored observations. The joint probabilities $P_s(X,Y)$ are not modeled, but they will lead to a different distribution of observations in the memory-based instance space. Although the joint probabilities are not always computed, they are the basis of machine learning experiments and the more $P_s(X,Y)$ resembles the joint probability $P_t(X,Y)$ from the target corpus, the easier and more efficient the labeling will be.

It is possible to factorize the joint probability in two ways:[1]

$$\begin{aligned} P(X,Y) &= P(X)P(Y|X) && (1.3)\\ &= P(Y)P(X|Y) && (1.4)\\ & && (1.5) \end{aligned}$$

A prerequisite to be able to have a good performance is that $P_s(X,Y) \approx P_t(X,Y)$. It is not always the case that this prerequisite is satisfied. When the source probabilities differ that much from the target probabilities, so annotators decide that they are not comparable, the probabilities are said to come from different domains. However, it is hard for human annotators to discern probability differences and most often domains are identified according to the semantic contents of the collected corpora. An example of a corpus containing domain labels referring to semantic contents is the Brown Corpus (Francis & Kučera, 1964), which is a corpus of texts that are grouped into informative prose and imaginative prose and both groups are more finely divided into various subgroups.

An example of a case where $P(X,Y)$ clearly differs in two separate domains comes from the British National Corpus. The token *fat* is mostly an adjective

[1]Which are also linked to Bayes' theorem: $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$ with $P(X) \neq 0$.

used to describe a person's figure in the *imaginative* domain and a noun used to refer to the actual substance in the *natural science* domain.[2] The actual occurrences are given in Table 1.1.

When training a machine learner on one domain and testing it on another domain, the experiment can be called an *out-of-domain* experiment as opposed to an *in-domain* experiment. *Domain adaptation* is any effort made to increase the performance for an out-of-domain experiment. The class labels in the source and target domain in a domain adaptation experiment are drawn from the same set.

When the class labels for the source and target domain are different, the task is no longer a domain adaptation task, but a *multi-task learning* problem. For example, a named-entity tagger for a flight reservation system would need to be able to identify information about timings, destinations and prices, while a named-entity tagger for a tourist guide system would need to be able to identify information about transport, accommodation and sightseeing trips (Jeong & Lee, 2009). Both tasks are similar, but the class labels are different, creating additional difficulties for a machine learner. Both domain adaptation and multi-task learning can be regarded as *transfer learning* problems (Jeong & Lee, 2009).

One can distinguish three setups for domain adaptation experiments: supervised and semi-supervised experiments and unsupervised (Daumé III, 2007). In the supervised setup, there is a large annotated source domain corpus and a small annotated target domain corpus, which will both be used to train the machine learner. In the semi-supervised setup, instead of a small annotated target domain corpus, a large unannotated target domain corpus is used. In both experimental setups, a test corpus coming from the target domain is to be used for the evaluation of the machine learner.

The definitions for the three domain adaptation setups are not fixed. Sometimes, semi-supervised, as defined above, is called unsupervised domain adaptation. A mixture of a large unannotated target corpus plus a small annotated target corpus during the training phase is then used to carry out domain adaptation in a semi-supervised manner (Plank, 2011).

[2]Mostly *adjective* means carrying the class label `AJ0` or `AJ0-NN1`; mostly *noun* means carrying the class label `NN1` or `NN1-AJ0`.

1.2.2 Domain adaptation

Domain adaptation is an established discipline in machine learning and has been approached in various ways. As can be inferred from the factorization $P(X,Y) = P(X)P(Y|X)$, there are two main approaches to make the source probabilities $P_s(X,Y)$ more similar to the target probabilities $P_t(X,Y)$: influencing the association between observations and class labels and changing (the probability of) the observations. Both approaches are not strictly divided – most probably $P(Y|X)$ will change also when $P(X)$ changes – but they will provide a framework for an overview of previous domain adaptation work. The reader is referred to the work of Jiang (2008) and Margolis (2011) for a different classification and a more extensive overview of domain adaptation research. The focus of this work is on quantifying and exploiting domain shifts rather than on domain adaptation itself.

An example of trying to influence the estimated conditional probability during the learning step is *instance weighting*. Examples of methods used to influence the estimated probabilities $P(X)$ are training data selection, training data creation, changing the feature space, and feature weighting. The methods are explained in the next subsections.

Instance weighting

One way to influence the association between an observation and a class label is by weighing the observations or instances. Assigning more or less weight to an observation during a learning step will change the estimation of $P(Y|X)$.

The theoretical basis for instance weighting is given by Jiang (2008). The technique introduces a factor $\frac{P_t(x_i^s,y_i^s)}{P_s(x_i^s,y_i^s)}$ to weigh the loss of an observation (x_i, y_i). It is not possible to compute the weighing factor from the data without any further assumptions. Jiang (2008) discerns two approaches in the domain adaptation literature: *class imbalance* and *covariate shift*.

In the case of class imbalance, it is assumed that the conditional probabilities of X are the same in the source and target domain: $P_s(X|Y) = P_t(X|Y)$. Under this assumption the weighing factor is reduced to $\frac{P_t(Y)}{P_s(Y)}$. The distribution of the class labels for the target domain has to be estimated in order to compute the weights – which is an unfeasible task.

In the case of covariate shift, the distribution of the labels is the same in the source and target domain for a given instance, $P_s(Y|X) = P_t(Y|X)$. Under this assumption the weighing factor is reduced to $\frac{P_t(X)}{P_s(X)}$.

Another version of instance weighing is instance pruning (Jiang & Zhai, 2007). As Dahlmeier & Ng (2010) explain, a small sample of labeled instances from the target domain is required when instance pruning is to be used. The instances from the source domain are labeled with a classifier, trained on this target domain sample. The top N instances that are predicted incorrectly – when ranked according to prediction confidence – are removed from the source domain. These instances are considered to differ too much from the target domain to contain helpful information. Finally, the system is trained on the adapted source domain. Dahlmeier & Ng (2010) compare an instance weighting and an instance pruning approach with the method explained in Daumé III (2007) for semantic role labeling. They find that, in general, instance pruning is a better method for adapting a semantic role labeler to a new domain.

The tree weighting approach for statistical parsing of Plank & Sima'an (2008) is related to instance weighting in the sense that parse trees, in contrast to instances, are given a weight according to their fitness for parsing sentences of a given domain. In this approach, a corpus is divided into subdomains by assigning a score to each tree, according to the likeliness that it belongs to a given subdomain. The result are domain-specific parsers trained on treebanks with weighted trees, which can then be combined into an optimal combination to parse an unseen sentence.

Training data selection

Instead of trying to alter the association between an observation and its class label, it is also possible to change the estimated probabilities of the observations, $P(X)$. Training data selection or filtering is an attempt to keep only those observations from the source domain that seem relevant for the target domain. This can be done in various ways. The most straightforward way is implemented by Gao et al. (2002) in order to adapt a statistical language model (SLM) to various domains. The method consists of four steps: (a) segmenting the training data, (b) ranking the segments according to a metric (perplexity), (c) selecting the top N training data segments, (d) reducing the language model by pruning it.

McClosky (2010) carries out domain adaptation that is an extension of training data selection by combining a mixture of models trained on different source domains into one model for parsing. A combined distance metric is used to assign weights to the different models in the combined model. The weighing of corpora in the final training set is conceptually similar to *model interpolation*, which is a special case of *maximum a posteriori adaptation* (Bacchiani & Roark, 2003). Maximum a posteriori adaptation means that the model that is learned, maximizes the probabilities of observations x given a model θ, along with the probability of the model, instead of using only the conditional probabilities $P(x|\theta)$. In the case of model interpolation, two models are learned: one for the source domain and one for the target domain. The probability for a class label y, given an instance x, is a linear combination – controlled by a λ parameter – of the probabilities taken from the two models:

$$P(y|x) = \lambda P_s(y|x) + (1 - \lambda)P_t(y|x) \tag{1.6}$$

Training data creation

Training data creation can help to adapt a machine learner to another domain. The creation of labeled data for the target domain can help domain adaptation in a supervised or semi-supervised way. The labeled data can be included in the training data, used during the learning phase of the machine learner.

A supervised way to adapt a machine learner to another domain is by active learning. Active learning is a technique that starts with selecting interesting instances from an unlabeled target data sample. Next, the selected instances are labeled by an annotator and are added to the training data, thus more efficiently increasing the performance of a machine learner. This approach is introduced by Lewis & Gale (1994) under the name of *uncertainty sampling*. The workflow for active learning is given in Figure 1.2 (Chan & Ng, 2007).

In the first step, all labeled source data is taken as training data and a model is inferred. Next, an instance from the target data is labeled with the trained model. If the machine learner is not very certain about the new label, the correct label is provided and the instance is added to the training data. If the machine learner is certain, it is assumed that the machine learner does not need this instance as additional evidence and the instance is discarded. The labeling and adding steps are repeated until a stopping criterion is met. Examples of stopping

```
INPUT    labeled source data, labeled target data sample
OUTPUT   adapted model
step 1   Train a model on the labeled source data,
step 2   Label an instance from the target data sample with the model,
step 3   If the certainty of the machine learner is below a given threshold, add the instance with the correct label to the training data,
step 4   Repeat from step 2 until a stopping criterion is met – e.g. a predefined number of cycles is reached or cross-validation accuracy does no longer increase significantly,
step 5   Train the final model on the extended training data.
```

Figure 1.2: The active learning algorithm, adapted from Chan & Ng (2007).

criteria are putting a threshold on the number of cycles and stopping when cross-validation accuracy does no longer increase significantly. The final model will be trained on the labeled source data, extended with selected instances from the target data.

Active learning becomes interesting during the corpus annotation phase when, initially, the target data sample is unlabeled. In step 3 of the algorithm of Figure 1.2 an annotator then provides the correct label. Active learning is a way to save annotation time, because only the interesting instances need the annotator's attention.

Instead of adding the instance with the correct label, it is also possible to simply add the instance with the predicted label. This is the concept of self-training (Charniak, 1997; McClosky, 2010; Sagae, 2010). During self-training, additional training data is created in a semi-supervised way. Jiang & Zhai (2007) present an example for part-of-speech tagging (POS-tagging). POS-tagging is a task that consists of labeling tokens of a sentence with information about the linguistic category to which they belong. Self-training in a POS labeling context is explored in Section 4.4. The self-training routine from Figure 1.3 is taken from McClosky (2010).

In a first step a model is trained on labeled data. Next, this model is used to label any amount of unlabeled data. The third step is optional and entails selecting

```
INPUT     labeled data, unlabeled data
OUTPUT    self-trained model
step 1  Train a base model on the labeled data,
step 2  Label the unlabeled data with the base model,
step 3  Select data from the newly labeled data [optional],
step 4  Train a self-trained model on the original labeled data and
        the selected data from step 3 while optionally assigning
        different weights to the labeled and the selected data.
```

Figure 1.3: The self-training algorithm, adapted from McClosky (2010).

those instances from the newly labeled data that contain useful information for the next training phase. This filtering step may reduce the noise that is introduced through inevitable misclassifications. In the last step, a new model is learned from the original labeled data extended with the newly labeled data. This new model is the self-trained model.

A second option for suppressing classification errors is by assigning different weights to the newly labeled training data with respect to the original data. Applying such a weighting scheme is called *count-merging* (Chan & Ng, 2007). For some machine learners, this can be done by simply duplicating instances in the training data. Bacchiani & Roark (2003) show that count-merging, like model interpolation, is a special case of maximum a posteriori adaptation.

Changing the feature space

Instead of acting on the number of instances in the training corpus it is also possible to change the probabilities $P(X)$ by remodeling the observations.

In most machine learning setups, observations are vectors and each element of the vector represents a feature. The first step of these machine learning tasks then consists of extracting features that contain the information required to solve the task. However, there is no well-defined set of features that can be reused for every NLP task. By adding or removing features, the nature of the observations change and this has an influence on the probability $P(X)$, as can be seen in Figure 1.4. The intelligent conversion of the features of an observation X can

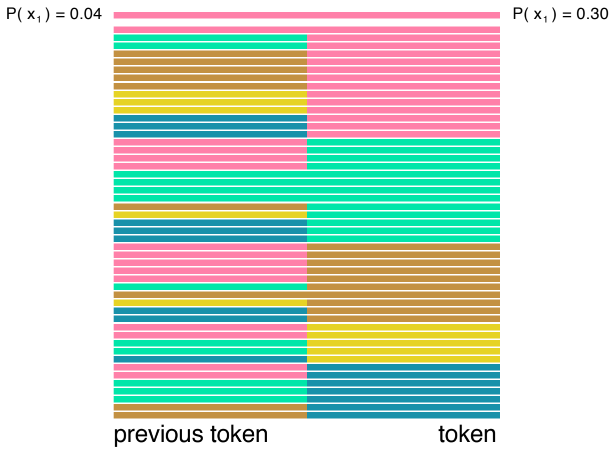


Figure 1.4: Example of how the probability of an observation changes when the feature space changes. Each line represents an observation, there are 50 observations. Each column represents a feature, there are two features: *token* and *previous token*. Each color represents a feature value, there are five different feature values: a vocabulary of five words. Both features have the same set of feature values. When only the *token* of an observation is taken into account, the first observation x_1 has an associated probability of 0.30. When the *previous token* is also taken into account – the feature space changes – the first observation x_1 has an associated probability of 0.04.

help to bring the source and target distributions closer together without losing the expressiveness of the system.

One of the better known adaptation procedures that draws upon the idea that adapting the features can lead to domain adaptation, is *structural correspondence learning* (SCL) (Blitzer et al., 2006).

Pivot features, which are features that are common in the source and target domain, are used to learn a mapping θ. This mapping then transforms the original feature space and the concatenation of the original and transformed feature space constitute the new feature space, which promises to better capture the similarities between the two domains. SCL has been shown to work for part-of-speech tagging (Blitzer et al., 2006) and sentiment classification (Blitzer et al., 2007). Plank (2009) shows that structural correspondence learning obtains better results than self-training when carrying out parse tree selection experiments.

Another example of changing the feature space is presented by Chen et al. (2009). They use a metric (maximum mean discrepancy) to transform the feature space in such a way that the distance between source and target domain is minimal without harming the in-domain accuracy on the source domain.

Globerson & Roweis (2006) argue that it is important not to assign too much weight to a feature for classifiers that incorporate feature weighting. Stressing a feature too much decreases the robustness of the classifier and, consequently, harms cross-domain usage. If a heavily weighted feature is not in the test corpus, the feature becomes useless and accuracy decreases. Globerson & Roweis (2006) try to create a classifier that deletes these sensitive features at test time, but without overly stimulating the negative effect on the accuracy. They showed that deleting features indeed increases the robustness. Missing features in the test corpus do not have a major influence on the accuracy, but, as a result, the accuracy of the adapted system on a test corpus without any missing features is lower than the accuracy of a classifier (SVM) trained on all features. However, they also show that for a spam detection task the system with feature deletion performs better than an SVM classifier, even on a complete test data. They argue that their procedure is related to, but dissimilar from, feature selection.

Although they do not change the feature space, the idea of using lexical information from the target domain in the model is also picked up by Hara et al. (2007). In an HPSG parser, they replace the source data probabilities for assigning lexical information to tokens with probabilities that are obtained from the target data.

Feature weighting

As Finkel & Manning (2009) point out, feature augmentation (Daumé III, 2007; Daumé III et al., 2010) is equivalent to their work on using a hierarchical Bayesian prior. The idea of the hierarchical Bayesian adaptation method is that feature weights can be learned from data from different domains and that they should provide corrections on the feature weights that are globally applicable. Or, more generally, language data is considered to have a common core of properties and domain-specificity of data comes from deviations from that core. For domain adaptation, it is important to capture those deviations as well as the common core.

As Finkel & Manning (2009) remark, the difference of their setup with the work of Daumé III (2007) is that in the latter paper the variances of each domain are considered to be the same. An example of hierarchical Bayesian adaptation for maximum entropy language modeling is described by Alumäe & Kurimo (2010).

1.2.3 Summary

In this section, the basic concepts of machine learning experiments are presented as they will be used in the following chapters. Together with the basic concepts, an overview is given of different approaches to domain adaptation with a focus on the various elements of the machine learning process that are available for adaptation. The second building block of this dissertation is the distance between the training and test corpus during an experiment. In the next section, two research lines that can be distinguished in distance metric research, are presented.

1.3 Measuring distance

One of the basic ideas of this work is that a measure of the difference between a source and a target corpus is needed to identify those situations in which domain adaptation is required. A good measure would also be informative to analyze the strength of domain adaptation methods. Indeed, an adaptation method that can increase the performance of an NLP tool when source and target domain are very divergent, is a more appealing method than a method than can only remedy small domain differences. In this section a short overview is given of methods to measure differences between text corpora. The focus of previous research can be purely descriptive or can be the relation between a distance measure and an NLP application. The next subsections will elaborate on the differences between these two lines of research.

1.3.1 Descriptive usage

Different kinds of properties of text corpora can be counted. A well-known approach is the Zipf curve, which is based on the counts of lexical items in a text (Zipf, 1972). However, the availability of parsers provides access to a whole range of more refined approaches. The primordial goal of counting is to characterize a corpus and express the deviation of the corpus from other corpora.

In the work of Verspoor et al. (2009) a whole range of properties of Open Access publications is compared with those of classic journal publications. Higher order syntactic phenomena like negation, passivization, conjunction, and pronominal anaphora are quantified, along with sentence length and a symmetric version of the Kullback-Leibler divergence based on token frequencies. Based on of these counts, Verspoor et al. (2009) conclude that both types of publications are similar.

Another example of the descriptive usage of measures can be found in Biber & Gray (2010). In this work, an attempt is made to find the difference between spoken register and professional academic writing by investigating discourse markers such as structural elaboration, structural compression and explicitness. Structural elaboration is measured by counting the number of finite complement clauses, non-finite complement clauses, finite adverbial clauses, finite relative clauses, and non-finite relative clauses. Structural compression is measured by counting attributive adjectives, noun as noun pre-modifiers, etc. They conclude that there are differences, but not necessarily those that were preconceived.

They show that academic writing is not more structurally *elaborated* than language with a spoken register. Elaborateness is expressed by the number of subordinate clauses in a corpus. Instead, academic writing is structurally *compressed*. Compression is expressed by the number of phrasal modiers embedded in noun phrases. They also argue that academic writing is less explicit in meaning than spoken language because of its compressed and less elaborated features.

The findings of Biber & Gray (2010) are based on the decision about what to count and how to annotate it. A working method not requiring annotator interference would not only lead to the production of more objective counts, it would also make processing of larger corpora feasible. Furthermore, automatic measures are more easily shared, enabling other researchers to apply them. One such automatic approach is clustering. It can be used to discern subdomains in a large corpus, although the descriptive dimension is somewhat lost when only clustered data is presented. Many natural language tools focus on biomedical corpora, therefore regarding *biomedicine* as a single domain, but this may be an oversimplification (Lippincott et al., 2010, 2011). Lippincott et al. (2011) choose a set of lexical, syntactic, semantic and discourse-related properties and use those properties to cluster the subdomains of a biomedical corpus.

1.3.2 In relation to an NLP task

In the previous subsection, the metrics are presented as self-contained. No external purpose is linked to the metric – the metric simply expresses distance. The issue of identifying subdomains without reference to an external purpose is addressed in the concluding section of Lippincott et al. (2011). Indeed, connecting a measure to a task gives more meaning to the measure. In this dissertation, attention is given to measures that are linked to an NLP task. Previous research that focuses on the reference of metrics to an external purpose is incorporated in the general overview of metrics in Chapter 2.

1.4 Research questions

For domain adaptation experiments, the actual domains are defined by an annotator, who bases the domain labels on his knowledge of the world. This link between domains and semantics can be easily understood by a human and may lead to sentences like:

The larger set of unlabeled data [...] *is closer to* [the target domain] *in general than to the source domain* [...] (Sagae, 2010).

This sentence may not draw any specific attention when reading a domain adaptation article, but there is an aspect that deserves more attention. What is meant when an article states that corpora are *closer* to one another? It seems trivial that a corpus of texts on genetics is closer to a corpus of microbiology texts than it is to a corpus of texts dealing with swimming contests, but what does this difference look like for a machine learner? And, how should this difference, as perceived by a machine learner, be understood?

A straightforward way to measure the distance between text corpora objectively is by running a machine learning experiment with a test and training set from the same corpus followed by an experiment with a test and a training corpus from different corpora. It is known that out-of-domain experiments decrease the performance of a machine learner, therefore the performance drop will be a measure of the distance between the different corpora. An advantage of measuring the distance in this manner is that the distance is defined using the context in which it will be used. As can be expected, different machine learning tasks may require a different notion of closeness. Two corpora that are very similar for a part-of-speech labeler may be very dissimilar for a dependency tree parser.

But there are also disadvantages associated with this manner of measuring distances: it requires labeled data and no externally defined measure of distance is provided. These two disadvantages will be explained in detail below.

One can easily think of a situation in which a distance measure that does not require labeled data becomes interesting. Consider a language engineer who has two large labeled corpora, both corpora being clearly different from the corpus on which he wants to apply a machine learner. There are unspecified constraints (storage limitations, price of the corpora, . . .) that force him to choose either one of the labeled corpora to train his system. The data that is fed to his application is unlabeled. Which training corpus should he choose? A distance measure that can work with unlabeled corpora, can help identify the training corpus that is closest to his data and, if the distance measure is well-designed, he will know that the performance of his tool will be the best given the limitations of the setup.

The absence of an externally defined distance measure leads to a one-dimensional view on the correlation between distance and performance. The missing extra dimension can become interesting in specific cases. If the correlation between a distance metric and the performance of a machine learner is well described, deviations from the correlation can help point out cases that deserve more attention. It is only by the juxtaposition of two variables – performance and distance – that cases that would otherwise go unnoticed, are picked up.

In summary, three reasons for automatic assessment of differences between unlabeled corpora can be given: (a) to estimate performance of NLP tools without the need of labeled data (Ravi et al., 2008; McClosky, 2010), (b) to be exploited during domain adaptation (Mansour et al., 2009; McClosky, 2010), (c) to provide more background information for domain adaptation experiments. A domain adaptation method that can increase the performance for an out-of-domain experiment with very different domains is more interesting than a domain adaptation method that can only remedy small domain differences.

Our research questions concerning the connection between machine learning performance and distance addressed in this work, are explained further in the remainder of this section.

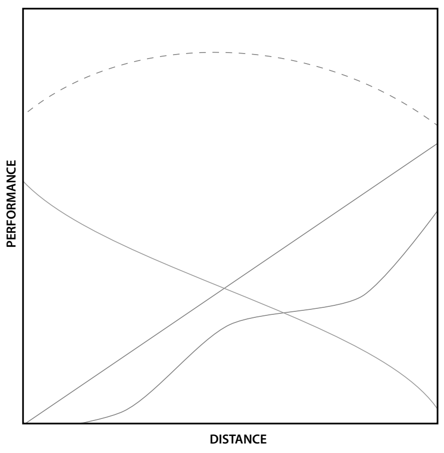


Figure 1.5: The solid lines all represent a monotonic correlation between distance and performance. The dashed line is a non-monotonic correlation.

1.4.1 Is there a monotonic relation between performance and distance?

Occasionally, distance measures are used explicitly in natural language processing studies concerning the challenges that domains impose on specific tools (Ravi et al., 2008; Chen et al., 2009; McClosky, 2010; Plank, 2011). The assumption that is made is that the distance measure that is used, is correlated monotonically (see Figure 1.5) with the performance of the natural language tool. The nature of the correlation is indeed of minor importance as long as the relation is monotonic, *i.e.* the correlation does not switch between positive and negative.

A non-monotonic correlation would hamper the usage of the distance, since more data points will be needed to infer the correlation, which could mean that finding the relation becomes unfeasible.

There are various questions related to this research question. Should the distance be expressed using only one metric from the many existing ones or should a combination of metrics be used? Are there conditions that should be satisfied to be able to observe the correlation? Is the correlation task-specific?

1.4.2 Is the correlation easily traceable?

Traceability of the correlation depends on the number of data points that are needed to retrieve the correlation, the amount of data needed to find reliable data points, and the generalizability of the correlation to other setups.

Linearity

An important question related to the previous research question is whether the connection between performance and distance can be approximated by linear regression. In fact, this question expresses a similar, but stronger condition than monotonicity. Linearity does not mean that performance should be directly linearly correlated with distance, but, ideally, there should at least be a transformation that would make it possible to relate both variables linearly.

The advantage of a linear correlation is that it can be easily captured. In the ideal case, two data points already suffice to find a linear correlation. Furthermore, a linear correlation permits using linear regression, enabling performance prediction with a certain degree of reliability.

Corpus size

Another question related to traceability is how quickly the relation can be found. Exploiting the correlation may become impractical if extremely large corpora are needed to find a correlation between performance and distance.

Machine learner independence

A last question covers the independence of the linear relation of the machine learner that is chosen for a given task. If a linear relation can be found for a machine learner adapted to a given task, does this relation persist when there is a change of machine learner?

1.4.3 Can the relation be exploited?

The most interesting question is whether the correlation between performance and distance can be used to adjust the machine learning process. With a linear correlation as the starting point, many experimental setups for the integration of a distance measure are conceivable. In this dissertation, three applications are investigated.

Training data selection

The distance between test and training corpus can be used to select only those segments of the training corpus that are likely to constitute a better training corpus.

Training data selection or instance selection can be carried out in three ways: incremental, decremental, and mixed (Jankowski & Grochowski, 2004). For incremental instance selection, the algorithm starts with a singe instance and adds instances as long as they are considered to be useful. For decremental instance selection, instances are removed from the training data when they are considered to be noise. The mixed approach makes use of both previous methods.

The reasons to reduce the size of the training corpus can be to increase the performance of a machine learner trained on the corpus, to speed up the training phase, to minimize memory requirements, or to obtain a set of prototypes. Prototypes are only a few instances that are considered to represent the entire training corpus.

Feature selection

The distance between corpora can be defined using the token frequencies of the texts, but this is not mandatory. The distance may as well be based on the features of the instances that are created for the learning task. Since each feature may capture a different aspect of a machine learning task, it may be possible to derive the quality of a feature from its influence on the correlation between distance and performance, when the feature is or is not included when calculating the distance measure.

Self-training

In a self-training setup, unlabeled data is added to the training data. The triage of unlabeled data could be done using the distance between the test corpus and the unlabeled data. The closer the corpora are to each other, the more likely adding the unlabeled data will improve the performance.

Since the basis of machine learning in NLP is that a corpus is a distribution, the information present in the differences between distributions could help treating the machine learner less as a black box, but more as an adaptive system that can change according to the data it is presented with.

1.4.4 Reproducibility of the research

Apart from the search for well-founded answers to the posed research questions, special attention has been given to reproducibility of the experiments that are carried out for this dissertation. The scripts that are used to obtain the results are made as accessible as possible, but within the boundaries that are imposed by the experimental coding routine that inevitably accompanies ongoing research. In Appendix B, on overview of the scripts is given and all scripts are available online. Attention is given to provide functioning scripts, but to fully reproduce all experiments adaptation of the source code can be needed.

1.5 Structure of this dissertation

The remainder of this work is structured as follows: Chapter 2 contains a definition of the concepts used in this work, together with an overview of various metrics. At the end of Chapter 2, some evaluation measures and statistical tests are discussed. Chapter 3 contains experiments with part-of-speech labelers and a prepositional phrase attacher in order to identify the performance-distance relation for these machine learners. Various experimental parameters are changed in order to get a clearer view on the subject. In Chapter 4, the correlation that is found in Chapter 3, is investigated by means of three applications that can benefit from the availability of a reliable distance metric. Finally, Chapter 5 contains the general conclusions of this dissertation.

Chapter 2

Divergences and other measures

2.1 Introduction

In this chapter, the definitions of various concepts that are used in this dissertation are given. Some definitions are descriptive – like the definition of a corpus – and others are mathematically defined.

First, the objects of study are described: text, corpus, distribution, and domain. An important remark is the interchangeability of the terms corpus, distribution, and domain in the context of this work. Following the descriptions of the concepts, a range of distance metrics are presented and the asymmetry property of the metrics is discussed in more detail. At the end of this chapter, all the basic building blocks of the experiments in the following chapters will have been surveyed.

2.2 Text, corpus, distribution, and domain

The terms domain, corpus, distribution, or even text need to be defined explicitly in order to avoid confusion when reading this work. These definitions are given for the use in our dissertation. No claim is made about their general validity.

Text

A text is a collection of tokens, ordered in such a way that they convey a meaning. Examples are news articles, novels, letters, but also shopping lists, information tables, or computer code.

Corpus

A corpus is a collection of texts. The collection may consist of only one text or different texts may be merged into one before calling it a corpus. In practice, a corpus will often consist of one data file.

Distribution

A distribution is a probability distribution extracted from a corpus. A distribution P can be described formally as:

$$P = \big\{p_k : p_k \in \mathbb{R}^+ \wedge \sum_i^n p_i = 1\big\} \tag{2.1}$$

with k a unique identifier and n the number of elements in P.

Extracting a distribution from a corpus can be done in various ways. Examples of distributions are token frequency, character frequency, and feature value distributions. Given a corpus, numerous different distributions can be obtained. More formally, the conversion of a corpus C to a distribution P can be stated as a mapping M on the corpus C:

$$M : \mathcal{C} \rightarrow \mathcal{D} \tag{2.2}$$

with $\mathcal{D}$ as the collection of distributions and $\mathcal{C}$ as the collection of corpora.

The mapping should be surjective (there should be a distribution for every corpus) and in most practical situations the mapping will be injective as well (every corpus is uniquely linked to a distribution and vice versa).

Although depending on the mapping, injectivity will not always hold, but it would often be the ideal situation. If injectivity would not hold, the same distance value could be obtained for pairs of corpora that are different. An example of a mapping that is not injective, would be a mapping that leads to a distribution only containing the count of the number of question marks in a corpus. For corpora of the same size, this mapping will often produce the same distribution and it is impossible to uniquely identify the original corpus if only the distribution is given. In general, this lack of expressiveness will cause the mapping to be impractical.

A commonly used mapping to extract a distribution, is getting the relative frequency counts of the tokens in a corpus, but the mapping used to create a distribution depends on the task at hand. As stated before, the number of different mappings is only limited by the imagination of the researcher. This is the reason why, for clarity, it should always be mentioned which mapping is used to extract a distribution from a corpus. However, in the practical chapters of this work, the relative token frequency mapping is assumed when the mapping is not explicitly mentioned.

Domain

The term domain is an ambiguous term and, in general, the term is used to express a difference between corpora. The exact nature of the difference depends on the context in which the term is used.

In computational linguistics, categorization of texts is done with *genre*, *register*, *text type*, *style*, and *topic* as discriminative features. Most authors that have written upon the subject, agree that these concepts are vague (Karlgren, 2010; Argamon & Koppel, 2010; Lee, 2001a). Lee (2001a) takes the definitions of the concepts as proposed by various authors, like Biber (1988) and EAGLES – a European initiative aiming at the establishment of language engineering standards (Sinclair & Ball, 1996) – and tries comparing and integrating them. The conclusions of Lee (2001a) are taken as the main source for the definitions given below.

The parameters to distinguish texts are considered to be external (intended audience, purpose, and setting) or internal (lexical or grammatical (co)occurrence features). Genres are then the outcome of the differences in the external parameters of a text. Illustrations of genre labels are: broadcast news, phone calls, letters and natural science articles. During genre labeling, the focus is on the whole text and on the practical categorization of the text. Genres are dynamic labels that are more or less based on social and/or ideological conventions. The rise of new genres, like the language used in social media, exemplifies the fact that the set of genres is dynamic. The forbidden books that were on the Index could be associated with an ideologically inspired genre label.

Register is regarded as being closely related to genre. The term *register* is used when the focus shifts more to the abstract level of internal and linguistic aspects of a text, in contrast to a focus on practical grounds, as it is for genre labels. The register is less tied to the text as a whole and is more independent from overall textual structures. Lee (2001a) gives the *legal register* as an example of register. When talking about the legal register, the focus is on the language. Parts of a text can be in the legal register while other parts can be colloquial. Texts in the legal register can belong to different genres like *testaments* and *law texts*. *Testaments* and *law texts* are social and practical categories for texts when focusing on the text as a whole. It seems unfeasible to come up with a text that belongs only partially to the testament genre.

A domain (a subject field: arts, science, religion) would be one of the attributes of a genre, along with medium (spoken, written, electronic), content (topic), purpose (informative, persuasive), type (description, narrative, argumentation), language (register, style) and form. Style would then arise from the actual linguistic choices made by an individual when producing texts within a given register and genre.

If genre (and register) are linked to the external parameters of a text, there should also exist a labeling system for the variation of the internal parameters of a text. Text types would be the outcome of differences in the internal parameters of a text. Illustrations of text types cannot be given, because it is hard to define different text types – text types should be defined crossing genres and based purely on internal parameters. For this reason text type is considered an elusive concept by Lee (2001a) and one that is hard to put into practice.

Overlooking all these definitions, it is clear that the concepts are vague, overlapping and mostly valued for their practicality in a given context. A common trait of the definitions is that they do not contain a scale to distinguish between

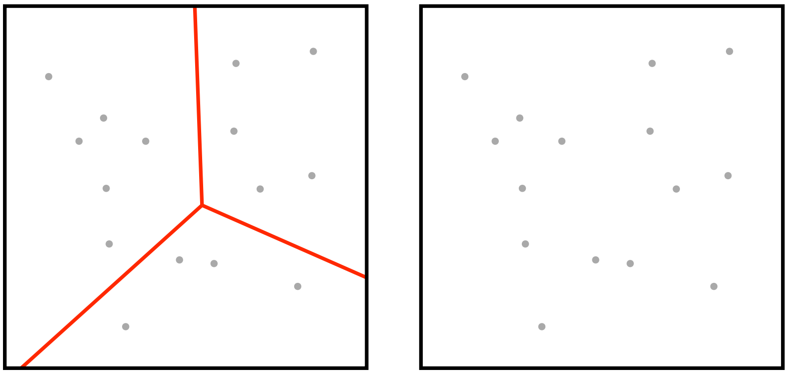


Figure 2.1: Domain classification and domain discovery – the dots represent corpora, the lines separate domains. For domain classification (left), the domains are given and the task consists of positioning a new corpus in the space. The domain of the new corpus can be retrieved from its position. For domain discovery (right), the domain boundaries between the corpora are to be found.

subgenres, genres and supergenres. For example, is the distinction between topic and domain (as defined in the previous paragraph) not merely an issue of looking at a text at a different scale?

Biber (1988) presented work on finding the dimensions that separate different genres. The dimensions are constructed from linguistics features such as the verb tense, the type of adverbials and pronouns, the use of passive voice and so on. All features are represented by relative frequency counts. The work is a nice example of how genres can be characterized objectively, but it focuses more on genre classification than on genre discovery. Figure 2.1 illustrates the difference between genre classification and genre discovery. The points in the figure depict corpora or texts. The lines are boundaries between genres. At the left of Figure 2.1, the boundaries are given by an annotator and the task consists of positioning a new text in the space. This is genre/domain classification. By positioning a text in the space, a genre label can be retrieved. There is no other ground to the genre label than the intuition of the annotator, who assigned genre labels to the texts that are already present in the space. Genre classification is a supervised task.

At the right of Figure 2.1, there are no boundaries and the task consists of finding the genre boundaries between the texts. This unsupervised task is called genre/domain discovery. Genre discovery is similar to clustering and the same problem occurs: what should be the threshold to separate one cluster from another? A second question that arises when carrying out genre classification and genre discovery is how the space is constructed. Are the dimensions of the space the combined linguistic dimensions of Biber (1988) or is a one-dimensional space built up by a single metric sufficient? These are relevant questions for which there may not be a general answer.

For our research, a pragmatic approach is adopted because of the difficulty of finding a theoretical ground for the presented concepts such as genre, domain, and register. The decisions made by the annotators while composing a text corpus containing any categorization are taken to be correct in the sense that we assume that the categories are distinct and on the same level of granularity. It is only by following the decisions of the annotators and not imposing our view on domains that we can introduce a sense of objectivity. By only preserving the notion of categories that the annotators considered to be distinctive and socially/pragmatically defendable, we also abstract from the difference between genre and domain and, for convenience's sake, all further categories in this work will be referred to as *domains*.

In the previous paragraph, it was shown that there is no other ground for the domain labels than the annotator's intuition. In search for an objective ground, a supplementary source is introduced for the experiments in the remainder of this work. It is the link between the accuracy of a machine learner that is designed for a given task and the discriminating factor that will provide the ground for the domain labeling. By exploring this link, it is possible to mix the domain reproducing task with the domain discovery task. This working method assures that the domains that are labeled by applying a discriminating factor (domain reproduction), are significant for the machine-learning task in the sense that the amplitude of the discriminating factor will result in different cross-domain accuracies. Putting a threshold on the accuracy differences between different domains leads to the opportunity to use the discriminating factor for domain discovery.

In the technical chapters, the terms domain, corpus, and distribution are used as synonyms because they are all different views on the same concept: A domain becomes a corpus when one wants to examine a domain and starts gathering texts. With a given mapping, a corpus can be uniquely linked to a distribution and vice versa – creating a sense of isomorphism between corpora and distributions.

2.3 Definition of a metric

Throughout this work, we will use the terms *metric*, *distance* and *divergence* as synonyms for any function that returns a real number when applied on any pair of corpora or on the derived distributions. The usage of these terms diverges from the usage in an exact, mathematical context. In a mathematical context a distance metric, d, is a symmetric, positive definite function that satisfies the triangle inequality for any pair of points.

$$d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$
$$\forall x, y, z \in \mathcal{X} :$$

$$\begin{aligned}
&d(x, y) \geq 0 \text{ (non-negativity)}\\
&d(x, y) = 0 \iff x = y \text{ (identity)}\\
&d(x, y) = d(y, x) \text{ (symmetry)}\\
&d(x, z) \leq d(x, y) + d(y, z) \text{ (triangle inequality)}
\end{aligned}$$

where $\mathcal{X}$ can be the collection of corpora or of the distributions extracted from the corpora, using a given mapping.

In statistics, a divergence can be defined by dropping the symmetry and triangle inequality conditions from the distance definition. Many functions in this dissertation do not satisfy all of the distance or divergence conditions. For example, Kendall's τ does not satisfy the non-negativity condition, perplexity does not satisfy the identity condition, and the Kullback-Leibler divergence does not satisfy the symmetry condition and the triangle inequality. As we will see later, not satisfying the symmetry condition makes a divergence better suited for the task at hand. On the other hand, not satisfying the triangle inequality condition makes a divergence unsuitable for drawing a similarity map of the corpora for a given task. Such a map could be used to infer a functional relation between a position on the map and the outcome of a machine-learning task. It is fairly easy to see why all conditions should hold in order to be able to produce a map. A visual clue of why triangle inequality is necessary for a topological representation of domains can be found in Appendix A on page 161.

In short, when the term distance, metric, or divergence is used in this work, the following function is targeted:

$$d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \tag{2.3}$$

2.4 Distance metrics for corpora

In this chapter we will give an overview of a collection of metrics that can be used to measure the distance between two corpora P and Q. A corpus can be a text, a tokenized text, a list of instances consisting of features of any kind, dependency trees, etc. In general, any collection of elements can be used, but in this work we will most of the time use the divergences on tokenized sentences.

Probability based metrics Since a corpus can be easily rewritten in a probability distribution of tokens it is not surprising that metrics using these probabilities are commonly used. In this category we list the metrics that are an uncomplicated combination of those token probabilities.

Let

$$P = \left\{ p_k : p_k = \frac{\text{count of token } k \text{ in corpus P}}{\text{total number of tokens in P}} \right\} \tag{2.4}$$

and

$$Q = \left\{ q_k : q_k = \frac{\text{count of token } k \text{ in corpus Q}}{\text{total number of tokens in Q}} \right\} \tag{2.5}$$

Then a well-known divergence metric has been developed by Kullback & Leibler (1951):

$$KL(P;Q) = \sum_k p_k log_2\left(\frac{p_k}{q_k}\right) \tag{2.6}$$

A nice example of the usage of the Kullback-Leibler divergence, KL, comes from Della Pietra et al. (1997) where the KL-divergence is used for feature selection by adding only those features that decrease the divergence. Another example can be found in Daumé III & Marcu (2006). They use the Kullback-Leibler divergence to compute the similarity between unigram language models.

The Kullback-Leibler divergence is asymmetric:

$$KL(P;Q) \neq KL(Q;P) \tag{2.7}$$

From the formula of the Kullback-Leibler divergence it is clear that the divergence is not defined when P and/or Q contains a token that is not in the other corpus. For natural language corpora this is often the case, making KL impractical. The Jensen-Shannon and the Skew divergence that are introduced below can offer solutions for this problem. Another option is to smooth the divergence in such a way that $log_2\left(\frac{p_k}{q_k}\right)$ is also defined for tokens where $p_k = 0$ or $q_k = 0$. For the experiments in the following chapters, we used smoothing:

$$log_2\left(\frac{p_k}{q_k}\right) = \begin{cases} 0 & \text{for } p_k = 0, \\ log_2\left(\frac{p_k}{q_k+\epsilon}\right) & \text{for } q_k = 0. \end{cases} \tag{2.8}$$

With ϵ set to a very small value. A more general form of the Kullback-Leibler divergence is the divergence developed by Rényi (1961):

$$\textit{Rényi}(P;Q;\alpha) = \frac{1}{(\alpha-1)} log_2\left(\sum_k p_k^{\alpha} q_k^{1-\alpha}\right) \text{ with } \alpha \geq 0 \tag{2.9}$$

This metric is equal to the Kullback-Leibler divergence when $\lim_{\alpha\to 1}$. In the selected situations when α is set to a value strictly greater than 1, the Rényi divergence may need smoothing in the same manner as the Kullback-Leibler divergence. Mansour et al. (2009) argue that the class of Rényi divergences are suitable in the context of combining multiple sources into one training set.

If α is set to $\frac{1}{2}$, the Rényi divergence becomes $-2log(Bhat)$ with $Bhat$ as the Bhattacharyya coefficient (Bhattacharyya, 1943). The Bhattacharyya coefficient can be turned into a true mathematical distance metric that obeys the triangle inequality (Kailath, 1967):

$$Bhat(P;Q) = \sqrt{1 - \Big(\sum_k \sqrt{p_k q_k}\Big)} \tag{2.10}$$

This distance may also be called the Hellinger distance (Hellinger, 1909), but we did not retrieve the articles for Hellinger (1909) and Bhattacharyya (1943) so we just follow Kailath (1967).

The Jensen-Shannon divergence (Lin, 1991) is an alternative to the KL-divergence that does not require every token in P to occur also in Q – making smoothing redundant.

$$JS(P;Q) = \frac{1}{2}\Bigg[KL\Big(P; \frac{P+Q}{2}\Big) + KL\Big(Q; \frac{P+Q}{2}\Big)\Bigg] \tag{2.11}$$

Grosse et al. (2002) use a version of the Jensen-Shannon divergence that contains weighting. The Jensen-Shannon divergence can be rewritten if we replace the KL-divergences by their definitions:

$$\begin{aligned}
JS(P;Q) &= \frac{1}{2}\Bigg[\sum_k p_k log_2\Big(\frac{p_k}{\frac{p_k+q_k}{2}}\Big) + \sum_k q_k log_2\Big(\frac{q_k}{\frac{p_k+q_k}{2}}\Big)\Bigg] && (2.12)\\
&= \frac{1}{2}\Bigg[\sum_k p_k log_2(p_k) - \sum_k p_k log_2\Big(\frac{p_k+q_k}{2}\Big) + \\
&\quad \sum_k q_k log_2(q_k) - \sum_k q_k log_2\Big(\frac{p_k+q_k}{2}\Big)\Bigg] && (2.13)\\
&= \frac{1}{2}\Bigg[\sum_k p_k log_2(p_k) + \sum_k q_k log_2(q_k) - \\
&\quad 2\sum_k \frac{p_k+q_k}{2} log_2\Big(\frac{p_k+q_k}{2}\Big)\Bigg] && (2.14)
\end{aligned}$$

It is easier to see where the weighting is included if the Jensen-Shannon entropy is expressed by means of the Shannon entropy H of a distribution P:

$$H(P) = -\sum_k p_k log_2(p_k) \tag{2.15}$$

This entropy can be included in the JS formula:

$$\begin{align} JS(P;Q) &= \frac{1}{2}\Bigg[-H(P) - H(Q) + 2H\Big(\frac{P+Q}{2}\Big)\Bigg] \tag{2.16}\\ &= H\big(\frac{1}{2}P + \frac{1}{2}Q\big) - \frac{1}{2}H(P) - \frac{1}{2}H(Q) \tag{2.17}\end{align}$$

A parameter λ can be introduced and the weighted version of the Jensen-Shannon divergence then becomes:

$$JS(P;Q;\lambda) = H\big(\lambda P + (1-\lambda)Q\big) - \lambda H(P) - (1-\lambda)H(Q) \tag{2.18}$$

with $0 < \lambda < 1$ and with $\lambda = \frac{1}{2}$, the weighted version becomes equal to the unweighted version.

Another alternative to the KL-divergence is the Skew divergence (Lee, 1999). The purpose of the Skew divergence is related to the purpose of the Jensen-Shannon divergence: it is a weighted version of the KL-divergence to enable the comparison between distributions when $p_k \neq 0$ and $q_k = 0$.

$$Skew(P;Q) = KL(Q; \alpha P + (1-\alpha)Q) \text{ with } \alpha \in [0,1] \tag{2.19}$$

Lee (2001b) discusses that the Skew divergence is better suited for estimating co-occurrence probabilities of words than the KL-divergence. In her work, she also presents an overview of other well-known metrics like Euclidean, Variational or L1, and Cosine divergence:

$$Euclidean(P;Q) = \sqrt{\sum_k (p_k - q_k)^2} \tag{2.20}$$

$$L1(P;Q) = \sum_k |p_k - q_k| \tag{2.21}$$

$$Cosine(P;Q) = \frac{\sum_k p_k q_k}{\sqrt{\sum_k p_k^2}\sqrt{\sum_k q_k^2}} \tag{2.22}$$

McClosky (2010) uses CosineTopK, which is a variant to the cosine divergence. Instead of taking the entire distribution only the K most frequent elements from the joint distribution are taken into account when computing the cosine divergence.

The recurrence metric (Kessler, 2001), R, can be linked to already mentioned metrics in the following way:

$$R(P;Q) = \sum_k |p_k - \frac{p_k + q_k}{2}| \tag{2.23}$$

which reduces to $R = \frac{L1}{2}$, and Rsq:

$$Rsq(P;Q) = \sum_k \left(p_k - \frac{p_k + q_k}{2}\right)^2 \tag{2.24}$$

which reduces to $Rsq = \frac{1}{4}(Euclidean)^2$.

The previous metrics are based on the token frequencies to produce a divergence score. The τ_α and τ_β scores, also called Kendall's τ, do not directly reflect the frequency difference, but they inform about the variable ordering of the token frequencies of two corpora P and Q. For this metric, both corpora should have the same number of tokens n (Kendall, 1970).

$$\tau_\alpha(P;Q) = \frac{2}{n(n-1)} \sum_{k_1} \sum_{k_2} sign\big((p_{k_1} - p_{k_2})(q_{k_1} - q_{k_2})\big) \tag{2.25}$$

$$\tau_\beta(P;Q) = \frac{2}{F} \sum_{k_1} \sum_{k_2} sign\big((p_{k_1} - p_{k_2})(q_{k_1} - q_{k_2})\big) \tag{2.26}$$

with

$$F = \sqrt{\big(n(n-1) - 2U\big)\big(n(n-1) - 2V\big)}$$
$$U = \text{number of ties in P, i.e. size of } \big\{(p_{k_1}, p_{k_2}) | p_{k_1} = p_{k_2}\big\}$$
$$V = \text{number of ties in Q, i.e. size of } \big\{(q_{k_1}, q_{k_2}) | q_{k_1} = q_{k_2}\big\}$$

From the formulas, it can be seen that $\tau_\alpha = \tau_\beta$ when there are no ties. The values of τ lie between 1 (when the ordering of the frequencies is the same for P and Q) and -1 (when the ordering of the frequencies in P is the reverse of the one from Q) and $|\tau_\alpha| \leq |\tau_\beta|$. In general, τ_β is used when both distributions are equivalent and τ_α is used when one distribution is a reference and the other an estimation of that reference. We refer to Appendix A on page 174 for an example adapted to token frequency counts.

Section A.4.3 in the appendix explains why Kendall's τ has some undesirable properties when trying to distinguish between domains. In addition, computing τ is computationally expensive and unfeasible for larger corpora. Therefore, τ will not be used in the experiments in this thesis.

Another set of divergence metrics is not rooted in the difference between probability distributions: average sentence length (ASL), simple unknown word rate (sUWR) and overlap, unknown POS trigram ratio (UPTR), and perplexity. Zhang & Wang (2009) and Ravi et al. (2008) apply them in an accuracy estimation context and define them as the following:

Average sentence length (ASL) (Zhang & Wang, 2009): The average length of the sentences of a corpus. All tokens, w, of the tokenized corpus, T, are counted and divided by the number of sentences, s, in the corpus, *i.e.*

$$ASL = \frac{|\{w | w \in T\}|}{|\{s | s \in T\}|} \tag{2.27}$$

The conversion of the average sentence length into a metric can be done in numerous ways. Four implementations are presented here:

$$ASL_1 = ASL_{testing} \tag{2.28}$$

$$ASL_2 = \left|ASL_{testing} - ASL_{training}\right| \tag{2.29}$$

$$ASL_3 = \frac{ASL_{testing}}{ASL_{training}} \tag{2.30}$$

$$ASL_4 = ASL_{testing} - ASL_{training} \tag{2.31}$$

ASL_1 is the version used in Zhang & Wang (2009).

Simple Unknown Word Rate (sUWR): This can only be computed by using the training corpus, S, and testing corpus, T. sUWR is the fraction of word types, t, in the testing corpus that are not in the training corpus, *i.e.*

$$sUWR = \frac{|\{t|t \notin S \wedge t \in T\}|}{|\{t|t \in T\}|} \tag{2.32}$$

(Zhang & Wang, 2009; Plank & van Noord, 2010)

Overlap: Overlap is a complement to the sUWR. The same annotation as for the sUWR definition is used:

$$Overlap = \frac{|\{t|t \in S \wedge t \notin T\}|}{|\{t|t \in S\}|} \tag{2.33}$$

Unknown POS Trigram Ratio (UPTR): This measure is the same as sUWR, except for the usage of POS trigrams instead of word types. It is possible to include sentence initial and sentence final markers to stress the position of a trigram in the sentence. (Zhang & Wang, 2009; Plank & van Noord, 2010)

Perplexity: Perplexity, *ppl*, of a corpus is computed with the formula:

$$ppl = 10^{\frac{-logprob}{w-oov+s}} \tag{2.34}$$

with w the number of words, s the number of sentences, and oov the number of tokens that occur in the test corpus, but not in the training corpus. *logprob* is the total probability of the sentences in the testing corpus when using a language model based on the training corpus. It is also possible to compute the perplexity without taking into account the number of sentences s. Perplexity has been used to measure distances for language modeling (Gao et al., 2002;

Klakow & Peters, 2002; Moore & Lewis, 2010). Cross-entropy is related to perplexity and Chen (2009) shows that it is possible to use training set cross-entropy together with other model statistics to model test set cross entropy for n-gram language models.

Plank & van Noord (2010), like Zhang & Wang (2009), find that applying three dependency parsers on data from different domains is rather robust to ASL_1, meaning that they didn't observe a correlation between the accuracy and the ASL_1. For sUWR, they found a high correlation for data-driven parsers and for UPTR, they observed no correlation, unlike Zhang & Wang (2009). Perplexity displayed the highest degree of correlation, with correlation coefficients between -0.64 and -0.57.

A more elaborate measure has been proposed by Blitzer et al. (2007): the $\mathcal{A}$-distance (Kifer et al., 2004). For this measure the actual instances, as they will be used in the machine-learning task, are extracted. Only the instances coming from the source domain are associated with a class label, but the class labels are not required to compute the $\mathcal{A}$-distance. Blitzer et al. (2007) propose a linear classifier as a means of measuring the similarity between the two domains. The Huber loss (Huber, 1964) then serves as a proxy of the $\mathcal{A}$-distance. The resulting proxy $\mathcal{A}$-distance correlates with the adaptation loss they observe when applying a sentiment classification tool on different domains. Originally, the Huber loss, ρ, has been stated as:

$$\rho(y, f(x)) = \begin{cases} \frac{1}{2}\left[y - f(x)\right]^2 & \text{for } |y - f(x)| < \delta, \\ \delta|y - f(x)| - \frac{1}{2}\delta^2 & \text{for } |y - f(x)| \geq \delta. \end{cases} \tag{2.35}$$

with y a (random) variable and $f(x)$ an estimation of that variable. δ is a parameter depending on the fraction of ys that do not follow the normal distribution (Huber, 1964). Huber loss can be adapted to the output of a linear classifier, like an SVM, if we take y to be the reference class labels and $f(x)$ to be the predicted class label. If we take $y, f(x) \in \{-1, 1\}$, $\delta > 2$, and divide the loss with a factor 2, then:

$$\rho(y, f(x)) = \begin{cases} 0 & \text{for } y = f(x), \\ 1 & \text{for } y \neq f(x). \end{cases} \tag{2.36}$$

To compute the proxy $\mathcal{A}$-distance, instances from the source domain are labeled with -1, instance from the target domain are labeled with 1. The instances of the two domains are concatenated, an SVM is trained upon this data set, and the Huber loss, is retrieved. The proxy $\mathcal{A}$-distance then evaluates to:

$$\text{proxy } \mathcal{A}\text{-distance} = 100(1 - \text{Huber loss}) \tag{2.37}$$

A more theoretical approach of the $\mathcal{A}$-distance, after renaming it to the $\mathcal{H}$-divergence, can be found in Ben-David et al. (2010).

Maximum mean discrepancy (MMD) is a test statistic introduced by Gretton et al. (2007) to find out if a distribution P is different from a distribution Q using only samples of both distributions. Such a sample can be a collection of unlabeled instances fit for a machine-learning task. When the MMD between the samples of two distributions is large, the distance between the distributions can be considered to be large and we expect a lower accuracy if one distribution is the training corpus and the other is the test corpus. The unbiased estimate of the square of the MMD is given by Gretton et al. (2007):

$$\text{MMD}^2 = \frac{1}{m(m-1)} \sum_{i \neq j}^{m} \Big[k(p_i, p_j) + k(q_i, q_j) - k(p_i, q_j) - k(p_j, q_i) \Big] \tag{2.38}$$

with

$$\begin{aligned} p, q &: p, q \in \mathbb{R}^d\text{; the instances from } P \text{ and } Q \\ d &: \text{the number of features of an instance } p, q \\ m &: \text{the size of the two distributions } P, Q \\ k &: \text{the reproducing kernel} \end{aligned}$$

When comparing corpora using only the frequency counts, we can say that $d = 1$, *viz.* the frequency value itself. Gretton et al. (2007) define constraints on the type of kernels that can be plugged in in the formula of the MMD^2. Gaussian and Laplace kernels meet these constraints. The Gaussian radial basis function kernel is defined as follows:

$$k(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{2.39}$$

The Gaussian radial basis function kernel is symmetric, leading to the reduction $k(p_i, q_j) = k(p_j, q_i)$ in the formula of MMD^2.

An implementation of the MMD for a document classification and an information extraction task is presented by Chen et al. (2009). In their work they minimize the MMD between the source and target domain during linear transformation of the feature space. Combined minimization of the MMD and minimization of the loss on the source domain ensures a better result in the final out-of-domain classification task.

Satpal & Sarawagi (2007) take the same approach as Chen et al. (2009), but they suggest a distance metric based on the expected value of every feature of an instance. For the source domain the expected value of the k^{th} feature is defined as:

$$E^k_{source} = \frac{1}{n}\sum_i^n f_k(x_i, y_i) \tag{2.40}$$

with n the number of instances for the source domain and $f_k(x_i, y_i)$ giving the value of the k^{th} feature of the i^{th} instance. In the paper of Satpal & Sarawagi (2007), f_k does not only depend on the instance, x, but also on the class label, y. Therefore they have to introduce a different formula to compute the expected value for the target domain:

$$E^k_{target} = \frac{1}{m}\sum_i^m \sum_y f_k(x_i, y)\Pr(y|x_i, w) \tag{2.41}$$

with m the number of instances for the target domain and with $\Pr(y|x_i, w)$ the probability of a class label y to be associated with feature vector x carrying weight w. E^k_{target} becomes the same as E^k_{source} if the actual value of a feature does not depend on the class label, $f_k(x_i, y_i) = f_k(x_i)$ .The actual distance, $\triangle E$, amounts to:

$$\triangle E = \sum_k^K d(E^k_{source}, E^k_{target}) \tag{2.42}$$

with K the number of features.

Satpal & Sarawagi (2007) use the square distance, $d(E^k_{source}, E^k_{target}) = (E^k_{source} - E^k_{target})^2$, as distance function d. It is clear that this metric is not straightforwardly usable for nominal features.

Daumé III & Marcu (2006) introduce a domain adaptation framework containing a parameter π. This parameter expresses the degree of relatedness of domains. Low values of specific π parameters indicate that the in-domain and out-of-domain data differ significantly. We do not go deeper into this parameter, since the parameter is tied to a specific framework, which makes it less generally applicable than other distance metrics.

The Mahalanobis distance (Mahalanobis, 1936) has been used in natural language tasks (Stamatatos et al., 2000; Shimizu et al., 2008; Dhillon et al., 2010). In its original form, it is suited for measuring the distance of one observation to a collection of observations rather than for measuring the distance between corpora as defined in this work. The original distance is defined as:

$$Mahalanobis(\mathbf{x}; Y) = \sqrt{(\mathbf{x} - \mu)^T S^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{2.43}$$

With $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ an n-dimensional point, Y an $m \times n$-dimensional matrix containing m n-dimensional points, $\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{bmatrix}$ the means of every coordinate of the points in Y, and S the covariance matrix of Y. In this work, we only have one vector $\mathbf{x}$ for corpus P and one for corpus Q making it impossible to compute the original Mahalanobis distance. For this reason, Weinberger & Saul (2009) introduce an extension to the notion of Mahalanobis distance by substituting $\boldsymbol{\mu}$ with an n-dimensional point $\mathbf{y}$ and by replacing the covariance matrix S by a positive semi-definite matrix M:

$$Mahalanobis(\mathbf{x}; \mathbf{y}; M)^2 = (\mathbf{x} - \mathbf{y})^T M (\mathbf{x} - \mathbf{y}) \tag{2.44}$$

If M is the identity matrix, the Mahalanobis distance equals the Euclidean distance. In fact, the Mahalanobis distance can be seen as a linear transformation of the Euclidean distance. When M is symmetric it can be decomposed as $M = L^TL$ and substituting this decomposition in the Mahalanobis equation gives:

$$\begin{aligned} Mahalanobis(\mathbf{x};\mathbf{y};M)^2 =& (\mathbf{x}-\mathbf{y})^T L^T L(\mathbf{x}-\mathbf{y}) && (2.45)\\ =& \big((\mathbf{x}-\mathbf{y})^T L^T\big)\big(L(\mathbf{x}-\mathbf{y})\big) && (2.46)\\ =& \big(L(\mathbf{x}-\mathbf{y})\big)^T\big(L(\mathbf{x}-\mathbf{y})\big) && (2.47)\end{aligned}$$

Which is a linear transformation L of the differences. When M is the identity matrix, so are L and L^T, and the distance becomes:

$$Mahalanobis(\mathbf{x};\mathbf{y})^2 = (\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y}) \tag{2.48}$$

which is the matrix representation of the Euclidean distance as defined on page 33 with $\mathbf{x} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix}$ and n the number of unique tokens in the union of corpus P and corpus Q. When applying the Mahalanobis distance on corpora, the matrix M can be regarded as a parameter that has to be learned or chosen. An algorithm – called ITML – to compute M in a supervised way, has been proposed and implemented by Davis et al. (2007). Dhillon et al. (2010) extend this algorithm to the usage of partially unlabeled data to assess M in a semi-supervised way. The Mahalanobis metric will not be used in this work, mainly because of computational restrictions, but an elementary comparison is made in Appendix A.2.

All metrics can be applied to any pair of distributions – they are not restricted to the use on token frequencies. Plank (2011) implements a distance that is based on the distribution of topics of segments (documents) in the training corpus. The topics are automatically extracted using *latent dirichlet allocation*(LDA, Blei et al. (2003)) and are distributions over words. Plank (2011) finds that using topic model-based divergences works slightly better than token-based divergences for parsing accuracy.

It can be expected that not every metric captures the same information about a corpus. To get a more detailed characterization of a corpus it may be a reasoned choice to construct a statistical model based upon a collection of metrics. Ravi et al. (2008) and McClosky (2010) build a linear model for a parser based upon cosine distance, average sentence length, sUWR and others. Plank (2011) tries to combine a token frequency based measure with a measure based on topic models to carry out training data selection.

We presented the above metrics because they can all be implemented using the token frequencies or other uncomplicated aspects of any natural language corpus. Since we are looking for a stable, readily employable, multipurpose metric, these are all interesting candidates. Nevertheless, many other metrics are conceivable. An example of a metric that uses node deletions/additions in dependency trees to calculate the superficial syntactic difference between languages can be found in Homola & Kuboň (2006). The drawback of such more complicated measures is that they rely on other resources besides the tokenized corpus. The extra resources, for example a dependency parser, may interfere with the measurement and may limit the applicability of the metric. It is trivial that measuring the distance using part-of-speech tags when trying to find a linear correlation to the outcome of a part-of-speech tagger is pointless – one may as well use the accuracy score of the part-of-speech tagger as a measure of distance. Because of these issues, we will not go deeper into the usage of measures based on automatic linguistic analysis in the remainder of this work.

To conclude this section, the most important metrics are presented in a table listing some of the properties. Table 2.1 contains information on whether the metric returns the same distance when P and Q are switched (column 1), whether the computed distance is bound to an interval or not (column 2), and whether the metric is defined when there are tokens that occur only in one corpus (column 3). For example, sUWR returns distance values between 0 and 1 and the value is unlinked to the order in which P and Q are used. The fact that the sUWR is defined for tokens that have a zero frequency is irrelevant, since the metric does not use token frequencies as such. An underscore in column 3 means that the metric as defined by its formula, can handle tokens that only occur in either P or Q.

2.5 Symmetry and asymmetry

All metrics presented in Section 2.4 have their own properties and they will highlight various aspects of a distribution – a metric can be sensitive to small changes in the distributions, may need lots of data to reach a stable value, etc.

metric	symmetry	distance range	zero frequency token
Kullback-Leibler	asymmetric	≥ 0	smoothing
Rényi	asymmetric	depending on α; not limited	depending on α
Bhattacharyya distance	symmetric	≥ 0	
Jensen-Shannon	symmetric	[0, 1]	–
Skew divergence	asymmetric	≥ 0	–
Euclidean	symmetric	≥ 0	–
Variational (L1)	symmetric	≥ 0	–
Cosine	symmetric	[0, 1]	–
τ	symmetric	[-1, 1]	–
ASL_1	asymmetric	≥ 1	irrelevant
ASL_2	symmetric	≥ 0	irrelevant
ASL_3	asymmetric	> 0	irrelevant
ASL_4	asymmetric	$]-\infty, +\infty[$	irrelevant
sUWR	asymmetric	[0, 1]	irrelevant
perplexity	asymmetric	≥ 0	irrelevant
proxy $\mathcal{A}$-distance	asymmetric	[0, 100]	irrelevant
MMD^2	symmetric	≥ 0	–

Table 2.1: Properties of various metrics – symmetry gives information about the importance of the order of the distributions when a metric is applied, the range of possible values of a metric is given in the distance range column, the zero token frequency column gives information about the needs when an element occurs in only one distribution. The value *irrelevant* means that the metric is not based on the probabilities.

One striking difference between the metrics is that there are symmetric and asymmetric metrics. This is also called the *commutative property.*

The potential importance of asymmetry is illustrated with Table 2.2. In this table, the label distribution of unspecified observations x is given according to two corpora coming from different domains. This is the same situation as in any cross-domain binary classification task. As can be seen, $P(A|x)$, the chance that observation x gets label A, is 99% for corpus P and 20% for corpus Q. This means that when corpus P is used as the training corpus there will be a high tendency to predict label A, harming the recall of instances with label B. The majority of the observations in corpus Q carry label B, meaning that the overall accuracy will suffer from this imbalance. On the other hand, when corpus Q is used as training and corpus P as test, there will be a high tendency to predict label B, harming the recall of instances with label A. Since the predominance of label A is even more outspoken than the predominance of B in the previous situation, the overall accuracy will suffer more. To sum up, the accuracy drop

	corpus P	corpus Q
label A	99%	20%
label B	1%	80%

Table 2.2: Different $P(y|x)$ in two corpora – a given observation x carries the A label for 99% of its occurrences in corpus P and for 1% of the occurrences it carries label B. For corpus Q, these figures are different and in reversed order.

can be expected to drop more or less depending on which corpus is used as training and as test.

Although the explanation above seems to be valid, a remark has to be made. If there would be no relation between an observation and its label, the asymmetry would disappear. Reconsider the probabilities of Table 2.2. If we use corpus Q as the training corpus, and assign labels to the instances of corpus P using the probabilities of Q – without doing any machine learning using features from the observations x – then 20% of 99% of the instances of corpus P will be labeled correctly with label A and 1%×80% instances would be correctly labeled as B. These figures do not undergo any influence when the test and training corpus are switched[1], making the *direction* of the experiment unrelated to the expected accuracy.

On the basis of the discussion above, it is possible to say that a machine learner that assigns labels with a probability that is strongly related to the probability of the occurrence of the labels in its training corpus, shows less of the asymmetry effect than a machine learner that tries to abstract from the training distribution by using the features of the observations. In the latter situation, a better correlation between the accuracy of out-of-domain experiments and the distance between training and test corpus can be expected if the distance metric takes into account this asymmetric behavior.

2.6 Implementation of the metrics

In the previous sections, the theoretical definitions of the metrics are given, but some decisions that are made during implementation can be equally important. For this reason, the implementation of the metrics as they are used, should be available. Some decisions are worth to be mentioned explicitly:

[1] 20% of 99% equals 99% of 20% and 1% of 80% equals 80% of 1%.

In the definition of the metrics, Q will be instantiated with the distribution derived from the training corpus and P will be the distribution derived from the test corpus.

The Rényi divergence has been implemented with the factor $\frac{1}{1-\alpha}$ instead of $\frac{1}{\alpha-1}$. This has no influence on the conclusions that can be drawn from the experiments. The only difference resulting from this adaptation is the sign of the Rényi divergence: the more negative the distance, the further the two measured distributions are from each other. The main advantage is that in a distance–performance plot the performance increases when moving towards the right of the plot, which is more intuitive for a left-to-right reader.

For the Kullback-Leibler divergence, a smoother version is used and the smoothing factor ϵ is set to 2^{-52}.

These are only a few choices that were made during research, but it already illustrates the importance of small implementation decisions: switching training and test corpus in an asymmetric metric leads to different conclusions, the sign of the Rényi divergence influences the graphs, and using a different smoothing ϵ in the Kullback-Leibler divergence leads to different distance values. To get a better insight in the metrics, the source code of the metrics is made available online and Appendix B contains an overview of the presented scripts.

2.7 Conclusions

In this chapter, some basic concepts and how they should be understood in the context of this work, are described, while the bulk of the chapter contains a survey of a range of distance metrics that will be used in the following chapters. The overview of metrics is followed by a discussion about the asymmetry property of metrics.

In the next chapter, the actual implementation and the applicability of the distance metrics will be tested by means of different corpora and different machine learners. The main point of interest will be to find an operable relationship between the metrics and machine learning performance.

Chapter 3

Domain distance and performance

3.1 Introduction

Although there are many ways to derive a distribution from a corpus, in many computational linguistics experiments a corpus is regarded as a sample of an underlying distribution of tokens. If this assumption is followed, a lot of methods become available to measure the divergence between different corpora, since there exist a lot of metrics to compute the divergence between probability distributions (Chapter 2). However, a question arises: do divergence metrics capture something *meaningful* when they are applied to linguistic corpora? With the term meaningful, we express the requirement that a metric can be used for a specific task, but also that there is a sense of logic, maybe only intuitively, when the metric is interpreted.

As discussed in Chapter 1, in general, a machine learning experiment consists of a training corpus and a test corpus. The performance of the machine learner can be estimated by training the machine learner on the training corpus and applying it to the test corpus. At the same time, it is possible to measure the distance between the training and the test corpus. Intuition tells us that there should be a correlation between the distance between corpora and the performance of a machine learner trained and tested on those corpora: the closer the training and test corpus are to each other, the higher the performance. This chapter is dedicated to testing that intuition and to finding to best measure for the distance.

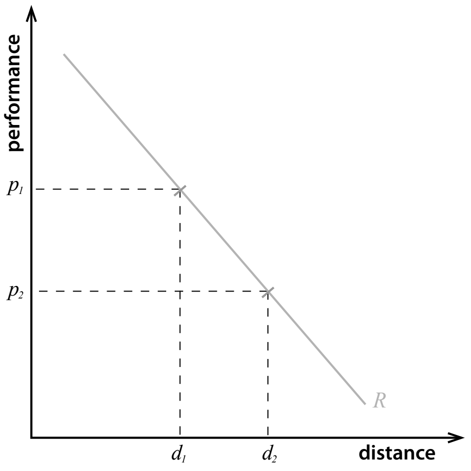


Figure 3.1: Data points on the distance–performance plot – a larger distance between the test and training corpus of an experiment leads to a lower performance score.

The relation between distance and performance can be explored in two different ways: absolutely and relatively. Absolute usage of the relation means that the value of the distance metric is used to predict the actual performance. This can be made more clear using Figure 3.1. An absolute usage of the relation $R : \mathbb{R} \rightarrow \mathbb{R}$ equals saying that the performance of an experiment with a distance d_1 between the test and training corpus would be p_1; $R(d_1) = p_1$. In this case, the nature of the relation R is not important – it should not be linear – as long as it is known and implementable. This type of usage is investigated by McClosky (2010) and Plank (2011).

When the relation is used in a relative manner, the value of the distance metric is only used to rank the performance of different experiments. A greater distance should lead to a proportionally greater difference in the ranking. Looking at Figure 3.1, a relative usage is related to expressing that a machine learner will perform better for a first test/training pair than for a second test/training pair because the distance d_1 between the first pair is smaller than the distance d_2

between the second pair; $d_1 < d_2 \Leftrightarrow p_1 > p_2$. To be able to rank experiments proportionally, the relation between distance and performance should be of a linear nature.

The absolute usage is more focused on (finding) the actual relation between distance and performance, while the relative usage is more focused on the consequences of the link between distance and performance. Because later in our dissertation, the distance will be used to select the best corpus from a pool of corpora, *i.e.* taking the best-ranked corpus, we will focus on the investigation of the relative usage of the distance–performance relation.

The first set of experiments in this chapter addresses the reliability of divergence metrics based on relative token frequencies. The size of the corpora used to measure the distance, may have an influence on the final distance value. If the corpora are too small, the distance value may vary substantially when only little extra data is added to a corpus, therefore making the distance value unreliable. In Section 3.3, the minimal corpus size needed to obtain a sufficiently stable distance value, is examined.

After defining the critical corpus size, a set of divergence metrics is investigated in relation to a machine learning task. The focus of the experiments is on finding the divergence metrics or combination of metrics that lead to the best linear relation between the divergence score and the performance of the machine learner. The first set of experiments consists of part-of-speech labeling experiments with a memory-based part-of-speech tagger (Sections 3.4.1 and 3.4.2). To investigate the general applicability of the linear relation, the experiments are repeated with varying labeling systems (Section 3.4.2), a different corpus (Section 3.4.4), varying homogeneity of the corpus (Section 3.4.5), and for a different machine learning task (prepositional phrase attachment, Section 3.5). In the final Section 3.6, the relation of the conditional probability $P(Y|X)$ with the performance is also investigated although the presence of the class labels Y in the conditional probability will make this factor unsuitable for techniques that rely only on unlabeled data.

3.2 Description of the corpora

Three different corpora are used: the British National Corpus (BNC), the OntoNotes corpus and the GENIA corpus. The well-known Wall Street Journal corpus is included in OntoNotes.

The reason for choosing these corpora is that they contain part-of-speech label information. In addition, OntoNotes and the GENIA corpus both contain syntactical trees, which are required to carry out prepositional phrase attachment experiments. Both the BNC and OntoNotes are divided into subcorpora based on domain labels, which makes them suited for domain specific experiments. An extra advantage of using the BNC is its size. Even the smallest domain in the BNC contains $\sim$3 million tokens.

3.2.1 British National Corpus

For the stability experiments and for the part-of-speech experiments further on in this chapter, data extracted from the British National Corpus (BNC, 2001) is used. The entire corpus consists of written books and periodicals.[1]

The BNC annotators provided nine domain codes (*i.e.* wridom codes), making it possible to divide the text from books and periodicals into nine subcorpora. These annotated semantic domains are: imaginative (wridom1), natural & pure science (wridom2), applied science (wridom3), social science (wridom4), world affairs (wridom5), commerce & finance (wridom6), arts (wridom7), belief & thought (wridom8), and leisure (wridom9).

The extracted corpus contains sentences in which every token is tagged with a part-of-speech tag, as defined by the BNC. Since the BNC has been tagged automatically, using the CLAWS4 automatic tagger (Leech et al., 1994) and the Template Tagger (Pacey et al., 1997), the experiments are artificial in the sense that they do not learn *real* part-of-speech tags, but rather part-of-speech tags as the automatic taggers assign them. The fact that the corpus is tagged automatically, is not considered to be problematic, since we are interested in the effect of domain differences and not in the POS-tagging task as such.

The tokens in the BNC may contain a whitespace. Whitespace inside a token may introduce undesired effects in the machine learners and, for this reason, the tokens are split into subtokens at the whitespace. Splitting the tokens instead of replacing the whitespace with an underscore ensures that the number of newly constructed tokens is minimized.

[1]This is done by selecting texts with BNC category codes for text type (*i.e.* alltyp3 (written books and periodicals)) and for medium (*i.e.* wrimed1 (book), wrimed2 (periodical), and wrimed3 (miscellaneous: published)).

domain	# tokens	# sentences
imaginative	19,507,596	1,333,450
world affairs	17,925,728	726,881
social science	13,481,239	542,410
leisure	11,088,447	560,094
arts	7,182,257	303,019
applied science	7,154,185	312,948
commerce & finance	6,787,847	302,455
natural & pure science	4,095,326	172,836
belief & thought	3,160,642	136,366

Table 3.1: Overview of number of tokens and sentences in each domain of the BNC.

domain	# tokens	# sentences
newswire (nw)	682,691	27,567
– Wall Street Journal (wsj)	355,641	14,786
– ECTB newswire (mz)	194,498	8,328
– extra newswire (enw)	132,552	4,453
broadcast news (bn)	226,272	12,146
broadcast conversation (bc)	209,346	14,412

Table 3.2: Overview of number of tokens and sentences in each domain of the OntoNotes corpus.

Table 3.1 shows the number of tokens and sentences for each domain. As can be seen in Table 3.1, the *belief* domain is the smallest domain and the *imaginative* domain has the most tokens. When random samples are taken to ensure that the same amount of data is used for each domain, the size of the *belief* domain will determine the maximal size of the samples. The version of the BNC that consists of domain samples of the same size is called the *normalized BNC*.

3.2.2 OntoNotes corpus

The OntoNotes corpus (release 3.0) has been collected by Weischedel et al. (2009). The OntoNotes corpus comprises various genres of text (news, conversational telephone speech, weblogs, use net, broadcast, and talk shows) in three languages (English, Chinese, and Arabic) with structural information (syntax and predicate argument structure), and shallow semantics (word sense linked to an ontology and coreference) (Weischedel et al., 2009), but in this work only the syntactic information for the English texts is used. The syntactic information has been manually annotated with the Penn Treebank tagset.

The corpus contains three domains: newswire (nw), broadcast news (bn) and broadcast conversation (bc). The newswire domain consists of texts from the Wall Street Journal (wsj), newswire data from the English side of the ECTB[2] (mz), and additional data. The subdivisions of the newswire domain are maintained by the annotators of OntoNotes.

The OntoNotes corpus will be used to extract a corpus adapted to part-of-speech labeling experiments (Section 3.4.4) and prepositional phrase attachment experiments (Section 3.5). Table 3.2 shows the number of tokens and sentences for each domain. As can be seen in Table 3.2, the *broadcast conversation* domain is the smallest domain.

3.2.3 GENIA corpus

The GENIA treebank corpus version 1 contains annotated data from 1999 Medline abstracts (Tateisi et al., 2005). The biomedical texts are segmented into sentences and each sentence is manually annotated for syntactic structure and part-of-speech tags with the Penn Treebank tagsets (Kim et al., 2006).

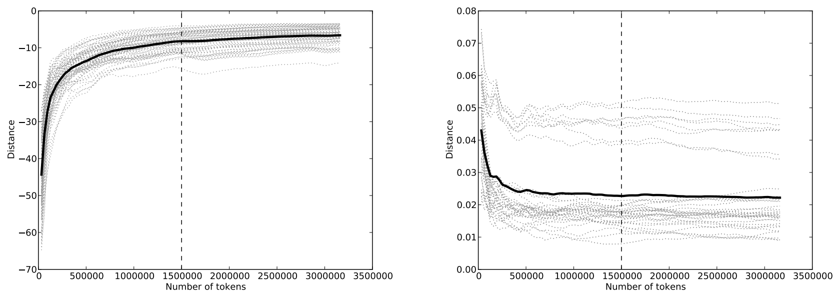
The corpus contains 486,630 tokens and 18,541 sentences and, because GENIA is annotated with syntactic structure, it can be used to extract a corpus fit for prepositional phrase attachment experiments (Section 3.5).

3.3 Metric stability

A first question that comes up when trying to apply any metric introduced in Chapter 2, is how large the corpus sample must be to obtain a stable and reliable value.

For the experiments in this section, each corpus of the British National Corpus is reduced to a bag of words and the distributions are computed as explained in Section 2.4 on page 30. To test for metric stability, the distance is computed with varying corpus sizes. The consecutive distances between two domains are computed on increasingly larger subsets of the corpora. Because the smallest

[2]English-Chinese Parallel Treebank.



(a) Rényi ($\alpha = 0.99$) **(b)** Euclidean

Figure 3.2: The evolution of the two divergences with increasing number of tokens – the 1,500,000 tokens threshold is marked with a dashed line.

domain has 3.2M tokens, see Table 3.1, the experiment finishes when the subsets contain this number of tokens. The corpora will increase in 100 steps to 3.2M tokens. This means that for every step the number of tokens on which the distance is computed, is increased with 32K tokens.

Figure 3.2 shows the evolution of the measured distance between corpora linked to increasing corpus size. The x-axis gives an indication of the size of the corpora expressed by the number of tokens, and the y-axis represents the distance values. Because there are nine domains, it is possible to repeat the size experiments $9 \times 8 = 72$ times. The dotted lines are linked to each separate experiment, the black line is the average of all experiments.

As the number of tokens increases, the distance decreases. For our implementation of the Rényi (Fig. 3.2a) divergence, a shorter distance is expressed with a less negative value. For the Euclidean distance (Fig. 3.2b), a shorter distance is expressed with a lower positive value. This observation can be expected because the chances that the token samples taken from two domains are divergent, diminish when those samples are larger: more data will often lead to more similarity.

A second observation is that the distance decrement slows down with increasing corpus size and continues to decrease for most corpora pairs. It is unclear if this continuing decrease should be interpreted as whether corpora of 3,160,642 tokens are not sufficiently large to reach a stable value, or as whether two text corpora

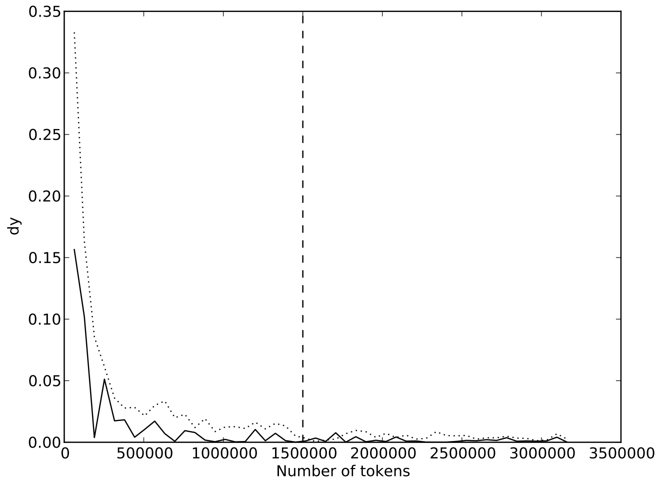


Figure 3.3: The evolution of the average distance difference of the Rényi ($\alpha = 0.99$) divergence ($\cdots$) and Euclidean distance (—) with increasing number of tokens drawn from the BNC. If the average distance difference reaches 0, the metric can be considered stable with regard to the number of tokens used to compute the distance value. The 1,500,000 tokens threshold is marked with a dashed line.

keep approaching each other when they grow larger, eventually becoming equal. The second interpretation may illustrate the point of view that all domains are biased samples of the entire language. The contents of a sample is deliberately chosen to come from a given domain, *i.e.* a specific subset of the language, and, as a consequence, the sample is unrepresentative of the language. This means that no random sampling is involved. When enlarging the sample, it becomes inevitably more representative of the language and, when this happens for all domains, all samples become more equal.

A last observation to make before trying to come up with a minimal corpus size is about the smoothness of the curve. A smooth curve would mean that the metric is more reliable, since changing the corpus size would not result in a quickly varying distance value. When a critical corpus size is reached, both metrics seems to be stable.

A second way to derive the stability of the average curves in Figure 3.2, is presented in Figure 3.3. In this figure, the dy_k-value belonging to a given x_k is computed with the formula:

$$dy_k = \left| \frac{y_k - y_{k-1}}{max(y_{k-1}, y_k)} \right| \tag{3.1}$$

The dy-value is the difference in distance at point k and can be regarded as a signless, relative version of the first order derivation of the graphs in Figure 3.2. If the dy equals 0, this means that the distance–vs–corpus-size plot has reached a plateau. The denominator of the formula makes the value scale independent, therefore enabling the comparison of different metrics on the same plot. Figure 3.3 shows that, after a steep beginning, the curves for both the Rényi and the Euclidean distance become stable. The Euclidean distance reaches the plateau earlier than the Rényi divergence, but when the size of the corpora falls in the range of 1,500,000 tokens, both metrics seem to return a reliable value. We will use this number of tokens as a rule of thumb for the minimal corpus size needed in our experiments.

The number of 1,500,000 tokens is a pragmatic, flexible and disputable threshold. It has been obtained based on the BNC; it may be different for other corpora. It has been derived by visual inspection of Figure 3.3. And, it is obtained by examining only two metrics. The main reason to use the number of 1.5M tokens is that corpora of 200K tokens or maybe even of 500K tokens are not sufficiently large to calculate a stable distance. A corpus of 1.5M tokens surpasses these substandard corpora in size without being unnecessarily large.

3.4 The distance–performance relation

When a machine learner is trained on a corpus from a given domain (source corpus) and tested on an out-of-domain test corpus (target corpus), the performance of the machine learner decreases. In Section 2.4, a range of metrics is presented. In this section, those metrics are applied to various source/target corpus combinations and the relation between the obtained distance and the performance of the out-of-domain experiment is investigated. The goal of these experiments is to unravel whether metrics capture useful information about a corpus given a specific task. A collection of useful metrics will be selected, based on the comparison of the linearity of the relation between distance and performance for different metrics and combinations of metrics.

The first set of experiments consists of part-of-speech labeling experiments with the British National Corpus and the performance will be compared with a single metric. An effort is made to attain a sense of generality for the linear relation by examining the relation for machine learners that are based on different theoretical grounds. Next, different metrics are linearly combined into a model and the combination is again compared to the performance of part-of-speech experiments. The claim that there is a linear relation between distance and performance can be made more general by repeating the first set of experiments on a different corpus, OntoNotes. Finally, the sensitivity of the distance metrics is tested by means of in-domain experiments. For in-domain experiments, there is also a difference between the test and the training corpus, although the difference will be small when compared to out-of-domain experiments. At the end of this section, we will have a clear view on how the relation between distance and performance works for part-of-speech labeling. In the next section, this relation will be investigated for a different task: prepositional phrase attachment.

3.4.1 Two part-of-speech taggers and a baseline algorithm

The British National Corpus described in Section 3.2.1 contains part-of-speech information which enables us to carry out a part-of-speech tagging experiment (Van Asch & Daelemans, 2010). The BNC corpus provides 91 different POS labels which are combinations of 57 basic labels. Three algorithms were used to assign part-of-speech labels to the words from the test corpus:

Memory-based POS-tagger (MBT, Daelemans & van den Bosch, 2005). MBT is a machine learner that stores examples in memory and uses a variation of the kNN algorithm to assign POS labels. The memory-based tagger consists of two kNN-classifiers: one classifier for tokens that are in the training corpus and one classifier for tokens that are not. The first classifier will be referred to as the *known words* classifier and the second will be referred to as the *unknown words* classifier. The default settings were used. This classifier is also used for the experiments in Section 4.

SVMTool POS-tagger (Giménez & Márquez, 2004). Support vector machines in a sequential setup are used to assign the POS labels. The default settings were used.

Majority algorithm. This algorithm assigns the POS label that occurs most frequently in the training set for a given token to the token in the test set. If the token did not occur in the training corpus, the overall most frequent tag is used.

For the two part-of-speech taggers no optimization has been carried out, because we are mainly interested in the effect of varying test/training pairs and the associated distance–performance correlation, and not in the actual performance. The effect that is investigated is not expected to change when performance increase has been obtained by means of optimization. Running the same machine learner with different settings could be considered similar to running two different machine learners. In this sense, optimization would increase the number of algorithms that can be tested. In this dissertation, we are only interested in the nature of the distance–performance correlation when different machine learners with a different theoretical basis are examined. The influence of different settings, which may be only minimal, can be the subject of future research.

3.4.2 Influence of algorithm choice

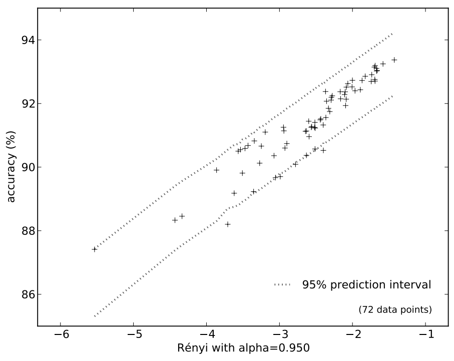
Random samples of equal size are taken from the nine corpora from the BNC in order to abstract from corpus size effects. During the sampling, the number of sentences is normalized, because the part-of-speech taggers should learn from complete sentences. The number of sentences of each domain is defined by the size of the smallest domain, *viz.* 136,366 sentences for the *belief* domain. Normalizing on the basis of sentences means that the number of tokens may differ, although every domain will contain approximately 3,000,000 tokens, which is more than the required minimal corpus size derived in Section 3.3.

Memory based tagger results

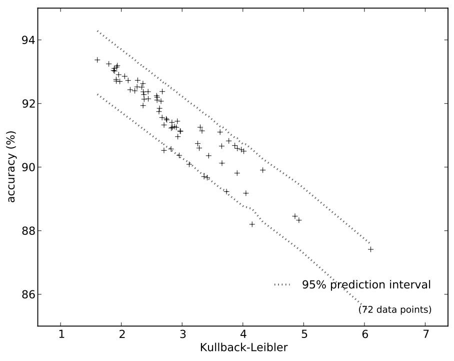
Each POS-tagging experiment consists of taking one of the nine domains as the training corpus and another domain as the test corpus. Because there are nine domains, the entire BNC can be covered doing 72 of such out-of-domain experiments. More reliable results can be obtained by repeating the experiment through cross-validation.

For out-of-domain cross-validation experiments, the scheme is more complicated than for in-domain experiments. For each of the nine domains in the BNC, the data is divided into five parts. For all pairs of domains, each part from the training domain is paired with each part from the testing domain. This results in a 25 fold cross-validation out-of-domain experiment. A data point in Figure 3.4

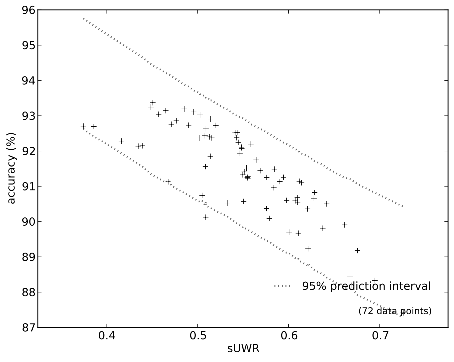
is the average outcome of such a 25 fold experiment and by using the entire BNC, 72 data points are yielded. For the plots in Figure 3.4, the x-value of a data point is the divergence score between the training and testing component of an experiment, the y-value is the accuracy of the part-of-speech tagger. Not every metric is represented in the figure, because the plots only present a coarse view of the relation between divergence and accuracy, resulting in an almost interchangeable appearance of most plots. Nevertheless, there are some trends that are worth mentioning.



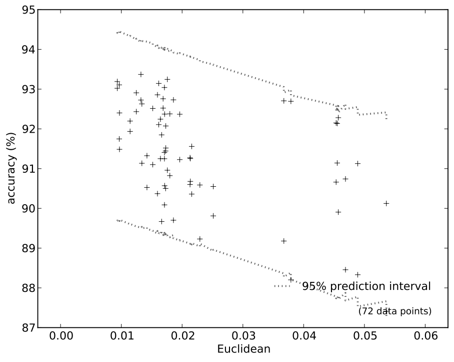
(a)



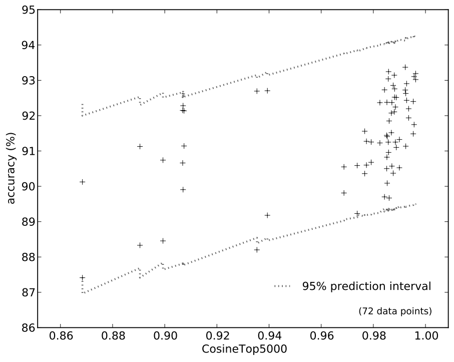
(b)



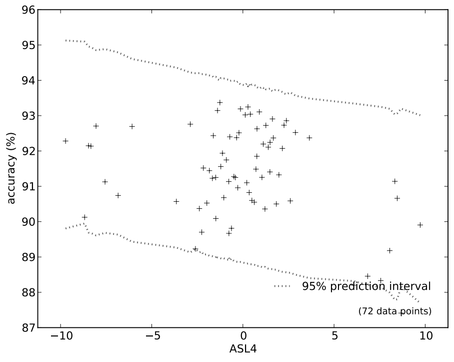
(c)



(d)



(e)



(f)

Figure 3.4: POS-tagger accuracy vs. metric value. The dashed lines indicate the 95% prediction interval. The smaller this interval, the better the relation between distance and performance.

First, the nature of some metrics is such that they produce a higher value for greater distances (like Kullback-Leibler, Fig. 3.4b), while other implementations of metrics produce a lower value for greater distances (like Rényi $\alpha = 0.95$, Fig. 3.4a). This results in increasing and decreasing trends in the plots of Figure 3.4. The easiest way to determine the *direction* of a metric is by looking at the highest accuracy, since all metrics return the shortest distance for the highest accuracy, regardless of the actual value of the metric. There are a few metrics that do not comply with this notion of direction. One example is average sentence length type 4, ASL_4 which is the difference between the ASL in the test and training corpus, Figure 3.4f. This means that a value of 0 indicates the shortest distance and other values, whether they are more positive or negative, indicate further distances. This behavior makes ASL_4 unsuitable for direct linear regression.

Another observation that can be made from the plots in Figure 3.4, is that some metrics are limited to an interval, see Table 2.1, third column. A good example is the cosine divergence, Fig. 3.4e. The value of the cosine divergence ranges from 0 (opposites) to 1 (entirely similar). As can be seen in the plot, the result of this limited range is that many data points cluster together at the shortest distance side of the plot. This is also an indication that the metric is not sensitive to small changes, since the cosine divergence labels most domain combinations as a combination between very similar domains.

A *prediction interval* is an accuracy interval that can be used to predict the accuracy of a machine learning experiment. The certainty of the prediction interval gives the certainty that the accuracy of the experiment will lie inside the prediction interval. The narrower the interval, the more precisely the accuracy of an experiment can be predicted.

Generally speaking, the width of the prediction interval is greater for symmetric divergences (Euclidean and Cosine divergence) than for asymmetric divergences (Kullback-Leibler, Rényi, and Overlap). For symmetric divergences, the accuracy for a combination of two domains is positioned at the same divergence score as the accuracy of the reverse combination of the same pair of domains. This collapse may be a cause of a wider prediction interval. The behavior is illustrated in Figure 3.5. The disadvantages of applying a symmetric divergence have been discussed in Section 2.5, but symmetry is certainly not the only cause for wide intervals.

When trying to predict the accuracy for an out-of-domain experiment, it is possible to first obtain a prediction interval by collecting the minimum and

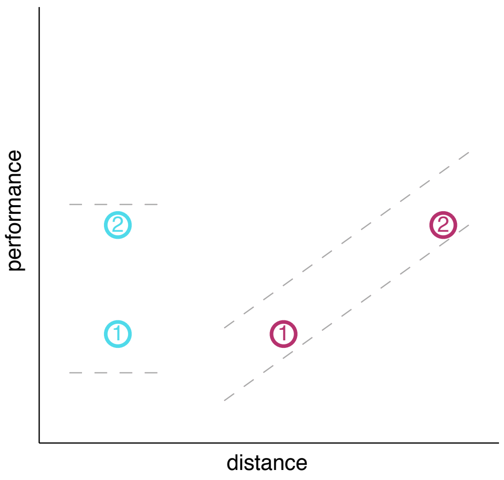


Figure 3.5: Illustration of the wider prediction interval for symmetric distance metrics. There are two machine learning experiments indicated with a circle and a number. The only difference between the experiments being a switch of the role of the training and test corpus in the metric. Both circles with number 1 are data points for the same experiment – hence the same performance – but the distance between test and training corpus is computed differently for both points. For the two data points at the left of the figure, the distance is computed with a symmetric distance metric. Since symmetric metrics are insensitive to the order of the measured corpora both data points are on the same vertical line. For the two data points at the right of the figure, the distance is measured with an asymmetric measure, leading to a different distance for these data points. The prediction intervals are indicated with dashed lines. It can be seen that the prediction interval for the symmetric distance metrics is wider.

```
INPUT    set of divergences paired with accuracy scores for
         different domain combinations; the divergence for a combination
         of domains that is not in the set.
OUTPUT   Accuracy for the unseen domain combination.
step 1   Compute the intercept ($\beta_0$) and regression coefficient
         ($\beta_1$) between divergences and accuracies using the least
         squares method,
step 2   Compute the accuracy for the divergence of the unseen
         domain combination; accuracy = $\beta_0 + \beta_1 *$ divergence,
step 3   Compute a 95% confidence interval for the predicted accuracy
         (See Appendix A).
```

Figure 3.6: Linear regression for accuracy prediction.

maximum accuracy scores for a set of out-of-domain experiments. It can then be expected that the accuracy for the new out-of-domain experiment will be in this prediction interval. There are two disadvantages to this working method:

The first disadvantage is linked to unexpected performance drops for out-of-domain experiments. One can expect that the accuracy of a new in-domain experiment will lie in the prediction interval derived from other in-domain experiments. For out-of-domain experiments, there is a higher risk that the accuracy for a new out-of-domain experiment will not lie in the interval because no knowledge about the difference between test and training corpus is present.

A second disadvantage is that the simple minimum/maximum prediction interval can be wide for out-of-domain experiments. A wide interval will have less predictive power.

Using linear regression with divergences as the regressors may overcome these two issues: it will provide knowledge about the difference between test and training corpus, and the prediction interval will be narrow if the linearity is strong. The linear regression can be carried out as depicted in Figure 3.6. A set of out-of-domain experiments is carried out and the divergence scores between the training and test corpus and the accuracies are collected. This collection of results can be used to carry out linear regression with the divergences as the regressor and the accuracies as the dependent variable. Next, the linear regression parameters can be used to predict the accuracy of a new divergence and the 95% confidence interval on this predicted accuracy can be computed.

The result of obtaining confidence intervals for every data point is a set of lower and upper boundaries. This set can be taken as the prediction interval. The size of the prediction interval gives an indication of the precision of the regression method. If the interval size is small, there is a good relation between the divergence and the accuracy, and it is possible to predict the accuracy for an unseen combination of domains with a high precision. The 95% prediction interval is also plotted as a dashed line in Figure 3.4. The ratio of the prediction interval based on regression and the minimum/maximum prediction interval is an indication of the improvement when using linear regression. The lower the ratio, the greater the improvement.

Table 3.3 on page 65 shows an overview of different metrics on the BNC corpus with the memory-based machine learner. For the experiment, the minimum/maximum prediction interval has a width of 5.945%. The first column of the table gives the metric that has been used. The second column (interval size) contains the width of the 95% prediction interval based on linear regression. The third column gives the ratio of the prediction interval to the minimum/maximum interval. The fourth column gives the Pearson product-moment correlation coefficient, or simply correlation coefficient (r) (Pearson, 1896), which is a measure of linear association. The last column repeats the correlation coefficient, but in an unsigned format, r^2.

We find that the Rényi divergence $\alpha = 0.95$ gives the best results, but the first three metrics give comparable results. It should be noted that the good result of the Rényi divergence does not mean that the divergence is best suited to draw statistically sound conclusions about the similarity of domains (Batu et al., 2000; Pathak & Nyberg, 2009). There are more advanced metrics and tests (Batu et al., 2000) available if the distance metric is to be used to compute a distance d between two samples of distributions, such that a distance $d < \epsilon$ means that the distributions are identical. An example of an advanced metric is the $\mathcal{A}$-distance (Devroye et al., 1996; Kifer et al., 2004), but the computationally expensive $\mathcal{A}$-distance is not used in this work, because we are not interested in tests for the distance as such, but only in the correlation between distance metric and performance.

Because of the setup of the experiment, a metric that can cope with asymmetry (see Section 2.5) will have an advantage over a metric that cannot. Indeed, the performance of an experiment for a given training and test corpus will most likely differ from the performance of the reverse experiment – switching the roles of the original training and test corpus. If data points, as well as their mirror points, are on the same plot, the value of the distance metric should be different for both points.

This property can be important for specific experimental setups, but for other experiments this difference is not important. The ability to handle asymmetry will not be required if an experiment consists of a fixed test corpus and a training corpus that increases in size during the experiment while the test-training distance is probed at defined intervals. The roles of the training and the test corpus in the metric are never switched. Therefore, for this type of experiment, the asymmetry property of metrics is not important.

It is possible to recalculate the correlations of Table 3.3 on page 65 to give more information about the practicability of the metrics in setups without asymmetry disruptions. The results are given in Table 3.4. These correlations are computed for the same data points as in Table 3.3 except that only data points for the same test corpus are plotted on the same plot rather than plotting all data points on a single plot. Since there are nine domains, this leads to nine plots for which the correlation can be computed. Averaging these nine correlations leads to the correlations reported in Table 3.4. Because a data point is not on the same plot as its mirrored data point, the influence of asymmetry is eliminated. In general, eliminating asymmetry improves the correlations – leading to an acceptable correlation between distance and performance for many metrics.

Given the setup, ASL1 – average sentence length of the test corpus – is the same for all data points on a plot leading to an incalculable correlation.

The difference between our setup and the one described in Klakow & Peters (2002), Zhang & Wang (2009), and Plank & van Noord (2010) is that they try to find a relation between a distance and a performance score, while we try to find a relation between the difference between two domains and the accuracy. The idea is that consistent errors in the distance metric are smoothed out when you work with the differences instead of with the absolute scores.

metric	interval size	ratio	r	r^2
Rényi with α =0.95	1.938	0.326	0.934	0.872
Kullback-Leibler	1.961	0.330	-0.932	0.869
Rényi with α =0.99	1.983	0.334	0.931	0.866
Rényi with α =0.90	2.270	0.382	0.908	0.824
Rényi with α =1.05	2.962	0.498	-0.837	0.701
Rényi with α =0.80	2.994	0.504	0.833	0.695
sUWR	3.077	0.518	-0.823	0.677
Skew with α =0.25	3.225	0.543	-0.804	0.646
Rényi with α =0.70	3.486	0.586	0.766	0.586
Skew with α =0.50	3.732	0.628	-0.725	0.525
Rényi with α =0.60	3.829	0.644	0.707	0.500
Skew with α =0.75	4.073	0.685	-0.659	0.435
Perplexity	4.074	0.685	-0.659	0.434
Bhattacharyya	4.080	0.685	-0.657	0.432
Rényi with α =0.50	4.100	0.690	0.654	0.427
Rényi with α =1.10	4.100	0.690	-0.653	0.427
Jensen-Shannon	4.139	0.696	-0.645	0.416
Skew with α =0.80	4.145	0.697	-0.644	0.414
Variational (L1)	4.275	0.719	-0.614	0.377
Skew with α =0.90	4.322	0.727	-0.603	0.363
Rényi with α =0.40	4.342	0.730	0.598	0.357
Skew with α =0.95	4.464	0.751	-0.566	0.321
Rényi with α =0.30	4.590	0.772	0.531	0.282
Rényi with α =3.00	4.665	0.785	0.508	0.258
Rényi with α =2.00	4.696	0.790	0.498	0.248
Euclidean	4.705	0.792	-0.495	0.245
Skew with α =0.99	4.725	0.795	-0.489	0.239
CosineTop5000	4.747	0.798	0.482	0.232
CosineTop500	4.751	0.799	0.480	0.230
CosineTop50	4.754	0.800	0.479	0.230
Rényi with α =0.20	4.876	0.820	0.436	0.190
ASL3	4.921	0.828	-0.418	0.175
ASL2	5.045	0.849	-0.364	0.132
ASL4	5.100	0.858	-0.337	0.114
Rényi with α =1.50	5.123	0.862	0.325	0.106
Rényi with α =0.10	5.212	0.877	0.273	0.075
Overlap	5.224	0.879	0.264	0.070
ASL1	5.412	0.910	0.043	0.002
Rényi with α =0.01	5.417	0.911	-0.000	0.000

Table 3.3: Linearity of different metrics with MBT accuracy on the normalized BNC. Interval size is the size of the average confidence interval when the performance is predicted based on the distance; ratio is the ratio of the interval size and the interval size without the use of metric-based linear regression; r and r^2 are measures for the degree of linearity between a distance metric and the performance.

metric	r	r^2
Rényi with α =0.99	0.995	0.990
Kullback-Leibler	-0.994	0.988
Rényi with α =0.95	0.994	0.987
Rényi with α =0.90	0.987	0.974
Skew with α =0.25	-0.973	0.947
Rényi with α =0.80	0.968	0.939
sUWR	-0.962	0.925
Skew with α =0.50	-0.954	0.912
Rényi with α =1.05	-0.948	0.904
Rényi with α =0.70	0.949	0.903
Variational (L1)	-0.944	0.893
CosineTop5000	0.929	0.865
Rényi with α =0.60	0.927	0.864
Euclidean	-0.928	0.863
Skew with α =0.75	-0.926	0.862
CosineTop500	0.927	0.862
CosineTop50	0.923	0.853
Skew with α =0.80	-0.917	0.846
Jensen-Shannon	-0.913	0.840
Rényi with α =0.50	0.899	0.818
Skew with α =0.90	-0.886	0.796
Bhattacharyya	-0.886	0.792
Rényi with α =0.40	0.862	0.758
Skew with α =0.95	-0.853	0.744
Rényi with α =2.00	0.842	0.729
Rényi with α =1.50	0.829	0.720
Rényi with α =1.10	-0.678	0.713
Perplexity	-0.803	0.688
ASL3	-0.766	0.675
Rényi with α =0.30	0.806	0.672
ASL4	-0.749	0.642
Skew with α =0.99	-0.771	0.624
ASL2	-0.720	0.591
Rényi with α =0.20	0.707	0.537
Rényi with α =3.00	0.639	0.512
Rényi with α =0.10	0.505	0.312
Overlap	0.274	0.187
Rényi with α =0.01	0.111	0.081
ASL1	–	–

Table 3.4: Symmetric version of Table 3.3. The difference between the scores in this table and the previous is that only data points based on the same test corpus are plotted on the same plot. Given the setup, ASL1 – average sentence length of the test corpus – is the same for all data points on a plot leading to an incalculable correlation.

SVMTool and majority baseline

	Majority	MBT	SVMTool
average accuracy	83.95	91.38	93.22
standard deviation	2.44	1.33	1.11

Table 3.5: Overview of the average performance of two different machine learners and a majority baseline on the POS-tagging experiment.

To investigate the possible influence of the machine learning method, the experiments from the previous subsection are also evaluated with two other machine learners. SVMTool is based on support vector machines and the majority baseline is a simple lookup algorithm. The average accuracies over all cross-domain experiments are shown in Table 3.5. It is clear that the simple majority algorithm obtains the lowest score and, as the larger standard deviation indicates, the accuracies differ more from each other. This means that the majority algorithm is more susceptible to domain shifts than the other learners. This can be expected, since the other learners have more sophisticated POS labeling mechanisms.

Table 3.6 presents the correlation for the accuracy of the majority algorithm and the SVMTool accuracy in the same way as it is presented in Table 3.3 for the memory-based accuracy. Only the four best metrics for each machine learner are included. There is a drop of 0.1 r^2-point after Rényi with α =0.990 for the majority algorithm. For the SVMTool results the drop is smaller. Combining the results in this table with the scores in Table 3.3 leads to the choice of Rényi (α = 0.99) as the best default metric because the highest average r^2 value of 0.877 is obtained with this metric. This means that the Rényi divergence will be used as the distance metric in further experiments. However, the collection of experiments is too small to give strong preference to one of the top 4 metrics.

3.4.3 Influence of combination of metrics

As pointed out before, there is no need to restrict an accuracy predictor to only one measure (Ravi et al., 2008; McClosky, 2010). A logical step to undertake is to linearly combine metrics in order to find a good fit. Some remarks should be made. Firstly, when more factors are added to a linear combination of factors, the fit with the data will intrinsically be tighter, *i.e.* producing a higher

metric	ratio	r	r^2
MAJORITY			
Rényi with α =0.95	0.308	0.925	0.855
Kullback-Leibler	0.311	-0.923	0.852
Rényi with α =0.90	0.329	0.913	0.834
Rényi with α =0.99	0.348	0.906	0.821
SVMTOOL			
Rényi with α =0.99	0.209	0.971	0.943
Rényi with α =0.95	0.329	0.927	0.859
Kullback-Leibler	0.334	-0.925	0.855
Rényi with α =0.90	0.397	0.892	0.795

Table 3.6: Linearity of top 4 metrics with SVMTool and majority accuracy on the normalized BNC. Ratio is the ratio of the interval size and the interval size without the use of metric-based linear regression; r and r^2 are measures for the degree of linearity between a distance metric and the performance.

correlation coefficient (r^2). This is not what we aim at, since repeatedly adding the same metric with slightly different parameters does not result in a better understanding of the nature of differences between corpora, but it simply results in a better fit to a data sample. Secondly, investigating high-level models may produce a powerful model tailored to a specific task, but in the current research we are interested in an answer to the question whether there is a simple metric that can be applied to a diverse collection of NLP tasks. In the remainder of this section we will investigate linear models based upon the metrics from the previous section, but without disregarding the practical interpretation of the final model.

Examining the formulas of the metrics in Table 3.3 helps to distinguish four groups:

- Probability distribution based (Kullback-Leibler, Rényi, Skew, Jensen-Shannon, Bhattacharyya, Variational, Euclidean, Cosine).
- Average sentence length (ASL).
- Unknown words count (sUWR, overlap).
- Perplexity (perplexity, reverse perplexity).

As the plots in Figure 3.4 on page 59 show, the observation that all probability distribution based metrics use the same building blocks, does not lead to interchangeability of the metrics with regard to accuracy prediction. In some metrics the linear relation is lost (see Euclidean[3]), while other metrics (Kullback-Leibler, Rényi) seem to be nicely correlated with accuracy. However, the common principle behind these metrics clearly differs from that of, for example, the average sentence length, which is computed without using token frequencies. This may lead to the plausible conclusion that every group of metrics captures a distinct aspect of corpora. This can be observed when examining the models in Table 3.7.

Nr	Parameters	r^2	$adjr^2$	gain
1	Rényi $\alpha = 0.95$	0.872	0.870	-
2	Rényi $\alpha = 0.99$, ASL	0.900	0.897	3.1%
3	Rényi $\alpha = 0.95$, ASL, sUWR	0.917	0.914	1.9%
4	Rényi $\alpha = 0.99$, ASL, sUWR, Rényi $\alpha = 0.01$	0.943	0.939	2.7%
5	Rényi $\alpha = 0.99$, ASL, sUWR, Rényi $\alpha = 0.01$, perplexity	0.952	0.948	1.0%
6	Rényi $\alpha = 0.90$, sUWR, ASL, Skew $\alpha = 0.50$, variational, perplexity	0.957	0.953	0.5%
7	Rényi $\alpha = 1.1$, ASL, Rényi $\alpha = 0.01$, sUWR, perplexity, cosine5000, cosine50	0.961	0.957	0.4%

Table 3.7: Overview of best fitting models with increasing number of parameters. r^2 and $adjr^2$ are an indication of the degree of linearity of the relation between the output of the model and the performance; gain is the relative $adjr^2$ difference when comparing a model with the previous model.

This table shows linear models[4] with a varying number of parameters fitting the same data from the experiments of Figure 3.4 on page 59. For every number of parameters, the best model according to r^2 is presented, along with its value for the squared correlation coefficient, r^2 and adjusted r^2 ($adjr^2$). The last column is an indication of the $adjr^2$ gain when adding a parameter, calculated with $\frac{adjr_i^2 - adjr_{i-1}^2}{adjr_{i-1}^2}$. The adjusted r^2 is more suited for comparison over models with a different number of parameters, but, since we are only interested in the nature of the parameters, the actual value of r^2 is of minor importance here. Moreover, when selecting these models, we did not split the data into a training and a test partition, so the correlation coefficients are not validated against unseen data. The parameters in Table 3.7 are sorted according to decreasing impact. The impact is measured by the student t test ratio on the estimated regression weight of a parameter. A parameter appears more in front of the list if the

[3]The Euclidean distance does not need to be probability distribution based like it is in our setup.

[4]Linear modeling has been carried out using the JMP 9.0.0 statistical software package.

probability that the parameter has a zero weight is smaller. For the model with five parameters, this means that Rényi $\alpha = 0.99$ has a greater influence on the model than perplexity.

Table 3.7 shows that a newly introduced parameter is member of a group different from the groups already present. If only one parameter is used, the best model is the one using a probability based metric, Rényi. Adding an extra parameter introduces average sentence length. Adding another parameter introduces unknown words. But it is only after adding two more parameters that perplexity is introduced and perplexity is ranked at the bottom. Considering the definition of perplexity – the degree of complexity when predicting the next word in a sentence – it is plausible that this metric is not that essential when predicting part-of-speech tags. McClosky (2010) has shown that perplexity is also not very useful when trying to predict parser accuracy. He ends up with a set of parameters consisting of probability based metrics (entropy, cosine50), sUWR and information about the composition of the training corpus, although perplexity was also tested. These two experiments may indicate that perplexity is not very well suited for accuracy prediction. Note that the fact that perplexity seems not to be useful for accuracy prediction does not mean that the metric is never useful. As Rehbein (2011) shows, perplexity is helpful when one is only interested in the similarity between corpora.

In the model with seven parameters, different probability based metrics are introduced, but the gain is diminishing. We did not compute models with more than seven parameters because of limited computational resources, but the trend is clear.

For later experiments we keep the top of the single metric models from Table 3.3 and the model with three parameters from Table 3.7. Although the gain for the model with four parameters is substantial, we choose the model with three parameters, because the model with four parameters contains the Rényi divergence twice. As can be seen in Figure A.5 on page 170, Rényi divergences vary in sensitivity, along with the value of α, but they do not behave contrastingly. In addition, two extra models are selected for their expected performance: an average model and a model selected by stepwise addition of parameters in a 6-fold cross-validation setup, in which the k-fold r^2 is the selection criterion.

Model averaging is a technique where the final model is not the single best model, but a combination of a collection of models. We take the average model of the collection of models with five terms, associated with an $adjr^2$ of 0.952. The average model will have 33 regressors.

In the cross-validation setup, the data is split into a test and training part. Parameters are gradually added to the model and evaluated on the test set. In the end, a model with sixteen parameters and an $adjr^2$ value of 0.961 is selected. Both selection procedures have their disadvantages. Using them in further experiments should show which model is more robust to changes in the setup of the experiments. Both the model obtained with the model averaging technique and the model obtained with the cross-validation setup, will be reused in the next subsection.

3.4.4 Influence of corpus

In the previous sections, it was shown that there is a good linear correspondence of some metrics (Kullback-Leibler, Rényi with $\alpha = 0.95$ and 0.99) and of combinations of those metrics with the POS-tagging accuracy. This relation has been observed with two part-of-speech taggers with a different theoretical basis and a baseline algorithm, but all experiments have been conducted on the same corpus, *viz.* the British National Corpus (BNC). The good correlation between distance and performance that has been found, could be due to the way the BNC corpus has been labeled. The annotation guidelines that were followed during construction of the corpus may differ from corpus to corpus. For this reason, confirmation of the results is needed using a different corpus. Despite that many other corpora have been annotated for parts-of-speech, the OntoNotes corpus (see Section 3.2.2) is selected to run confirmation experiments, because the annotators provided a division of the corpus into domains.

Excerpts have been taken from the OntoNotes corpus: broadcast conversation (bc), broadcast news (bn), extra newswire (enw), newswire data from the English side of the English-Chinese Parallel Treebank (mz), and the Wall Street Journal (wsj). The newswire (enw) domain is the smallest with 132,552 tokens so all excerpts are limited to this number of tokens. The data for the five domains leads to twenty cross-validated out-of-domain experiments: each domain is combined once with each other domain.

For reasons of clarity, it should be noted that 132,552 tokens is below the minimal corpus size as established in Section 3.3. Figure 3.3 shows that a corpus of 132,552 tokens is right at the end of the initial drop of the first order derivation of the divergences. This means that the divergence scores will be less reliable compared to working with larger corpora, although it is still possible to look at some trends. The conclusions drawn from the OntoNotes experiments are informative – certainly when they are combined with the conclusions from the more reliable BNC experiments.

metric	interval size	ratio	r	r^2	$adjr^2$
Kullback-Leibler	6.672	0.492	-0.894	0.800	0.789
Rényi $\alpha = 0.99$	7.452	0.550	0.866	0.750	0.736
Rényi $\alpha = 0.95$	7.966	0.587	0.845	0.715	0.699
Jensen-Shannon	12.16	0.896	-0.575	0.330	0.293
Rényi $\alpha = 0.95$, ASL, sUWR	6.563	0.484	0.898	0.807	0.771
average model	7.085	0.523	0.880	0.774	–
6-fold cross-validation model	7.758	0.572	0.854	0.729	–

Table 3.8: Overview of different metrics on the normalized OntoNotes. Interval size is the size of the average confidence interval when the performance is predicted based on the model; ratio is the ratio of the interval size and the interval size without the use of model-based linear regression; r, r^2 and $adjr^2$ are measures for the degree of linearity between a distance value from the model and the performance.

The correlation between distances computed with the top three metrics of Table 3.3 and the POS-tagging accuracy, can be seen in Table 3.8. The table also contains the results for two models that are developed for the BNC in Section 3.4.3: the average model (33 regressors) and the 6-fold cross-validation (16 regressors). There is no $adjr^2$ reported for the average and 6-fold cross-validation model, since the number of regressors in these models exceeds the number of observations (20).

There is a 13.560% difference between the highest and the lowest accuracy score. The third column of Table 3.8 shows that using metrics tends to reduce this range with approximately 50%. The r^2 scores are lower than for the BNC experiments, but a value of 0.8 is sufficiently high to speak of a linear relation. From Table 3.8, it can be derived that Kullback-Leibler and the model with a linear combination of three divergences show the best linear relation when looking at r^2. The adjusted r^2 has been developed to be able to compare models with a different number of regressors and the KL-divergence comes out best when looking at the $adjr^2$. This is an indication that combining divergences can lead to overfitting. A second indication of overfitting with divergence combinations is that the average and cross-validation models do not perform best for OntoNotes, while they do so for BNC, on which they are fitted.

In the experiments in the next chapters and sections, we will focus only on a single metric to compute the distance between training and test corpus because of the risk of overfitting when estimating the extra parameters for a linear model and because the experiments in this subsection show that there is no guarantee that the linear model will clearly outperform a single metric.

3.4.5 Influence of homogeneity of corpora

In the previous part-of-speech labeling experiments, we focused on out-of-domain experiments. The judgement of the annotators on what are separate domains has been followed to differentiate between domains. This does not mean that the domains in the previous experiments are measurably different. The notion of *domain* is not essential to the experiments, since they are designed to study the effect of differences in token frequencies between corpora, when labeling tokens with POS tags. The actual domain labels are only of interest for users that need the domain classification for external reasons. This means that domain labels provided by human annotators can be well suited. However, an objective way to label texts with domain labels can still be interesting. In this subsection, the link is studied between, on the one hand, the linear relation between distance and performance and, on the other hand, domain labeling.

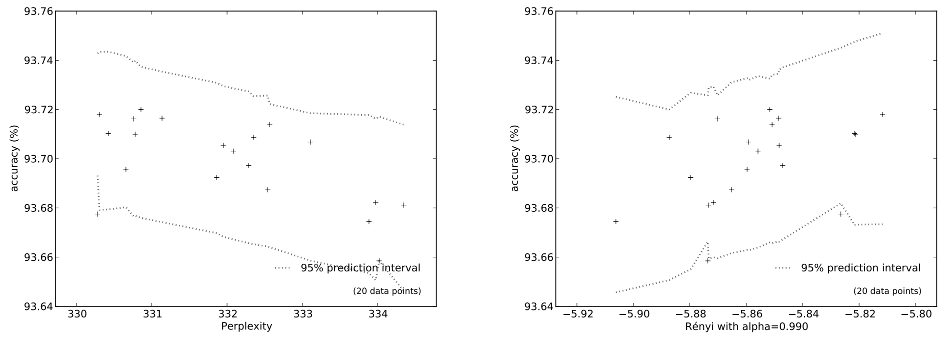
The linear relation between distance and performance that exists for out-of-domain experiments, should also exist for in-domain experiments. The single difference between in-domain and out-of-domain experiments is that in-domain training and test corpora are labeled with the same domain label by human annotators. For this reason, the experiments of the previous subsections can be repeated for in-domain experiments because of this arbitrary difference between in-domain and out-of-domain experiments.

Annotators intend to group similar texts into one domain, therefore it can be expected that the distance between the test and training corpus for in-domain experiments will be smaller, maybe even too small for the distance metric to overcome noise effects. The noise effect could break the linear relation between divergence and accuracy and the disappearance of the linear relation could be a means to define domains in an objective manner. A set of corpora that exhibits a linear relation would then be a set of corpora from different domains. A set of corpora that does not exhibit a linear relation, would consist of corpora belonging to the same domain.

Comparing in-domain with out-of-domain experiments

In a first set of in-domain experiments, the setup is chosen such that it reflects the setup of the out-of-domain experiments. For these experiments, the British National Corpus is used, see Section 3.2.1.

The BNC-subcorpus with the domain tag *world* is split into five random parts of approximately 145,285 sentences (> 3,000,000 tokens). These parts are treated in the same way as the domains in the out-of-domain experiments of Section 3.4.2, so they are split into five parts (approx. 29,057 sentences) for the 25-fold cross-validation experiments. In this way, it is possible to obtain in-domain results that are comparable with the out-of-domain experiments. The only difference being the slightly different corpus size, since the out-of-domain corpora contain 136,253 sentences instead of 145,285 sentences. Randomizing the data before splitting it into five parts, guarantees that the data can be considered as truly in-domain. Not randomizing the data would leave the possibility that the first data in the corpus is different from the last data. If such a corpus is split into parts, *subdomains* could be created.



(a) Perplexity **(b)** Rényi ($\alpha = 0.99$)

Figure 3.7: In-domain POS-tagger accuracy vs. perplexity and Rényi divergence. The dashed lines indicate the 95% prediction interval. The smaller this interval, the better the relation between distance and performance is.

The average accuracy of the in-domain experiment is 93.70% (0.02% standard deviation). The perplexity shows the highest correlation with the accuracy and it is associated with an r^2 value of 0.384. Perplexity has not been a good metric in previous experiments and is not a good metric in this experiment because of the low r^2. The linear relation between metric scores and accuracy is lost, but a hint of it remains noticeable as can be seen in Figure 3.7. The figure also shows the plot for Rényi ($\alpha = 0.99$), which is the best metric for the out-of-domain experiments. The test and training corpus are too close to each other for the metric to pick up information, but a remark about the sensitivity of the metric has to be made: Despite the observation of lost linear correlation, there may still be a linear correlation between distance and performance for in-domain experiments. If this were the case, the lost correlation could be attributed to the insensitivity of the metric. The linear correlation may depend on small distance differences, too subtle to be picked up by the metric. A more sensitive metric may be able to pick it up. To exclude this option, a second set of experiments is performed.

Testing sensitivity

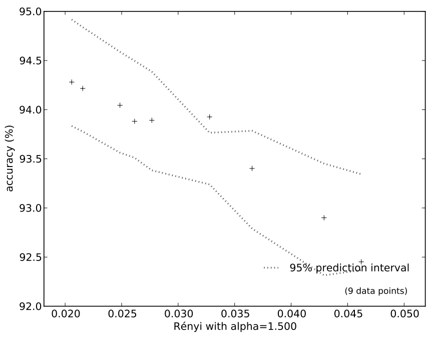
To test the sensitivity of the metrics, another POS-tagging experiment has been set up. In the previous subsection, it has been found that the linear correlation is lost for small differences in distance and the associated small difference in performance, dissolving the linear correlation in a cloud of noise. It is possible to set up an experiment with small distance differences, but with larger performance differences. The key feature of these experiments being that there is a clear difference between the data points because of the different performance, while there are only small distance differences. The result is that a metric that is not sufficiently sensitive, will not show a linear correlation.

A round of in-domain POS-tagging experiments is set up, using the domains from the British National Corpus. We showed that the domains are different, which means that the performances of the nine in-domain experiments are expected to be sufficiently different when compared to each other. This ensures the large performance differences.

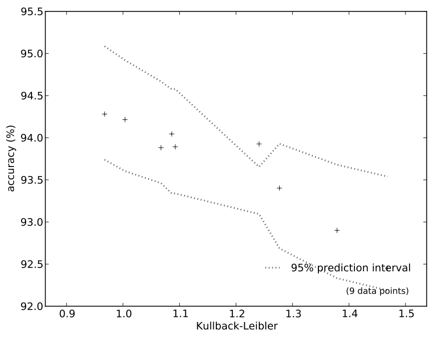
For the in-domain experiments, the data from a single domain is split into five parts of approximately 27,273 sentences.[5] Each part is combined with every other part, once as training corpus and once as test corpus.

For one domain, the result is one data point based on a 20-fold cross validation and consisting of an average POS labeling accuracy and an average divergence

[5]Each domain has been randomly downsampled to 136,366 sentences – the size of the smallest domain (*belief*).



(a) Rényi



(b) KL

Figure 3.8: In-domain POS-tagger accuracy vs. Rényi ($\alpha = 1.5$) and Kullback-Leibler divergence. The dashed lines indicate the 95% prediction interval. The smaller this interval, the better the relation between distance and performance is.

score. Because the distance is based on purely in-domain test/training pairs, the distance is expected to be small.

The relation between these experiments and the in-domain experiment in the previous subsection is that the previous experiment tries to enlighten what happens inside a data point in Figure 3.8, hence the smaller performance differences in the previous experiment.

By running the experiments with the nine domains of the BNC, nine data points can be obtained. Figure 3.8 depicts the data points when the Rényi divergence ($\alpha = 1.5$) and Kullback-Leibler divergence are used as the metric. The x-axis shows the metric value, the y-axis shows the POS labeling accuracy.

metric	interval size	ratio	r	r^2
Rényi with α=1.50	1.143	0.632	-0.952	0.905
Kullback-Leibler	1.232	0.681	-0.943	0.889
Rényi with α=1.10	1.270	0.702	-0.940	0.883
Rényi with α=1.05	1.280	0.707	-0.939	0.881
Rényi with α=0.01	1.290	0.713	0.938	0.879
Rényi with α=0.99	1.290	0.713	0.938	0.879
Skew with α=0.01	1.295	0.716	-0.937	0.878
Rényi with α=0.95	1.296	0.716	0.937	0.878
Skew with α=0.05	1.302	0.719	-0.936	0.877
Rényi with α=0.90	1.302	0.720	0.936	0.877
Rényi with α=0.10	1.302	0.720	0.936	0.877
Skew with α=0.10	1.309	0.723	-0.936	0.875
Rényi with α=0.80	1.313	0.725	0.935	0.875
Rényi with α=0.20	1.313	0.725	0.935	0.875
Rényi with α=0.70	1.320	0.729	0.935	0.873
Rényi with α=0.30	1.320	0.729	0.935	0.873
Rényi with α=0.40	1.324	0.731	0.934	0.873
Rényi with α=0.60	1.324	0.731	0.934	0.873
Rényi with α=0.50	1.325	0.732	0.934	0.872
Skew with α=0.99	1.325	0.732	-0.934	0.872
Skew with α=0.25	1.326	0.733	-0.934	0.872
Skew with α=0.95	1.337	0.739	-0.933	0.870
Skew with α=0.90	1.343	0.742	-0.932	0.869
Jensen-Shannon	1.344	0.743	-0.932	0.869
Skew with α=0.50	1.344	0.743	-0.932	0.869
Skew with α=0.75	1.350	0.746	-0.931	0.868
Bhattacharyya	1.388	0.767	-0.927	0.860
Variational (L1)	1.672	0.924	-0.893	0.797
Rényi with α=3.00	1.958	1.082	0.848	0.718
Overlap	2.224	1.229	-0.800	0.640
sUWR	2.224	1.229	-0.800	0.640
Perplexity	2.999	1.657	-0.564	0.318
Rényi with α=2.00	3.264	1.803	0.472	0.223
ASL3	3.622	2.001	0.432	0.186
CosineTop50	3.362	1.858	0.403	0.163
ASL2	3.523	1.947	0.380	0.144
CosineTop5000	3.595	1.987	0.289	0.083
CosineTop500	3.587	1.982	0.285	0.081
Euclidean	3.640	2.011	-0.246	0.060
ASL1	3.971	2.194	0.122	0.015

Table 3.9: Linearity of different metrics with MBT accuracy on the normalized BNC for in-domain experiments. Interval size is the size of the average confidence interval, when the performance is predicted based on the distance; ratio is the ratio of the interval size and the interval size without the use of metric-based linear regression; r and r^2 are measures for the degree of linearity between a distance metric and the performance.

The average accuracy for an in-domain experiment is 93.67% (0.63% standard deviation), which is higher and more stable than the 91.38% (1.33% standard deviation) accuracy score from the out-of-domain MBT experiment. An overview of the linear correlation for various metrics for the in-domain experiments is given in Table 3.9.

When comparing the scale of the x-axis of the in-domain plot for Kullback-Leibler, Figure 3.8b, with the x-axis scale for out-of-domain experiments, Figure 3.4b on page 59, it is clear that the distance between train and test for in-domain experiments is smaller, ranging between 0.9–1.5 instead of between 1–7. Nonetheless, the plots show a good linearity which is confirmed by the r^2 figures in Table 3.9.

The experiments indicate that the metrics are sufficiently sensitive to capture meaningful information about small distance differences between training and test corpus. The combination of the finding that the metrics are sufficiently sensitive to capture small distance differences together with the lost correlation observation from the previous subsection, leads to the conclusion that a lost correlation between distance and performance can be attributed to a homogenous set of corpora rather than to insensitivity of the metric.

Apart from this conclusion, two extra remarks have to be made:

(i) The observation that the Rényi divergences with α values that ranked rather low in out-of-domain experiments Table 3.3 on page 65 are at the top of the in-domain Table 3.9 on page 77 seems rather peculiar. A possible explanation may be found in the analysis made in Section A.4.1. Increasing α decreases the sensitivity of the Rényi divergence to tokens that are in one corpus, but missing in the other. At the same time, an increased sensitivity for token frequency differences is observed. For out-of-domain experiments, a large number of missing tokens and greater frequency differences between training and test corpus can be expected, meaning that the α parameter can have a big influence. For in-domain experiments, the factors upon which the α parameter acts, can be expected to be less prominent, making the Rényi divergence robust to the value of α. Only for the larger values of α ($\alpha = 2$ and 3), enough emphasis is put on the factors to show a detrimental effect.

(ii) The fact that the r^2 value for $\textit{Rényi}(P; Q; \alpha = \delta)$ is the same as for $\textit{Rényi}(P; Q; \alpha = 1 - \delta)$ for α values lower than 1, is due to the setup of the experiment. For one data point, an experiment is run as well as its reverse and the outcome is averaged. This averaging has as effect that the following equality holds:

$$\textit{Rényi}(P;Q;\alpha=\delta) = \frac{\delta}{1-\delta}\,\textit{Rényi}(P;Q;\alpha=1-\delta) \tag{3.2}$$

Since the correlation is indifferent to scaling, the r^2 values for divergences with $\alpha = \delta$ and $\alpha = 1 - \delta$ are the same. This behavior also means that Figure 3.8 is not directly comparable to Figure 3.4 on page 59.

Conclusion

In a first set of in-domain experiments, it is shown that the metrics are not able to pick up meaningful information in an in-domain experiment, see Figure 3.7 on page 74. The linear correlation between distance and performance is lost. In a second set of in-domain experiments, it is shown that the best metrics are sufficiently sensitive to pick up small distance differences between test and training corpus and the linear relation between distance and performance is preserved, see Figure 3.8 on page 76.

The conclusion from these two sets of experiments is that the disappearance of the linearity in the first set of experiments is not due to insensitivity of the method, but to the absence of significant differences – differences in distance as well as in performance – between the training and test corpus – the only difference being sampling noise. This would mean that the five parts of the randomized *world* corpus could be considered as being homogenous.

3.5 A different machine learning task

In the previous section, is has been shown that a divergence metric based on relative token frequencies captures information that is linearly related to the performance of part-of-speech labeling systems. The general applicability of the method has been tested by varying the labeling system (Section 3.4.2), the corpus (Section 3.4.4), and the homogeneity of the corpus (Section 3.4.5)

In this section, the setup for the POS labeling experiments from Section 3.4 is repeated for the prepositional phrase attachment task (PP-attachment). The GENIA corpus and domains of the OntoNotes corpus are used to extract corpora that can be used for the PP-attachment task. The goal of the experiments in this section is to see whether the linearity is preserved when switching from the POS labeling task to the PP-attachment task.

In the next subsection, the PP-attachment task is first described in more detail before reporting on the experiments.

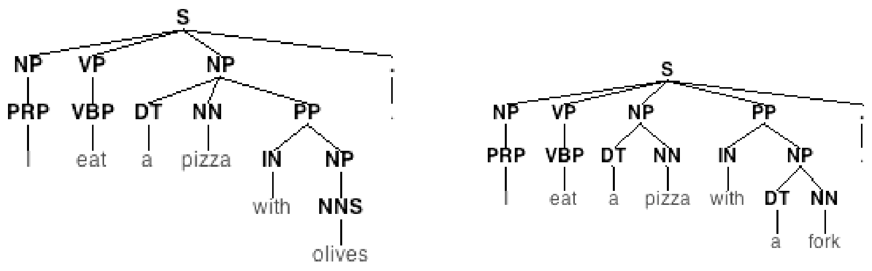
3.5.1 Description of the PP-attachment task

The core of the PP-attachment task is to link a prepositional phrase – mostly a preposition followed by a noun phrase – to its head. The task is a subtask of parsers that construct constituent trees. To be able to carry develop and evaluate a PP-attacher, a corpus of constituent trees is needed to extract a suited PP corpus. OntoNotes (Weischedel et al., 2009), Wall Street Journal (Marcus et al., 1994), and GENIA (Tateisi et al., 2005) are all fit for the task. The OntoNotes and GENIA corpus are used for the experiments in this section, since the goal of the PP-attachment experiments is to get a better insight into the influence of domain shifts and OntoNotes provides data annotated for three different domains: broadcast conversation (bc) , broadcast news (bn), and newswire texts (nw). The newswire (nw) domain also contains the data from the Wall Street Journal corpus (See Section 3.2.2). GENIA contains texts from the biomedical domain (genia) (See Section 3.2.3).

As Atterer & Schütze (2007) state, the classic formulation of the PP-attachment task, as defined by Ratnaparkhi et al. (1994) and Hindle & Rooth (1993), is a simplification. The classic formulation uses quadruples (v, n_1, p, n_2)[6] that were manually selected from a corpus. This improves the performance of PP-attachment systems, but for a natural language application these quadruples are not available. In the experiments of Atterer & Schütze (2007), the evaluated PP-attachment systems did not significantly improve on a state-of-the-art parser, the Collins parser (Collins, 2003), indicating that the use of quadruples is harmful when trying to draw conclusions for realistic PP-attachment tasks. The PP-attacher system that is used in this section does not make use of this simplified representation and therefore can be regarded as more fit to the task of natural language PP-attachment. To this end, a corpus is to be extracted from the OntoNotes and GENIA corpus to be able to train and test a more realistic PP-attacher.

[6] v, n_1, p, n_2 is a verb–noun–preposition-noun pattern like e.g. *receives book from Salinger.*

The OntoNotes and GENIA corpora consist of tree structures representing the syntactic structure of sentences, as shown in Figures 3.9a and 3.9b. The trees are transformed into a flat representation in order to be able to define one unique attachment site (anchor) for every prepositional phrase (PP). A flat representation of an anchor–PP pair consists of a pair of indices. The first element of the pair is the index in the sentence of the anchor; the second element is the index of the preposition. For the sentence in Figure 3.9a the representation is (3, 4) – token count starts at zero. For the sentence in Figure 3.9b the representation is (1, 4). The output of the memory-based prepositional phrase attacher consists of this type of index pairs. Evaluation is done by checking if the returned index pairs are present in the reference.



(a) I eat a pizza with olives. **(b)** I eat a pizza with a fork.

Figure 3.9: Constituent tree structures.

The memory-based PP-attacher (MBPA) (Van Asch & Daelemans, 2009) has been used for the experiments in this section. The PP-attacher is based on a *k*NN-classifier TiMBL (Daelemans & van den Bosch, 2005). This PP-attacher has been incorporated in MBSP.[7] As Van Asch & Daelemans (2009) showed, the memory-based PP-attacher obtains slightly lower but comparable scores when compared to a state-of-the-art statistical full parsing approach, *viz.* the Collins parser (Collins, 2003; Bikel, 2004).

3.5.2 Experimental setup

For the experiments, three domains from OntoNotes (bn, bc, nw) and the GENIA domain (genia) are used to extract the needed anchor-pp pairs. For com-

[7]Memory-based shallow parser (De Smedt et al., 2010; Daelemans & van den Bosch, 2005; Daelemans et al., 1999) – http://www.clips.ua.ac.be/pages/MBSP [October 2011].

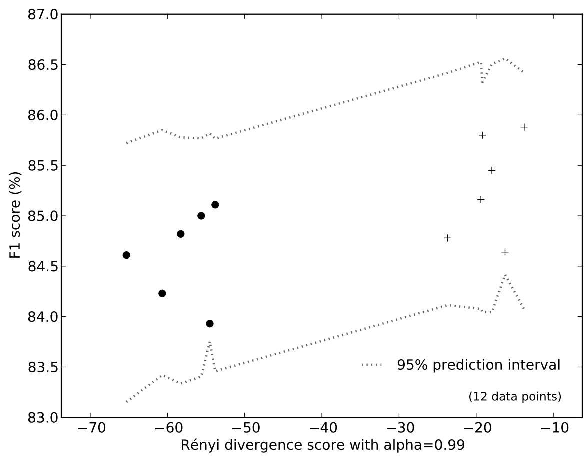


Figure 3.10: F1 score against Rényi divergence for out-of-domain prepositional phrase attachment experiments. The data point is marked with a + if both source and target domain come from the OntoNotes corpus. If the GENIA corpus is either used as source or as target domain the data point is marked with a •.

parison, the number of tokens in each domain is normalized. For each domain, random samples of equal size are taken. The samples have the same number of tokens as the smallest domain, which is broadcast conversation with 209,346 tokens (see Table 3.2). During sampling, the sentences are kept intact, resulting in a corpus of 209,346 tokens for broadcast news, a corpus of 209,350 tokens for newswire, and a corpus of 209,358 tokens for GENIA.

Each domain is used once as the training corpus for the MBPA, while another domain serves as the test corpus. A combination of two domains results in 25 test/training experiments, because each domain is split into five parts and each part of the training corpus is once combined with each part of the test corpus. Because there are four domains, the outcome is twelve data points obtained by 25-fold cross-validation. Evaluation of an experiment is done by computing the F1-score for the output.

Figure 3.10 shows the outcome of the PP-attachment experiments on the GENIA and OntoNotes corpora. The figure is built up in the same way as Figure 3.4. The x-axis is the Rényi 0.99 divergence between the test and training corpus and the

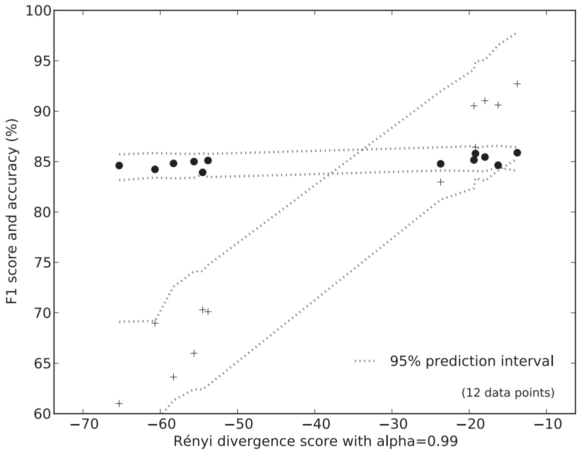


Figure 3.11: Performance versus Rényi divergence. The PP-attachment F1-score (•) is depending less on the divergence than the POS accuracy (+).

y-axis gives the PP-attachment F-score. The data points that are represented by a cross (+) are training and test combinations of strictly OntoNotes domains. The data points that are represented by a bullet (•) are combinations of an OntoNotes domain and the GENIA corpus.

The correlation coefficient r^2 is 0.398, which is low. Visual examination of the figure supports the hypothesis that there is a weak linear relation between the divergence and the F-score. The visual linear relation mainly comes from the F1-score drop when GENIA is included in the experiment. When examining only the experiments with OntoNotes (+) r^2 value drops to 0.197 and the relation between divergence and F1-score seems to be lost.

Domain insensitivity or robustness can be a possible explanation for the weak relation between divergence and F1-score. The effect of the robustness can be seen in Figure 3.11, which reproduces the PP-attachment (•) figures from Figure 3.10, but the outcomes of POS-tagging experiments using the same corpora are included (+). The r^2 value for the POS experiment is 0.966 (0.792 OntoNotes only), which is very good. The data clusters at the left of the Figure 3.11 are the data points when GENIA is introduced as training or as test corpus. The clusters at the right are the result of *pure* OntoNotes experiments. Figure 3.11

shows that the POS experiment accuracy drops considerably when GENIA is part of the POS experiment, although the performance drop is much less for the PP-attachment experiment.

Plank & van Noord (2010) argue that domain sensitivity appears to be relying heavily on lexical information and this is what can be observed here. For both the POS labeling task and the PP-attachment task, the distance between training and test corpus is computed using relative token frequencies, meaning that the distance is based on the same information for both tasks. The distance between GENIA and the OntoNotes domains is rather large. The scale of Figure 3.7 can be taken as a reference for the claim that the distance is large. The lexical variation between biomedical and journalistic texts puts a huge strain on the POS labeler, but does not do so for the PP-attacher. The PP-attacher appears to be robust to the lexical variation and consequently the linear relation between the divergence and the performance is lost.

3.5.3 Lexical robustness

An additional experiment has been conducted to obtain a second view on the importance of lexical features when carrying out PP-attachment. The core of the PP-attacher system is a kNN based algorithm that indicates if a given prepositional phrase should be linked to a given candidate anchor point or not. The instances that are fed to the machine learner contain three lexical features on a total of fifteen features: the lemma of the candidate anchor, the lemma of the head of the prepositional phrase, and the lemma of the token in front of the prepositional phrase. The other features of the instances contain non-lexical information such as POS tags, presence of a hyphen, etc.

For the experiment, the same corpora as in the previous experiment are used: the three domains from OntoNotes and the GENIA domain. Twelve out-of-domain experiments can be run if every domain is once used as test and once as train in combination with every other domain. The average micro-averaged F-score for the twelve out-of-domain experiments with all features included is 93.31% (standard deviation 0.82%). It has been argued that lexical features are responsible for diminished performance on out-of-domain test corpora (Plank & van Noord, 2010). In a collection of 455 experiments, three of the fifteen features are omitted and the twelve out-of-domain experiments are rerun, amounting to a total of 5,460 experiments.

The number of omitted lexical features is used to divide the experiments into groups. The average micro F-scores for each of those four groups are given in Table 3.10. The first column shows the number of lexical features that are removed. For every experiment, three features are omitted, meaning that omitting three lexical features is the upper limit. The second column contains the average micro F-score for all experiments that contain the same number of lexical features. The third column contains the standard deviation (σ) on the micro F-score. Note that the standard deviation can be fairly large because some domain pairs are more compatible than others.

The last column contains the number of experiments that are averaged. For example, if three features from the instances are omitted simultaneously, then in $\frac{220}{455} = 48.35\%$ of the cases none of the omitted features will be a lexical feature. Since there are three lexical features, in $\frac{1}{455} = 0.22\%$ of the cases all three omitted features will be lexical. The last column stresses that a larger standard deviation does not necessarily come from a larger number of experiments in that group. The larger standard deviation comes mainly from the quality difference of the excluded features.

omitted lexical features	micro F-score	standard deviation	share
0	89.00%	6.12%	48.35%
1	90.01%	8.06%	43.52%
2	92.00%	7.24%	7.91%
3	94.64%	0.42%	0.22%

Table 3.10: Effect of omitting lexical features on performance. Averaged micro-averaged F-score and the standard deviation are given along with the percentage of experiments that fall into each group.

It can be seen that ignoring lexical features increases out-of-domain performance to such a degree that ignoring all lexical features improves the micro F-score when all features are used. This is the situation for out-of-domain experiments and it illustrates that the PP-attacher learns mainly from non-lexical features. During the development of the PP-attachment system, it has been optimized for in-domain usage and for in-domain experiments lexical features are more likely to contain helpful information.

3.5.4 Conclusion

In this subsection, the relation between distance and performance has been tested for a task that differs from the POS labeling task of the previous section. It has been found that the PP-attachment task is more robust to a domain shift than the POS labeling task. From the experiments in this section, it can be seen that NLP tasks can be positioned on a scale ranging from domain sensitive to domain insensitive NLP tasks.

In the case of domain sensitive machine learning, domain sensitivity comes from lexical differences between training and test corpus and a distance metric is able to capture these differences, thereby providing information about out-of-domain performance. There is no sense in measuring the distance between training and test corpus in settings, when the machine learner is robust to domain shifts. The performance will hardly vary and the distance metric cannot provide useful information. It is even shown that removing lexical features for the domain insensitive PP-attachment task improves the performance for out-of-domain experiments.

Lexical features have a great influence on the domain sensitivity of a task. In this subsection, it was shown that measuring the distance is only useful for domain sensitive tasks. This means that, if measuring the distance is useful for domain sensitive tasks and if the domain sensitivity is influenced by lexical features, it is a logical decision to base the distance on lexical information, *viz.* the relative token frequencies. Regardless of this conclusion, it may still be useful to base the distance between corpora on information different from token frequencies. In Section 4.3 on feature selection, this idea of basing a distance on other information than token frequencies is further investigated.

3.6 Conditional probability and performance

From a statistical point of view, the machine learning problem can be expressed by stating that the probabilities of the instances in the target domain, $P_t(X,Y)$, are estimated by a model based on the probabilities of the instances in the source domain, $P_s(X,Y)$. A joint probability can be resolved into factors: $P(X,Y) = P(Y|X)P(X)$. The token frequencies in the previous sections are an instantiation of the probability factor $P(X)$ and these sections test the hypothesis that $P(X)$ can be linked to machine learning performance. The first condition that has to be satisfied for this hypothesis to be true is that there should be an actual performance difference. The second condition is that the probabilities of the target domain, $P_t(X)$, should be compared to the probabilities of the source domain, $P_s(X)$ using a suitable metric. If these two conditions are satisfied, there is a link between token frequencies and performance and the strength of the method comes from the fact that no class labels are needed when computing $P(X)$.

There is, however, a second factor that has an influence on machine learning performance, which has been disregarded so far in our discussion. $P(Y|X)$ expresses how the use of tokens may vary between domains. An example from the British National Corpus is the token *fat*, which is mostly an adjective to describe a person's figure in the *imaginative* domain and a noun to refer to the actual substance in the *natural science* domain.

Because the class labels must be known to find $P(Y|X)$, $P(Y|X)$ cannot be used for performance prediction, but there should be a relationship with performance as well as there is one for $P(X)$. In this section, this relation is briefly investigated.

A subset of the corpora used for the POS experiments in Section 3.4, has been selected. The BNC corpora have been reduced to contain only those sentences that consist of tokens with a POS label that is associated with more than 1% of the tokens in the entire BNC.[8] This has been done to minimize the influence of low frequency tags, which may obfuscate general trends when looking at the data in detail. Filtering out low frequency POS labels does not radically change the nature of the experiment, as can be deduced when a divergence-performance plot similar to Figure 3.4 on page 59 is made. The r^2 correlation is 0.956 before filtering out the low frequency labels and remains unchanged after filtering. Eight POS labeling experiments have been conducted with the memory-based tagger. For each experiment, the *imaginative* domain is taken as the target domain and one of the other eight BNC domains is taken as the source domain.

After running the experiments, the effect of $P(Y|X = x)$ for a given token x on the predicted POS labels for that token, can be analyzed. Table 3.11 shows the effect of the different ordering of class label frequencies in source and target domain for two tokens: *straight* and *locked*. The *straight* token is chosen because it has two different class labels in all domains. The *locked* token is chosen because it is missing from some domains and it has multiple class labels in other domains.

[8]These labels are: VVG, VBZ, DPS, VM0, CJS, PUQ, VVD, TO0, CRD, VVN, DT0, VVI, PRF, CJC, PNP, NP0, AV0, NN2, AJ0, PRP, AT0, PUN, NN1.

	straight			*locked*	
source	accuracy	τ_b	source	accuracy	τ_b
applied	31.6%	-1	arts	20.0%	-0.82
commerce	42.1%	-1	leisure	20.0%	-0.82
natural	42.1%	-1	world	40.0%	0.50
arts	57.9%	0	commerce	60.0%	0.82
social	73.7%	1	belief	60.0%	–
belief	78.9%	1	applied	80.0%	–
leisure	78.9%	1	social	80.0%	0.82
world	78.9%	1	natural	100%	–

Table 3.11: Influence of $P(Y|X{=}\textit{straight})$ and $P(Y|X{=}\textit{locked})$ on performance. The accuracy of assigning POS labels is given for two tokens and split out for different source domains. The *imaginative* domain is taken as the target domain. The Kendall τ values are an indication of the changed order of the probabilities of the POS labels in the source and target domain.

A large portion of the tokens has only one associated class label, which makes them less interesting to examine. The examples in Table 3.11 are mainly chosen to illustrate the general trend that a reordering of the conditional frequencies of the class labels in source and target data will lead to a decreasing performance. However, as we will see, the performance is influenced by more factors.

The first three columns of Table 3.11 contain the data for the token *straight*, the last three columns are the data for the token *locked*. For every experiment, the target domain consists of the *imaginative* texts. The source domain is indicated in the first column. For every source domain, the accuracy of the POS labeling experiment for the token is given in the second column. The third column contains the value of Kendall's τ_b. τ_b has been calculated with a distribution consisting of all $P_t(Y|X = \textit{straight})$ and the corresponding distribution from a source domain. A τ value of 1 indicates that the order of the class labels is the same for the target domain (imaginative) and the source domain when the labels are sorted according to frequency. A value of -1 indicates that the order is reversed and intermediate values indicate partially reversed orders. When a token does not appear in the source domain, τ cannot be calculated.

As can be seen in Table 3.11, source domains for which the ordered values of $P_s(Y|X = \textit{straight})$ are the same as for the target domain, are best in assigning a POS label to the token *straight*. However, $P(Y|X)$ is not the only factor that is involved. The POS labeler does not only use information about a token, it also uses low-level morphological features (last characters of the token) and

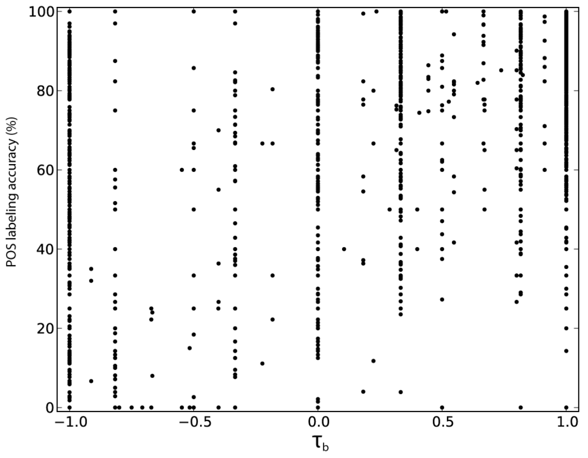


Figure 3.12: The general trend induced by a reordered class label frequency distribution in target domain, if compared to the source domain. Each data point is a token in a given source domain/target domain setting. The x-axis is the τ_b value; the y-axis is the POS labeling accuracy. The general trend is that tokens for which the class labels frequencies are ordered in the same manner in source and target corpus ($\tau_b \approx 1$), obtain higher accuracies. The trend is obfuscated because class label order is not the only factor that influences the labeling process, but nevertheless a r^2 value of 0.684 is obtained.

information about the context of a token. This way the POS labeler manages to obtain an 80% accuracy for the token *locked*, although the token does not appear in the *applied science* corpus. This can only be explained by the influence of morphological and contextual features, but the trend that reversing the order of the class labels for a token is harmful, is still noticeable.

Because it is impossible to investigate all tokens in Table 3.11, a plot is made of all tokens in all source/target settings. Figure 3.12 contains those data points and it can be seen that a similar class label order in test and training corpus has a positive effect on performance.

Another way of looking at the data is by summing the τs for all tokens in the target corpus to get a sense of the combined effect of all differences in $P(Y|X)$. The result is that the sum of all τs is correlated with the overall performance

($r^2 = 0.972$), meaning that $P(Y|X)$ is an important factor during POS labeling.

Instead of Kendall's τ, every divergence could be used to determine the variance in $P(Y|X)$, but Kendall's τ is the metric of choice, because the order of the class labels can be considered to be more important than the actual frequency of the class labels. This intuition is confirmed when the correlation of the Kullback-Leibler divergence with the overall performance is computed and proves to be only 0.406.

The observation that $P(Y|X)$ is an important factor, has led to many domain adaptation techniques that try to mitigate the variance of $P(Y|X)$ between source and target domain and, as is empirically shown in this section, performance gain can be expected from this approach.

3.7 Conclusions

In this chapter, the quality of the distance metrics from Chapter 2 is investigated. In a first section, the minimal corpus size needed to find a stable metric value, has been set to 1,500,000 tokens. This figure has been deduced from an experiment on the BNC with the Rényi divergence and the Euclidean divergence, serving merely as an indicative figure.

Based on various part-of-speech labeling experiments, it is shown that a probability-based divergence metric based on relative token frequencies, captures information that is linearly related to the performance. The probability-based divergence metrics that came out best, are the Kullback-Leibler and Rényi with an α value a little under 1. The general applicability of the method has been tested by varying the labeling system (Section 3.4.2), the corpus (Section 3.4.4) and the homogeneity of the corpus (Section 3.4.5)

The homogeneity experiments of Section 3.4.5 indicate that the relation between distance and performance may be the beginning of a method to assign domain labels objectively, *i.e.* without depending on the judgement of a human annotator. If the relation is linear for a set of corpora and a second set of corpora does not exhibit this linear relation, the second set of corpora may be internally homogenous and could therefore be labeled as a single domain.

Repeating the POS experiments for the prepositional phrase attachment task led to the formulation of two conditions that should be satisfied before the relation between distance and performance can be explored: The first condition is that there should be an actual performance difference when carrying out an experiment. The second condition is that the probabilities of the target domain, $P_t(X)$, should be compared to the probabilities of the source domain, $P_s(X)$, using the right metric. If these two conditions are satisfied, there is a link between token frequencies and performance and the strength of the method comes from the fact that no class labels are needed when computing $P(X)$.

From a statistical point of view, the machine learning problem can be expressed by stating that the probabilities of the instances in the target domain, $P_t(X,Y)$, are estimated by a model based on the probabilities of the instances in the source domain, $P_s(X,Y)$. A joint probability can be resolved into factors: $P(X,Y) = P(Y|X)P(X)$. The token frequencies in the previous sections are an instantiation of the probability factor $P(X)$ and the sections handle about the hypothesis that $P(X)$ can be linked to machine learning performance. In the experiments of Section 3.6, the influence of $P(Y|X)$ on the performance is investigated. It has been found that the conditional probabilities are an important factor, but $P(Y|X)$ cannot be used for performance prediction, because the class labels are needed to compute the conditional probabilities.

In this chapter, the focus was on the first two research questions of this dissertation, namely, (1) is there a monotonic relation between distance and performance, and (2) is the correlation easily traceable? The experiments showed that a linear relation between distance and performance could be found for a large selection of metrics. The quality of the linear relation varies, although, with the top metrics, high correlation values can be obtained. Secondly, the traceability of the correlation depends not only on the nature of the correlation, but also on the corpus size and the machine learner that has been used. Experiments are carried out to find a rule-of-thumb value for the minimal corpus size needed. The value remains an approximation, since it may be tied to the corpus from which it was derived. The correlation is preserved when different algorithms are used to carry out POS-tagging. The fact that most variables, except the NLP task, do not seem to interfere with the linear correlation between distance and performance indicates that using a distance can be a robust element in domain adaptation methods.

In the next chapter, the conclusions of this chapter will be used to test the Rényi distance metric in different types of natural language processing applications. The predictive power of the relation between distance and performance may be used to select a good training corpus, to select good sentences for self-training, or even to select good features.

Chapter 4

Applying distances

4.1 Introduction

In the previous chapter, the correlation between a metric and the performance was investigated. In this chapter, three applications that can benefit from the use of a metric are carried out: training data selection, feature selection, and corpus selection.

Before the experiments are presented, it may be interesting to focus again on the two different aspects of metrics for natural language corpora, discussed in Section 1.3 on page 14:

Aspect *a*: quantifying the (dis)similarity of the corpora,
Aspect *b*: linking distance to performance.

Aspect *a* simply expresses the distance between two corpora. For example, a distance between two corpora can have a value of 4 and the distance between two other corpora can be 6. This means that the first two corpora are closer to one another than the second pair, although there is no connotation linked to this distance difference.

Aspect *b* is the link with performance and provides additional meaning to distance differences. Looking at the example, the first pair of corpora should lead to a better performance than the second pair, because they are closer to each other. This predictive dimension can only be added to a distance metric, if there is a good correlation between distance and performance for a given task.

The distinction between the two aspects may seem an unnecessary complication, but in this section it is shown that the importance of the second aspect varies depending on the exact usage of the metric in the experiment. Aspect *b* is important if the experiment consists of ranking several training corpora according to the performance, when they are used to label a test corpus. The tighter the link between distance and performance, the better the ranking. If a training data selection experiment like in Section 4.2 is carried out, the link between distance and performance is not the only influencing factor. The stability of the metric, the *gamut* of the metric, and the sensitivity to different corpus sizes are only three factors on which the successful application of the metric will depend. All these factors determine the way the metric will quantify (dis)similarity and will cause faster/slower, more eager/economical data selection.

The correlation between distance and performance can be used as a selection method to find the most suitable distance metric for an experiment, if the importance of aspect *b* is pronounced. If aspect *b* is less important, the correlation will be less informative and the most useful distance metric should be selected on other grounds such as sensitivity to infrequent tokens, sensitivity to corpus size, etc., none of which can be easily modeled analytically.

In Section 4.2, a training data selection experiment is used to illustrate the importance of aspect *a*. In Section 4.3, the relation between performance and a distance is investigated for a distance that is computed using the features of instances, rather than using token frequencies as in Chapter 3. This setup will provide information about the use of a distance metric for feature selection. Finally, in Section 4.4, unlabeled corpora are selected according to the expected performance gain when the corpora are added to a training corpus during self-training. The experiment in Section 4.4 involves performance prediction, so it can be expected that the correlation between distance metric and performance will play a recognizable role.

4.2 Training data selection

4.2.1 Training data selection algorithm

Selecting data from a large corpus in order to keep only the interesting sentences, is the first application of divergence metrics that we investigate. First, we will describe some related research.

A similar experiment for language modeling has been carried out by Moore & Lewis (2010), who use the difference in cross-entropy of a sentence, according to an in-domain language model and a model trained on a random sample from a large training set, to select those sentences that will help decrease test set perplexity. An elaborate training selection experiment is presented by Gao et al. (2002). They present a training set selection experiment for the construction of a language model. The algorithm of Gao et al. (2002) consists of four steps that show a resemblance to the algorithm in Figure 4.1. An example of training data selection that is not designed for language modeling, but for natural language parsing, can be found in Plank (2011). In this work, data is added to the training set for a parser, adding the closest data first. Different metrics were applied to quantify the closeness of the data and she found that Jensen-Shannon leads to the best results, while the Rényi divergence leads to the worst results. As we will discuss later in this chapter, the preference of using the Jensen-Shannon divergence cannot be derived straightforwardly from the correlation of the metric with performance. In addition, parsing is a different task from POS-tagging. We showed in the previous chapter that the distance–performance correlation varies for different tasks.

The algorithm for the experiments in this section is presented in Figure 4.1. Consider a large corpus of labeled data that consists of data from various domains, but not from the test domain. It should be possible to select only those sentences from the labeled corpus that are relevant for tagging the test corpus. The relevance of a sentence is defined by the effect on the distance between the training corpus and the test corpus when the sentence is added. If adding the sentence decreases the distance, the sentence is retained, otherwise the sentence is dismissed. A threshold value ϵ is used to influence the decision when adding a sentence decreases the distance. Applying this algorithm, a new training corpus is created from the large corpus by sequentially testing all sentences against the test corpus.

```
INPUT    labeled (out-of-domain) data, test data
OUTPUT   subset of labeled (out-of-domain) data
step 1   Select a random sentence from the labeled data and call
         it the training corpus,
step 2   Compute the distance between the training corpus and
         the test data,
step 3   Add a random sentence from the labeled data to the train-
         ing corpus,
step 4   Compute the distance between the training corpus and
         the test data,
step 5   If the newly computed distance is lower than the previous
         distance, keep the sentence in the training corpus, other-
         wise remove the sentence again,
step 6   Repeat from step 3 until all labeled data is tested.
```

Figure 4.1: The training data selection algorithm.

The underlying idea of this experiment is that the effect of the linear relation between accuracy and distance should be that a higher accuracy is obtained when the distance is being minimized. A similar idea can be found in McClosky (2010), in which a combined distance is used to weigh corpora from a set of training corpora from different domains.

In the experiment of this section, the data consists of sentences tagged with POS information, extracted from OntoNotes (Weischedel et al., 2009). For more details about the OntoNotes corpus, see Section 3.5.

4.2.2 Selection experiment

For the experiments, the three domains from OntoNotes (bc, bn, and nw) are rewritten into a format that can be read by the memory-based tagger, which also has been used in previous sections. This enables us to conduct three rounds of experiments: each domain is taken once as the test domain and the other two domains are combined into one large, out-of-domain training set. The training set is randomized at the sentence level and the algorithm from Figure 4.1 is implemented (with $\epsilon = 0$).

Quality of the selected subset

The selection from the labeled out-of-domain data that is obtained through the training data selection experiment, is called the *selected subset*. It is possible to repeat the experiments using any distance metric in step 4 of the algorithm in Figure 4.1. For the first set of experiments, the distance metric is the Rényi divergence ($\alpha = 0.99$), but experiments have also been run with the Jensen-Shannon divergence, the sUWR[1], and the overlap metric.[2] The size of the final selected subset depends on the type of metric that is used.

The selected subset can be used as the training corpus in a memory-based POS-labeling experiment, with the test corpus that has been used during training data selection as the test corpus. The POS-labeling accuracy can be compared with two reference accuracies: the accuracy when the entire out-of-domain corpus is used as the training set and the accuracy when a random subset with the same size as the selected subset is taken from the out-of-domain data.

Table 4.1 shows the outcome of the experiments with Rényi as the distance metric. There are three columns for each domain: the first column (full corpus) shows the figures when all data from the other domains is used as training data, the second column (selected subset) shows the figures when training data selection has been carried out on the full corpus, and the third column (random subset) shows the figures when a random sentence sample is taken from the full corpus. The random sample is of the same size as the selected subset. The first row gives the number of sentences in the training corpus and the second row gives the same number, but relative to the full corpus. The third row shows the Rényi ($\alpha = 0.99$) divergence between training and test corpus. The higher the value (less negative), the closer training and test corpus are to each other. The last row shows the overall accuracy of a default POS-tagging experiment with the memory-based tagger.

From Table 4.1, it can be seen that – for the three domains – the training corpus can be compressed to approximately 40% through training data selection with only little accuracy loss.[3] There is a small accuracy gain when comparing the accuracy of the training data selection to the accuracy of a random sample. This

[1]The fraction of tokens in the test corpus that are not in the training corpus.

[2]The fraction of tokens in the training corpus that are not in the test corpus.

[3]In a preliminary version of this experiment the Wall Street Journal data was not included in the full training data for bc and bn. As a consequence, the distance between training and test data was slightly larger. In contrast to the experiments in Table 4.1, the result of these preliminary training data selection experiments was a 0.1% accuracy gain for both bc and bn.

	full corpus	selected subset	random subset
broadcast conversation (bc)			
size	39,719	14,753	14,753
relative size	100%	37.1%	37.1%
distance	-15.03	-14.70	-16.85
accuracy	88.96%	87.37%	86.97%
broadcast news (bn)			
size	41,984	16,655	16,655
relative size	100%	39.7%	39.7%
distance	-5.58	-5.38	-8.14
accuracy	95.28%	95.02%	94.32%
newswire (nw)			
size	26,559	11,044	11,044
relative size	100%	41.6%	41.6%
distance	-13.16	-12.94	-17.70
accuracy	92.29%	92.24%	91.08%

Table 4.1: Training data selection with Rényi ($\alpha = 0.99$) divergence as the distance metric for POS labeling experiments. The accuracies for the selected subsets are significantly smaller than those for the full training corpus on a 0.01 significance level with approximate randomization tests. The random subset is the average of ten folds. Significance cannot be computed but the size of the accuracy difference can be compared with those between the full corpus and the selected subset: most probably the accuracy differences between the selected subset and the random subset are significant.

indicates that there is an advantage in using a distance metric for training data selection.

The experiment is repeated with $\epsilon = 1$ (a new sentence is allowed to increase the distance with a value of 1) and $\epsilon = -1$ (a new sentence should decrease the distance with at least -1). Changing ϵ did not improve on the final result. With $\epsilon = 1$, all original sentences are included in the selection. With $\epsilon = -1$, too few sentences are included in the selection. Increasing ϵ to -0.01 did not lead to the selection of more sentences than with $\epsilon = -1$.

From the results of the training data selection experiment, it can be concluded that a large training corpus can be compressed to approximately 40% without overly harming the performance. Training data compression can be useful when data storage is limited, but it can also speed up algorithms. Many machine-learning algorithms are hindered by a time-consuming training phase. For eager learning, the training phase will get considerably shorter when the training data is half its original size. For lazy learning, the test phase will get shorter. Training data compression as it has been carried out in this experiment, is not suited to increase performance. It is possible to obtain a compressed training corpus that is closer to the test data than the full training corpus, although the effect is not sufficient to overcome normal variation.

Using other distance metrics

The Rényi divergence score is used for the experiments that are reported in Table 4.1. The same experiments are repeated with Jensen-Shannon divergence as the distance metric. The Jensen-Shannon divergence is chosen for two reasons: contrary to the Rényi divergence it is a symmetric divergence $\big(JS(P;Q) = JS(Q;P)\big)$, and Plank (2011) reports that Jensen-Shannon clearly outperforms the Rényi divergence – albeit for a different task. The results are reported in Table 4.2.

In general, the selected subsets acquired with Jensen-Shannon divergence as the distance metric are smaller than those for the Rényi divergence. The accuracies are also slightly lower than for the Rényi divergence, but this can be attributed to the smaller subset size and should not be attributed to a lower quality of the selected data as will be shown in the next subsection. The fact that the selected subsets obtained with the Jensen-Shannon divergence, are smaller than the ones obtained with the Rényi divergence can be attributed to a different way of quantifying the distance.

	full corpus	selected subset	random subset
broadcast conversation (bc)			
size	39,719	8,165	8,165
relative size	100%	20.6%	20.6%
distance	0.219	0.118	0.226
accuracy	88.96%	86.82%	85.75%
broadcast news (bn)			
size	41,984	11,902	11,902
relative size	100%	28.3%	28.3%
distance	0.147	0.089	0.155
accuracy	95.28%	94.77%	93.72%
newswire (nw)			
size	26,559	6,677	6,677
relative size	100%	25.1%	25.1%
distance	0.183	0.118	0.205
accuracy	92.29%	91.75%	90.32%

Table 4.2: Training data selection with Jensen-Shannon divergence as the distance metric for POS labeling experiments.

	full corpus	selected subset	random subset
sUWR			
size	39,719	5,378	5,378
relative size	100%	13.5%	13.5%
distance	0.218	0.218	0.456
accuracy	88.96%	86.47%	85.36%
Overlap			
size	39,719	2,932	2,932
relative size	100%	7.4%	7.4%
distance	0.760	0.004	0.501
accuracy	88.96%	84.22%	84.13%

Table 4.3: Training data selection for the broadcast conversation (bc) domain with sUWR and overlap as the distance metric for POS labeling experiments.

	selected subset	random subset	relative size	r^2
Rényi	87.37%	86.97%	37.1%	0.990
JS	86.82%	85.75%	20.6%	0.840
sUWR	86.47%	85.36%	13.5%	0.925
overlap	84.22%	84.13%	7.4%	0.187

Table 4.4: Results for the broadcast conversation (bc) domain regrouped from Tables 4.1, 4.2, and 4.3 together with the correlation r^2 from Table 3.4.

Table 4.3 shows the results for two other distance metrics: sUWR and overlap. The sUWR is chosen because it has a different theoretical basis than the Rényi divergence and because McClosky (2010) has reported on its qualities. Overlap is included because of its very low correlation with the performance of a POS labeling task, see Table 3.4.

The results are reported for the broadcast conversation (bc) domain only, because the other domains show similar trends. As can be seen, the selected subsets are smaller than for the Jensen-Shannon and Rényi divergence. Nevertheless, the scores in Table 4.3 indicate that the usage of a distance metric leads to a more qualitative subset than a random sample of comparable size, although the accuracy difference between selected and random subset for the overlap metric is not encouraging.

The data for the broadcast conversation (bc) domain from previous tables is repeated in Table 4.4. Examination of Table 4.4 leads to the conclusion that a better correlation between distance and performance does not guarantee a better training data selection – sUWR is better correlated than Jensen-Shannon divergence, but the accuracy for the selected subset is higher for Jensen-Shannon divergence. Despite this observation, there appears to be a positive influence of a better correlation on the improvement of the selected subset over the random subset. The correlation for the overlap metric is very low, but even then the accuracy of the selected subset hints at the fact that using a distance metric – albeit a badly correlated metric – helps when data has to be selected from a larger corpus. The difference between the accuracy of the selected subset for overlap (84.22%) and the random subset (84.13%) is not significant[4] though.

Using a metric helps when selecting data, regardless of the quality of the correlation between the metric and performance. The fact that the correlation has

[4]At 95% significance level with approximate randomization testing on sentence level.

	selected subset	random subset
Rényi	84.87%	84.13%
JS	84.87%	84.13%
sUWR	84.43%	84.13%
overlap	84.22%	84.13%

Table 4.5: POS-tagging accuracy with training data selection for the broadcast conversation (bc) corpus with Rényi, Jensen-Shannon (JS), sUWR, and overlap as the distance metric. All training corpora are samples of 2,932 sentences. According to approximate randomization tests (99% significance level), Rényi and JS perform significantly better than sUWR. sUWR performs significantly better than overlap. The difference between the overlap metric and a random selection is not significant.

an influence on the outcome of the training data experiment can also be seen in Table 4.5 despite the non-trivial relation between correlation and the outcome of the experiment. In order to calculate the scores in this table, a sample of the size of the smallest subset, *i.e.* the subset obtained with the overlap metric, is taken from the selected subsets for the broadcast conversation (bc) domain for four metrics (Rényi, JS, sUWR, and overlap) from previous experiments. The reason for this is to eliminate corpus size effect. The disadvantage of downsizing is that a training corpus of 2,932 sentences is rather small. The samples are used to carry out a POS labeling experiment. The scores in Table 4.5 illustrate that better correlated metrics (Rényi, JS, and sUWR) tend to perform better than a badly correlated metric (overlap).

The observations of this subsection can be summarized as follows: using a metric during training data selection helps when compared to random selection. Although the link between the quality of the correlation and the outcome of the experiment is not very strong, the link seems to indicate that well correlated metrics lead to better results during a training data selection experiment.

Learning curves

In the previous subsection, we found that the Rényi divergence was able to select a subset that obtained the best accuracy for a POS labeling experiment, although the selected subset is also the largest subset of all alternatives that is selected. We found that well correlated metrics tend to perform better, but the link between

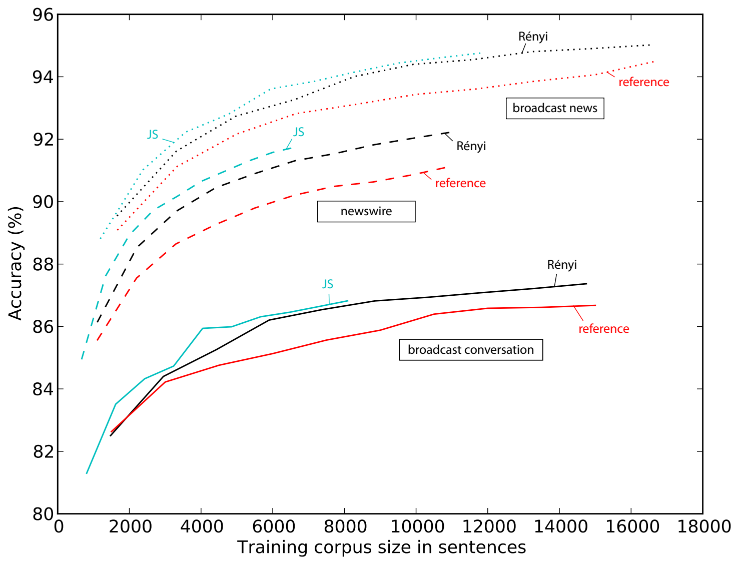


Figure 4.2: The learning curves for training data selection experiments. The x-axis represents the number of sentences in the increasing training corpus selection. The y-axis represents the overall accuracy of the POS-labeling experiment. The Rényi lines are the training selection experiments for which the Rényi divergence was used to measure the distance between test corpus and the subset selected from the full corpus. The lines for the experiments with Jensen-Shannon as the distance metric and the reference lines are also given. For the reference lines, a sample of a given size is taken from the full corpus, without taking into account any distance metric. The dotted lines are experiments on the broadcast news (bn) domain, the dashed lines are for the newswire (nw) domain, and the solid lines are for the broadcast conversation (bc) corpus.

correlation and the final result is not straightforward. It is interesting to have an additional view on the training data selection experiments.

During the experiments, the size of the selected training corpus continuously increases as long as a newly added sentence decreases the distance between the training and the test corpus. This setup enables us to draw learning curves.

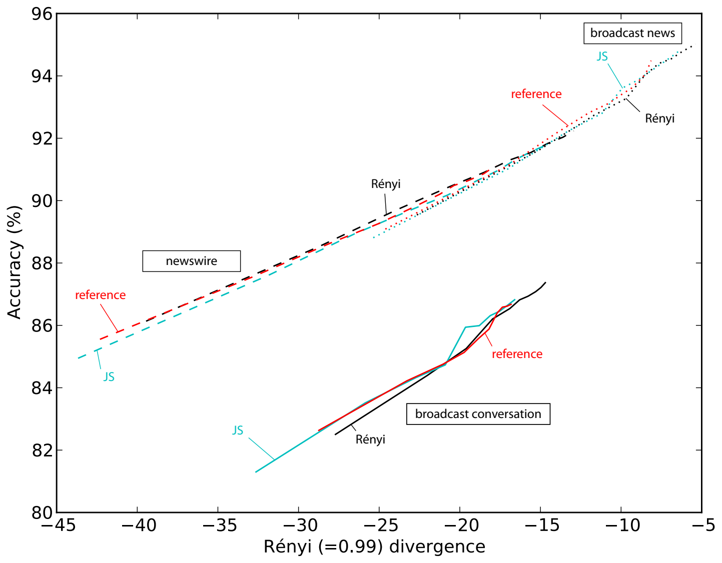
The curves of Figure 4.2 and 4.3 are based on ten samples collected during the data selection process. Since the selected subset for the broadcast conversation (bc) domain contains 14,753 sentences, the size of the selection increases with steps of 1,475 sentences. The other curves are constructed in the same manner, except that a data point is created by taking smaller or bigger steps according to the size of the selected subset.

The first learning curve is obtained by plotting the size of the selected subset versus the accuracy when that subset is used in a memory-based POS labeling experiment. Figure 4.2 contains the learning curves for the experiments of Tables 4.1 and 4.2. The experiments are carried out for three domains, which are represented by different line styles: solid lines (bc), dotted lines (bn), and dashed lines (nw). The Rényi lines are for the selected subsets of Table 4.1, *viz.* with Rényi as the distance metric, the JS lines are for Table 4.2, *viz.* with the Jensen-Shannon divergence. Reference lines are also given. These reference lines are obtained by random sampling the full corpus, although a sample is always contained in the next (larger) sample.

The learning curves of Figure 4.2 show that the Jensen-Shannon (JS) curves are positioned higher than the Rényi curves. Both curves show quicker learning than the reference, obtained through random sampling. The curves for the Jensen-Shannon divergence do not reach as far as the curves for the Rényi divergence because fewer sentences are selected for the final subset. For example, for broadcast conversation (bc): 14,753 sentences are selected for Rényi against 8,165 sentences for Jensen-Shannon.

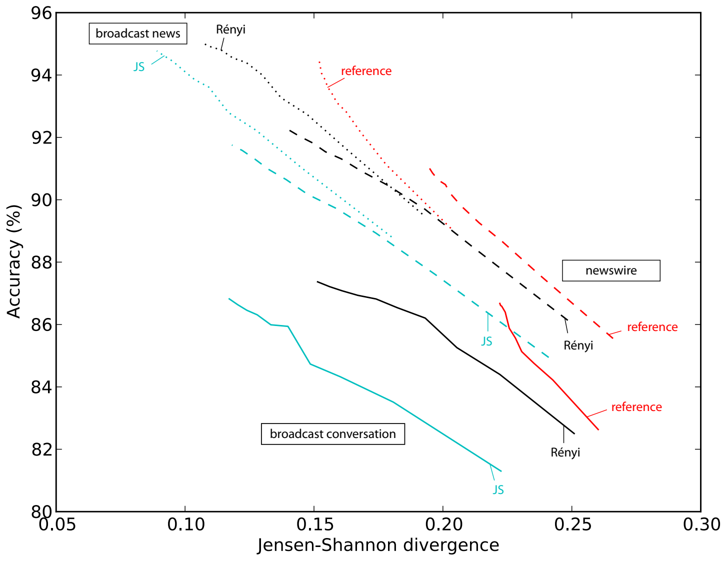
The fact that the JS curve is positioned above the Rényi curve is an indication that the Jensen-Shannon divergence is better at quantifying the distance between test and training corpus for the memory-based POS labeling experiment. At any moment, the Jensen-Shannon divergence is able to select a better sentence than the Rényi divergence. This behavior cannot be linked to a better correlation between the divergence and the performance (aspect *b*), but can be attributed to the different internal structure of the Jensen-Shannon divergence when compared to the Rényi divergence (aspect *a*).

A second view on the experiments can be obtained by plotting the distance between the test corpus and the snapshots of the selected training corpus versus the accuracy. Figure 4.3a contains the plot where the Rényi divergence is used to compute the distance. Note that the JS lines are still experiments with Jensen-Shannon as the distance metric, but here the distance at any given moment is expressed in terms of the Rényi divergence. Figure 4.3b contains the plot with JS distances. During the data selection experiment, the distance between the test



(a) Rényi divergence

Figure 4.3: The learning curves for training data selection experiments. The x-axes represent the distance value of different metrics. The y-axes represent the overall accuracy of the POS-labeling experiment. The Rényi lines are the training selection experiments for which the Rényi divergence was used to measure the distance between test corpus and the subset selected from the full corpus. The JS lines are for the experiments with Jensen-Shannon as distance metrics. The reference lines are also given. For the reference lines, a sample of a given size is taken from the full corpus, without taking into account any distance metric. The dotted lines are experiments on the broadcast news (bn) domain, the dashed lines are for the newswire (nw) domain, and the solid lines are for the broadcast conversation (bc) corpus. The plots are different views on the same experiments of Figure 4.2. (continues).



(b) Jensen-Shannon divergence

Figure 4.3: (continued)

corpus and the current selection decreases (becomes less negative in Figure 4.3a – less positive for Figure 4.3b).

Shorter lines – measured along the x-axis – in Figure 4.3 may represent slower learning. This is most clear when comparing the bc-line for random selection on Figure 4.3b with the bc-line for Rényi. As can be seen in Figure 4.2, the final corpus size of both experiments is similar, but in Figure 4.3b it can be seen that the JS distance varies less per step for the reference line than it does for the Rényi line, indicating that adding approximately the same amount of data does bring the training and test corpus more quickly together when the Rényi metric is used.

A second observation that can be made, is that the lines for Figure 4.3a coincide when the same test corpus is used. All solid lines coincide because for each line the bc test corpus is used to measure the distance. This is not the case for Figure 4.3b, in which lines for the same test corpus are only approximately similar. This difference can be attributed to the different nature of the metrics. If the solid lines would be extended by adding all data until the full corpus is used as the training corpus, all lines would end in the same point. On Figure 4.3a this point is at Rényi distance 15.03, accuracy 88.96%, see Table 4.1. On Figure 4.3b, the same point is at JS distance 0.219, accuracy 88.96%, see Table 4.2.

If the Rényi divergence is used to look at the experiment (Figure 4.3a), the way the end point is approximated only depends on the test corpus. If the test corpus and the distance between test corpus and training corpus are known, the accuracy is determined – all solid lines coincide. The nature of the Jensen-Shannon divergence is different. Figure 4.3b shows that there is not only an influence from the test corpus, but the training corpus also has an influence – the solid lines do not coincide. This can be illustrated by reconsidering the definitions of the Rényi divergence and the Jensen-Shannon divergence:

$$Rényi(P;Q;\alpha) = \frac{1}{(\alpha-1)} log_2 \left(\sum_k p_k^{\alpha} q_k^{1-\alpha} \right) \text{ with } \alpha \geq 0 \tag{4.1}$$

$$JS(P;Q) = \frac{1}{2}\left[KL\Big(P; \frac{P+Q}{2}\Big) + KL\Big(Q; \frac{P+Q}{2}\Big) \right] \tag{4.2}$$

For the Rényi divergence, there is only one factor expressing the difference between P and Q, namely the factor $p_k^{\alpha} q_k^{1-\alpha}$. If the test corpus P is fixed, the

	full corpus	selected subset	random subset
Adding 5 sentences			
size	41,984	26,790	26,790
relative size	100%	63.8%	63.8%
distance	-5.58	-5.52	-6.68
accuracy	95.28%	95.20%	94.90%
Adding 10 sentences			
size	41,984	31,850	31,850
relative size	100%	75.9%	75.9%
distance	-5.58	-5.55	-6.14
accuracy	95.28%	95.22%	95.09%
Adding 20 sentences			
size	41,984	37,080	37,080
relative size	100%	88.3%	88.3%
distance	-5.58	-5.57	-5.76
accuracy	95.28%	95.25%	95.15%

Table 4.6: POS-tagging accuracy with training data selection for the broadcast news (bn) corpus – adding five, ten, and twenty sentences simultaneously.

only variance comes from the divergence of Q from P. For the Jensen-Shannon divergence, there are two factors expressing a distance, namely $log_2(\frac{p_k}{\frac{1}{2}(p_k+q_k)})$ and $log_2(\frac{q_k}{\frac{1}{2}(p_k+q_k)})$.This means that the variance of the JS divergence does not only come from the difference between (a fixed variant of) Q and P (the first factor), but also from the way the training corpus Q differs from a fixed variant of itself (the second factor). This second factor is the reason that the evolution of the JS divergence during training corpus selection depends not only on the test corpus P, but also on the training corpus Q. This observation emphasizes the importance of the different ways the distance between two distributions is quantified by different metrics.

Adding more sentences per step

In the previous experiments, only one sentence was added per step, in this subsection the influence of taking steps of more sentences at a time is investigated.

Table 4.6 contains results for three experiments with the broadcast news (bn) corpus – similar to the experiments of Table 4.1 – except that five, ten, or twenty sentences are added simultaneously during step 3 of the algorithm presented in Figure 4.1. The Rényi divergence is used as distance metric.

The results show that adding more sentences per step leads to a higher accuracy compared to adding only one sentence per step. The disadvantage of adding more sentences is the final corpus size: when adding one sentence the full corpus is reduced to 39.7% of its sentences, when more sentences are added the size of the selection ranges from 63.8% to 88.3% of the original training corpus. If distance based training corpus selection is to be used in an application, the minor accuracy gain at the cost of a bigger corpus may favor the implementation with adding one sentence per step instead of adding more sentences, since in the former setup the reduction of the training corpus will be more marked.

The experiment has also been repeated adding 100 sentences per step, but adding 100 sentences to the selected subset always makes the selected subset more similar to the test corpus, leading to a selection that is equal to the full training corpus.

4.2.3 Conclusion

The Rényi metric is found to produce the subset with the highest accuracy, although the learning curves reveal that the Jensen-Shannon divergence is most interesting when it comes to efficiently selecting sentences from a large corpus. The fact that Jensen-Shannon quantifies the distance between test and training corpus in a more efficient manner, cannot be deduced from the correlation of the divergence with the performance of POS labeling experiments, because it is rooted in a different aspect of the metric. These results show that it is important to know which is the major aspect influencing the usage of the metric. For the experiments in this section the way in which distance is measured, is important for the quick selection of interesting instances. The self-training experiments of Section 4.4 are an example of experiments where the distance–performance correlation has a more pronounced influence.

4.3 Feature selection

4.3.1 Introduction

The relation between token frequency distributions and machine learning performance has been the focus of the previous chapter. As defined in Chapter 3, the relation between a corpus $\mathcal{C}$ and a distribution $\mathcal{D}$ is defined by the mapping M:

$$M : \mathcal{C} \rightarrow \mathcal{D} \tag{4.3}$$

In this section, the mapping will use the features of the actual machine learning instances instead of relative token frequencies. Additional information becomes available when feature instances are used to compute distances between corpora. We test whether this additional information can be used to carry out feature selection.

Because the mapping takes into account the separate features, detailed information about the behavior of features may become available. Access to this fine-grained information about separate features leads to the following concept that is to be tested and that is explained in more detail later:

A useful feature influences the correlation between a distance and the performance of a feature-based machine learner in a different way than a superfluous or harmful feature.

Feature selection algorithms have been studied before and can be divided into two groups depending on the availability of class label information during selection. Supervised feature selection uses class label information and can be divided into three groups: filters, wrappers, and a hybrid based on both previous methods (Liu & Yu, 2005; Gheyas & Smith, 2010). An example of a hybrid method is a combination of a simulated annealing algorithm and a genetic algorithm (Gheyas & Smith, 2010). Both algorithms are based on processing cycles in which new feature subsets are evaluated on a test set.

If no class information is used during feature selection, the method is called unsupervised feature selection or feature selection for clustering (Liu & Yu, 2005). The research of unsupervised feature selection can be classified in two categories: (1) clustering features into categories, and (2) eliminating redundant features by investigating their interdependence (Mitra et al., 2002). The feature selection method that we propose, is similar to the second category in the sense that candidate features are discarded according to the similarity between a feature subset with and without the candidate feature.

The setup of the experiments is given next in this section, together with the definition of an instance. What is meant by an instance and a feature is described in detail, because of the ambiguous usage of these terms in natural language processing. The definition of an instance serves as an introduction to the definition of instance distance. Finally, the introductory subsection is followed by two subsections with experiments: one with experiments based on synthetic data and one with experiments based on real data, *viz.* the British National Corpus.

4.3.2 Experimental setup

In previous experiments, the correlation between a memory-based POS-tagger and its performance is found to be linear. The input of the POS-tagger consists of tokens, grouped into sentences. This input is internally rewritten into instances that can be used by the machine learning components of the POS-tagger. For the experiments in this section, these instances will be used to compute the distances and these distances will be compared to the performance of the machine learner that uses the instances and not to the overall performance of POS-tagger. The goal of the experiments is to see whether removing features from the instances has an influence on this distance–performance correlation and whether the influence contains clues about the usefulness of the removed feature.

Before the correlation is tested, more details are presented on how an instance distance can be obtained.

Obtaining feature distributions

Many natural language processing tools are based on instances. An instance is constructed by applying feature functions f_i to observations x in order to obtain feature values. The combined output of all feature functions is an instance. If there are n feature functions, the instance will contain n feature values. The term *feature function* can also be abbreviated to *feature.* In the POS experiments, two instances are created for each token in the corpus: one instance for the *known words* classifier and one instance for the *unknown words* classifier.

The TiMBL[5] implementation of the kNN algorithm is used for the experiments (Daelemans et al., 2010). The TiMBL implementation is specially designed to facilitate natural language processing. For TiMBL, the feature values are categorical as opposed to the feature values for SVMs, which are – in their most basic form – binary. Since the feature values for TiMBL are categorical, it is easy to obtain a distribution P_i based on the ith feature function for a corpus of instances:

$$P_i = \{p_{i,k} : \text{relative frequency of feature value } k \text{ returned by } f_i\} \tag{4.4}$$

If a feature function f_i would return the token on which the instance is based, the set of feature values would cover all tokens in the corpus and P_i would be the relative token frequency, as it has been used in previous experiments.

Instance distance

When only relative token frequencies are used to compute the distance between corpora, only one distribution is extracted and the distance between two such corpora can be obtained by applying any metric d to the extracted distributions. When features instead of only tokens are used to extract distributions, it is possible to yield a distribution for every feature. If an instance has n features, this leads to n distributions. In order to reduce the distance to a single number, the instance distance between two instance corpora P and Q is computed as follows:

$$\text{instance distance}(P, Q) = \sqrt{\sum_{i}^{n} d_i\big(P_i, Q_i\big)^2} \tag{4.5}$$

with

$$\begin{aligned} P_i &= \text{distribution based on feature } x_i\text{; extracted from corpus } P \\ Q_i &= \text{distribution based on feature } x_i\text{; extracted from corpus } Q \\ d_i &= \text{metric for feature } i \\ n &= \text{number of features} \end{aligned}$$

[5] http://ilk.uvt.nl/timbl

Defined in this way, the instance distance can be regarded as the Euclidean norm of a vector for which each d_i defines a dimension. In the experiments in this section, each metric d_i is the Rényi divergence ($\alpha = 0.99$).

Note that this is not the only way to compute a distance based on instances: an alternative definition could include the harmonic mean of the feature distances instead of the root of the summed squares. Neither is it necessarily the case that the feature distance d_i is the same for each feature.

Correlation based on instance distance

One condition has to be tested before the instances can be used for feature selection experiments: there should be a linear correlation between a distance based on the instance and the performance of the machine learner.

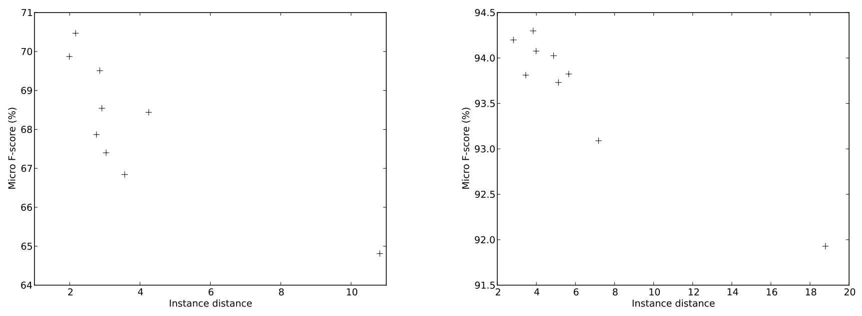
The input of the memory-based tagger is tokens grouped into sentences. In previous experiments, we found that a token-based distance is well correlated with the performance of the memory-based tagger. As explained on page 56, the memory-based tagger consists of two components: a classifier for *known words* and a classifier for *unknown words*. The input of the memory-based tagger is rewritten into feature instances and these instances are classified by the two classifiers.

The condition that there should be a correlation between the instance distance and the performance of a memory-based tagger component can be tested using the definition for instance distance presented above. Both internal classifiers of the memory-based tagger are a TiMBL classifier and the condition can be tested for the *unknown words* classifier and *known words* classifier.

Figure 4.4 contains the plots for both classifiers of the memory-based tagger. The x-axes represent the instance distance between the training corpus and the test corpus. The y-axes show the performance of the kNN-classifier.

There is a data point for each of the nine domains of the British National Corpus. One domain is taken as the test corpus and all other domains are grouped into one large training corpus. More background information about the experimental setup and the classifiers is given in Section 4.3.4.

Figure 4.4 shows that there is a reasonably good correlation between the instance distance and the performance for both the *unknown words* and the *known words* kNN classifier.



(a) Unknown words **(b)** Known words

Figure 4.4: Correlation between instance distance and performance for two memory-based classifiers.

The r^2 correlation for the *unknown words* classifier is 0.669; for the *known words* classifier, it is 0.910, which indicates a good relation between instance distance and performance. At the right of Figures 4.4a and 4.4b, there is an *outlier*, namely when the *imaginative* domain is taken as the test corpus. Removing this outlier would decrease the correlation between the instance distance and the micro-averaged F-score. Although there is no reason to exclude the *imaginative* domain, the lower correlation would indicate that the instance distance is not sufficiently sensitive to capture smaller differences between domains.

The result shows that the correlation between distance and performance for a memory-based tagger also holds for the internal classifiers of the tagger. Now that the correlation is confirmed for the instance distance, the influence of separate features on the correlation can be investigated and the instance distance may be used for feature selection, if it can be showed that useful features influence the correlation in a different way than superfluous or harmful features.

4.3.3 Analysis using synthetic data

Experiments on real data are essential, but it may be harder to get deeper knowledge about the problem because the inevitable variance of real data can introduce a lot of artifacts. In this first subsection, a range of experiments with highly controlled data should give more insight into the strengths and weaknesses of the techniques, before they are applied to real data. The synthetic data in this subsection consists of instances that are created using a random generator and by applying some basic rules.

Feature construction

In this section, synthetic data is produced by using an instance generator. The instance generator can produce good features, bad features, and random features. The different types of features are defined as follows:

Good feature. A good feature is a feature that increases the performance when it is inserted into the instances.

A good feature is constructed by taking the same number of feature values as there are class labels in the corpus of instances and each feature value is uniquely linked to a class label. An instance with a given class label will have the same associated feature value in the training corpus and in the test corpus. With this setup, a simple look-up strategy would suffice to solve the problem, because the feature value is directly linked to the class label.

Ambiguity is created by introducing a small chance α of inserting an *incorrect* feature value. The α value is a measure of the goodness of the feature. The smaller α, the better the association between the feature value and the feature. With real data, a feature can be better for one class label and another feature can be better for another class label. To mimic this behavior, α should vary along with the class labels.

Bad feature. A bad feature is a feature that introduces classification errors when it is inserted into the instances.

A bad feature is constructed by inverting the small chance of selecting an incorrect feature value to $1 - \alpha$ for the test corpus, while the chance of inserting an incorrect feature remains α for the training corpus. Because a feature value is associated with a class label with a high chance $1 - \alpha$ in the training corpus and with a small chance α in the test corpus, the machine learner will mistakenly assume that it should depend heavily on the feature. This will introduce a lot of classification errors.

Random feature. A random feature does not contain any information about the classification task and has no or little influence on performance.

A random feature is constructed by taking the same number of feature values as there are classes. However, there is no link between a feature value and a class label and the feature values are distributed randomly over the instances.

Instance construction

For the experiments, a set of five class labels is chosen in such a way that there is no class imbalance. This means that 20% of the instances in the test and training corpus carries class label `A`. The other class labels `B`, `C`, `D`, and `E` are represented to the same extent. To mimic the varying quality of a feature for different class labels, class label `A` is associated with an α value of 0.10. For class labels `B`, `C`, `D`, and `E`, α will be 0.20, 0.30, 0.40, and 0.50.

The random feature has a random value with no link to the class label.

For the good feature (the second feature), there is a good chance for the feature value to be the same in both training and test data. For class label `A`, the chance to have a good feature value in both test and training is $(1 - \alpha)(1 - \alpha)$ or $0.90 \times 0.90 = 0.81$.

The chance that the bad feature has the same feature value in training and test data is small. For class label `A`, the chance to have a good feature value in a test instance is α and to have a good feature value in a training instance is $1 - \alpha$. The chance to have a good feature value in both training and test instance then becomes $\alpha(1 - \alpha)$ or $0.10 \times 0.90 = 0.09$.

Because of the high correlation $1 - \alpha$ of a good feature value with the class label in the training data, the machine learner will learn to depend heavily on both the good feature and the bad feature, although during test time the bad feature will mislead the classifier.

In the first set of experiments, 2,000 TiMBL instances with five features are created per data file. The first three features are random features, the fourth feature is a good feature and the last feature is a bad feature. The fact that not one but three random features are included is to minimize the number of instances in the test corpus that can be exactly matched with an instance in the training corpus.

Corpus construction

The corpora for the experiments are organized as follows: nine test corpora of 2,000 instances with five features are created and to each test corpus a training corpus of 2,000 instances with five features added. The result is that nine default TiMBL experiments can be run with these corpora.

The number of instances is limited to 2,000 to minimize the number of exact matches between instances from a test corpus and from a training corpus. This is necessary to retain the need for a machine learner in a context with few different feature values per feature.

TiMBL has an option to exclude a feature from processing, making it possible to repeat the nine TiMBL experiments, while excluding one or more features.

Experiments

Nine test/training pairs are created and nine default TiMBL experiments are carried out. The average micro F-score for these nine experiments is 42.89% when all five features are included. The goal of the experiments is to find features that cause an increase in performance, when removed from the instance. The effect of removing one feature is studied by rerunning the nine experiments, while removing one feature at a time. The average F-scores for the nine experiments when a feature is excluded are shown in the second column of Table 4.7. The third column contains the difference with the performance when all features are included. The performance with all features is given in the last row of the table. A negative difference signals a negative effect on performance when the feature is excluded from the instances. As can be seen, removing the good feature (feature 4) is harmful, removing the bad feature (feature 5) is beneficial and removing a random feature (features 1, 2, and 3) has almost no effect.

First, Table 4.7 can be used to carry out a sanity check. It can be seen that excluding the good feature (feature 4) causes the performance to drop with 16.79%. Excluding the bad feature (feature 5) leads to a similar performance increase and excluding a random feature has only a minor effect on the performance. The minor effect of the random features may come from the fact there are only five feature values. This small set of feature values offers the machine learner the chance to learn something from the random features: any feature value may co-occur a little bit more often with a certain class label. Since there is nothing

excluded feature	F-score	performance difference	r^2	distance deviation	description
1	43.69%	0.80%	0.228	0.7%	random feature
2	43.15%	0.26%	0.209	0.5%	random feature
3	43.58%	0.69%	0.226	0.7%	random feature
4	26.10%	-16.79%	0.208	0.6%	good feature
5	59.52%	16.63%	0.038	79.7%	bad feature
none	42.89%	–	0.214	–	all features

Table 4.7: Micro-averaged f-score and r^2 when removing features.

to be learned, all inferred knowledge is deceptive. Removing the random feature will lead to a small performance gain.

Now that the influence of feature exclusion on performance is established, the relation between the instance distance and the performance can be investigated. To get a better insight into this relation, a plot of the instance distance versus the performance is presented in Figure 4.5. The information present in this figure will be discussed in the next paragraphs.

Because there are nine experiments per excluded feature, it is possible to compute the correlation r^2 between the test/training distance and the performance. It is possible to recompute this correlation for every set of experiments for which the same feature is excluded. These correlations are given in the fourth column of Table 4.7. The data points of the nine experiments when the bad feature (feature 5) is excluded are inside the grey selection of Figure 4.5. According to Table 4.7, the correlation between these nine data points is 0.038. The data points when one of the good or random features is excluded and when no features are excluded, do all coincide at the right of Figure 4.5.

From the data points in Figure 4.5, it can be seen that there is information in the instance distance that separates the bad feature (in the grey selection) from the other features. There is no information separating the random features from the good feature. The experiments needed to compose Table 4.7 and Figure 4.5 have been repeated three times to see if the correlation contains useful information for selecting bad features. The conclusion is that the correlation that differs the most from the correlation for the experiments with the complete instance, is mostly a bad feature. But this observation does not always hold. The reason for this is clear when looking at Figure 4.5. The correlation – which is a single

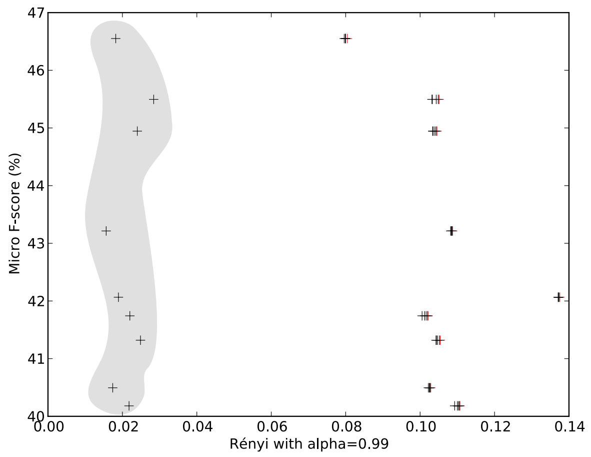


Figure 4.5: Effect on instance distance–performance plot when removing a feature. The x-axis is the Rényi divergence based instance distance. The y-axis is the micro F-score of the synthetic kNN experiment. The data points of the nine experiments when the bad feature (feature 5) is excluded are inside the grey selection.

figure describing the linearity of data points – is not the best way of describing the difference between the data points inside the grey selection and the other data points.

A better way to describe the difference between the data cloud when feature 5 is excluded, is by computing the average relative difference between the distance of the entire instance and the distance when the feature is excluded:

$$\begin{aligned} \text{distance deviation } &= \frac{d - d_i}{d} \qquad (4.6) \\ &\text{with} \\ &\quad d = \text{distance when all features are included} \\ &\quad d_i = \text{distance when feature } i \text{ is excluded} \end{aligned}$$

The distance deviations are given in the fifth column of Table 4.7. The bad feature (feature 5) is behaving clearly different from the other features when the distance deviation is used to describe the data. The bad feature has the highest distance deviation, namely 79.7%.

More features

The instances in the previous experiments only have five features. Many problems require instances with more features in order to capture the information needed to solve the problem. To investigate the effect of more features, the previous experiment is repeated with instances that have four good and four bad features. All good features are created with the same instance generator. This means that they are similar in nature. The same holds for the bad features. In addition to the good and bad features, four random features are also included.

The data is used twice. Table 4.8 contains data for an experiment that is an exact replica of the experiment of the previous subsection; only one feature is excluded when computing the distances. The results are presented in the same manner as in Table 4.7, except that the correlation r^2 is not reported because we have shown that r^2 does not contain interesting information. Secondly, Figure 4.6 shows the results when four features are excluded simultaneously.

excluded feature	F-score	performance difference	distance deviation	description
1	75.00%	0.01%	0.2%	random feature
2	75.03%	0.01%	0.2%	random feature
3	75.07%	0.05%	0.1%	random feature
4	75.02%	0.00%	0.2%	random feature
5	64.46%	-10.56%	0.2%	good feature
6	64.35%	-10.67%	0.1%	good feature
7	65.36%	-9.66%	0.1%	good feature
8	64.17%	-10.85%	0.0%	good feature
9	76.99%	1.97%	13.5%	bad feature
10	77.08%	2.05%	11.8%	bad feature
11	77.19%	2.17%	15.1%	bad feature
12	76.96%	1.94%	12.7%	bad feature
none	75.02%	–	–	all features

Table 4.8: The effect of removing one feature on performance and distance.

In Table 4.8, the third and fourth column are the most interesting. The fourth column contains the distance deviation – *i.e.* the difference between the test/-training distance of the complete instances and the test/training distance of the instances when a feature is removed – and the results show that the distance deviation of the bad features (features 9, 10, 11, and 12) is still clearly different from the distance deviation from the good features (features 5, 6, 7, and 8) and random features (features 1, 2, 3, and 4). It has to be noted that the effect is less pronounced when compared to the experiment with only one bad feature. This can be attributed to the fact that the bad features are all the same, while removing only one feature, still leaves three bad features to have an influence on the distance measure.

Another aspect that could be an influence is the fact that TiMBL does not take correspondences between features into account.

The distance deviation can still be used to identify bad features when there are more bad features in an instance. It would be interesting to see the effect of removing more features simultaneously.

The instances of the experiment have twelve features in total. This means that there are 495 combinations of four features. Presenting such an amount of data in a table does not help to gain insight. We therefore present the results in a graph.

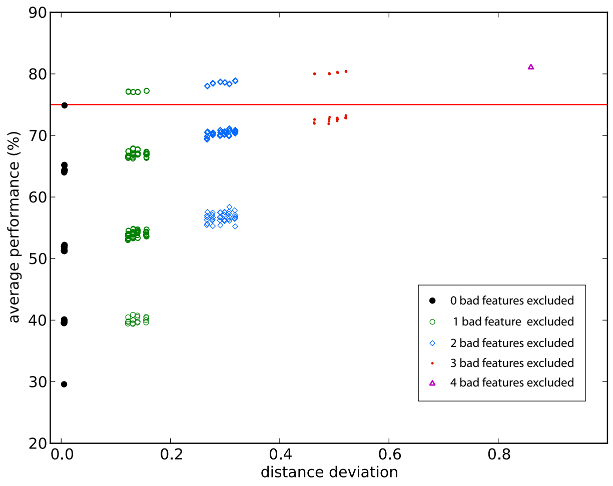


Figure 4.6: Effect of removing four features on performance and distance.

Figure 4.6 shows the result when four features are excluded simultaneously. A data point in Figure 4.6 is the averaged result of nine TiMBL-experiments.

The distance deviation is on the x-axis of Figure 4.6; the average performance is on the y-axis. The horizontal line is the average performance when all the features are used. Data points above this line come from sets that are a combination of features that harm performance when they are included in the instances.[6]

When focusing on the distance deviation alone, it can be seen that there are five discernible groups. Each group has its symbol in Figure 4.6 and the last group (triangle) consists of only a single data point. These groups coincide with the number of features in the set of four removed features that are bad. The first group at the left (circles at distance deviation ∼0.0) are the combinations that do not contain any bad feature. This means that for these experiments all bad features are still included in the instances. The second group (open circles, at distance deviation ∼0.14) are the combinations that contain one bad feature. As can be expected, the average performance of these experiments is higher than the

[6]The performance for these sets is higher because the performance is computed by removing the selected set from the set of all features: only good features remain.

performance of the first group. The third and fourth group are the groups when two and three bad features are removed. The last data point at distance ∼0.86 are the experiments when all four removed features are bad features.

These groups can be further divided into subgroups when the average performance is taken into account. The distance deviation does not vary for these subgroups, but the performance does. The varying performance comes from the different proportions of random and good features that are in the set of removed features. The more good features there are in the set of excluded features, the lower the performance. All subgroups of a given group have the same distance deviation value. This observation supports the previous one that the distance deviation technique can detect bad features, but cannot detect differences between good and random features.

Different number of feature values

In the previous subsection, the effect of the number of features per instance is discussed. In this subsection, the effect of the number of feature values will be discussed.

The experiments are set up as in the previous sections except for the creation of the instances. The instances have six features:

2 random features: one random feature that is a sample from a set of five feature values and one random feature that is a sample from a set of 50 feature values,

2 good features: one good feature with five feature values (one for each class label) and one good feature with 250 feature values (50 for each class label),

2 bad features: one bad feature with five feature values (one for each class label) and one bad feature with 250 feature values (50 for each class label).

The features with five feature values (feature 1, 3, and 5) are exactly the same as the features in the previous experiments. The features with 50 feature values (feature 2, 4, and 6) only differ in the number of feature values that are associated with a given class. For example, there would be only one feature value a_1 referring to class label `A` for feature 1, but there would be 50 feature values $\{a_1, \ldots, a_{50}\}$ for class label `A` in feature 2.

Table 4.9 contains the data from the experiment – excluding one feature at a time. It can be seen that the performance difference for the features with 50

excluded feature	F-score	performance difference	distance deviation	# feature values	description
1	47.41%	1.26%	$0e^{-6}$	5	random feature
2	46.23%	0.08%	17.5%	50	random feature
3	30.55%	-15.60%	$0e^{-6}$	5	good feature
4	44.65%	-1.51%	17.2%	50	good feature
5	63.12%	16.97%	$1e^{-6}$	5	bad feature
6	46.24%	0.08%	20.9%	50	bad feature
none	46.15%	–	–	–	all features

Table 4.9: The effect of removing one feature on performance and distance when features have a varying number of feature values.

feature values behaves as expected, but that the extent is much smaller than the performance for features with only five feature values. The bad features have the highest distance deviations: $1e^{-6}$ for the bad feature with five feature values and 20.9% for the bad feature with 50 feature values.

When taking only into account the features with 50 feature values, the bad feature is the feature with the highest distance deviation. This is consistent with previous observations. For the features with five feature values, the observation also holds, but the values approach the significant digits threshold, which makes the observation less strong.

The conclusion of this experiment is that mingling features with a different number of feature values can make the distance deviation impractical to use. Features with many feature values tend to weigh too heavily on the distance score compared to features with fewer values. As can be seen in Table 4.9, the features with many feature values have an influence on the distance deviations of the features with five values. The distance deviations of the latter are considerably smaller than the comparable ones reported in Table 4.7 on page 118. When features with many feature values are mixed with features with fewer values, the distance deviation for the latter may become too small to use.

Differentiating random and good features

Although the most interesting application for feature selection is an application that identifies features that harm performance, it may be interesting to be able

feature	information gain	gain ratio	description
1	0.02	0.01	random feature
2	0.84	0.12	random feature
3	0.93	0.41	good feature
4	1.47	0.21	good feature
5	0.92	0.40	bad feature
6	1.47	0.21	bad feature

Table 4.10: The information gain (IG) and gain ratio (GR) of the features in Table 4.9.

to separate random from good features.

The relation of feature distributions with the performance is examined in the previous experiments, without considering the relation of a feature with the class labels of the instance. This information about the class label/feature relation is available for the training corpus and can be used for feature selection. A common way of exploring the amount of information that a given feature contributes to the knowledge of the correct class label, is by computing the gain ratio (Daelemans et al., 2010). The gain ratio (GR) (Quinlan, 1993) is a normalized version of the information gain (IG) (Quinlan, 1986) in order to abstract from the number of feature values per feature.

Since there is a training corpus that contains class labels, the gain ratio can be computed for the training corpus. The gain ratio from the training corpus can then be used to infer the *randomness* of a feature.

The information gain and gain ratio of the features for the experiments of Table 4.9 are given in Table 4.10. The random features have a lower gain ratio when compared to features with the same number of feature values. For example, random feature 1 has a gain ratio of 0.01 while the good feature 3 and bad feature 5 have values of 0.4. The gain ratio difference between random and good/bad features becomes smaller when there are more feature values involved.

If the information in Table 4.9 and Table 4.10 is combined, good, bad and random features can be identified. Based on the distance deviation for features with the same number feature values, it can be inferred from Table 4.9 that feature 5 and 6 are bad features for the task. Hence, features 1 through 4 are either random or good features. Based on the gain ratios in Table 4.10, it is possible to say that feature 1 and 2 contain less information about the correct class labels – are more random – than features 3 and 4. Since it is known that feature 3 and 4 are not bad features they should be good features.

The gain ratio captures different information than the distance deviation does. Gain ratio is a measure of how good a feature is linked to its class label. The gain ratio gives information about the *randomness* of a feature. The distance deviation captures how different an instance is when a given feature is in- or excluded. In previous sections, we showed that this difference can be linked to the *badness* of a feature. Both variables are required to separate good, bad, and random features from each other.

Conclusion

The feature with the highest distance deviation from the distance between test and training corpus when the entire instance is used, is a feature that contributes the least to a good performance – or will even harm performance. The conditions that should be satisfied for this conclusion to hold are: (1) that the number of different feature values is the same for every feature, (2) that the quality of a feature is different for different class labels (here expressed by the α parameter), and (3) that the performance decrease induced by a bad feature, is sufficiently large.

The link between distance and performance can be used to get an indication of a bad feature, but discriminating between good and random features is not possible this way. Information gain and gain ratio can be of help when investigating the amount of information a feature contains, thus becoming a tool to distinguish between good and random features.

The conclusion also holds when excluding more than one feature simultaneously, although the contribution of the separate features may get obscured. The conclusion holds to a lesser extent when condition (1) is not met. The influence of a different number of feature values per feature may obscure the information present in the distance deviation.

The conclusions of this section are all reached by controlling the exact nature and the amount of features and feature values involved. The clear boundaries between random, good and bad features can most probably not be drawn when conducting real data experiments because real data will contain features that are a mix of good and bad characteristics. When instances are created for real data, there will be interdependences between the features. These interdependences can be hard to detect and they may alter the correlation between instance distance and performance.

In the next subsection, it is examined whether the techniques from this subsection are still relevant for real data.

4.3.4 Analysis using real data

In the previous subsection, the applicability of distance deviation when establishing the quality of features is investigated by means of synthetic data. In this subsection, the method is put to the test with real data. Real data will not necessarily satisfy the conditions that are responsible for the clear separation of bad features from other features. For real data, in general, the number of feature values will not be the same for each feature. Nor will it be guaranteed that a feature contains a different amount of information about each class label. And most importantly, in real data the extent to which a bad feature will harm performance will not be as large as it is for synthetic data. Because some conditions cannot be satisfied with real data, additional experiments are needed to see to what extent the technique can provide information in less controlled setups.

Experimental setup

For the experiments in this section, the memory based POS experiments on the British National Corpus from Section 3.4 are recreated. The memory-based tagger consists of two kNN-classifiers: one classifier for tokens that are in the training corpus and one classifier for tokens that are not. The first classifier will be referred to as the *known words* classifier and the second will be referred to as the *unknown words* classifier.

The instances for the *known words* classifier contain 14 features:[7] POS-tags of the five tokens at the left of the focus token, the token at the left, the ambitag and token of the focus token, the token at the right, and the ambitags of the five tokens at the right.

[7]MBT pattern: dddddwfWwaaaaa.

The instances of the *unknown words* classifier contain 15 features:[8] POS-tags of the three tokens at the left of the focus token, the token at the left, the token at the right, the ambitags of the three tokens at the right, the begin character of the focus tokens, the three last characters of the focus token, and three binary indicators: an indicator for the presence of a capital, the presence of a hyphen, and whether the focus contains a number or not.

The results of the experiments are obtained through cross-validation experiments of the same type as for the synthetic data excluding one feature at a time. For every domain, all other corpora are merged into one large training corpus.[9] The test corpus and the training corpus are split into three parts and a TiMBL experiment is run with all possible test/training pairs. The performance and distance scores reported in the tables below are based on the averages of those nine TiMBL experiments.

Experiment with real data

In a first setup, the corpora for the *unknown words* and *known words* classifier are used, as explained in the experimental setup. The outcome of these experiments is presented in Table 4.11 and Table 4.12.

The first column of the tables contains the index of the feature that is excluded and for which the other columns report the results. The last column contains a description of the excluded feature.

For the *unknown words* classifier, the average micro F-score when no features are excluded, is 68.20% (std. deviation 1.73%). For each of the 15 features, this experiment is repeated while one feature is excluded from the instances. The performance difference of these TiMBL experiments with the performance using all features, is presented in the second column of Table 4.11. A negative value in column 2 indicates that the given feature is a good feature and removing it from the instances causes performance loss. A positive value is an indication of a bad feature.

For the *known words* classifier, the average micro F-score when no features are excluded is 93.67% (std. deviation 0.74%). Column 2 of Table 4.12 contains the performance deviation from this value when a feature is excluded.

[8]MBT pattern: dddwFwaaachnpsss.

[9]Because the corpora for *known words* are too large, a sample is taken of 200,000 instances per domain.

excluded feature	performance difference	distance deviation	# feature values	gain ratio	description
13	-2.80%	$0.5e^{-3}$%	2	0.25	hyphen
14	-2.05%	$1.6e^{-3}$%	2	0.39	capital
15	-0.29%	$2.0e^{-3}$%	2	0.73	numeric
11	-4.58%	$8.4e^{-3}$%	76	0.16	-2 end character
12	-12.04%	$12.0e^{-3}$%	73	0.25	-1 end character
10	-2.12%	$12.2e^{-3}$%	76	0.12	-3 end character
1	-0.68%	$17.2e^{-3}$%	72	0.14	initial character
4	0.13%	$4.9e^{-3}$%	92	0.02	-3 left context tag
5	0.01%	$6.0e^{-3}$%	92	0.05	-2 left context tag
6	-2.72%	$10.9e^{-3}$%	92	0.13	-1 left context tag
2	-0.46%	0.8%	169	0.10	left context token
3	-0.05%	4.4%	166	0.10	right context token
8	0.12%	15.6%	3967	0.04	+2 right context ambitag
9	0.14%	16.9%	3889	0.03	+3 right context ambitag
7	-0.88%	16.9%	3589	0.12	+1 right context ambitag

Table 4.11: *unknown words* feature selection experiments.

The third column of the tables contains the distance deviations. As has been seen for the experiments with synthetic data, a high distance deviation is an indication of a bad feature, although mixing features with a different number of feature values may interfere with this rule of thumb. The features in the tables are sorted on increasing distance deviation, but are loosely grouped on the basis of the number of feature values. The fourth column gives the number of different feature values for a feature. Finally, the fifth column contains the gain ratio values.

The results in Table 4.11 and Table 4.12 cannot be easily interpreted. At first sight, the conclusion that a higher distance deviation is reserved for bad features (a positive performance difference) cannot be deduced from the tables. The fact that features with a high number of feature values have a more outspoken distance deviation can also be seen in the tables.

Something that can be concluded from comparison of the tables for the real data with Table 4.9 for the synthetic data, is that the extent to which a feature harms the performance is much smaller for the real data than it is for the synthetic data.

excluded feature	performance difference	distance deviation	# feature values	gain ratio	description
4	0.02%	$3.0e^{-3}$%	92	0.01	-5 left context tag
5	0.02%	$3.1e^{-3}$%	92	0.01	-4 left context tag
6	0.02%	$3.3e^{-3}$%	92	0.02	-3 left context tag
7	-0.06%	$3.5e^{-3}$%	92	0.07	-2 left context tag
8	-1.33%	$3.9e^{-3}$%	92	0.23	-1 left context tag
3	-0.05%	1.0%	164	0.21	right context token
1	-0.25%	1.1%	164	0.20	left context token
2	-0.57%	1.2%	163	0.67	focus token
14	0.03%	6.3%	2895	0.03	+5 right context ambitag
13	0.03%	7.0%	2953	0.03	+4 right context ambitag
12	0.03%	7.7%	3014	0.04	+3 right context ambitag
11	-0.01%	8.5%	3062	0.08	+2 right context ambitag
10	-0.56%	9.3%	3136	0.20	+1 right context ambitag
9	-20.82%	9.8%	3277	0.72	focus ambitag

Table 4.12: *known words* feature selection experiments.

Indeed, for Table 4.11 the worst feature yields only a performance difference of 0.14%, which is small compared to 16.97% in Table 4.9. For Table 4.12, the worst feature only harms the performance with 0.03%.

This observation touches on the heart of the problems that occur, when distance deviation is applied on real data in a feature selection setup: the real natural language data used in this experiment, does not contain features that are as harmful as the bad features in a synthetic setup. Most real feature values will not be highly informative for the class label in the training data and highly disinformative in the test data, as is the case for the bad features in synthetic data. The result is that bad features for real data can harm the performance, but not sufficiently so for the distance deviation method to pick it up.

Mixed real/synthetic data experiment

To test whether extremely bad features would behave in the same manner as in the synthetic data, a bad feature – feature 16 – is added to the real data. For these experiments, only a small subset of the BNC corpora is taken. The bad feature is constructed in the same manner as the bad features in the synthetic

excluded feature	performance difference	distance deviation	# feature values	performance difference	distance deviation
13	-2.14%	$0.2e^{-3}$%	2	-1.14%	$0.2e^{-3}$%
14	-2.75%	$0.5e^{-3}$%	2	-2.96%	$0.4e^{-3}$%
15	-1.21%	$13.2e^{-3}$%	2	-2.87%	$12.3e^{-3}$%
16	–	–	65	29.37%	2.5%
10	-0.83%	$18.7e^{-3}$%	73	-0.73%	$17.6e^{-3}$%
12	-10.65%	$19.7e^{-3}$%	70	-4.80%	$18.5e^{-3}$%
11	-3.14%	$29.4e^{-3}$%	71	-1.49%	$27.6e^{-3}$%
1	-0.01%	$39.1e^{-3}$%	69	-0.36%	$36.7e^{-3}$%
4	0.07%	$6.6e^{-3}$%	91	-0.11%	$6.2e^{-3}$%
5	-0.09%	$8.1e^{-3}$%	90	-0.54%	$7.7e^{-3}$%
6	-2.51%	$14.6e^{-3}$%	87	-1.85	$13.8e^{-3}$%
2	-0.53%	0.3%	145	-0.73%	0.3%
3	0.22%	0.6%	144	-0.64%	0.6%
8	0.14%	17.3%	940	-0.19%	16.3%
7	-0.70%	18.3%	865	-0.87%	17.3%
9	0.16%	18.6%	942	0.01%	17.6%

Table 4.13: *Unknown words* feature selection experiments on a small subset of the corpus. With and without a bad synthetic feature 16 included.

data experiments: one feature value for each class label (with $\alpha = 0.05$). The results are in Table 4.13. The first column contains the excluded feature; the second and third column contain the performance difference and distance deviation when only the real features are used. The fourth column contains the number of feature values for a feature. The fifth and the sixth column contain the performance difference and distance deviation when a synthetic feature is included.

As can be seen, feature 16 has a higher distance deviation (2.5%) when compared to features with a comparable number of features ($17.6e^{-3}\% - 36.7e^{-3}\%$) . This observation is consistent with the observation for synthetic data: bad features have a higher distance deviation.

This latter experiment has been repeated with a good and a bad synthetic feature with approximately 830 feature values. The bad synthetic feature behaves also as expected. Excluding the bad feature leads to a performance gain of 1.28% and the distance deviation for the bad feature is 67.1%, which is higher than the distance deviation of comparable features $7 - 9$. Because the bad feature with 830 feature values has a higher number of feature values than feature 16 of Table 4.13, the performance difference is less outspoken. The good synthetic feature also behaves as expected. Excluding the good feature leads to a performance loss of 31.5%, while the distance deviation is 1.9%. A possible conclusion that can be made, based on these extra experiments, is that for the part-of-speech instances, the performance difference linked to a single feature is too small to be picked up by the distance deviation, since including a synthetic bad feature leads to the same observations as for entirely synthetic data.

Feature selection based on distance deviation

Using distance deviation for feature selection might be useful in some specific situations: when one expects that sufficiently harmful features are present. Next, we suggest how the feature selection could be put into practice.

Feature selection could be carried out as follows. First, a random and a bad synthetic feature are introduced for each group of features with a comparable number of features. These synthetic features will function as reference points for the distance deviation. Next, for a set of test/training pairs, the distance between the test and training corpus is computed, using the full feature vectors and excluding each of the features. Based on these distance values, all distance deviations can be computed. When all distance deviations are computed, it is possible to compare the deviations of a real feature with the deviations of the bad and the random feature for each group of features. We have seen that bad features always have a distance deviation that deviates maximally from the distance deviation of random features. If the deviation of a real feature approaches the deviation of the bad feature, the real feature is probably a harmful feature too.

Another application could be the identification of features that deserve more attention during instance construction. If a feature is associated with a high distance deviation value, it may be a good idea to investigate its role during machine learning and maybe change to way the feature is defined.

4.3.5 Conclusion

In this section experiments were designed to test the following idea:

A useful feature influences the correlation between a distance and the performance of a feature-based machine learner in a different way than a superfluous or harmful feature.

It has been found that computing the instance distance by taking the length of the distance vector that consists of a distance per feature, sufficiently preserves the correlation between the distance metric and performance.

Contrary to the idea that has been tested, it is not the useful feature that stands out when the distance is used for feature selection. Instead, it is the harmful feature that becomes traceable. The distance deviation is an indicator that can be used to separate bad features from good features and superfluous features. This conclusion is based on experiments on synthetic data as well as on real data.

The knowledge of this different behavior leads to a feature selection method based on instance distances, but, when real data is used, parameters such as a different number of feature values and dependencies between features have a detrimental effect on the information provided by the distance deviation. For the moment, the influence of these parameters makes it impractical to use the distance deviation as a general indicator good and bad features.

A second disadvantage of distance deviation-based feature selection is that for real data, the performance loss due to a single feature is not sufficiently pronounced to be picked up by the distance deviation.

The experiments in this section confirmed the existence of a relation between the harmfulness of features and a distance metric for memory-based experiments, although further research is needed for finding a generally applicable way of exploring this relation for feature selection.

4.4 Selecting corpora for self-training

Adding more data to a training corpus is found to be beneficial, but adding only a small amount of carefully chosen data can also help a lot. This conclusion is drawn by Chelba & Acero (2006), while presenting work on adapting maximum entropy Markov models. Similar reasoning is the basis of active learning, as it is presented by Lewis & Gale (1994), see Figure 1.2 for the algorithm. The uncertainty of a classifier is used to identify instances that can be interesting for the classifier to have in its training corpus.

Another way of determining the suitability of an instance could be through the use of a distance metric. In this section, self-training experiments are set up to investigate the usefulness of a distance metric in selecting unlabeled corpora that would increase the performance of a classifier, if they were added to the training corpus. Or expressed in a more condensed manner: a distance metric is used to select the best-suited unlabeled data for self-training.

4.4.1 Introduction

Self-training is a semi-supervised learning method, intended to create additional training data for supervised learning in resource-scarce situations. The basic self-training algorithm is explained in Figure 1.3 on page 11.

The experiments in this section will show that adding an unlabeled corpus to the training corpus of a memory-based POS labeler through self-training, will not always have a positive influence on the performance. For this reason, this section is dedicated to investigating whether distances can help self-training by indicating which unlabeled data should or should not be added to the training corpus.

Sagae (2010) argues that self-training is only beneficial in those situations where the training and test data are sufficiently dissimilar, but other factors – such as labeling accuracy of the unlabeled data – may have an influence too. The question is: when are the test and training corpora too similar to draw benefit from self-training and, more fundamentally, when is an unlabeled corpus no longer suited for self-training? It can be expected that the answer to these questions will be task-specific, but nevertheless this section contains an attempt to formulate a general answer.

First, a performance indicator is designed to be able to benefit from the information that may be present in the distance metric, because, later in this section, it will be shown that simply using the distance is not an option for a self-training setup. The performance indicator should be based on unlabeled data and it should indicate whether a given combination of test corpus, training corpus and unlabeled corpus can be expected to benefit from self-training.

In the remainder of this section, the design of the performance indicator is explained. Next, self-training experiments for a memory-based POS-labeler are carried out in order to test the performance indicator.

4.4.2 Designing a performance indicator

The goal of a self-training experiment is to find the corpus that should best be added to the training corpus, meaning that adding the extra data would result in an increase of the accuracy. A performance indicator is needed to find the best corpus to add to the training corpus. As has been shown in Chapter 3 for a straightforward experiment, a distance based on relative token frequencies is already a good performance indicator. For the self-training experiments, a new performance indicator will be proposed. The performance indicator should provide information about the performance of a POS-tagger after the complex process of self-training. As a consequence of the self-training cycle, the performance indicator cannot consist of a single distance metric. Nevertheless, the observation that there is a good relation between distance and accuracy for regular POS labeling experiments, remains the theoretical starting point for the design of the performance indicator.

The relation between distance and performance

In Section 3.4, we saw that there is a good linear relation between the Rényi divergence and the labeling accuracy of (memory-based) POS-taggers. The corpora that will be used in the section are only slightly different from those in Section 3.4, but nevertheless, the relation with the distance metric is recomputed and presented in Figure 4.7. The distance is the Rényi divergence ($\alpha = 0.99$) between a training corpus (*applied sciences*) and the respective test corpora – whose labels are alongside the associated data points in Figure 4.7. As expected, there is a good correlation ($r^2 = 0.890$).

During self-training, data is added to the training corpus, but nothing is changed to the experimental setup of the previous paragraph. The only difference is that the training corpus is extended. The black dots in Figure 4.8 represent the experiments where *applied sciences* is used as the training corpus and *imaginative texts* is used as the test corpus. For each data point, one domain[10] is added to the training corpus before running the POS labeling experiment. The names of

[10]With gold standard labels.

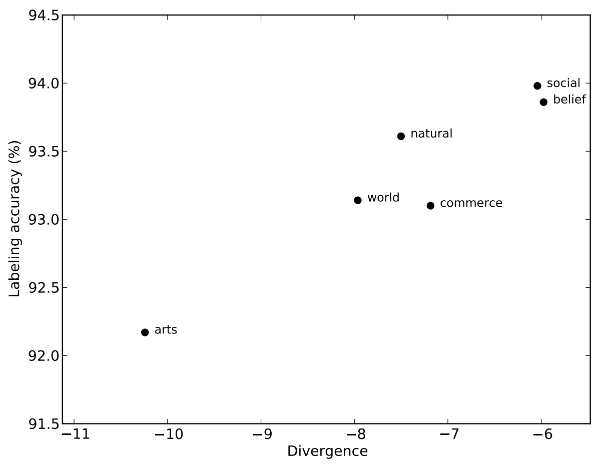


Figure 4.7: Correlation between Rényi (0.99) distance and POS labeling performance. The *applied science* domain is the training corpus for each experiment. The names of the test corpora are given alongside the data points.

which corpora are added to the training corpus, are given alongside each data point. As expected, there is a very good correlation between the divergence metric and the labeling accuracy, $r^2 = 0.967$.

Until now, the data points from Figures 4.7 and 4.8 that are discussed, are only a repetition of the experiments in Chapter 3. The diamonds in Figure 4.8 are obtained in a different way, namely through self-training. For these data points, the corpus that is added to the training corpus is not provided with the gold standard labels, but with the labels as they are assigned in the labeling step, step 2 of the self-training algorithm. Because the distance between training and test corpus is computed without the use of the class labels, all distances remain the same: each diamond is on the same vertical line as its gold standard counter part. As can be seen, adding the newly labeled corpora as opposed to adding the gold standard does harm the performance substantially. The dotted line represents the performance when no extra data is added to the training corpus. Adding newly labeled corpora sometimes even harms performance to such extent that adding data through self-training leads to worse results than plainly using the training corpus to label the test corpus. The r^2 correlation

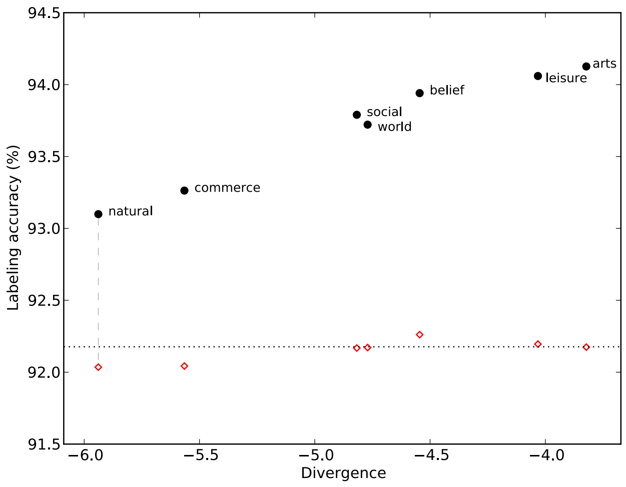


Figure 4.8: Correlation between Rényi (0.99) distance and POS labeling performance. The *applied science* domain is the training corpus for each experiment; the *imaginative texts* domain is the test corpus. Different corpora are added to the training corpus and their names are given alongside the black data points. The black dots are obtained when the added corpus contains the gold standard labels. The diamonds are obtained when the corpora are added through self-training. Each diamond is exactly below its gold standard counter part – as indicated by the vertical dashed line. The dotted line is the accuracy when no additional data is added to the training corpus before using it to label the test corpus.

of the diamonds is 0.637, which is lower than for the black data points. This implies that the correlation between distance and performance gets lost when self-training is involved.

Additional observations

In the previous experiment, we found that the linear correlation between distance and performance gets lost when self-training is involved, but it remains a fact that the labeling step – step 2 of the self-training algorithm – will be better when the unlabeled data is closer to the training corpus. This is a first observation forming a basis for the application of distance metrics in self-training. The accuracy of step 2 will be referred to as the labeling accuracy in the remainder of this section.

A second observation is that the correlation between distance and the accuracy for train/test pairs does not get lost when a particular corpus is added to each of the training corpora.

In a self-training setup, one expects a good correlation between the final accuracy and the distance between the test corpus and the expanded training corpus, when the added data has gold standard labels (the black dots in Figure 4.8). The expanded training corpus is a term denoting the training corpus after the addition of extra data to the original training corpus. It would be interesting if this correlation would still hold when the distance is computed on the test corpus and the unlabeled data alone.

In Figure 4.9, the accuracy is the POS labeling accuracy when the expanded training corpora are used to label the test corpus (*natural & pure science*). The training corpus is *arts* and the labels of the added unlabeled data are indicated in the figure.

The distance is computed twice: once as the divergence between the extended training corpora and the test corpus[11] (●), and once as the divergence between the unlabeled corpora and the test corpus[12] (▶). For the expanded training corpora the r^2 correlation is 0.964, for the unlabeled corpora the correlation is 0.960.

Because the measurement of the distance is the only difference between the bullets and the triangles, the performance is the same for a triangle and its

[11]In the next subsection, this distance will be called d_3.
[12]In the next subsection, this distance will be called d_4.

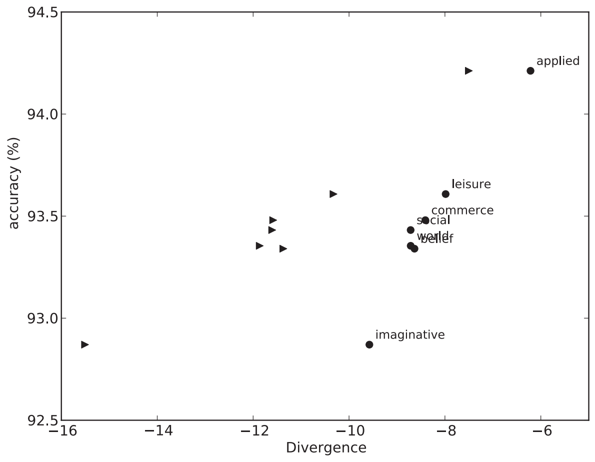


Figure 4.9: Correlation between two distances and (gold) self-training performance. The distance is computed twice: once as the divergence between the extended training corpora and the test corpus (d_4, ●), and once as the divergence between the unlabeled corpora and the test corpus (d_3, ▶). For the expanded training corpora the r^2 correlation is 0.964, for the unlabeled corpora the correlation is 0.960.

bullet counterpart. The result is that the triangles are on the same horizontal line as the bullets. The data points are shifted to the left of the graph when only the unlabeled corpus is used to compute the distance. Or, putting it differently, the distances increase when only the unlabeled corpora are used to compute the distance.

Instead of looking at the data points as being the result of adding extra data to a training corpus, one may also interpret them as being the result of adding a fixed, constant corpus (the original training corpus, *arts*) to a set of experiments with a different training corpus (the gold standard unlabeled corpora). The conclusion that can be drawn from this interpretation is that including a fixed corpus when computing distances, does not influence the degree of correlation between a distance metric and the performance. Indeed, the correlation changes from 0.960 to 0.964.

Therefore, when proportionality[13] is the only matter of interest, computing the distance using only the variable part of a training corpus, is sufficient to find the proportionality between distance and performance. In terms of the self-training experiments that will be conducted later, this implies that the distance between the unlabeled data and the test data can act as a proxy for the distance between the extended training corpus and the test data: $d_3 \approx d_4$. This will make it easier to compute the distances that are used in the performance indicator that will be derived in the next subsection.

The performance indicator

In the previous subsections it has been found that in self-training, the distance between training and unlabeled data is of importance in the labeling step (observation 1) and that the distance between the self-trained model and the test data is important in the final usage of the model. The latter distance is equivalent with the distance between the unlabeled data and the test data (observation 2). A good performance indicator for unlabeled corpus selection would incorporate all these distances[14] that are involved during self-training. The distances are conceptually represented in Figure 4.10:

[13]Proportionality: accurracy $\propto \frac{1}{\text{distance}}$.

[14]The distances are expected to be strictly positive and a higher value means that the corpora are more dissimilar.

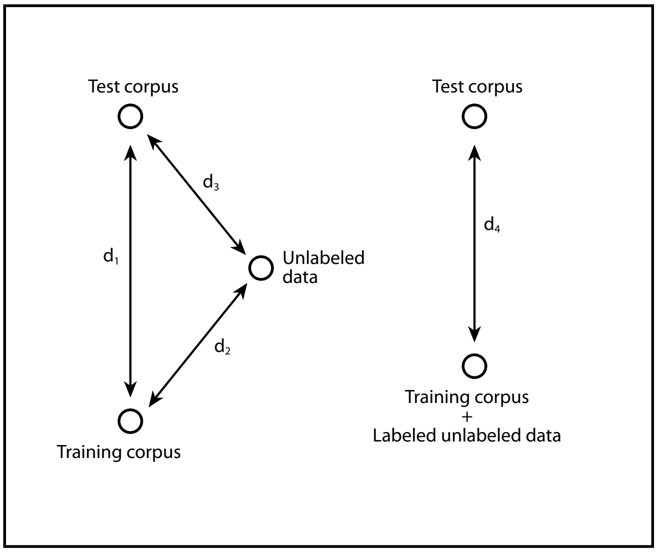


Figure 4.10: The various distances when self-training.

Test/training distance d_1: This is the distance between the training corpus and the test corpus. This distance is inversely proportional to the accuracy of the default experiment involving only training and test corpus.
Unlabeled/training distance d_2: This is the distance between the training corpus and the unlabeled data to be used during the labeling step of self-training. This distance is inversely proportional to the accuracy of the labeling step.
Test/unlabeled distance d_3: This is the distance between the unlabeled corpus and the test corpus. Following the observation in the previous subsection, *viz.* $d_3 \propto d_4$, this distance can act as a proxy for the distance between the extended training corpus and the test corpus (d_4). This means that the distance d_3 is inversely proportional to the accuracy of the situation when the labeling step is 100% accurate.
Test/unlabeled+training distance d_4: This is the distance between the extended training corpus and the test corpus.

The idea of how to combine all distances into one performance indicator, is illustrated with Figure 4.11. The black column at the left of the Figure (acc_4)

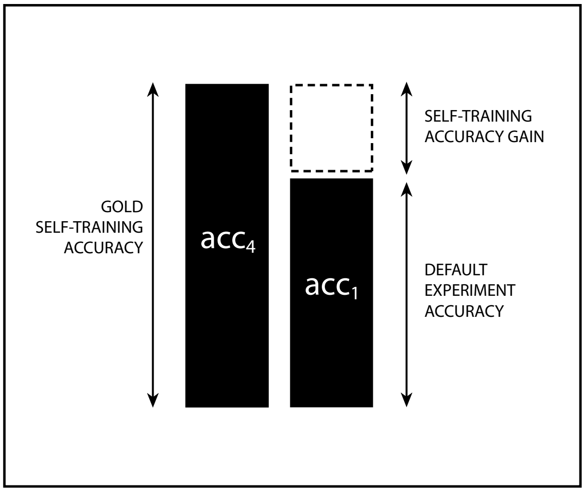


Figure 4.11: Comparing accuracies for self-training.

indicates the accuracy when the correctly labeled unlabeled data are added to the training data. This situation is the ideal self-training situation, if we presuppose that adding the correctly labeled data to the training data will maximally increase the testing accuracy. The column at the right of Figure 4.11 reveals the internal components of self-learning. The total accuracy is the sum of the accuracy of the experiment without any self-training (default experiment, acc_1) and the accuracy gain that comes from self-training. If the labeling step would be perfect, this sum adds up to the same accuracy as the column at the left.

If the accuracy of the default experiment would increase, the self-training accuracy gain would be smaller, since we presupposed that the sum of acc_1 and the gain equals the maximally attainable accuracy acc_4. This behavior can be expressed as:

$$\frac{\text{gold self-training accuracy } (acc_4)}{\text{default experiment accuracy } (acc_1)} \propto \text{self-training gain} \tag{4.7}$$

and gain can be expected from self-training as long as:

$$\frac{\text{gold self-training accuracy } (acc_4)}{\text{default experiment accuracy } (acc_1)} > 1 \tag{4.8}$$

This condition can be expressed by the factor:

$$\frac{\left|\frac{acc_4}{acc_1} - 1\right|}{\frac{acc_4}{acc_1} - 1} \tag{4.9}$$

which will be +1 if there is gain to be expected and -1 otherwise.

acc_4 is inversely proportional to the distance d_4, but only because the correctly labeled unlabeled data is used to compute acc_4. acc_1 is inversely proportional to distance d_1. This means that the proportion of the accuracies can be rewritten to:

$$\frac{d_1}{d_4} \propto \text{self-training gain} \tag{4.10}$$

A linear relation has the format $y = ax + b$ and for the distance–performance relation this becomes:

$$acc_1 = k_1 d_1 + c_1 \text{ with } k_1 < 0 \tag{4.11}$$

$$acc_4 = k_4 d_4 + c_4 \text{ with } k_4 < 0 \tag{4.12}$$

Equation 4.8 can then be rewritten to a formula that expresses when self-training gain can be expected:

$$\frac{d_1}{d_4} > K \text{ with } K = \frac{k_4}{k_1} + \frac{c_4 - c_1}{k_1 d_4} \tag{4.13}$$

From the formula for K, it can be seen that the actual switching point between self-training gain and self-training loss depends on the distance between the expanded training corpus and the test corpus (d_4), and on the distance between the training corpus and the test corpus (d_1).

When the performance indicator is being used in a real self-training experiment, the performance is the actual object of investigation and consequently it is not available to compute the slopes k_1 and k_4. It is possible to estimate a general value for K by introducing some approximations.

Both k's reflect the proportionality between distance and accuracy for the same type of experiment, therefore it can be expected that $k_1 \approx k_4$. The second term in the definition of K depends on the accuracies – at zero distance between test and training corpus – when using the training corpus together with the unlabeled data (c_4) and without the unlabeled data (c_1). In both cases, there is no difference between test and training corpus and it can be expected that the accuracies will be similar, $c_1 \approx c_4$. These approximations lead to the reduction: $K \approx 1$.

Distances d_1 and d_4 may be incomparable because of corpus size effects. The distance d_1 is measured between the training corpus and the test corpus in contrast to d_4, which is measured between the expanded training corpus and the test corpus. Based on Figure 4.9, it has been found that $d_4 \approx d_3$ with d_3 the distance between the unlabeled data and the test corpus. It is possible to choose the amount of unlabeled data to be equal to the amount of training data, leading to a d_3 that is comparable to d_1. This comparable d_3 can be used as a proxy for d_4. Rewriting the self-training gain now becomes:

$$\frac{d_1}{d_3} > 1 \tag{4.14}$$

This condition can be expressed by the factor:

$$\frac{\left|\frac{d_1}{d_3} - 1\right|}{\frac{d_1}{d_3} - 1} \tag{4.15}$$

This factor will be +1 if there is gain to be expected from self-training and -1 when no gain is expected. This factor is deduced for a situation where the unlabeled data is perfectly labeled. This cannot be expected in a realistic self-training experiment. The labeling accuracy is inversely proportional to the distance between the training corpus and the unlabeled corpus (d_2). We assume that a higher labeling accuracy leads to a higher self-training gain:

$$\frac{1}{d_2} \propto \text{self-training gain} \tag{4.16}$$

The same proportionality holds for d_1 and d_3 and the harmonic mean of all distances can be included in the performance indicator to express the dependency of self-training accuracy on the various distances.

The final performance indicator now becomes:

$$\text{performance indicator} = \frac{\left|\frac{d_1}{d_3} - 1\right|}{\frac{d_1}{d_3} - 1}\left(\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3}\right) \tag{4.17}$$

The absolute value of this performance indicator is expected to be proportional to the self-training accuracy and the sign of the performance indicator signals whether self-training gain can be expected or not. In the next subsection, the usage of the performance indicator is tested for self-training experiments with a memory-based POS-labeler.

4.4.3 Experiments

Evaluating the performance indicator

The nine part-of-speech corpora from British National Corpus are used for the memory-based POS-tagger experiments in this section. To abstract from size effects, samples are taken from the corpora. The size of the samples approximates 1,500,000 tokens per domain, keeping sentences intact.

For the self-training experiments, one domain is taken as the labeled training corpus and another domain is taken as the test corpus. Because there are nine domains, this leaves seven domains to be used as unlabeled data. Each of those seven domains is used in a separate self-training experiment. With this setup, 72 self-training experiments can be run, in which each experiment is designed to find the best corpus out of the seven unlabeled corpora. The result is 504 data points that should be split into seven groups: the group of experiments with a positive self-training outcome and the group with negative outcomes.

For the experiments, all distances are computed using the Rényi divergence (α=0.99).

In the previous subsection, a performance indicator for the suitability of unlabeled data for self-training was developed. This performance indicator is now empirically validated with 504 self-training experiments. The data points for all experiments can be seen in Figure 4.12, in which the performance indicator

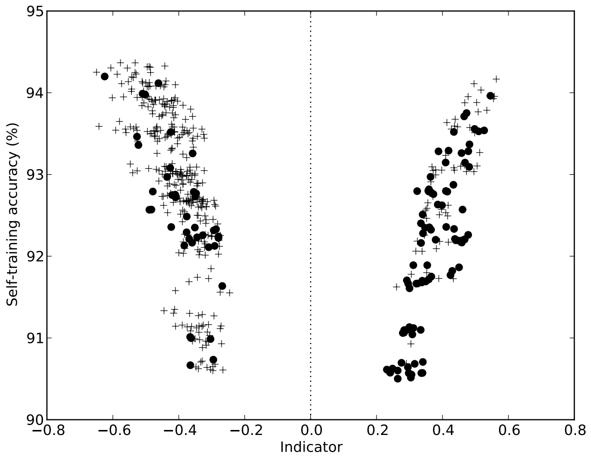


Figure 4.12: Correlation between performance indicator and self-training performance.

is plotted against the final self-training accuracy. The experiments for which self-training is beneficial compared to the default test/training experiment, are indicated with bullets (•), the experiments where self-training is harmful are indicated with crosses (+).

As can be seen, positive self-training experiments have a positive performance indicator value. This means that the performance indicator can be computed using the distances when a choice of unlabeled data is available. If the performance indicator is positive, there is a good chance that adding the unlabeled data through self-training will increase performance.

It can also be seen that the self-training performance is correlated with the absolute value of the performance indicator. A larger absolute value of the performance indicator signals a higher self-training performance. Note that a higher self-training performance does not represent better self-training, since the default performance can already be high. For the right side of Figure 4.12 the r^2 correlation is 0.689, for the left side 0.498.

The right correlation is reasonable; the left correlation is not so good. The difference in r^2 may be linked to the fact that the performance indicator has

	negative indicator	positive indicator	total
self-training loss	234	142	376
self-training gain	18	110	128
total	252	252	504

Table 4.14: Contingency table based on Figure 4.12.

Selection metric	precision	recall	F1-score	r^2
Overlap	23.20	45.31	30.53	0.187
Jensen-Shannon	43.25	85.16	57.37	0.840
sUWR	43.25	85.16	57.37	0.925
Kullback-Leibler	43.65	85.94	57.89	0.988
Rényi (0.99)	43.65	85.94	57.89	0.990
Reference	25.40	100	40.51	–

Table 4.15: Evaluation scores for different metrics. The Rényi scores (0.99) are based on contingency Table 4.14. The distance–performance correlation r^2 comes from Table 3.4.

been designed, starting from a situation where self-training helps. Since the main goal of the performance indicator is to separate suitable from unsuitable unlabeled data, no effort has been made to mediate this low correlation.

Table 4.14 is the contingency table for Figure 4.12. The precision for the positive self-training experiments can be computed from this contingency table. The precision for this experiment is given in Table 4.15 together with the recall and the F1-score. This precision of self-training using Rényi as the metric is 43.65%. When every unlabeled corpus is expected to contribute to a better performance, the reference precision would be 25.40%. The result is that the chance of adding good unlabeled data is 1.7 times higher, when the performance indicator is used.

The recall for the positive self-training experiments is 85.94%, leading to an F1-score of 57.89%. Recall would be 100% if all unlabeled data were expected to contribute to a better performance, leading to a baseline F-score of 40.51%.

It is possible to apply a χ^2 test for independence on the contingency Table 4.14. The null hypothesis is that the sign of the performance indicator is independent of the outcome of the self-training experiment. Based on the contingency table, the χ^2 test statistic is 88.64. This is associated with a p-value of $5e^{-21}$ (1 degree of freedom), which is lower than 0.05: the null hypothesis is not accepted. This

Selection metric	precision	recall	F1-score	r^2
Rényi (0.10)	36.51	71.88	48.42	0.312
Rényi (0.20)	40.08	78.91	53.16	0.537
Rényi (0.30)	42.46	83.59	56.32	0.672
Rényi (0.40)	42.46	83.59	56.32	0.758
Rényi (0.60)	42.46	83.59	56.32	0.864
Rényi (0.50)	43.25	85.16	57.37	0.818
Rényi (0.70)	43.25	85.16	57.37	0.903
Rényi (0.80)	43.25	85.16	57.37	0.939
Rényi (0.99)	43.65	85.94	57.89	0.990
Rényi (0.90)	44.05	86.72	58.42	0.974
Rényi (0.95)	44.05	86.72	58.42	0.987
Optimized Rényi (0.99)	58.78	67.97	63.04	0.990
Optimized Rényi (0.90)	61.61	65.62	63.40	0.974
Optimized Rényi (0.95)	63.85	64.84	64.34	0.987

Table 4.16: Evaluation scores for different values of the α parameter in the Rényi metric. The distance–performance correlation r^2 comes from Table 3.4.

indicates that the sign of the performance indicator contains information about the outcome of the self-training experiments.

Using other distance metrics

It is possible to substitute the Rényi divergence in the performance indicator with any other distance metric. Table 4.15 contains the evaluation scores when overlap, Jensen-Shannon, sUWR and Kullback-Leibler are used to calculate the performance indicator. The fifth column contains the correlation of the distance with the performance as it has been found in Chapter 3.

As can be seen, there is a correlation between the distance–performance correlation of a metric and the F1-score. As a result, the correlation can be used to select the distance metric that is best included in the performance indicator.

To get an idea of the correlation, the experiment is repeated with Rényi divergences with α values that vary between 0.1 and 0.99. The reason to use only variants of the Rényi metric is that the quantification of the distance will gradually change between the different version without changing the entire nature of the metric. Rényi divergences with slightly different α's will compute the distance in a more similar way than e.g. the sUWR does. This is done to ensure that no other influence than the varying distance–performance correlation can be expected to be at play in the figures of Table 4.16.

Table 4.16 contains the evaluation metrics for the self-training experiment together with the distance–performance correlation. It can be seen that the F1-score increases together with the r^2 score – the correlation between the two variables being 0.904.

Optimizing K

During the deduction of the performance indicator, some approximations were introduced to estimate the value of parameter K. Since in the experimental setup, the self-training accuracies are known, it is possible to optimize the value of K.

$$K = \frac{k_4}{k_1} + \frac{c_4 - c_1}{k_1 d_4} \tag{4.18}$$

As can be seen from the definition, K depends on the task at hand and no conclusions about the value of K in other machine learning experiments can be linked to the outcome of the optimization, as it is carried out here.

For the optimization, the calculation of the performance indicator, based on the Rényi (0.99) divergence of Figure 4.12, is recalculated varying the value of K from 0 to 3 with steps of size 0.05. The F1-score is plotted against the value of K in Figure 4.13.

The best K value is 1.15, which leads to a positive self-training precision of 58.78%, a recall of 67.97%, and an F1-score of 63.04%. Repeating the optimization for the top 2 divergences from Table 4.16, shows that the best score can be obtained with an α value of 0.95. The F1-score of this setup is 63.34 as can be seen at the bottom of Table 4.16. For both $\alpha = 0.90$ and $\alpha = 0.95$, the best K value is 1.20. In a practical situation, K cannot be estimated, but if future research would lead to a way to better estimate the value of K these results could become obtainable.

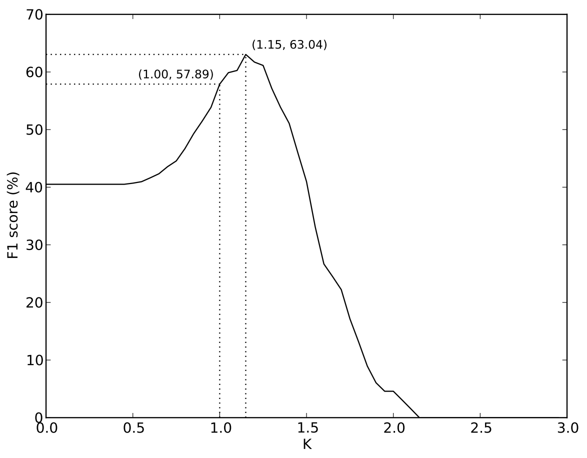


Figure 4.13: Evolution of F1-score with a varying parameter K.

4.4.4 Conclusion

In this section, we showed that the correlation between a distance metric and the accuracy can be exploited to find the best unlabeled corpus to be used in a self-training setup. For this reason, a performance indicator is designed and it is applied to memory-based POS labeling experiments. We found that using the performance indicator increases the chance of beneficial self-training 1.7 times. With ideal parameters, this factor increases to 2.5 times. It is also found that metrics that are better correlated with the performance of the POS-tagger in a simple test-training setup, are better suited to be used in the performance indicator.

A slightly different application of the performance indicator is to select unlabeled corpora in resource-scarce situations. Consider a human annotator that wants to annotate additional data for a domain for which no extra resources are available because there is no digital or not even written data available. The performance indicator could be used to select corpora from other domains that are most likely to increase performance when they are annotated and added to the small existing corpus.

4.5 Conclusions

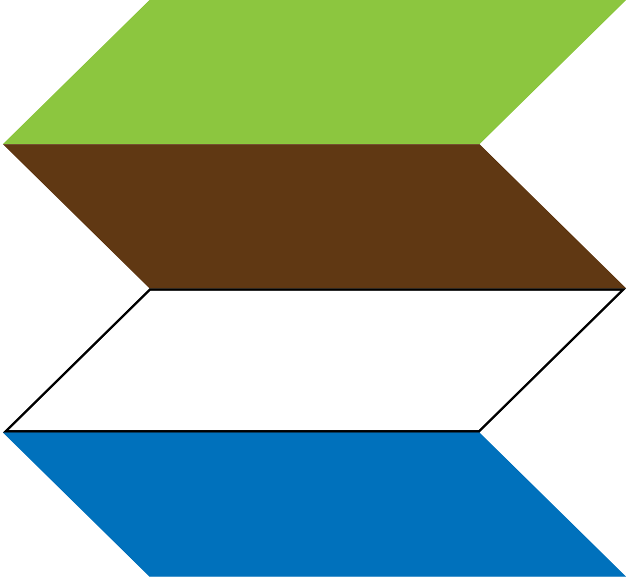
In this chapter, three applications that can benefit from the use of a distance metric were carried out: training data selection (Section 4.2), feature selection (Section 4.3), and corpus selection (Section 4.4).

We found that the Rényi metric produces the subset with the highest accuracy during the training data selection experiment, but examination of the learning curves reveals that the Jensen-Shannon divergence is most interesting when it comes down to efficiently selecting sentences from a large corpus. In this work, this difference is attributed to the fact that a metric has two aspects: aspect *a*, quantifying the (dis)similarity of the corpora, and aspect *b*, linking distance to performance. The preference of one metric over the other depends on which aspect is more important for a given task. For the training data selection experiment, it seems to be that aspect *a* is more important, although the influence of aspect *b* is noticeable.

In the section about feature selection, we found that harmful features influence the correlation between distance and performance in a different way than useful or superfluous features. Superfluous features are features with little information content. This result shows that not only do distance metrics contain information when token frequency distributions are used, but also when a different mapping from corpus to distribution is used. In these experiments a feature-based instance distance was used. A disadvantage of using the distance deviation for an experiment with real data is that few features are truly harmful for real data.

In the corpus selection experiments, we found that the usage of the metric depends more on the relation with the performance (aspect *b*) than it does for the training data selection experiments. We showed that a performance indicator that is based on a well-correlated metric, helps to detect combinations of a test corpus, a training corpus, and an unlabeled corpus that will result in a performance gain, when they are combined in a self-training setup.

We can conclude that distance metrics can be successfully used in various applications and, to a certain extent, it is possible to select a metric that is best for a given task on the basis of the degree of correlation between the distance values, produced by that metric and the performance of a given machine learner.



Chapter 5

Conclusions

A summary of the experimental results and the insights acquired from a close inspection of the experiments is given in this final chapter. In addition to the summary, some more speculative ideas are addressed; how can the domain notion be understood? The final section of this chapter contains ideas for further research and is dedicated to the limitations of this dissertation.

The introductory Chapter 1 contained a definition of some basic concepts, a general overview of related literature and the research questions that are addressed. Chapter 2 was an introduction to various distance metrics and how these metrics can be applied to natural language corpora. Chapter 3 was dedicated to the correlation that exists between relative token frequency-based distance metrics and the performance of NLP tools. In Chapter 4, the usage of the distance metrics in three different applications was investigated. Because one of the aims of this dissertation is to make the experiments reproducible, an overview of the scripts is given in Appendix B and the source code is made available online.

5.1 Summary

Distance metrics have been used in domain adaptation techniques in various ways. A basic assumption that gives this research a theoretical grounding, is that there is a correlation between the distance between two corpora and the performance of a natural language processing tool for which the domain adaptation technique has been designed. Although this assumption is intuitive and practice shows that the assumption is valid, the actual choice of the right distance metric is often not based on insight into the actual behavior of the metric. The main motivation of our research was to gain a better understanding of the weaknesses and strengths of distance metrics in a given setup.

The first chapters were dedicated to the behavior of the metrics in different settings. A first influential factor is (a)symmetry. Metrics like the Kullback-Leibler divergence and the unseen word ratio (sUWR) are asymmetric in the sense that $d(x, y)$ does not necessarily equal $d(y, x)$. This behavior follows from the definition of the metrics and is commonly known. In the introductory Chapter 2, detailed attention was given to asymmetry and its significance for the usage of the metric in machine learning experiments.

Metric properties that cannot be straightforwardly derived from the definition, were investigated more closely in Chapter 3. In a first experiment, the British National Corpus (BNC) was used to investigate the correlation between various distance metrics and the performance of a part-of-speech tagger for symmetric and asymmetric usage of the metrics. We found that the distance – between a test corpus and a training corpus from different domains – is linearly correlated with the performance of a memory-based POS-tagger using those corpora. Depending on the metric, this correlation is better or worse. The fundamental research question of this dissertation was whether the distance–performance relation is monotonic. A linear relation is a monotonic relation. Monotonicity is important in the applications: It ensures that the distance metric can be used to rank experimental setups.

Entropy based metrics (like Rényi divergence and Kullback-Leibler divergence) exhibit the best linear correlation in the asymmetric and symmetric setup, but, in the symmetric setup, the differences between the bulk of the selected metrics decreases.

After finding the monotonic relation between distance and performance, the traceability of the relation was investigated. The template for these part-of-speech tagging experiments was the same as for the previous experiments, but the focus switched: going from the influence of the algorithm and the influence of the corpus to the influence of linearly combining different metrics.

Running the experiments with different algorithms revealed that using a baseline algorithm, an SVM-based POS-tagger or a memory-based POS-tagger does not have a major influence on the linearity of the correlation. This indicates that the observed performance drop during out-of-domain POS-tagging experiments is directly linked to the training data and the influence of the actual processing of the data is of minor importance. It should be noted that none of the three selected algorithms uses external information like, for example, additional wordlists or semantic information that is not present in the training corpus. Using a more complex setup may influence the easily traceable linear correlation between distance and performance. In our research, we found that the more elaborate processing cycle belonging to self-training, is already sufficiently different from the simple test/training cycle to distort the linear correlation.

Apart from the experiment with the BNC corpus, the single metrics were also used for the experiments with the OntoNotes corpus and the presence of the linear relation was confirmed. The linear correlation appears to be independent from the corpus. Also, the POS experiments were repeated using a combination of single metrics. The metric combinations perform better than the single metrics for POS-tagging experiments on the BNC, but when the same combinations were tested on the OntoNotes corpus they were no longer performing best. This may be an indication that combining metrics introduces overfitting.

In-domain POS-tagging experiments were the last of the series of POS-tagging experiments in Chapter 3. The distance between training and test corpus is smaller for in-domain experiments than for out-of-domain experiments. We found that the linear correlation is lost for in-domain experiments and that the loss of the linear correlation could not be attributed to insensitivity of the metrics. This observation could be a starting point for the objective assessment of the homogeneity of a set of corpora.

Prepositional phrase (PP) attachment experiments have been carried out in addition to POS-tagging experiments. Switching from the POS-tagging experiment to the prepositional phrase attachment experiment, shows that the latter is less sensitive to domain shifts. Our point of view is that the insensitivity comes from the minor importance of the lexical features when carrying out out-of-domain PP attachment experiments. The basis for this view comes from the observation that excluding lexical features does not harm the out-of-domain performance or may even increase it. Previous research has shown that for in-domain experiments, the lexical features do contribute to a better attachment accuracy. This may be an indication that lexical features lead to overfitting to the domain on which the PP attachment system is trained. This is not a major problem, since the PP attachment system does not rely heavily on lexical features, but for systems that do need lexical information, domain adaptation may yield the best results when focusing on the lexical features.

In the final part of Chapter 3, a shortcoming of a distance based on relative token frequencies alone, was investigated. For a machine learner, the joint probabilities $P(X, Y)$ are important for the obtainable performance. The joint probability can be resolved into two factors $P(X, Y) = P(X)P(Y|X)$. The relative token frequency distance only takes into account the first factor $P(X)$, because we wanted to use the distance in an unsupervised manner, but experiments showed that the second factor $P(Y|X)$ cannot be disregarded if one wants to accurately predict the performance of a POS-tagging system.

In the first section of Chapter 4, the importance of the two aspects of a distance metric became clear. These aspects are:

Aspect *a*: quantifying the (dis)similarity of the corpora, Aspect *b*: linking distance to performance.

The training data selection experiments showed that the Jensen-Shannon divergence quantifies the distance between test and training corpus in a more efficient manner, although the metric with the better distance–performance correlation (Rényi divergence) produces the best performing training data subset. This can be attributed to a lesser importance of aspect *b*, when compared to the importance of aspect *a* for this type of usage of the metrics.

For the self-training experiments, the importance is reversed. For these experiments, a performance indicator was designed, based on the assumption that there is a linear correlation between distance and performance for POS-tagging experiments. It can be expected that basing the performance indicator on this assumption will make aspect *b* more important. The results confirmed this expectation.

The feature selection experiments of Chapter 4 drew upon the idea that it is possible to obtain a distance based on other distributions except relative token frequencies. We found that an instance distance based on the feature value distributions of an instance, can be used to retrieve a linear relation between this instance distance and the performance of a memory-based classifier. The research question that was investigated is whether a useful feature influences the correlation between a distance and the performance of a feature-based machine learner in a different way than a superfluous or harmful feature. We found that it is possible to identify harmful features based on the deviation of the distance computed with and without the harmful feature. This observation came from synthetic data. For real data, the identification of harmful features is hampered by the fact that the different features tend to be associated with a different number of feature values and by the fact that, for real data, the negative influence of a single feature on the performance is limited.

Finally, we can formulate cautious answers to the research questions presented at the beginning of this dissertation.

1.4.1 : Is there a monotonic relation between performance and distance? We found that there is a monotonic correlation between performance of POS-taggers and relative token frequency distances between the training and test corpus. The nature of the correlation depends on the task. This monotonic correlation is manifest for some metrics, but many metrics hint at a similar correlation. Depending on the context in which the metric is used (symmetric or not), the degree of correlation of many metrics can be increased.

1.4.2 : Is the correlation easily traceable? For POS labeling, the correlation is shown to be linear, independent of the corpus, and independent of the labeling algorithm. The correlation is found not to be easily traceable for an NLP tool, *viz.* a PP attachment tool, that does not extensively use lexical information, while the distance is being calculated using relative token frequencies.

1.4.3 : Can the relation be exploited? The correlation between distance and performance can be exploited in domain adaptation setups. An important remark is that there are two aspects to a metric: (a) quantifying the (dis)similarity of the corpora, and (b) linking distance to performance. When choosing a metric for an application, it depends on the exact usage of the metric which aspect will be predominant. The degree of correlation between distance and performance (r^2) can be used to select a suitable metric only for those applications for which aspect b is more important.

5.2 Subjectivity of the domain notion

Several chapters of this dissertation have been assigned to research that tries to answer questions concerning the relation between the test/training corpus distance and the performance of an NLP tool. There is also a question that is not explicitly investigated in those chapters because answering it has less practical value and may be subjective. The question is whether the notion of

	test 1	*test 2*	*test 3*
training 1	$d_{1,1}$	$d_{1,2}$	$d_{1,3}$
training 2	$d_{2,1}$	$d_{2,2}$	$d_{2,3}$
training 3	$d_{3,1}$	$d_{3,2}$	$d_{3,3}$

Table 5.1: Distances d between all domains in out-of-domain experiments. *training i* and *test i* are the same domain, only the role in the experiment differs. E.g. the distance between *domain 1* as the training corpus and *domain 2* as the test corpus is $d_{1,2}$. All combinations are given because distances can be asymmetric.

domain is merely an expansion of the notion of test and train set difference or whether the notion contains additional information about corpora.

The question is not whether domain labels do contain information. The fact that they do contain information, can be easily derived from the experiments. Very few machine learning experiments will obtain 100% accuracy because of the inconsistencies between the test and training sets. These inconsistencies can be, for example, unseen words, unseen structures, or the ambiguous usage of words. When annotators are assigning domain labels to corpora, they will derive the labels from the semantic content or the source of the corpora. Grouping texts into domains is a method to minimize the inconsistencies between a test and a training corpus that belong to the same domain. The fact that domain labeling provides information, can be seen when the out-of-domain experiments in this dissertation are compared to the in-domain experiments. Removing out-of-domain texts from a domain effectively leads to a better in-domain performance. This means that domains do capture information, but the labels are subjective and discontinuous – there are no fixed gradations defined for domains.

From our point of view, it would be better to supplement the domain labels with a table containing the distances between the domains. Table 5.1 is an example of such a table for three domains. E.g. the distance between *domain 1* as the training corpus and *domain 2* as the test corpus is $d_{1,2}$. When the distance information is given, domain labels are no longer considered equivalent to each other: one pair of domain labels may cover corpora that are actually much more similar than the corpora covered by a second pair of domain labels. A continuous gradation is introduced. In addition, providing the distance information is an objective way of differentiating between corpora. Information about the distance may be interesting for domain adaptation experiments: it may be easier/harder to carry out domain adaptation for corpora that are further apart. This behavior would go unnoticed when only domain labels are used.

It should be noted that the distances of Table 5.1 are not necessarily obtained using distance metrics. They may as well be the performances of the NLP tool under investigation, since the performance can be a good measure for the difference between corpora. This would mean that distance $d_{1,1}$ is the in-domain performance.

When domain labels are supplemented with distance tables, the notion of *domain* becomes a subjective expansion of the objective concept *distance*. Using domain labels can be considered to be a concise way of saying that the dissimilarity between the test and training corpus is large and may be so large that even humans can see that they are different – for example, because the semantic content of the corpora is different. If a researcher would adhere to this view on domains, the notion would lose its connotation of domain as a clearly distinctive subdivision of a language. It would also mean that any technique devised to carry out domain adaptation, could also be useful for in-domain usage.

Finding an objective way to define domains would consist of finding a threshold on the distance scale. Corpora with a mutual distance below this threshold would belong to the same domain, corpora with a larger distance would belong to different domains. Such a threshold may even not exist, but if the threshold exists, it will not be trivial to find its exact value.

5.3 Further research

This dissertation is an attempt to obtain a better understanding of distance metrics in natural language processing. There are many applications of distance metrics conceivable and further research is needed.

When designing a natural language processing tool, it is not necessary to restrict the training data to a single source. Information is increasingly becoming available and there are no restrictions on the incorporation of this additional information into the NLP tool. Using such a complex setup may influence the relation between the core of the training data and the performance of the tool. The influence of the use of external information is yet to be investigated and may lead to interesting results. For example, how can the additional information be incorporated in the distance metric and how will this affect the linearity of the correlation?

Reconsidering the two aspects of a metric, it is clear that the degree of correlation between distance and performance only contains information about the second aspect – *viz.* linking distance to performance. Would it be possible to find a predictor, apart from the actual application of the metric, that provides information about the first aspect – *viz.* quantifying the (dis)similarity of the corpora? Or would it be possible to find a method to assess the relative importance of the two aspects for a machine learning task? These are fundamental and interesting questions that are not addressed in this dissertation.

A final indication for further research may deal with the unsupervised feature selection. We found that distance-based feature selection is achievable in a controlled situation. Obstacles arise when the feature selection method is applied to real data and distance-based feature selection becomes impractical. Further research is needed to clarify the various parameters that cause the feature selection method to become ineffective. It is only by getting a clearer view of these parameters that a realistic distance-based feature selection method may become available.

Appendix A

Background information

A.1 Triangle inequality and its application

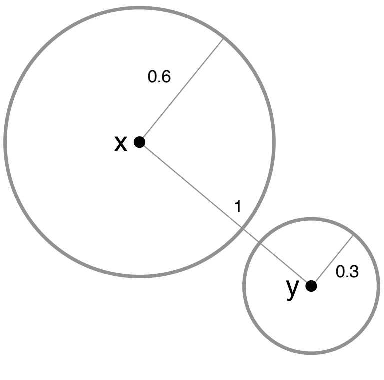


Figure A.1: Two corner points, x and y, of a triangle and the arcs for an undefinable third point, z.

Figure A.1 shows two corner points, x and y, of a triangle (x, y, z). Each point represents a text corpus. A divergence, d, gave the following distances:

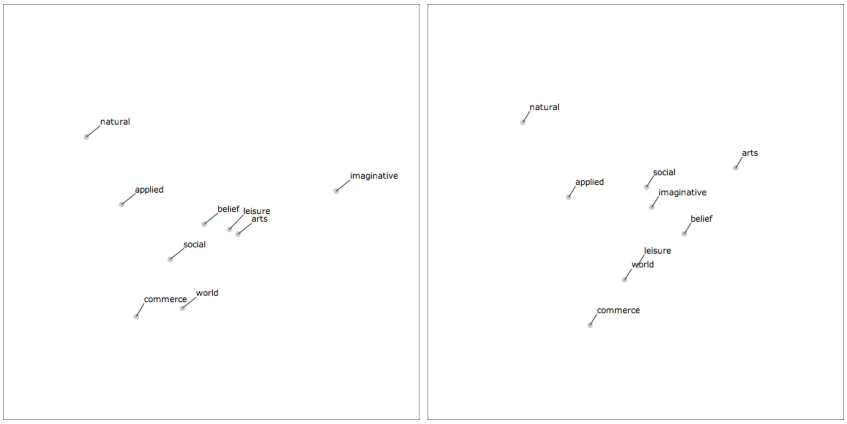
$$\begin{aligned} d(x,y) &= 1 \\ d(x,z) &= 0.6 \\ d(y,z) &= 0.3 \end{aligned}$$

Points x and y are drawn in Figure A.1. Point z should be on a circle with radius 0.6 centered at point x, but it should also be on a circle with radius 0.3 centered at point y. As can be seen in the figure, the circles do not intersect, leaving the position of point z undefined. This is due to the fact that the triangle inequality does not hold for every pair of points, *viz.* $d(x, z) + d(y, z) < d(x, y)$. If the triangle inequality would hold, the circles would intersect and the third point z could have been found.

A true mathematical metric (see Section 2.3) can be used to build up a multidimensional space. This means that a set of points for which all pairwise distances are given, can be plotted, although dimensionality reduction may be needed. In the collection of metrics in Table 3.3, the best metric, *i.e.* a metric in the mathematical sense, is the Bhattacharyya distance (r^2 0.425). Disappointingly, the relation between the metric score and the accuracy is not that strong. This means inferring POS labeling accuracy from a plot of the domains from the British National Corpus, is not reliable. Nevertheless, the plot can be made and can be seen at the left of Figure A.2.

The first step in the construction of the plot in Figure A.2 is the computation of the coordinates for every domain. Coope (2000) presents an algorithm that can compute these coordinates based on a set of distances. The basis is the triangle inequality in the sense that after establishing coordinates for two points, the third point can always be found in the manner as explained earlier (after adding a dimension to the space). For three points, a fourth point can be found and so on. For n domains, a space is build with at maximum n dimensions.

A multidimensional space with $n > 3$ cannot be plotted easily. That is why a second step is introduced. The multidimensional space is projected onto two dimensions using principle component analysis (PCA). The pitfall of using a PCA-driven projection in order to be able to plot a multidimensional space, is that through the loss of essential information two data points may get positioned very close to one another even though they are far apart.



(a) Bhattacharyya distance **(b)** sUWR

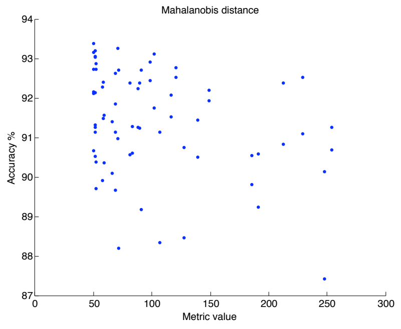
Figure A.2: A mapping of the Bhattacharyya metric and the sUWR in a 2-dimensional plane.

Even when keeping the weak relation with the accuracy and the projection issue in mind, it is possible to consider Figure A.2 as an interesting insight in the space of domains, but no claims can be made based on the visual representation alone. The algorithm that is used to construct the Bhattacharyya plot can be applied to distance values from metrics that are not truly mathematical metrics and the algorithm may finish without errors. The sUWR (r^2 0.677) was the best metric for which this is the case and the plot is included at the right of Figure A.2. As can be seen, both plots look similar (in both cases there is some relation with the accuracy), but they are sufficiently different to illustrate that sUWR may perform better than the Bhattacharyya distance.

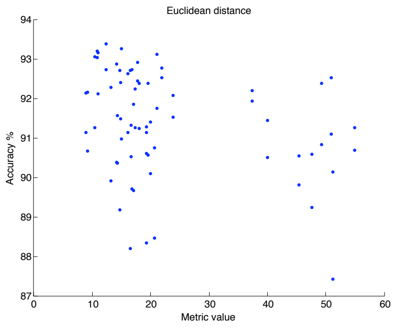
A.2 Computing the Mahalanobis distance

As defined earlier, the Mahalanobis distance is given by the formula:

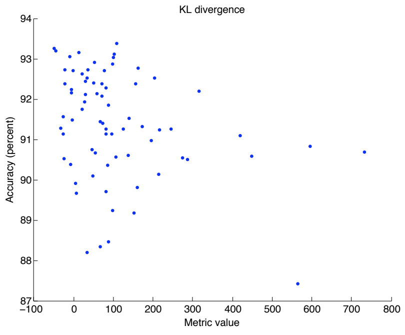
$$Mahalanobis(\mathbf{x}; \mathbf{y}; M) = \sqrt{(\mathbf{x} - \mathbf{y})^T M (\mathbf{x} - \mathbf{y})} \tag{A.1}$$



(a) Mahalanobis distance



(b) Euclidean distance



(c) Kullback-Leibler divergence

Figure A.3: POS-tagger accuracy vs. metric value for samples of 500 random tokens.

in which the matrix M is a parameter that has to be chosen or learned. In this section, the ITML[1] algorithm (Davis et al., 2007) is used to learn the matrix M, when computing the Mahalanobis distance between domains. This algorithm aims to minimize the difference between the Mahalanobis and the Euclidean distance, while stating that the distance between data points from the same class must be smaller than a given threshold u and the distance between data points from different classes must be greater than a threshold l. It is already clear that the Mahalanobis distance is not well suited to compare domains, since there is only one data point for every domain – *viz.* a vector containing

[1]http://www.cs.utexas.edu/∼pjain/itml [Last visited: January 2012].

the relative token frequencies – meaning that the threshold u cannot be applied. Nevertheless, it is possible to run the ITML algorithm to compute the matrix M using token frequency vectors.

The corpora for the BNC domains, see Section 3.3, are used to construct one token frequency vector $\mathbf{x}$ for each of the nine domains. Each vector gets a different class label – the domain. The matrix M is computed and is used to establish the Mahalanobis distance between the domains. Because of computational limitations, only 500 tokens are taken into account when constructing the vectors. This is done twice: once taking the 500 most frequent tokens and once taking 500 random tokens. This working method is far from ideal, since finding the distance between domains, requires the entire domain. After obtaining the distance between the domains, the data points can be plotted against the accuracy of the memory-based tagger resulting in plots similar to the ones in Figure 3.4 on page 59. The plot for the Mahalanobis distance is plotted alongside the plot for the Euclidean distance and the Kullback-Leibler divergence in Figure A.3. The x-axis represents the value of the distance metric and the y-axis represents the POS-tagger accuracy. At first glance, it is obvious that the correlation is weaker than in Figure 3.4. This can be attributed to the constraint of taking only 500 tokens into account. The exact correlation coefficients are 0.483 for the Mahalanobis distance, 0.296 for the Euclidean distance, and 0.482 for the KL-divergence. These unsigned figures can also be found in Table A.1 along with the figures for the experiments with the 500 most frequent tokens and with all tokens.[2]

The Mahalanobis distance could not be computed for all tokens due to computational constraints.

corpus	Euclidean	Mahalanobis	Kullback-Leibler
500 random tokens	0.296	0.483	0.482
500 most frequent tokens	0.475	0.514	0.097
All tokens	0.477	–	0.804

Table A.1: Pearson correlation coefficients for three metrics and two distributions.

From this simple comparison, we conclude that, apart from computational constraints, the Mahalanobis distance can be used to compute the distance between

[2]There is a small difference with the r value in Table 3.3, because, in this section, the corpora were not pruned to contain an equal amount of sentences, neither are the distances computed with a cross-validation technique.

domains, and most likely it will show a better correlation with POS-tagging accuracy than the Euclidean distance. The second conclusion is that the Kullback-Leibler divergence is the metric of choice when looking at the correlation of the distance metric with the accuracy because of its practicality and better results when applied on the entire corpus.

A.3 Confidence interval for a predicted observation

The result of a linear regression analysis is an equation of the form:

$$f(x) = \beta_0 + \beta_1 x \tag{A.2}$$

This formula can be used to predict the response for a new point, x_{new}. This response is then called y_{new}. Because β_0 and β_1 are estimations, there is a degree of uncertainty associated with the newly obtained y_{new}. This uncertainty can be expressed through the determination of a confidence interval, [*lower*, *upper*], that will contain y_{new} with a confidence of $1 - \alpha$:

$$y_{new} \in [lower,\ upper] \text{ with } 1 - \alpha \text{ certainty} \tag{A.3}$$

The prediction interval is defined by the formula

$$\begin{aligned} &[y_{new} - t_{1-\frac{\alpha}{2}} SE_{y_{new}},\ y_{new} + t_{1-\frac{\alpha}{2}} SE_{y_{new}}] \\ &\text{with} \\ &\quad n: \text{the number of observations used to construct} \\ &\qquad \text{the regression equation} \\ &\quad t: \text{the student } t \text{ distribution with } n-2 \text{ degrees of} \\ &\qquad \text{freedom} \end{aligned} \tag{A.4}$$

$SE_{y_{new}}$ can be regarded as a proxy for the classical standard deviation in common confidence intervals. It can be computed as the following:

$$SE^2_{y_{new}} = MSE + MSE\left(\frac{1}{n} + \frac{(x_{new} - \bar{x})^2}{\sum_i (x_i - \bar{x})^2}\right) \tag{A.5}$$

$$MSE = \sum_i \frac{\big(y_i - f(x_i)\big)^2}{n-2} \tag{A.6}$$

$$\bar{x} = \frac{1}{n}\sum_i x_i \tag{A.7}$$

For example, the plots in Figure 3.4 are based on 72 combinations of domains. If we want to obtain the confidence interval for the prediction of one data point using all other data points, we have 71 observations. This means that the value of $t_{1-\frac{\alpha}{2}}$ with 69 degrees of freedom ($71-2 = 69$) for a confidence of 95% ($\alpha = 0.5$) becomes 1.9949. The only thing missing to compute the confidence interval is the $SE_{y_{new}}$, which should be computed using the values for the individual data points.

Note that the confidence interval is computed for the new observation x_{new}. The computation differs from computing the confidence interval for the mean response at x_{new}.

A.4 Divergence metrics on artificial data

In this section, artificial distributions are used to gain insight on the subtle variety in the extensive collection of divergences. The advantage of using self-created distributions is that they can be kept very small, such that interpreting the difference between two distributions can be done by only looking at the frequencies themselves. The possibility to pair evaluation by human interpretation and distance values produced by the metrics, provides a technique to get a deeper understanding of the nature of the metrics. Like for all empirical experiments, the conclusions of this section depend on the hypotheses that the researcher had in mind when setting up the experiment. Those hypotheses are decisive in selecting the distributions in Table A.2. While creating the distributions, efforts have been made to cover a number of case studies, but nevertheless the collection is tentative.

The basic idea for the experiments in this section is that a reference distribution, R, is selected. The reference distribution – which would be the distribution based on the training corpus in a practical application – consists of three tokens, see Table A.2. The first row in Table A.2 gives the exact figures for the frequencies as they have been chosen. Token 1 has a relative frequency of 9.1%, token 2 has a share of 11.1% in the distribution and the most prevailing token has a frequency of 79.8%. Needless to say that the sum of the frequencies totals 100%. One token of the reference distribution prevails clearly over the other two. Setting up the distribution this way has as a result that the distribution becomes an extremely simplified version of a Zipfian distribution – which is the expected distribution for the tokens of a textual corpus.

In the experiments, the distance between the reference distribution and a distribution from a collection of dissimilar distributions is computed. In Table A.2 the frequencies of these distributions are presented as well as an interpretation of the property that accounts for the dissimilarity to the reference distribution.

id	token1	token2	token3	description
R	0.091	0.111	0.798	Reference distribution (training set)
1	0.495	0.505	–	Absence of most frequent token
2	0.141	–	0.859	Absence of average frequent token
3	–	0.141	0.859	Absence of least frequent token
4	0.081	0.121	0.798	Small difference for less frequent tokens
5	0.101	0.101	0.798	Less frequent tokens are equally frequent
6	0.111	0.091	0.798	Switched frequencies for less frequent tokens
7	0.798	0.111	0.091	Reverse frequencies
8	0.091	0.121	0.788	Small decrease for most frequent token
9	0.091	0.101	0.808	Small increase for most frequent token
10	0.273	0.333	0.394	Same frequency order, but smaller differences
11	0.333	0.333	0.333	Equally frequent tokens

Table A.2: Artificial distributions.

Figure A.4 presents the distributions from Table A.2 visually. The x-axis is the three tokens and the y-axis contains the associated frequencies. Although the lines from the distributions that barely diverge from the reference, appear superimposed, it can be deduced from the figure that some distributions greatly differ from the reference. Most notably, the lines for the distribution with the absence of the most frequent (id 1), with reverse frequencies (id 7), with smaller differences (id 10), and with equally frequent tokens (id 11) do not resemble the line for the reference distribution.

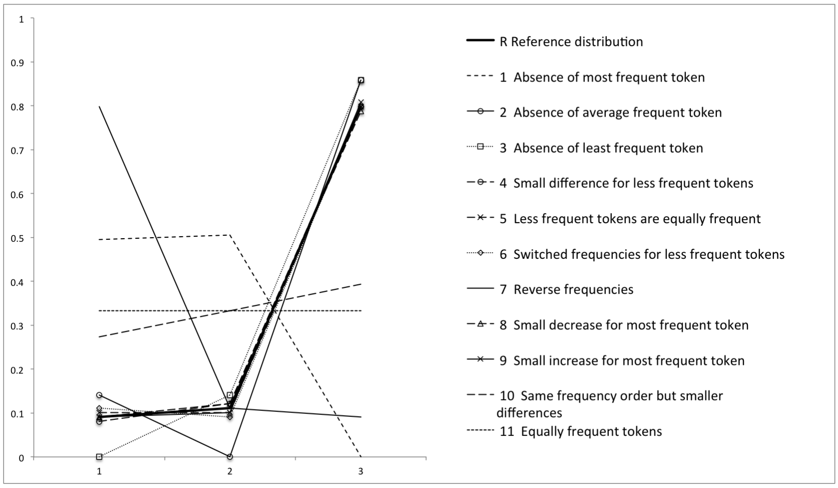


Figure A.4: A visual representation of the distributions in Table A.2.

A.4.1 Kullback-Leibler and Rényi divergence

Figure A.5 shows a comparison of the distances as computed with the Kullback-Leibler and the Rényi divergence (with varying α)). The various distributions are put on the x-axis; the y-axis shows the relative distance value for a given metric. The relative distance value is 1 for the distribution that is furthest apart from the reference, according to a given metric and 0 for the distribution that is closest. The distance values for the other distributions are linearly scaled to fall in this interval. The reason for the rescaling is that it permits to compare different metrics.

A first observation that can be made from Figure A.5 is that the distance values per distribution show a smooth transition with varying α's and that the Kullback-Leibler divergence blends in nicely. This can be expected, since the Kullback-Leibler divergence is the same as the Rényi divergence with $\lim_{\alpha\to1}$.

Since Kullback-Leibler and Rényi are closely related, it is interesting to know which properties the α parameter emphasizes and which ones are de-emphasized. Inspecting the figure with the relative distance values, we see that there is a downward trend with increasing α for the distributions with an absence of a token (id 1, id 1, and id 3). This means that an increase of α lowers the sensitivity of

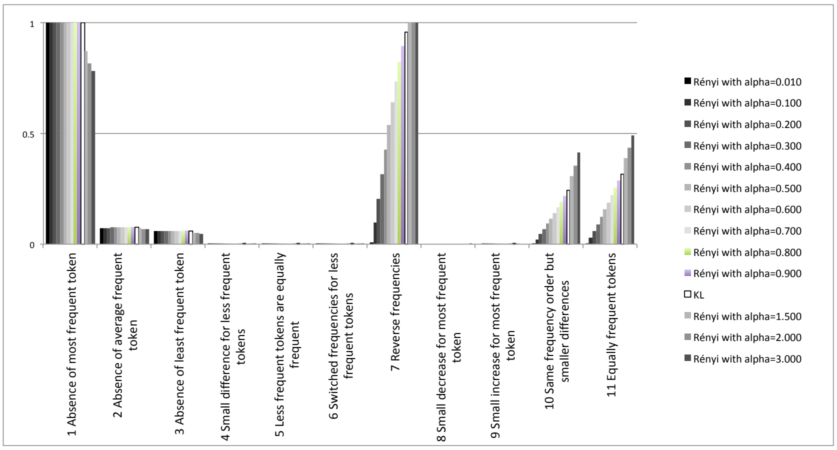


Figure A.5: Relative values of Kullback-Leibler and the Rényi divergence.

the metric with respect to missing tokens – a missing token enlarges the distance less quickly. This effect is amplified when surpassing the value $\alpha = 1$.

Decreasing the sensitivity to missing values is coupled with an increase of the sensitivity to the relative magnitudes of the token frequencies. This can be seen when looking at the distributions with reverse frequencies (id 7), with smaller frequency differences between the tokens (id 10), and with equally frequent tokens (id 11). The trend of increasing sensitivity to relative magnitude is also present for the corpora for which no bars are shown in Figure A.5, but the relative distance values are too small to be visible. The common property of all these distributions is that all tokens are present, but that the frequencies differ more or less from the reference distributions. An increasing α stresses the frequency differences.

Finally, a note has to be made on this increasing sensitivity. The evolution of the outcome of the Rényi divergence is more complicated than discussed here. A hint of the complexity can be captured by plotting the formula for the Rényi divergence while using α as the independent variable and with e.g. $p_k = q_k = 0.5$ while restricting both distribution to one value, $|P| = |Q| = 1$. In our opinion though, the most important values of α are included in our analysis.

In the next section, we will only report on the Kullback-Leibler divergence, since the Rényi divergence shows the same behavior, as has been shown in this section.

A.4.2 Difference based metrics, proportion based metrics and others

In this section, some token frequency based metrics are compared using the same setup as in the previous section. The simple unknown word rate (sUWR) can also be regarded as being based on token frequencies, but this metric is too crude for the small sizes of the artificial distributions. The sUWR has the maximum value when a token is missing and has 0 otherwise.

The metrics that we do look at, are the KL-divergence as the exemplar of divergences that contain the proportion of the token frequencies in their formula, the Euclidean distance as the exemplar of divergences that contain the difference of the token frequencies in their formula and the cosine distance, which is a more complicated metric. Figure A.6 is setup in the same way as Figure A.5.

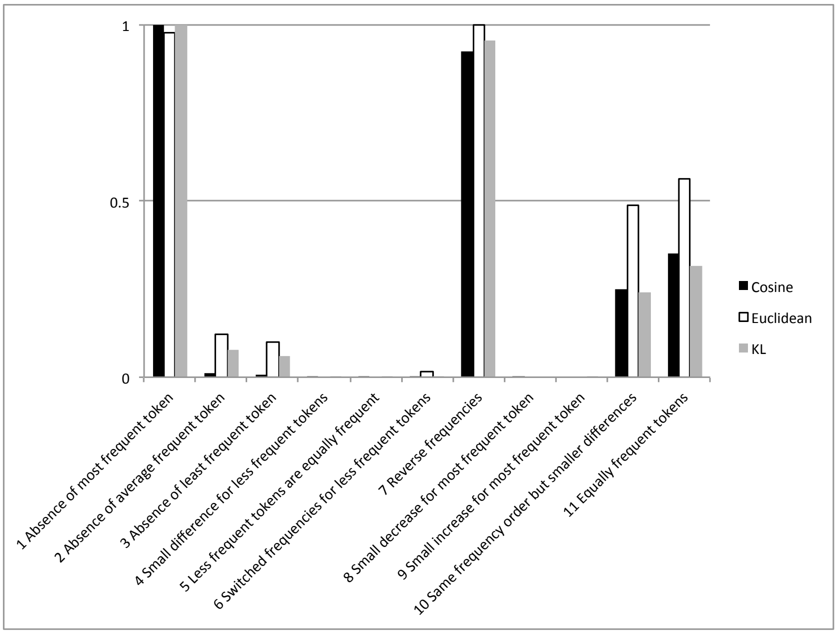


Figure A.6: Relative values of Kullback-Leibler, Cosine and Euclidean divergence.

A first observation that can be made from Figure A.6 is that small changes (id 4, 5, 6, 8 and id 9), even if they occur with the most frequent token, are not captured by any metric. This can be regarded as a suitable property, because corpora – *i.e.* the distributions – are mere samples of a domain and thus subject to normal deviations typical of taking samples. Overall the distance measures follow the visual difference between the corpora in FigureA.4 – the less its frequency line in the figure resembles the reference line, the higher the distance value.

A second observation is that the cosine divergence is almost equally sensitive to small changes (id 4, 5, 6, 8 and id 9) compared to the absence of less frequent tokens (id 2 and id 3). The KL-divergence and the Euclidean distance are also insensitive to small changes, but they do pick up the absence of a token. This means that for the cosine divergence, change is not different from complete absence. From a domain distance measuring perspective, there may be an important difference between these two situations: a token frequency change from 0.798 to 0.788 may be unimportant to distinguish domains, but the absence of a token with a frequency of 0.111 may indicate that this is a low frequent domain specific token and it may be good to capture this in the final distance value.

In the beginning of this section, the sUWR was mentioned. The absence of tokens is exactly the type of behavior that the sUWR measures, so when the cosine divergence is used to measure the distance between distributions, it may be combined with the sUWR to make up for the cosine divergence's insensitivity to missing tokens. And in fact, this is what comes up from the experiments in McClosky (2010). In his experiments, he finds that combining the cosine divergence with, among other metrics, the sUWR has the most outspoken predictive power for parser accuracy.

The next observation that can be made from Figure A.6, is that the Euclidean distance is more sensitive to changes than the cosine and KL-divergence. The different sensitivity of the Euclidean and KL-divergence can be read from the curves in Figure A.7. In this figure, the evolution of the Euclidean divergence and the KL-divergence with increasing deviation of the measured distribution from the reference distribution from Table A.2 on page 168, is visualized. On the x-axis the decrease of the token frequency of token 3 is given – which is also double the increase of the token frequency of token 1, and double the increase of the token frequency of token 2. The choice of decreasing the token frequency of token 3 and distributing the loss evenly among the other two tokens, is arbitrarily made, but the frequencies are tied to each other in the sense that they always must sum up to 1.

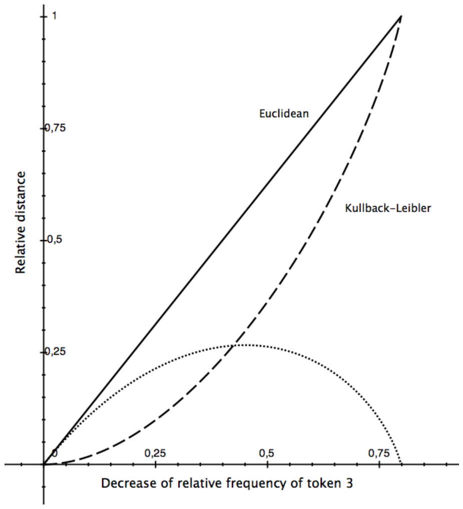


Figure A.7: Evolution of divergences with increasing token frequency deviation. The dotted line is the difference between the two divergences.

On the y-axis the relative distance value can be read. For the maximal distance this is 1. The decrease is limited to 0.798, because this is the token frequency of token 3, so a greater decrease is impossible. The dotted line is the difference between the Euclidean and KL-divergence. It can be seen that the Euclidean divergence is more sensitive to changes: it indicates a greater relative distance between a distribution and the reference distribution at any decrease. The difference is the highest when token 3 has lost approximately half of its frequency count. This effect can also be retrieved from Figure A.6. For the distributions where token 3 has lost approximately half of its frequency count (id 10 and id 11), Euclidean distance is discernibly greater that the KL-divergence. For the other distributions, the difference with the KL-divergence is smaller. The curves in Figure A.7 can be drawn for any pair of divergences and they will all disclose the different sensitivities of the divergences. It is hard to state a theoretical preference of one divergence over the other, based on these curves. Is it better to have a metric that is a bit more sensitive in the beginning or is the opposite better? This type of questions can only get an answer in a practical application.

In this section, we showed that most divergences are dissimilar in the way they weigh frequency differences, but often they show the same trends. Sometimes it is possible to state a theoretic preference based on the differences, but mostly a preference will not emerge until using the divergences in a practical application. In Chapter 4, we explored the possibility to use the Pearson's correlation coefficient to select the best-suited metric for a given task.

A.4.3 Kendall's τ on token frequencies

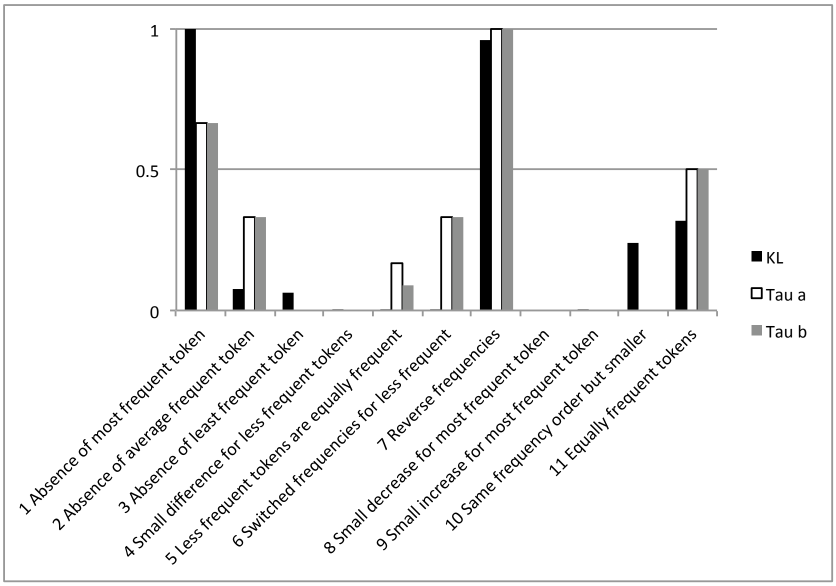


Figure A.8: Relative values of Kullback-Leibler and Kendall's τ_α and τ_β.

A pair of metrics not included in the previous section, are Kendall's τ_α and τ_β. The nature of this metrics is such that it measures ordering rather than the difference between token frequencies. Because of this property, the τ's catch characteristics of the distributions that seem undesirable when measuring the distance between domains, as can be seen in Figure A.8. The figure is build up in the same way as Figure A.5, but the distance value is rescaled such that the original τ values closer to -1 are mapped to the greatest relative distances (*viz.* 1) and values closer to 1 are mapped to the smallest relative values (*viz.* relative distance 0). It can be seen that the τ's are sensitive to situations when there are only minor changes to the reference distribution – more specific, when the distributions of the lesser frequent tokens are equal (id 5) or switched (id 6). Another less desirable property is that τ fails to capture changes in the token frequencies, if the ordering remains the same (id 10). The fact that τ does not capture the absence of the least frequent token (id 3) is not harmful, since for larger corpora not capturing the loss of only the least frequent token will not have such an influence on the measured distance, as it has with the artificial distributions. Nevertheless, it is implausible that small changes in lesser frequent tokens are essential when trying to distinguish between domains so we disregard the usage of the τ's.

k	p_k	q_k	r_k	s_k	t_k
token 1	0.01	0.02	0.40	0.00	0.37
token 2	0.09	0.08	0.34	0.43	0.37
token 3	0.10	0.10	0.16	0.27	0.16
token 4	0.30	0.25	0.07	0.17	0.07
token 5	0.50	0.55	0.03	0.13	0.03
$n = 5$	1.00	1.00	1.00	1.00	1.00

Table A.3: Artificial distributions with varying token frequency ordering.

To gain more insight in the different behavior of τ_α and τ_β, we introduce a new set of artificial distributions. Consider the collection of corpora P, Q, R, S and T with token frequencies in Table A.3. When knowing that τ varies between -1 and 1, we can make the following observations:

- For corpus P and corpus Q, there are no two tokens in a corpus that have the same token frequency, so there are no ties. The order of the tokens, when sorted according to their frequency, is the same for P and Q, so $\tau_\alpha = \tau_\beta = 1$.

- There are no ties in corpus R. The order of the tokens in R is the reverse of the one in P (and in Q), so $\tau_\alpha = \tau_\beta = -1$.

- There are no ties in corpus S. The most frequent token in R, token 1, is not in S. The order of the other tokens is similar in both corpora, this gives $\tau_\alpha = \tau_\beta = 0.2$. If token 2 would be missing in S, τ would be 0.4. Missing token 3 increases τ to 0.6 – meaning that a less frequent value has a lower impact on the final distance value, τ, since without any missing values tau would be 1 when measuring the distance between R and S.

- There is one tie in corpus T, $t_{token1} = t_{token2}$. This gives $\tau_\alpha = 0.9$ and $\tau_\beta = 0.95$ when compared with R. τ_β is less sensitive to ties, which is a

	Label A	Other label
Predicted label A	true positive	false positive
Predicted another label	false negative	true negative

Table A.4: Performance table for instances labeled with a class label A .

desirable property, because, when comparing corpora, the fact that two tokens occur equally frequent does not seem to be an important fact.

The list of observations shows some good properties of τ, but overall it is hard to argue why τ would be a better metric than the other metrics presented in Chapter 2. Furthermore, calculating Kendall's τ for large corpora is a computationally challenging task, which makes it an unfavorable metric.

A.5 Evaluation measures

Over the years, various evaluation measures for machine learning experiments have been devised. Many of them are common knowledge among computational linguists, but nevertheless it is necessary to specify them in order to avoid confusion. Also in this section, the Pearson correlation coefficient is introduced.

A.5.1 Performance scores

Performance scores are indispensable when it comes to developing natural language processing tools. Although the use of specific performance scores for a task may be under discussion, the fact that a performance score is required, is undisputed. Making use of performance scores will not only supply a more objective basis to evaluate NLP tools, it will also facilitate the development of NLP tools by making comparison available. Recall, precision, F-score and accuracy are all well-known performance scores that are employed throughout the research here presented.

van Rijsbergen (1975) specifies recall (R) and precision (P) for information retrieval purposes. The idea is that a query is carried out and that one wants to know what proportion of answers is correct (precision) and what proportion of potentially good answers was found (recall). For the class labeling experiments that were conducted in this dissertation, the query may be ‘What are the instances with class label A?’. The labeling system should then indicate which are those requested instances. This query can be repeated for every class label defined in the ongoing task. Looking at Table A.4, one can easily deduce how to compute the precision and recall. The columns give the two mutually exclusive options for an instance: either it carries the class label A or it carries another (undefined) label. The rows give the labels as they are assigned by the labeling system. The cells of the table are filled with instance counts; represented by names for the sake of generality. The precision is the proportion of answers that is correct. The complete answer of the labeling system consists of all instances that are predicted to carry class label A, *i.e.* the sum of the true and false positives. The instances that are assigned another label, are not an answer to the question which instances carry class label A. The proportion of answers that is correct, is then given by:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{A.8}$$

The number of potentially good answers is given by all instances that carry the class label A. The proportion of potentially good answers that is indeed found is given by:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{A.9}$$

Because both recall and precision are interesting measures a parameterized combination has been proposed by van Rijsbergen (1975), E, the effectiveness. The effectiveness is commonly used in a slightly rewritten form: the F-score or F-measure :

$$\text{F-score}(\beta) = \frac{(1+\beta^2)PR}{\beta^2 P + R} \tag{A.10}$$

The F-score is the parameterized version of the harmonic mean of precision and recall.

As van Rijsbergen (1975) explains, when $\beta \to 0$, the influence of the recall becomes less pronounced in the final value. When $\beta \to \infty$, precision becomes less important. This may not be clear when looking at the formula for F, but the original definition of effectiveness E in van Rijsbergen (1975) may be of help.

$$\text{Effectiveness}(\alpha) = 1 - \frac{1}{\frac{\alpha}{P} + \frac{1-\alpha}{R}} \tag{A.11}$$

with $\alpha = \frac{1}{\beta^2+1}$ and $E = 1 - F$.

Throughout this work, we may use the letter P to refer to precision, R to refer to recall and $F1$, or simply F, to refer to F-score with $\beta = 1$.

For classification tasks with multiple class labels, the F-scores can be averaged. It is possible to obtain a micro-averaged F-score and a macro-averaged F-score:

$$\begin{aligned} \textit{macro F-score} &= \frac{1}{n}\sum_{c\in\mathcal{C}} F(c) && \text{(A.12)} \\ \textit{micro F-score} &= \sum_{c\in\mathcal{C}} P(c)F(c) && \text{(A.13)} \end{aligned}$$

$$\begin{aligned} &\text{with} \\ F(c) &= \text{the F-score for class label c} \\ \mathcal{C} &= \text{the collection of class labels} \\ n &= |\mathcal{C}|\text{, the number of different class labels} \\ P(c) &= \frac{\text{the number of instances with class label c}}{\text{the total number of instances}} \end{aligned}$$

Note that when there is a test and training division of the corpus, $\mathcal{C}$, n, and $P(c)$ can be taken from the test or from the training corpus. In this work, they will be taken from the training corpus.

The accuracy is a general measure, like averaged F-score, and is defined as:

$$\textit{accuracy} = \frac{\text{tp}}{N} \tag{A.14}$$

$$\begin{aligned} &\text{with} \\ \text{tp} &= \text{true positive for any class label} \\ N &= \text{the total number of instances} \end{aligned}$$

A.5.2 Correlation coefficient

Various methods to quantify the linear correlation between two variables have been developed, making it possible to express the degree of correlation with a single figure, *viz.* a correlation coefficient. For experiments that focus on finding a correlation between two variables – e.g. between corpus distance and machine learner performance – this correlation coefficient can be used to identify the distance measure, is most informative about the machine learner performance.

A well-known correlation coefficient is presented in the work of Pearson (1896). This coefficient of correlation is defined as

$$r = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{n\sigma_x\sigma_y} \tag{A.15}$$

with

$$\begin{aligned} x, y &= \text{paired observations} \\ n &= \text{the number of observations} \\ \bar{x}, \bar{y} &= \text{the average value of } x \text{ and } y \\ \sigma_x^2 &= \frac{1}{n}\sum_i^n (x_i - \bar{x})^2 \text{ and } \sigma_y^2 = \frac{1}{n}\sum_i^n (y_i - \bar{y})^2 \end{aligned}$$

r is also called the Pearson product-moment correlation coefficient. The coefficient ranges between -1 and 1. A value of 0 indicates that there is no correlation between property x and property y of the subjects. A value of 1 indicates a perfect positive linear relation, and -1 means that there is a perfect negative correlation. There are other correlation coefficients available – like Spearman's ρ – but because we want to test for linearity rather than for a monotonic correlation, Pearson's coefficient will be used throughout this work. The assumptions for the application of the correlation coefficient on data are that both x and y are normally distributed and that a xy-plot does not show a pattern indicating a complex non-linear correlation.

If the sign of r is not important, one can report r^2. Of course, the behavior of r^2 is slightly different of r because of the quadratic proportion.

During linear model fitting, adding a regressor will increase r^2, although adding a regressor does not always mean that the model is better. While adding regressors, there is a risk of overfitting. The adjusted r^2, $adjr^2$, has been devised to make r^2 more comparable over models with a different number of regressors. This is done by taking into account the degrees of freedom. The relation between r^2 and the adjusted r^2 is:

$$adjr^2 = 1 - \frac{df_{tot}}{df_{err}}(1 - r^2) \tag{A.16}$$

$$\begin{aligned}
&\text{with}\\
df_{tot} &= \text{the number of observations } - 1\\
df_{err} &= df_{tot} - \text{ the number of regressors}
\end{aligned}$$

The adjusted r^2 is more robust with respect to the addition of regressors, meaning that $adjr^2$ is particularly useful when carrying out linear model selection.

A disadvantage to the usage of Pearson's correlation coefficient is that it contains only information about the degree of linearity of the correlation. It is possible to have two variables that are correlated in a non-linear manner. The correlation coefficient is not able to detect such non-linear correlations. Because of this, it is always necessary to inspect a plot of the data in addition to the computed correlation coefficient.

A.5.3 Statistical tests

Statistical tests can be used to obtain more objective judgements about the difference between outcomes produced by different NLP tools. In this dissertation, two statistical tests are used: the χ^2 test of independence and approximate randomization testing. The first test is used to establish the dependence between an indicator, deduced in Section 4.4 and the outcome of self-training experiments. The second test is used in Section 4.2 to compare POS labeling systems.

χ^2 test of independence

The χ^2 test of independence[3] can be used to test for the independence of two nominal variables on the basis of a contingency table, such as Table A.5. If the

[3]A description of the χ^2 test of independence can be found in many statistical handbooks. On online example is at http://udel.edu/~mcdonald/statchiind.html [October 2011].

	left	right	**total**
top	18	110	**128**
bottom	234	142	**376**
total	**252**	**252**	**504**

Table A.5: Example contingency table to illustrate the χ^2 test of independence. There are two nominal variables (left/right) and (top/bottom). The counts of all possible combinations are given in the table, together with the totals.

χ^2 test is used in this works, it is carried out as explained below.

Consider the contingency table A.5, it contains two nominal variables (left/right and top/bottom) and the frequency count of the presence of each of the two different values of the variables. The counts in the table are the observed values, with $O_{1,2}$ the value in the first row and the second column, *viz.* 110. The χ^2 test is designed to indicate, if observed values differ significantly from expected values. The null hypothesis is that they do not differ significantly, meaning that the two nominal variables are independent. An expected value is calculated as:

$$E_{i,j} = \frac{\sum_k O_{i,k} \ \sum_k O_{k,j}}{\sum_{k,l} O_{k,l}} \tag{A.17}$$

χ^2 is then calculated as:

$$\chi^2 = \sum_{i,j} \frac{\left(O_{i,j} - E_{i,j}\right)^2}{E_{i,j}} \tag{A.18}$$

In the example, χ^2 is 88.6. The probability of this χ^2 value can be retrieved from tables or software. The degrees of freedom for a 2×2 contingency table is 1, computed as: (number of rows)–1 × (number of columns)–1. It can be found that the probability associated with the χ^2 value is $5e^{-21}$, which is a lot smaller than the generally accepted 0.05 significance level. This means that the null hypothesis is rejected and that the two nominal variables are not independent.

Significance of accuracy differences

A good statistical test developed for assessing the significance level of the difference between two complicated evaluation measures, such as F-score, is the approximate randomization test (Noreen, 1989). This test gives an answer to the question whether the outcome of an experiment using one technique could be obtained when the experiment was run with another technique, while assuming that both techniques are equal (Yeh, 2000).

The approximate randomization test is well suited for situations with a high number of responses (> 20), as will be the case in a machine learning setup with thousands of instances. If the number of responses is small, an exact randomization test becomes available (Yeh, 2000). A second advantage of the test is that it does not make any assumption about the nature of the underlying distribution of the responses. The last advantage is that an approximate randomization test can handle paired observations – in the sense that the test can compare the performance of two different systems labeling the same set of instances.

Inter-instance dependencies appear when the label assigned to a first instance, has an influence on the label predicted for the following instance. This is the case for conditional random fields (CRFs), machine translation systems and the memory-based tagger (MBT), when looking at the token level. If the previous token is labeled correctly, the likeliness increases that the current token will be labeled correctly to boot. This property makes the approximate randomization test unsuited for significance testing on the token level, since the inter-dependencies will make that probability of the outcomes of two labeling systems will no longer be uniquely defined by the possible permutations of the predicted labels. A good illustration of this behavior is given by Yeh (2000).

A possible solution for the inter-dependency problem is not to take tokens as the minimal unit to shuffle, but to shuffle sentences. Indeed, when the labeling system is stateless, the labels assigned to one sentence will not depend on the labels given to the previous sentence. An example for machine translation can be found in Riezler & Maxwell III (2005).

In the remainder of this subsection, an example of exact randomization testing for two instances is given.

For exact randomization testing, the labels given by two different systems are randomly shuffled between the two systems, see Table A.6. For a test set of two instances, it is possible to shuffle the predictions in four different manners.

system1	label1 label1	100%
system2	label2 label2	0%
		100%
system1	label1 label2	50%
system2	label2 label1	50%
		0%
system1	label2 label1	50%
system2	label1 label2	50%
		0%
system1	label2 label2	0%
system2	label1 label1	100%
		100%
reference	label1 label1	–

Table A.6: All permutations of the labels given to two instances by two systems. The first row contains the labels as they are given by system1. Both labels are named *label1*, but the label given to the first instance (column 2) does not have to be the same as the label given to the second instance (column 3). The index merely indicates that the label has originally been given by system1. The last column contains the accuracy when the two predicted labels are compared to the reference (last row). The absolute accuracy difference between the two systems is indicated in bold.

A label assigned to one instance can never be switched with a label given to another instance, only labels assigned to the same instance by a different systems are switched. The first two rows of Table A.6 represent the original systems. The absolute accuracy difference between the two systems is 100%. The H0 hypothesis to test is that system1 performs equally to system2 and that the predicted labels are interchangeable. Under H0, all permutations of Table A.6 are equally likely to occur.

From Table A.6, it can be seen that the accuracy difference between the two systems is greater than or equal to the original accuracy difference for two shuffles out of 4. This means that under H0, the original accuracy difference of 100% has a probability of 0.5. The H0 hypothesis would be rejected if the probability drops below a predefined significance level (often 0.05 or 0.01). This means that H0 is not rejected and that there is no significant difference in the accuracy of system1 and system2.

The number of possible permutations rapidly increases with the number of instances. In practice, this means that exact randomization testing becomes impractical for larger test sets and approximate randomization testing can be used.

During approximate randomization testing, the permutations of Table A.6 are not computed, but the predicted labels are shuffled randomly between the two systems – keeping the label–instance relation intact. When the number of shuffles is sufficiently large, the probability of the original accuracy difference can be estimated with the formula:

$$prob = \frac{1 + nge}{1 + \text{number of shuffles}} \tag{A.19}$$

with *nge* the number of shuffles for which the absolute accuracy difference is greater than or equal to the original accuracy difference.

For part-of-speech accuracy, the elements that should be shuffled are not the single part-of-speech labels, but the labels of entire sentences. Meaning that in this case *label1* in Table A.6 stands for all labels of a single sentence. This is due to the already mentioned inter-dependencies.

Appendix B

Software

In this Chapter, the scripts that are written to carry out the experiments in this dissertation are listed. The scripts are the result of evolving research and are implemented, while obtaining progressing knowledge about the subject. This leads to inefficient and graceless coding. The scripts are provided with all idiosyncrasies included and no support is provided. Nevertheless, when presenting results in computational linguistics research, the influence of specific decisions during code writing cannot be underestimated. For this reason, the code must be accessible to the reader of this work thus offering her/him the minimum that is needed to validate, to reproduce or even to ameliorate the presented research. The scripts are made available at:

http://www.clips.ua.ac.be/∼vincent/thesis-software

More information about a script can be found when running the script with the `-h` option:

```
$ python script.py -h
```

but in most cases looking at the source code will provide more interesting information.[1]

[1]Some of the scripts depend on software that is freely available on the Internet. The links in this chapter were all visited in January 2012.

Scripts that are more generally applicable are available at:

http://www.clips.ua.ac.be/∼vincent/software.html

B.1 Divergences and other measures

`getdistances.py` :
A script to compute some distances between two files that contain data in a format that can be used by the memory-based tagger (MBT). This script depends on `distribution.py` and `metric.py`. `distribution.py` contains a collection of functions to read distributions from files. `metric.py` contains one function to compute a collection of distances based on distributions.

`asl.py` :
A script to compute the average sentence length of files that contain data in a format that can be used by the memory-based tagger (MBT).

`cosinetopk.py` :
A script to compute the cosine divergence of files that contain data in a format that can be used by the memory-based tagger (MBT). The cosine divergence is computed with vectors of predefined length. Depends on Numpy.[2]

`perplexity.py` :
A script to calculate the perplexity of an MBT file when another MBT file is used to construct the language model. Based on SRILM.[3]

`mmd.py` :
A script[4] to compute the maximum mean discrepancy (mmd) for vectors of integers as defined by Gretton et al. (2007). Depends on Numpy.

`hdivergence.py` :
A script to compute the $\mathcal{H}$-divergence as presented by Ben-David et al. (2010). Depends on SVM^{light}[5] (Joachims, 1999) and `binarize.py`.[6] To compute the $\mathcal{H}$-divergence, the Huber loss is needed. From the output of *svm_learn* of

[2]http://numpy.scipy.org
[3]http://www.speech.sri.com/projects/srilm
[4]The script is validated against http://people.kyb.tuebingen.mpg.de/arthur/mmd.htm
[5]http://svmlight.joachims.org/
[6]http://www.clips.ua.ac.be/∼vincent/software.html#binarize

SVM^{light}, the Huber loss can be approximated with $\frac{\text{L1 loss}}{2n}$, with n the total number of instances. An overview of various loss functions can be found in Rosasco et al. (2004).

`art.py` :
A script to carry out approximate randomization tests.

B.2 Domain distance and performance

`wordcount.py` :
A script to count the number of tokens and sentences in MBT style files.

`approx.py` :
A script to crop and randomize MBT style files to a given number of sentences or tokens.

`bncpos.py` :
A script to extract MBT style files from BNC version 2 for nine domains.

`nfold.py` :
A script to split an MBT file into parts that can be used for cross-validation.

`ontonotes.py` : A script to extract MBT style files from OntoNotes 3.0. Depends on NLTK.[7]

`size_experiment.py, size_report.py` :
Two scripts to produce the plots for the metric stability experiments. They depend on matplotlib[8], Numpy and various divergence scripts.

`experimentloop.py, experiment_report.py` :
The scripts to investigate the correlation between distance metrics and machine learner performance. Depends on
`nfoldmbtmetrics.py`, Scipy[9], Numpy, and MBT. More information about how to use the scripts can be found in `correlation_readme.txt`.

`learner.py, baseline.py, SVMTool.py, selfloop.py` :
Three scripts to switch from MBT as the POS-tagger to SVMTool[10] or the

[7]http://www.nltk.org/
[8]http://matplotlib.sourceforge.net
[9]http://www.scipy.org
[10]http://www.lsi.upc.edu/∼nlp/SVMTool

majority algorithm in `experimentloop.py`. `selfloop.py` can be used to carry out in-domain experiments.

`statistic2.py` :
A script to compute confidence intervals for new observations. This script contains the function called check(), an extended version of the analyzedata() function presented in `experiment_report.py`.

`erroranalysis.py` :
A script to investigate the effect of $P(Y|X)$ on performance. More information about how to use the script can be found in
`erroranalysis_readme.txt`.

`robust.py, timbl.py` :
A script to run TiMBL experiments with different sets of ignored features. Can be used to investigate the effect of omitting lexical features. Depends on TiMBL and Numpy.

B.3 Applying distances

`selection.py selection_report.py` :
Two scripts for training data selection experiments. Depends on MBT. The textfile `trainingdata_readme.txt` contains more information about how to use the scripts.

`featureselection.py, featureselection2.py, getdists.py` :
Scripts to investigate the correlation between distance and performance for separate features. `featureanalyzer.py` can be used to analyze the data. `addfeature.py, adddummy.py` can be used to construct files with synthetic data. Depends on Numpy. More information about the experimental procedure can be found in `featureselection_readme.txt` and an example of how to create synthetic data can be found in `prepare.sh`.

`selftrain.py, selftrainloop.py, selftrain_report.py` :
Scripts to carry out self-training experiments. More information about how to use the scripts can be found in `selftraining_readme.txt`. Depends on Numpy and matplotlib.

B.4 Background information

`nspheres.py, topology.py` :
Two scripts to visualize the distances, as produced by a metric. Depends on visual.[11]

`prepare_for_itml.py` :
Computing the Mahalanobis distances involves finding the matrix M with the ITML algorithm[12] (Davis et al., 2007).
`prepare_for_itml.py` can be used to extract a matrix of token frequency vectors for the BNC domains and to put the output from a POS labeling experiment into a matrix. The created files can then be used with the MATLAB scripts: `RelatedDistance.m`, `MetricLearningAutotuneCorr.m`, `KL.m`. Some of these scripts are adaptations of the original ITML scripts and should be used together with these original scripts. The steps involved in computing the Mahalanobis distance are given in `mahalanobis_readme.txt`.

`compdiv.py` :
A script that computes distances between artificial distributions.

[11] http://www.clips.ua.ac.be/~vincent/software.html#visual
[12] http://www.cs.utexas.edu/~pjain/itml

Bibliography

Alumäe, T., & Kurimo, M. (2010). Domain adaptation of maximum entropy language models. In *Proceedings of the ACL 2010 Conference Short Papers*, (pp. 301–306). Uppsala, Sweden: Association for Computational Linguistics.

Argamon, S., & Koppel, M. (2010). The rest of the story: Finding meaning in stylistic variation. In S. Argamon, K. Burns, & S. Dubnov (Eds.) *The Structure of Style*, (pp. 79–112). Springer-Verlag, Heidelberg.

Atterer, M., & Schütze, H. (2007). Prepositional phrase attachment without oracles. *Computational Linguistics*, *33*(4), 469–476.

Bacchiani, M., & Roark, B. (2003). Unsupervised language model adaptation. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, (pp. 224–227). Hong Kong, China: IEEE.

Batu, T., Fortnow, L., Rubinfeld, R., Smith, W. D., & White, P. (2000). Testing that distributions are close. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, (pp. 259–269). Redondo Beach, CA, USA: IEEE.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. (2010). A theory of learning from different domains. *Machine Learning*, *79*, 151–175.

Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, *35*, 99–109.

Biber, D. (1988). *Variation across speech and writing*. Cambridge, UK: Cambridge University Press.

Biber, D., & Gray, B. (2010). Challenging stereotypes about academic writing: Complexity, elaboration, explicitness. *Journal of English for Academic Purposes*, *9*(1), 2–20.

Bikel, D. (2004). Intricacies of Collins' parsing model. *Computational Linguistics*, *30*(4), 479–512.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Blitzer, J., Dredze, M., & Pereira, F. (2007). Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the* 45th *Annual Meeting of the Association of Computational Linguistics*, (pp. 440–447). Prague, Czech Republic: Association for Computational Linguistics.

Blitzer, J., McDonald, R., & Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, (pp. 120–128). Sydney, Australia: Association for Computational Linguistics.

BNC (2001). The British National Corpus, version 2 (BNC world). Available at http://www.natcorp.ox.ac.uk (Last accessed: March 2012).

Chan, Y. S., & Ng, H. T. (2007). Domain adaptation with active learning for word sense disambiguation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, (pp. 49–56). Prague, Czech Republic: Association for Computational Linguistics.

Charniak, E. (1997). Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, (pp. 598–603). Rhode Island, USA: MIT Press.

Chelba, C., & Acero, A. (2006). Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech and Language*, *20*(4), 382–399.

Chen, B., Lam, W., Tsang, I., & Wong, T.-L. (2009). Extracting discriminative concepts for domain adaptation in text mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, (pp. 179–188). Paris, France: ACM.

Chen, S. (2009). Performance prediction for exponential language models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (pp. 450–458). Boulder, CO, USA: Association for Computational Linguistics.

Cohen, A., Mantegna, R. N., & Havlin, S. (1997). Numerical analysis of word frequencies in artificial and natural language texts. *Fractals*, *5*(1), 95–104.

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, *29*(4), 589–637.

Coope, I. D. (2000). Reliable computation of the points of intersection of n spheres in $\mathbb{R}^n$. *Australian and New Zealand Industrial and Applied Mathematics Journal (ANZIAM)*, *42*(E), C461–C477.

Daelemans, W., Buchholz, S., & Veenstra, J. (1999). Memory-based shallow parsing. In *Proceedings of the Third Conference on Computational Natural Language Learning (CoNLL)*, (pp. 53–60). Bergen, Norway: Association for Computational Linguistics.

Daelemans, W., & van den Bosch, A. (2005). *Memory-Based Language Processing*. Studies in Natural Language Processing. Cambridge: Cambridge University Press.

Daelemans, W., Zavrel, J., van der Sloot, K., & van den Bosch, A. (2010). Timbl: Tilburg memory-based learner, version 6.3. Tech. Rep. ILK 10-01, Tilburg University.

Dahlmeier, D., & Ng, H. T. (2010). Domain adaptation for semantic role labeling in the biomedical domain. *Bioinformatics*, *26*(8), 1098–1104.

Daumé III, H. (2007). Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*. Prague, Czech Republic: Association for Computational Linguistics.

Daumé III, H., Kumar, A., & Saha, A. (2010). Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, (pp. 53–59). Uppsala, Sweden: Association for Computational Linguistics.

Daumé III, H., & Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, *26*, 101–126.

Davis, J. V., Kulis, B., Jain, P., Sra, S., & Dhillon, I. S. (2007). Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, (pp. 209–216). Corvalis, OR, USA: Omnipress.

De Smedt, T., Van Asch, V., & Daelemans, W. (2010). Memory-based shallow parser for Python. CLiPS Technical Report Series 2, CLiPS, University of Antwerp.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4), 380–393.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*, vol. 31 of *Stochastic Modelling and Applied Probability*. Heidelberg, Germany: Springer-Verlag.

Dhillon, P. S., Talukdar, P. P., & Crammer, K. (2010). Learning better data representation using inference-driven metric learning. In *Proceedings of the ACL 2010 Conference Short Papers*, (pp. 377–381). Uppsala, Sweden: Association for Computational Linguistics.

Finkel, J. R., & Manning, C. D. (2009). Hierarchical Bayesian domain adaptation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, (pp. 602–610). Boulder, Colorado: Association for Computational Linguistics.

Francis, W. N., & Kučera, H. (1964). A standard corpus of present-day edited American English, for use with digital computers. Brown University. Providence, Rhode Island, USA. Revised in 1971, 1979.

Gao, J., Goodman, J., Li, M., & Lee, K.-F. (2002). Toward a unified approach to statistical language modeling for chinese. *Transactions on Asian Language Information Processing*, *1*(1), 3–33.

Gheyas, I. A., & Smith, L. S. (2010). Feature subset selection in large dimensionality domains. *Pattern Recognition*, *43*, 5–13.

Giménez, J., & Márquez, L. (2004). SVMTool: A general POS tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, (pp. 43–46). Lisbon, Portugal: European Language Resources Association.

Globerson, A., & Roweis, S. (2006). Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, (pp. 353–360). New York, USA: ACM.

Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., & Smola, A. J. (2007). A kernel method for the two-sample-problem. In *Advances in Neural Information Processing Systems 19*, (pp. 513–520). Cambridge, Massachusetts, USA: MIT Press.

Grosse, I., Bernaola-Galván, P., Carpena, P., Román-Roldán, R., Oliver, J., & Stanley, H. E. (2002). Analysis of symbolic sequences using the Jensen-Shannon divergence. *Physical Review E*, *65*(4), 041905–1—041905–16.

Hara, T., Miyao, Y., & Tsujii, J. (2007). Evaluating impact of re-training a lexical disambiguation model on domain adaptation of an HPSG parser. In *Proceedings of the Tenth International Conference on Parsing Technologies*, (pp. 11–22). Prague, Czech Republic: Association for Computational Linguistics.

Hellinger, E. (1909). Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen. *Journal für die reine und angewandte Mathematik (Crelle's Journal)*, *1909*(136), 210–271.

Hindle, D., & Rooth, M. (1993). Structural ambiguity and lexical relations. *Computational Linguistics*, *19*(1), 103–120.

Homola, P., & Kuboň, V. (2006). A structural similarity measure. In *Proceedings of the Workshop on Linguistic Distances*, (pp. 91–99). Sydney, Australia: Association for Computational Linguistics.

Huber, P. J. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics*, *53*(1), 73–101.

Jankowski, N., & Grochowski, M. (2004). Comparison of instances seletion algorithms I. algorithms survey. In *Artificial Intelligence and Soft Computing (ICAISC 2004)*, vol. 3070 of *Lecture Notes in Computer Science*, (pp. 598–603). Zakopane, Poland: Springer.

Jeong, M., & Lee, G. G. (2009). Multi-domain spoken language understanding with transfer learning. *Speech Communications*, *51*, 412–424.

Jiang, J. (2008). *Domain Adaptation in Natural Language Processing*. Ph.D. thesis, University of Illinois at Urbana-Champaign, Illinois, USA.

Jiang, J., & Zhai, C. (2007). Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, (pp. 264–271). Prague, Czech Republic: Association for Computational Linguistics.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In *Advances in kernel methods: support vector learning*, (pp. 169–184). MIT Press.

Kailath, T. (1967). The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology*, *15*, 52–60.

Karlgren, J. (2010). *Textual Stylictic Variation: Choices, Genres and Individuals*, (pp. 113–125). Springer-Verlag, Heidelberg.

Kendall, M. G. (1970). *Rank correlation methods*. London, UK: Griffin, 4th ed.

Kessler, B. (2001). *The significance of word lists: Statistical tests for investigating historical connections between languages*. Chicago, USA: CSLI publications, The University of Chicago Press.

Kifer, D., Ben-David, S., & Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the 30th Very Large Data Bases Conference (VLDB'04)*, (pp. 180–191). Toronto, Canada: VLDB Endowment.

Kim, J.-D., Ohta, T., Tateisi, Y., & Tsujii, J. (2006). GENIA corpus manual - encoding schemes for the corpus and annotation. Tech. Rep. TR-NLP-UT-2006-1, Tsujii Laboratory, University of Tokyo.

Klakow, D., & Peters, J. (2002). Testing the correlation of word error rate and perplexity. *Speech Communication*, *38*, 19–28.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, *22*(1), 79–86.

Lee, D. Y. W. (2001a). Genres, registers, text types, domain, and styles: Clarifying the concepts and navigating a path through the BNC jungle. *Language Learning & Technology*, *5*(3), 37–72.

Lee, L. (1999). Measures of distributional similarity. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, (pp. 25–32). Maryland, USA: Association for Computational Linguistics.

Lee, L. (2001b). On the effectiveness of the skew divergence for statistical language analysis. In 8^{th} *International Workshop on Artificial Intelligence and Statistics (AISTATS 2001)*, (pp. 65–72). Florida, USA. Online repository http://www.gatsby.ucl.ac.uk/aistats/aistats2001 (Last accessed: April 2010).

Leech, G., Garside, R., & Bryant, M. (1994). CLAWS4: The tagging of the British National Corpus. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING 94)*, (pp. 622–628). Kyoto, Japan: Association for Computational Linguistics.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, (pp. 3–12). Dublin, Ireland: ACM.

Li, W. (1992). Random texts exhibit Zipf's-law-like word frequency distribution. *IEEE Transactions on Information Theory*, *38*(6), 1842–1845.

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, *37*(1), 145–151.

Lippincott, T., Ó Séaghdha, D., & Korhonen, A. (2011). Exploring subdomain variation in biomedical language. *BMC Bioinformatics*, *12*(212), 37–72.

Lippincott, T., Ó Séaghdha, D., Sun, L., & Korhonen, A. (2010). Exploring variation across biomedical subdomains. In *Proceedings of the 23rd International Conference on Computational Linguistics*, (pp. 689–697). Beijing, China: Association for Computational Linguistics.

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, *17*, 491–502.

Mahalanobis, P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, *2*(1), 49–55.

Mansour, Y., Mohri, M., & Rostamizadeh, A. (2009). Multiple source adaptation and the Rényi divergence. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, (pp. 367–374). Montreal, Quebec, Canada: AUAI Press.

Marcus, M., Kim, G., Marcinkiewicz, M., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., & Schasberger, B. (1994). The Penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology (HLT)*, (pp. 114–119). Plainsboro, NJ, USA: Association for Computational Linguistics.

Margolis, A. (2011). *Automatic Annotation of Spoken Language Using Out-of-Domain Resources and Domain Adaptation*. Ph.D. thesis, University of Washington, Seattle, WA, USA.

McClosky, D. (2010). *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph.D. thesis, Department of Computer Science, Brown University, Rhode Island, USA.

Mitra, P., Murthy, C., & Pal, S. K. (2002). Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*, 301–312.

Montemurro, M. A. (2001). Beyond the Zipf-Mandelbrot law in quantitative linguistics. *Physica A: Statistical Mechanics and its Applications*, *300*(3–4), 567–578.

Moore, R. C., & Lewis, W. (2010). Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, (pp. 220–224). Uppsala, Sweden: Association for Computational Linguistics.

Noreen, E. W. (1989). *Computer-intensive methods for testing hypotheses*. New York, NY, USA: John Wiley.

Pacey, M., Fligelstone, S., & Rayson, P. (1997). How to generalize the task of annotation. In *Corpus Annotation: Linguistic Information from Computer Text Corpora*, (pp. 122–136). London, UK: Longman.

Pathak, M. A., & Nyberg, E. H. (2009). Learning algorithms for domain adaptation. In *Proceedings of the 1st Asian Conference on Machine Learning: Advances in Machine Learning*, (pp. 293–307). Nanjing, China: Springer-Verlag.

Pearson, K. (1896). Mathematical contributions to the theory of evolution. – III. Regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London (Series A)*, *187*, 253–318.

Plank, B. (2009). A comparison of structural correspondence learning and self-training for discriminative parse selection. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*, (pp. 37–42). Boulder, Colorado: Association for Computational Linguistics.

Plank, B. (2011). *Domain Adaptation for Parsing*. Ph.D. thesis, University of Groningen, the Netherlands. Groningen Dissertations in Linguistics 96.

Plank, B., & Sima'an, K. (2008). Subdomain sensitive statistical parsing using raw corpora. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*

(LREC'08). Marrakech, Morocco: European Language Resources Association.

Plank, B., & van Noord, G. (2010). Dutch dependency parser performance across domains. *Computational Linguistics in the Netherlands 2010: selected papers from the 20th CLIN meeting*, (pp. 123–138).

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ratnaparkhi, A., Reynar, J., & Roukos, S. (1994). A maximum entropy for prepositional phrase attachment. In *Proceedings of the workshop on Human Language Technology (HLT)*, (pp. 250–255). Plainsboro, NJ, USA: Association for Computational Linguistics.

Ravi, S., Knight, K., & Soricut, R. (2008). Automatic prediction of parser accuracy. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, (pp. 887–896). Honolulu, Hawaii: Association for Computational Linguistics.

Rehbein, I. (2011). Data point selection for self-training. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages*, (pp. 62–67). Dublin, Ireland: Association for Computational Linguistics.

Rényi, A. (1961). On measures of information and entropy. In *Proceedings of the* 4^{th} *Berkeley Symposium on Mathematics, Statistics and Probability*, vol. 1, (pp. 547–561). Berkeley, California, USA: University of California Press.

Riezler, S., & Maxwell III, J. T. (2005). On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, (pp. 57–64). Ann Arbor, MI, USA: Association for Computational Linguistics.

Rosasco, L., De Vito, E., Caponnetto, A., Piana, M., & Verri, A. (2004). Are loss functions all the same? *Neural Computation*, *16*(5), 1063–1076.

Sagae, K. (2010). Self-training without reranking for parser domain adaptation and its impact on semantic role labeling. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, (pp. 37–44). Uppsala, Sweden: Association for Computational Linguistics.

Satpal, S., & Sarawagi, S. (2007). Domain adaptation of conditional probability models via feature subsetting. In *Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, (pp. 224–235). Warsaw, Poland: Springer-Verlag.

Shimizu, N., Hagiwara, M., Ogawa, Y., Toyama, K., & Nakagawa, H. (2008). Metric learning for synonym acquisition. In *Proceedings of the 22nd International Conference on Computational Linguistics*, (pp. 793–800). Manchester, UK: Association for Computational Linguistics.

Sinclair, J., & Ball, J. (1996). Preliminary recommendations on text typology. Expert Advisory Group on Language Engineering Standards (EAGLES) EAG—TCWG—TTYP/P, Consiglio Nazionale delle Ricerche, Istituto di Linguistica Computazionale, Pisa, Italy. Available at http://www.ilc.cnr.it/EAGLES96/texttyp/texttyp.html (Last accessed: June 2011).

Stamatatos, E., Fakotakis, N., & Kokkinakis, G. (2000). Automatic text categorization in terms of genre and author. *Computational Linguistics*, *26*(4), 471–495.

Tateisi, Y., Yakushiji, A., Ohta, T., & Tsujii, J. (2005). Syntax annotation for the GENIA corpus. In *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP), Companion volume*, (pp. 222–227). Jeju Island, Korea: Asian Federation of Natural Language Processing.

Van Asch, V., & Daelemans, W. (2009). Prepositional phrase attachment in shallow parsing. In *Proceedings of the 7th International Conference on Recent Advances in Natural Language Processing (RANLP)*, (pp. 12–17). Borovets, Bulgaria: Association for Computational Linguistics.

Van Asch, V., & Daelemans, W. (2010). Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, (pp. 31–36). Uppsala, Sweden: Association for Computational Linguistics.

van Rijsbergen, C. J. (1975). *Information Retrieval*. London, UK: Butterworths.

Verspoor, K., Cohen, K. B., & Hunter, L. (2009). The textual characteristics of traditional and open access scientific journals are similar. *BMC Bioinformatics*, *10*, 1–16.

Voloshynovska, I. A. (2011). Characteristic features of rank-probabilty word distribution in scientific and belletristic literature. *Journal of Quantitative Linguistics*, *18*(3), 274–289.

Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, *10*, 207–244.

Weischedel, R., Pradhan, S., Ramshaw, L., Kaufman, J., Franchini, M., El-Bachouti, M., Xue, N., Palmer, M., Marcus, M., Taylor, A., Greenberg, C., Hovy, E., Belvin, R., & Houston, A. (2009). OntoNotes release 3.0. Linguistic Data Consortium, Philadelphia.

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th International Conference on Computational Linguistics*, vol. 2, (pp. 947–953). Saarbrücken, Germany: Association for Computational Linguistics.

Zhang, Y., & Wang, R. (2009). Correlating natural language parser performance with statistical measures of the text. In *Proceedings of the 32nd annual German conference on Advances in artificial intelligence*, (pp. 217–224). Paderborn, Germany: Springer-Verlag.

Zipf, G. K. (1972). *Human behavior and the principle of least effort: an introduction to human ecology*. New York, NY, USA: Hafner.

Index