Measuring emotion

Measuring emotion

Exploring the feasibility of automatically classifying emotional text

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

Frederik Vaassen

Promotor
Prof. dr. Walter Daelemans

Faculteit Letteren en Wijsbegeerte
Departement Taalkunde
Antwerpen 2014

Universiteit Antwerpen

Frederik Vaassen

Faculteit Letteren en Wijsbegeerte
Departement Taalkunde

MEASURING EMOTION

Exploring the feasibility
of automatically classifying emotional text

EMOTIE METEN

Onderzoek naar de haalbaarheid
van het automatisch classificeren van emoties in tekst

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

Frederik VAASSEN

Promotor: Prof. Dr. Walter Daelemans

Antwerpen, 2014

The hardest part of writing a PhD is not the research. It's not setting up the experiments, it's not analyzing the results. It's not even writing it all down. The hardest part is to keep convincing yourself, year after year, that you can pull it off, and that what you found out is worth writing about.
The support of the people around me has been invaluable, and I'm grateful that they were there.
I am grateful to my family, who never doubted me for a moment. At home, my wife Jennifer remained more than patient with me, even when I sometimes failed to put work aside to tend to more important things. She also helped putting everything into perspective, which turned out to be essential to deliver this thesis. Then there's of course my parents, who guided me to this point, but who always gave me the freedom to choose my own path.
My supervisor, Walter Daelemans, has always been available for inspiration, feedback and support. He trusted me with a project five years ago, and since then, he has trusted me again and again when it was most needed. I am also grateful to all my colleagues at the CLiPS Research Center. I imagine some of you will pick up this book —possibly with no intention of actually reading it, but I forgive you!— to scan these acknowledgements hoping to find their name. Well, sorry, I'm not going to mention anyone in particular! There have been quite a few of you over the years, and you have all been great to work with, and many of you have become good friends. I will always fondly remember the table tennis tournaments and the Little Red Book and the coffee beers and the hamster house...

Abstract

This thesis explores methods for the automatic supervised categorization of text according to the emotions of its author. Since "emotions" are by definition subjective, we predict that this task will present a number of significant challenges. We know that there is low inter-annotator agreement on gold-standard data, since humans find it hard to identify emotions in text, which results in datasets that are noisy and difficult to learn from. In addition to (and maybe partially as a consequence of) this annotation difficulty, existing emotion datasets are relatively small. This makes the task even more challenging, as there is very little data from which a classification algorithm can infer a model.

To explore the possibilities and limitations of automatic emotion classification, we develop two very different case studies in detail.

In a first case study, we attempt to automatically classify sentences from business conversations according to a dimensional emotion model called *Leary's Rose*, or the *Interpersonal Circumplex*. We discretize this model into eight emotion categories, and attempt to find the best features to capture the problem using a Support Vector Machines-based classification system. Considered feature types include word tokens and lemmas, character n-grams, part of speech tags, lexicon entries from two different subjectivity lexicons, and contextual information regarding the previous positions of speaker and hearer on Leary's Rose. We determine that character n-grams are good at capturing relevant information, and that they are well supplemented with carefully selected emotion keywords. In theory, a speaker's previous emotional state makes for an excellent predictor of their current position on the Circumplex, but in practice, we find that performance is too low for contextual class information to be useful. We compare the classifiers' performance to the inter-annotator agreement on the dataset, and conclude that the ambiguity is also present in the gold standard data, as even human agreement is very low.

The second case study concerns the automatic identification of emotions in suicide notes. We again classify on a sentence by sentence basis, but the class system is resolutely discrete (meaning there is no explicit dimensional relationship between the considered emotions, as there was on the Interpersonal Circumplex). It is also possible for one sentence to contain more than one emotion. To solve this multi-label classification problem, we use a SVM-based system which attributes a class label if its probability exceeds an experimentally determined threshold. We experiment with token and character n-grams, and again conclude that character n-grams capture relevant features well. The results of the case study also show very clearly that the number of training instances for a class has a very large influence on the classifier's performance on that class. Sparse classes are extremely difficult to predict well. This is hardly a surprising finding, but given the limited size of existing emotion datasets, sparse classes are a problem that needs to be tackled.

Having determined that low-agreement training data and data sparseness are significant hurdles for emotion classification, we try to find solutions to work around these problems, or to at least mitigate their negative impact. While it seems unrealistic to build a large, high-agreement corpus for every new emotion classification task, it might be possible to build a single large, reliable corpus, which can then be used in a variety of other tasks. To determine if this approach is a valid one, we experiment with cross-domain classification and domain adaptation. We conclude that while there cannot be a single, universal, domain-independent "emotion classifier", it is possible to build a cross-domain classifier that outperforms a classifier built on target data only. When a dataset is especially skewed and difficult, taking a larger, more stable dataset as training data can improve performance significantly, especially when this cross-domain data is supplemented with part of the target dataset. This way the learner can adapt to particularities of the target dataset, while still having enough general information to build a stable emotion model. We also show that applying a few simple domain adaptation techniques can improve cross-domain performance even further.

Encouraged by these results, we attempt to limit training noise and data skewedness in the training set even further, in the hope that a limited but well-balanced and representative dataset will serve as a good training set —when supplemented with some target data— for cross-domain emotion classification. We attempt to counter the negative effects of unbalanced class distributions by applying class weights: strengthening smaller classes, and weighing down large classes. We conclude that this technique helps with highly skewed datasets, but that it is detrimental in most other cases.
We also attempt to build a hierarchical classification system: a top-level classifier determines if an instance is neutral or if it contains an emotion; the bottom-level classifier —trained only on emotional instances— assigns emotion labels, but only to those instances that were accepted by the top-level classifier. This technique turns out not to be effective, as the hierarchical structure only serves to multiply classification errors: an instance that was wrongly classified by the top-level classifier, can never be corrected at the bottom level.
Finally, we attempt to limit the training data to only those instances that are typical for their class. Filtering out "noisy" atypical instances in this way, however, turns out to be problematic, as this technique cuts down the number of instances in an already very small dataset, which accentuates the problem of data sparseness.

We conclude our exploration of automatic emotion classification with some pointers for the present and hopes for the future: it is usually a good idea to have a larger dataset with higher-confidence emotion annotations, even if this training dataset and the target data do not belong to the same domain. When supplemented with a part of the target data, it will serve as a good basis for a cross-domain emotion classifier. In terms of features, we have found that character n-grams capture a lot of relevant information, and that it is a good idea to supplement these with a limited

set of carefully selected emotion keywords (from an emotion lexicon, for instance). Dealing with any remaining noise and data skewedness is best left to the classifier, as manipulating the instance space prior to training will likely result in performance loss.
For the future, we hope to the see available emotion datasets grow both in size and in quality, but we believe text-based emotion classification can only go so far. This is why we hope to see more and more multi-modal approaches to emotion detection, approaches that do not abstract away from the treasure of emotional information that is to be found in intonation, facial expressions, body language...

Samenvatting

In "Emotie meten" onderzoeken we methodes voor het automatische identificeren van emoties in tekst. Gegeven een tekstfragment, in welke mate is het mogelijk om een computer te trainen om de emotie van de auteur af te leiden, zonder terug te grijpen naar niet-tekstuele informatie?

Om op deze vraag te antwoorden, worden er twee verschillende casestudies in detail uitgewerkt. Deze casestudies wijzen ook meteen op de potentiële applicaties van het onderzoek naar automatische emotieclassificatie.

In de eerste casestudy wordt er een systeem ontwikkeld dat zinnen uit business-conversaties classificeert volgens een dimensioneel emotiemodel: de Roos van Leary, of het Interpersonele Circumplex. Dit systeem werd geïntegreerd in een proof-of-conceptapplicatie, deLearyous, in de vorm van een virtuele conversatiepartner die kan gebruikt worden om communicatieskills te trainen. Zinnen uit conversaties worden met behulp van machine learners (Support Vector Machines) automatisch in één van acht mogelijke octanten geplaatst, waarbij elk octant overeenkomt met een mogelijke emotie. Experimenteel werd vastgesteld dat de machine learners de emotie-informatie het best kunnen vatten wanneer ze getraind worden met specifieke emotietermen en met character n-grams (karaktersequenties) die geëxtraheerd worden uit een dataset van reeds geclassificeerde conversaties. In theorie is de emotie van dezelfde spreker in een vorige zin in de conversatie ook een zeer sterke indicator, maar de kwaliteit van de voorspellingen ligt te laag om daar in de praktijk efficiënt gebruik van te kunnen maken. We stellen vast dat de voorspellingen van het systeem vaak te wensen overlaten, maar tonen ook aan dat mensen ook slecht scoren op het identificeren van emoties in tekst. Dit nuanceert de resultaten van de automatische systemen, maar werpt ook vragen op over de betrouwbaarheid van emotiedatasets in het algemeen.

In een tweede casestudy worden er emoties geclassificeerd in zelfmoordbriefjes. Een systeem dat emoties in dit soort teksten kan terugvinden, zou gebruikt kunnen wor-

den in medische context; om te helpen bij diagnose van depressie, of zelfs bij het voorkomen van zelfmoordpogingen. Ook hier wordt de tekst zin per zin geclassificeerd in één van 15 mogelijke klasses, en ook hier worden machine learners getraind op vooraf gelabelde teksten. Character n-grams scoren goed, op voorwaarde dat er genoeg training data aanwezig is voor de te voorspellen klasse. Dit wijst op het belang van uitgebreid en goed gebalanceerd trainingsmateriaal.

Uit de casestudies concluderen we dat de onbetrouwbaarheid (die logisch volgt uit de subjectieviteit van emoties) en de schaarsheid van trainingsdata de grootste struikelblokken zijn voor automatische emotieclassificatie. We zoeken dan ook naar oplossingen om deze belemmeringen uit de weg te ruimen, of alleszins te minimaliseren.

Hoewel het ondoenbaar lijkt om voor elke emotieclassificatietaak één uitgebreide en betrouwbare dataset te bouwen, kan er wel gewerkt worden op één "universele" emotiedataset. We tonen aan dat het aan de hand van enkele simpele domeinadaptatietechnieken mogelijk is om een systeem dat getraind is op één dataset, aan te passen aan een andere dataset, met in sommige gevallen zelfs een verbetering in classificaties als gevolg. We tonen ook aan dat enkele technieken die direct ingrijpen op de featurevectoren minder aangewezen zijn, daar deze leiden tot overfitting en slechtere voorspellingen.

We concluderen ons onderzoek met enkele "best practices" omtrent trainingdataselectie en training features. We kijken uit naar de combinatie van de technieken die toegepast werden tijdens dit onderzoek –puur gebaseerd op tekstuele informatie– met technieken die aanspraak doen op niet-tekstuele bronnen: intonatie, gezichtsuitdrukkingen, lichaamstaal, etc.

Contents

Chapter 1

Introduction

What if your computer could understand how you feel? What if —based on the e-mails you wrote today, based on your posts on social networks, on the reviews you wrote online— your computer could deduce your mood, and react accordingly? "Feeling down? Here's some upbeat music you like!", "Stressed out? Allow me to change your desktop's color scheme to a more soothing setting."

The above scenario probably sounds far-fetched. And it is, in a way; we are still far from reaching a point where a machine can use textual information to reliably identify something as complex and subtle as human emotions. Let's not even mention the trial-and-error and fine-tuning it would require to determine how and when a system should react to whatever emotion it has identified. (One can imagine, for instance, that the last thing one would want when feeling grumpy, is a computer trying to do things it was never asked to do.) Despite all this, automatic emotion classification —which, as the name implies, involves automatically categorizing text fragments according to the emotions they contain— is most definitely an active field of research. And one day, perhaps by combining what we will have learned from text with information from other sources (biometric input devices are in full development, for instance), we will no longer see a "feeling" system as preposterous.

As things are now, however, automatic emotion classification of text is in its infancy. This thesis is our contribution to the field, and as the reader will realize, we will not refrain from bringing bad news if we deem it necessary. After all, reducing something that is by definition subjective to something that can be processed through objective (unfeeling?) algorithms and statistics, sounds like an uphill battle.

1.1 Quantifying the unquantifiable

There are inherent problems with wanting to automatically identify emotions. Speaking strictly from a computational linguistics point of view, we run into our first, significant roadblock before even attempting to classify anything: putting together a good dataset is problematic.
We take a "good" dataset —at least in the case of supervised text classification— to imply two things. First, it needs to be large enough so that all categories we want to learn are sufficiently represented. Put simply, if we have not seen that a certain feature (say, a word like *down*) can point to a certain class (for instance the *sadness* category), our model will not be able to learn that there is an association between the two. Secondly, the dataset needs to contain reliable information. While most machine learning algorithms are designed to deal with a significant amount of noise, too many noisy examples will eventually confuse any system.
The biggest problem for emotion datasets is to be found in the latter point: reliability. We will see in-depth examples of this in the following chapters, but emotion classification of text is a difficult task, even for humans. The likelihood that multiple annotators agree on the emotional contents of a text fragment is very slim, and even the average agreement between pairs of annotators will be low. How we perceive emotions in a text fragment is strongly correlated with the context the text occurs in, but it is also influenced by personal biases such as personality and mood. Additionally, a lot of information regarding a speaker's (or writer's) emotional state is to be found in non-textual data. In the case of text transcribed from speech, we lose intonation, body language, facial expressions, etc., which in some cases make the difference between one emotion and the next. As a result, text collections annotated according to emotion usually have relatively low inter-annotator agreement, and the "gold" labels —which we train our classifiers on— are hard to pin down. On some texts, disagreement may have been caused by an annotator making a mistake or misinterpreting a text fragment, but in most cases, disagreement is simply that: annotators disagree on the emotions present, because emotions are inherently subjective.

The fact that it is difficult to acquire reliable emotion data indirectly means that it is also difficult to acquire *a lot* of emotion data. Annotation is a lengthy and difficult process, and not many researchers are willing to invest the time or resources to produce a dataset which is likely to have less-than-optimal reliability. An alternative is to harvest data from the Internet. The web is full of textual information, and in some rare cases texts are tagged with moods or emotions. Facebook[1] has the ability to tag status updates with emotions, and LiveJournal[2] has long allowed users to select a mood tag to accompany their blog entries. It would be possible to put

[1] http://www.facebook.com - Last accessed on July 17th, 2014.
[2] http://www.livejournal.com - Last accessed on July 17th, 2014

together a large emotion dataset by mining these online resources (Mishne (2005) and Keshtkar & Inkpen (2009), amongst others, have used LiveJournal data to train their mood classifiers), but again, reliability is a problem. There is no guarantee that the authors of the online texts use these emotion tags reliably or consistently, and the interpretation of the tags may vary from author to author.

In short, there are no perfect datasets for automatic emotion classification, because emotions are in themselves subjective. This, however, will not stop us from investigating the possibilities. In the following chapters, we will explore what is currently achievable in emotion classification. We will use a variety of datasets, which all have their own strength and flaws, to answer questions regarding the feasibility of automatic emotion classification, and the robustness of emotion classifiers.

Chapter 2 will start by placing "emotion classification" in a broader context. We will briefly outline the fields of text categorization and sentiment mining, and we will see how emotion classification of text borrows extensively from both. We will focus on the "emotion" in emotion classification, as it is anything but a unambiguous term. What "emotions" are and which emotions should be seen as distinct is a subject of heated debate among psychologists. Should emotions be seen as discrete, or should we see an emotion as a position in a continuous space? If we adopt the former approach, how many discrete emotions should we consider? What dimensions —should we adopt the dimensional hypothesis— define emotional space? The answers to these questions should influence how computational linguists choose to represent their emotion classification tasks, but we will see that the choice of model is often driven by convenience and practical considerations, rather than by findings in psychology. The chapter will also serve to outline the state of the art in the automatic emotion classification of text, as we will give a few salient examples of how the problem is most often tackled.

The next two chapters are case studies, in which we describe our approach to carrying out two very different emotion classification tasks.

The first case study is described in Chapter 3. The deLearyous dataset is a relatively small dataset of transcribed conversations from a business environment, annotated according to Leary's Rose, also known as the Interpersonal Circumplex, a dimensional model for emotion classification. In this case study, the task is to determine the position of the participants in a conversation on this Rose, on a sentence-by-sentence basis, based only on textual information. To our knowledge, this is the first time that conversations have been analyzed automatically for their participants' positions on the Interpersonal Circumplex.

Chapter 4 describes the second case study: the automatic classification of emotions in suicide notes. The dataset in this study uses a discrete emotion framework, and the nature of the texts is radically different from the more restrained business conversations transcribed in the deLearyous dataset. Furthermore, we do not assume

that emotions are mutually exclusive, and carry out multi-label emotion classification.

The case studies will back up our introductory claims that the size of the training set and low human agreement are problematic for emotion classification. In the next two chapters, we will explore ways to circumvent these problems. We will investigate whether or not is it possible to use a dataset from a source domain that is different from the domain of the target dataset, to improve classification performance on the target task. While in general, one would expect a classifier to perform best when trained on data from the same domain as the test documents, we hypothesize that when the training dataset is small and noisy, a classifier trained on a larger, more reliable dataset —even if the data is from a different domain— will give significantly better results.

In Chapter 5, we will determine to what extent a model trained on a single dataset can be called a "universal" emotion classifier. How well will a classifier from one domain perform in a different domain, and why? What can we do to make a model generalize better, and how does this influence performance of the model in the source domain? We will see our hypotheses confirmed, as performance on a smaller, more difficult dataset is significantly improved by using a larger training set from a different domain, especially after applying a few simple domain-adaptation tricks.

Chapter 6 will dig into slightly more advanced techniques with the hope of improving in-domain and cross-domain emotion classification even further. We will attempt to limit the noise in the training dataset, and we will try to limit the skewedness of the data distribution, in the assumption that a more uniform class-distribution and more typical instances will result in a more stable training set. We will adopt a variety of approaches to test these theories, including adapting class weights in the classification algorithms, applying a hierarchical classifier ensemble, and using typicality-based instance resampling. We will however notice that class weighting only works in very specific cases, that hierarchical classification increases classification errors, and that only retaining typical instances accentuates data sparseness problems, which results in a performance drop.

We will wrap up our findings in Chapter 7. While we will by no means have solved the emotion classification problem, we will be able to give some pointers on how to maximize performance (or on how to minimize performance loss across datasets) given the datasets currently available for emotion classification. We will also give our thoughts on what we believe the future of emotion classification will look like, and what we believe will be the necessary steps to take in order to build emotion classifiers that are accurate enough to serve as the basis for practical applications.

Chapter 2

Related research

The automatic identification of emotions in text is a problem that is traditionally interpreted as a text categorization task. To properly introduce emotion classification, it is therefore necessary to take a step back, and to first introduce text categorization in general. This is what we will do in section 2.1. Section 2.2 will zoom into sentiment analysis, as many of the tools (section 2.2.1) and techniques (section 2.2.2) used in sentiment analysis are also used in emotion classification. Finally, section 2.3 will describe emotion classification proper. We will first investigate how emotions are classified in psychology (section 2.3.1), and if (and how) these classification frameworks are useful for automatic emotion classification. We will investigate a set of lexicons constructed specifically to help in the categorization of emotional text (section 2.3.2), and we will give a short overview of the state of the art in emotion classification (section 2.3.3).

2.1 Automated text categorization

It is impossible to write an overview of the work in automated emotion classification without first referring to the more general framework of automated text categorization. The goal of a text categorization task is to automatically associate natural language texts with predefined thematic categories, or, more formally, given a domain of documents D and a set of predefined categories $C = (c_1, ..., c_{|c|})$, the goal is to determine for each document-category pair $\langle d_j, c_i \rangle$ whether or not the document d_j should be filed under category c_i (Sebastiani, 2002). In practice, this may imply automatically labeling a set of news articles with their topics (*politics*, *sports*...) or classifying e-mails as either *spam* or *ham*. Less straightforward applications include determining part of speech tags of words (should the word "trip" be categorized as a

noun or as a verb?) or word-sense disambiguation (does the word "trunk" in a given context refer to the luggage or to a tree?).

Automated text categorization (also known as text classification) has been around since the early 60s (Maron, 1961), but with the increasing prevalence of electronic documents, the possible applications for text categorization, as well as the possibilities for research, have multiplied. Sebastiani (2002) convincingly summarizes the research in text categorization, regrouping a fragmented field and providing a solid foundation for future researchers to build upon. While research in text classification has by no means stalled in the last decade, the definitions and methods outlined in Sebastiani's work are still very relevant today. Preferences in machine learning methods have shifted over the years, and current research often isn't averse to incorporating information sources that are external to the source documents (lexicons, knowledge bases...), yet the experimental methods have largely remained the same.

As restating what is elegantly described in Sebastiani (2002) would be redundant, we will instead focus on where our research starts to diverge from the canonical "text categorization" as he outlined it. Both the terms "text" and "category" can be interpreted in a variety of ways. A text may vary in length, meaning the goal of a text categorization task may be to attribute single words to a given category, just as it is possible to categorize sentences, paragraphs, documents or entire monographs. The term "category" is even more open to interpretation. At its most straightforward, a category can be defined as a "topic", i.e. all documents that belong to a single category are concerned with the same subject matter. It is also possible to categorize texts on a more abstract level. What if the goal is not to determine the topic of the document, but to determine the opinion the author holds regarding the topic he writes about? This is what the more specific domains of opinion mining and sentiment analysis are concerned with.

2.2 Sentiment analysis

"Sentiment analysis" and "opinion mining" (also sometimes "subjectivity analysis") are terms that can be used interchangeably (Pang & Lee, 2008). Sentiment analysis is a specific subtask within the broader field of text categorization, in that the goal is to classify a text according to the sentiment or opinion that it expresses. Popular applications of sentiment analysis are determining the sentiment expressed in user comments on online blogs (Chesley et al., 2006), the automatic classification of product reviews into positive and negative reviews (Morinaga et al., 2002; Dave et al., 2003), or determining which aspects of a product are positively or negatively reviewed (Baccianella et al., 2009).

The difference between topic classification and sentiment analysis isn't just a matter of class definitions, however. Topic classification usually comes down to determining the topic of a document using simple, low-level features such as the presence of specific keywords. For instance, to decide whether a document belongs to the *sports* or the *politics* category, a good approach would be to identify key terms occurring in the text. If the document contains words like "ball", "player" or "homerun", one might reasonably expect the text to deal with sports. If, however, words like "presidency", "liberal" or "conservative" pop up, the *politics* category might be a safer bet.

To determine the sentiment orientation (also termed *semantic* orientation) of a text, the key features are rarely related to its topic. It is possible to write about politics in a positive or negative way, just as it is possible to express a positive or a negative opinion about sports. The relevant features used to determine the sentiment orientation of a text dealing with any topic will most likely be very similar: "disappointing" is usually negative while "stunning" is positive, and the subject of the text will do little to influence this.

The relatively domain-independent nature of sentiment features is why a significant part of the research in sentiment analysis has been concerned with generating subjectivity lexicons. Words that have strong associations with positive or negative sentiment are catalogued, and optionally associated with a score reflecting their sentiment orientation. The resulting lists can then be used to gauge the sentiment of a text document, either by simply asserting the presence (or absence) of these keywords (often in combination with negation rules: "far from stunning" should not be seen as contributing to a positive sentiment), or by using these keywords as features for a more complex machine learning system. We will use a number of sentiment lexicons in our emotion classification experiments in the next chapters, which is why we summarily describe the most important lexicons in the following paragraphs.

2.2.1 Subjectivity lexicons

There are numerous lexicons for sentiment analysis ("subjectivity lexicons", as we will term them), and their creation processes can differ greatly. The most straightforward construction method is to compile the list of sentiment keywords manually. While this approach allows for total control over the contents of the lexicon, it is also very time-consuming. This problem can be alleviated using a semi-supervised approach, where a limited set of "seed" words are selected manually, but where the seed list is expanded through automatic means. It is also possible to construct lexicons fully automatically by identifying and extracting opinion-carrying words from large corpora.

As an example of the first approach, the lexicon used in the SO-CAL Semantic Orientation Calculator (Taboada et al., 2011) was constructed manually. Adjectives, verbs, nouns, and adverbs taken from a set of 400 reviews were ranked on a scale from

-5 (strongly negative) to +5 (strongly positive) by human annotators. The authors show that by using their lexicon and taking into account intensifiers (adverbs such a "slightly", "very", "extraordinarily" etc.) and negation, it is possible to calculate the polarity of an entire text with a simple rule-based system, without ever resorting to more complex classification algorithms.

Turney & Littman (2003) adopted a semi-supervised method to construct their lexicon. They used two different statistical measures for word association –Pointwise Mutual Information and Latent Semantic Analysis– to expand an original set of positive and negative words. Their methods were tested on a collection of 3,596 manually labeled nouns, adjectives, verbs and adverbs, and reached accuracies up to 95%, indicating that this approach is indeed a valid way to construct a reliable lexicon.

SentiWordNet (Baccianella et al., 2010), a sentiment lexicon based on WordNet, was also created using a semi-supervised approach. Starting from two small seeds of 7 positive and 7 negative synsets, the semantic orientation values were propagated through WordNet using the semantic relations present in the network, either keeping the same orientation in the case of synonymy-type relations, or flipping the orientation for antonymy-like links. The resulting synset lists were further expanded using a learner trained on descriptions of known synsets, and in a final pass, sentiment orientation scores were refined based on the intuition that if the gloss for a synset contains many positive terms, the synset itself is highly likely to be positive.

The approach adopted by Mohammad et al. (2009) also makes use of an initial set of seed words. Their seed set was partially made up of a pre-existing semantic orientation lexicon –the General Inquirer lexicon (Stone et al., 1966)– and partially of automatically retrieved pairs of words that fit predefined affixation patterns ("honest"-"*dis*honest", "possible"-"*im*possible", "sense"-"*non*sense"). The motivation for this second source of terms is to be found in the idea that, given a pair consisting of an explicitly marked word and its unmarked counterpart, the marked word will tend to have the negative semantic orientation (as can be easily verified in the aforementioned word pairs). Given the set of seed words, the authors set out to label words in the Macquarie Thesaurus (Bernard, 1986), assigning positive orientation if a thesaurus "paragraph" (one such paragraphs consists of a set of near-synonyms) contains more positive seed words than negative ones, and assigning negative semantic orientation if the opposite was true. The MSOL (Macquarie Semantic Orientation Lexicon) resulting from this relatively straightforward procedure appears to have a higher coverage than the aforementioned SentiWordNet, and a higher accuracy compared to other subjectivity lexicons (Mohammad et al., 2009).

Finally, it is possible to create a subjectivity lexicon in a fully unsupervised manner, deducing the semantic orientation of words from their usage in large corpora. Hatzivassiloglou & McKeown (1997) searched the Wall Street Journal corpus for

conjunctions of adjectives. Adjectives that occur conjoined will most likely carry the same polarity (in the case of the conjunctions "and" or "or"), or opposite polarities (in the case of "but"). Consider for instance *corrupt and brutal* vs. **corrupt and legitimate*. Given the fact that unmarked word forms are usually most common and the fact that the unmarked forms of a word often carry the positive connotation (knowledge which is also exploited by Mohammad et al. (2009) in the method described in the previous paragraph), it is assumed that the cluster with the highest average word frequency will be the cluster of positively oriented adjectives.

Since part of the experiments that will be described further in this work (see Chapter 3) will be carried out on Dutch data, it is also interesting to mention that similar sentiment lexicons have been constructed for Dutch (Jijkoun & Hofmann, 2009; Jijkoun et al., 2010). The creation of the DuOMAn lexicon (Jijkoun & Hofmann, 2009) was inspired by the technique used by Esuli & Sebastiani (2007), where a PageRank-like algorithm is applied to WordNet synsets to rank them according to their opinion-related properties. The authors start by automatically translating the positive and negative polarity words in the English subjectivity lexicon in OpinionFinder (Wilson et al., 2005) into Dutch. Using the relations present in Cornetto (Vossen et al., 2007), an expanded Dutch Wordnet, the initial sentiment orientation scores ($+1$ or -1, depending on the polarity in the original English lexicon) are propagated to related synsets. Synonymy relations keep the sign of the score, while antonymy relations flip it. A dampening factor is also at play, ensuring that the seed words retain more weight than words reached through semantic links. The final result of this iterative procedure is a list of words ordered by their polarity score.

Another subjectivity lexicon for Dutch was generated by Maks & Vossen (2012). Using three different corpora containing three different text types –a corpus of Dutch Wikipedia articles, a corpus of Dutch news articles, and a corpus of reader comments— subjective words are automatically extracted. An important difference compared to previous lexicons, is that the authors make the distinction between speaker/writer subjectivity ("Bush is bad for the economy.") and actor subjectivity ("Bush is angry over Obama's leaking of private conversations"). Their hypothesis is that speaker/writer subjectivity will be most prevalent in reader comments, and actor subjectivity in the news articles. Wikipedia articles serve as a dataset of neutral, objective texts. Comparing word frequencies across the three datasets allowed the authors to determine which words were most strongly associated with subjective text, and whether they were associated with speaker/writer subjectivity or actor subjectivity.

Finally, De Smedt & Daelemans (2012b) created a subjectivity lexicon of Dutch adjectives by manually annotating 1,100 adjectives according to polarity and subjectivity strength, and by automatically expanding this manually annotated set using two different unsupervised techniques: distributional extraction and Wordnet expansion. The first step relies on the fact that words that occur in similar contexts will likely

have similar meanings. After analysing the distribution of nouns and adjectives in a large corpus, for each of the 1,100 adjectives in the seed set, the adjectives that are calculated to be most similar inherit the polarity, subjectivity, and intensity of their parent. In a second step, the resulting enhanced lexicon is further expanded by traversing Cornetto (Vossen et al., 2007) relations, passing on subjectivity scores from synset to related synset. The result of this two-step process is a Dutch subjectivity lexicon of 5,407 adjectives with values for polarity, subjectivity, intensity, and reliability.

We will use both the DuOMAn lexicon by Jijkoun & Hofmann (2009) and the Pattern lexicon by De Smedt & Daelemans (2012b) in Chapter 3, where they will serve as sources for extra features for our Support Vector Machines (SVM)-based emotion classifiers.

2.2.2 Determining the sentiment of text

Armed with the aforementioned subjectivity lexicons, it seems more feasible to classify documents according to their semantic orientation. Still, there are various ways of integrating these lexicons in the classification process. Pure lexicon-based methods use only the lexicon to determine the sentiment of a given text, and no machine learning is involved (Balahur et al., 2010; Kim & Hovy, 2004; Mohammad et al., 2009; Taboada et al., 2011). A lexicon can also serve as a source of features for a machine learning algorithm (Councill et al., 2010; de Albornoz et al., 2010; Rentoumi et al., 2010; Reyes & Rosso, 2011; Wawer, 2010; Balahur, 2013), which is also the approach we will use in Chapter 3. Using subjectivity lexicons is obviously not an absolute requirement for sentiment classification, and many researchers choose to instead rely on different feature types altogether to train a machine learner (van Atteveldt et al., 2008; Pak & Paroubek, 2010; Tsur et al., 2010; Bermingham & Smeaton, 2011). We will outline only one of each type of application, as giving an overview of all experiments carried out within the domain of sentiment classification would take us much too far, but interested readers may want consult the aforementioned sources for more examples.

Lexicon-based methods

Balahur et al. (2010) realize the full potential of sentiment lexicons while attempting to determine the sentiment carried by a speaker in fragments of reported speech from newspaper articles. They make use of (combinations of) four different sentiment lexicons: WordNet Affect (Strapparava & Valitutti, 2004), SentiWordNet (Baccianella et al., 2010), MicroWNOp (Cerini et al., 2007) and their own JRC Tonality. The sentiment scores from each of these sources are remapped to a value on a scale from

−4 (highly negative) to 4 (highly positive), so as to obtain a collection of uniformly labeled terms. The overall subjectivity present in a fragment of reported speech is then simply calculated by taking the sum of the sentiment scores of the words in a window around the opinion target. One possible problem with this approach, however, is that one might end up detecting good vs. bad news instead of a positive vs. negative opinion on the news. To avoid this, the authors experimented with removing all sentiment words which were also potentially category words (relating to a news topic). For some sentiment lexicons, this resulted in a significant improvement, while for others no change was noted. Balahur et al. (2010) report that the best results were obtained using JRC Tonality and MicroWNOp with a window of 6 words around the opinion target. The authors also note that most resources did not reach baseline scores, and that the size of a lexicon is not necessarily related to its performance in sentiment classification tasks.

Combining machine learning and sentiment lexicons

As we outlined in the previous section, it is possible to use subjectivity lexicons as the only resource to determine the sentiment of a text. But why not bundle the advantages of a lexicon with the power of a machine learner and see subjectivity lexicons as an extra source of features to learn from?

An example of this is to be found in Wawer's 2010 work. They investigate the performance of machine learning algorithms (focusing specifically on Support Vector Machines) trained on features extracted from existing lexical resources. They evaluate features extracted from the General Inquirer dictionary (Stone et al., 1966), the Dictionary of Affect in Language (DAL) (Whissell, 1989) and SentiWordNet (Baccianella et al., 2010). The authors evaluate the different lexicons and pay special attention to the interaction between the size of the feature set and the C parameter in the SVM classifier. The authors' conclusions seem to illustrate that using only the terms or categories present in a lexicon is not a valid alternative to using bag-of-words-like features, as a classifier trained with token or synset unigrams (even without disambiguation) performed significantly better than any of the classifiers that used only lexicon features. Noticing that SentiWordNet is consistently outclassed by other lexicons, Wawer hypothesize that assigning subjectivity scores on the synset level may not be ideal, since different senses of the same synset may carry different connotations (e.g. "a quack" vs. "medically unqualified").

Pure machine learning methods

It is perfectly possible to perform sentiment classification without resorting to lexicons, and to rely instead only on machine learning and features extracted from the

documents themselves. Bermingham & Smeaton (2011), for instance, after having determined in previous research (Bermingham & Smeaton, 2010) that using supervised learners for sentiment classification worked better than using sentiment lexicons, decide to adopt the same approach to model sentiment in tweets concerning the Irish general elections of 2011, aiming to use this information to predict election results. Starting from a set of manually annotated tweets relating to parties and candidates for the election, they trained a task-specific classifier which discriminates between positive, neutral or negative tweets. The authors report that the initial performance of a simple SVM or Naïve Bayes classifier was disappointing, mostly due to the strong class imbalance (the positive class contained only 12% of all tweets). They managed to mitigate this problem by using their Multinomial Naïve Bayes classifier in a boosting setup (Adaboost), which increases the weight of instances from minority classes to boost their performance. The tweets that were labelled as being positive by the classifier turned out to be useful for the prediction of the election results.

Pak & Paroubek (2010) also turn to Twitter for sentiment analysis, and they too adopt supervised learning for sentiment classification. They start by gathering a large corpus of tweets that is automatically split into three subsets: tweets containing positive emotion, tweets containing negative emotion, and neutral tweets not containing either. This automatic division of the data happens quite simply by identifying emoticons. If a positive emoticon is identified (*:)*, *:-)*, *=)*, *:D*...), the tweet is allocated to the positive subset. If a negative emoticon is found (*:(*, *:-(*, *=(*, *;(*...), the tweet ends up in the negative subset. Objective tweets were gathered from the Twitter accounts of popular newspapers or magazines. Analysis of the corpus showed that part of speech tags have different distributions for subjective tweets and objective tweets, making them valid indicators of subjective text. Strong pointers for tweets carrying sentiment include interjections, personal pronouns and the superlative forms of adjectives, whereas objective tweets contain more common and proper nouns.

2.3 Emotion classification

Whereas sentiment classification focuses on placing a document in two or more classes positioned between a "positive" and a "negative" pole, emotion classification takes a text document (the term "document" can, again, refer to a text of any length, ranging from an entire monograph to a single phrase) and places it into one or several emotion classes. What these classes are, however, depends on the underlying emotion framework one decides to use. We will outline the most important emotion classification frameworks in the following paragraphs.

2.3.1 Emotion frameworks

Emotions are, by definition, highly subjective. Consequently, researchers in the fields of psychology and emotion science have long debated the way in which emotions are to be defined (Gendron & Barrett, 2009). Many different frameworks exist, and it is not our goal to list or describe them all. However, it is important to be aware of the fact that the emotion frameworks can be grouped into two sets according to their defining viewpoints: *discrete* frameworks and *dimensional* frameworks (Zachar & Ellis, 2012). This distinction is relevant in the context of this thesis, since in our experiments in Chapter 3, we will use a dimensional framework of emotion, whereas the other chapters (Chapters 4, 5, and 6) will focus on discrete emotion frameworks.

Discrete frameworks

Proponents of the discrete frameworks see emotions as fundamentally different and separate. According to them, there is a set of "basic" emotions which have a psychological and biological basis (Ekman, 1971). These emotions are often considered universal, and can be distinguished based on our physiological reactions when we experience them. The most popular set of "basic" emotions are without a doubt Ekman's emotions (Ekman, 1971): *anger*, *disgust*, *fear*, *happiness*, *sadness* and *surprise*. Ekman determined that these emotions must be "basic" after discovering that the members of an isolated, preliterate tribe in Papua New Guinea appeared to associate facial expressions to emotion-provoking situations in a similar way as we do in Western cultures. Since this tribe could not have been influenced by our culture in any way, he concluded that there must be universal emotions which are innate and not the result of cultural influences. Ekman later added a number of emotions that are not reflected through facial expressions to his initial list, but much of the literature in emotion research and computational linguistics alike prefers to stick to the aforementioned "big six". Even among proponents of a discrete emotion framework, there is disagreement on whether these emotions are a product of *nature* (evolutionary psychology) or *nurture* (social constructionism). Some even argue that we have a limited set of innate emotions which are molded and expanded by our interactions with others, thus reconciling the "nature" and "nurture" viewpoints. (Prinz, 2004)

Regardless of the heated psychological debate, discrete emotion classification frameworks are popular, and especially so among text categorization researchers. Automatically classifying text documents according to their emotional contents is a compelling task, but having a limited, clearly defined set of possible emotion classes is clearly a blessing, as it makes the task easier to tackle. A significant part of emotion classification research adopts this discrete approach, and the most popular

framework in this respect is indeed Ekman's set of six basic emotions. Among the emotion classification papers we will discuss, a great many classify documents into the "big six" (Bellegarda, 2010; Ghazi et al., 2010; Strapparava & Mihalcea, 2008; Purver & Battersby, 2012; Strapparava et al., 2007; Ghazi et al., 2012; Strapparava & Mihalcea, 2007; Aman & Szpakowicz, 2007; Mohammad, 2012b).

Alm et al. (2005) slightly adapt the Ekman emotions in that they divide *surprise* into *positive surprise* and *negative surprise*. Tokuhisa et al. (2008) use a set of 10 discrete emotions. *Happiness*, *fear*, *sadness* and *anger* overlap with the "big six", but *surprise* and *disgust* are nowhere to be found, instead being replaced by *pleasantness*, *unpleasantness*, *relief*, *disappointment*, *loneliness* and *anxiety*. It is not immediately apparent why the authors chose these emotion classes for their experiments.

Lin & Chen (2008) let their class system be determined by their data. They use news articles from the Chinese Yahoo! Komo News service. This website has the particuliarity that users can vote on one of eight possible emotions (*heartwarming*, *happy*, *sad*, *surprising*, *angry*, *boring*, *awesome*, *useful*) according to how the news article makes them feel. Thanks to this rating feature, the Yahoo! News site is actually a large, crowd-annotated dataset.

Roberts et al. (2012) supplement the six basic emotions with *love*, hypothesizing that in their informal data, which are comprised of tweets reflecting on a FIFA World Cup, *love* would be a common emotion.

De Albornoz et al.'s framework (2012) is a combination of the discrete models by Arnold (1960), Plutchik (1980) and Parrott (2001). Whether this is acceptable from a psychological point of view is debatable, but the authors nevertheless obtain a set of 16 possible emotions. Expanding these emotions so each emotion has an antonym, and then again reducing the framework when some emotions weren't present in any of their annotated data, de Albornoz et al. were left with 14 emotional categories, which are linked pair-wise with antonymy relations.

Inkpen et al. (2009), Jung et al. (2006) and Mishne (2005) do not perform emotion classification in the strictest sense, but instead attempt to identify "moods" in text. That mood and emotion are not entirely unrelated, though, is immediately apparent when we see that the LiveJournal[1] hierarchy of moods, which is used in all three papers, has many of the "basic" emotions at its top level. Jung et al. (2006) only use four of these top-level classes and classify blog posts into *happy*, *sad*, *angry* or *fearful* texts. Inkpen et al. (2009) use all top-level moods and include a variety of moods that do not overlap with emotion frameworks from psychology research. These include moods such as *confused*, *determined*, *enthralled* or even *working*. As far as we know, LiveJournal's moods have no basis in any psychological processes, but it is a convenient source of crowd-annotated data to use in classification experiments.

[1] http://www.livejournal.com – Last visited on July 17th, 2014

Finally, in the context of a shared task on emotion classification in suicide notes (Pestian et al., 2012), participating teams were instructed to classify individual sentences into one or several emotion classes. The possible classes were *hopelessness*, *love*, *guilt*, *blame*, *thankfulness*, *anger*, *sorrow*, *hopefulness*, *fear*, *happiness/peacefulness*, *pride*, *forgiveness*, and somewhat more controversially, *instructions*, *information* and *abuse*. These emotion classes cannot immediately be linked to any existing emotion classification frameworks, but they are instead intimately linked to the emotions one tends to find in the rhetoric of individuals with suicidal thoughts. We will discuss the technical details of the participants' approaches to emotion detection in Chapter 4, which deals specifically with this shared task.

Dimensional frameworks

The thesis that our emotions are distinct and separable is a daring one. Not much in our biology is discrete, and many researchers argue that the same applies to our psychology, and to the way emotions function. The dimensional models argue that emotions are the surface realizations of two or more underlying dimensions. What exactly these dimension are, however, is again the subject of much debate (Rubin & Talarico, 2009).

A popular dimensional model of emotion is the **circumplex model** (Russell, 1980; Feldman Barrett & Russell, 1998; Russell & Barrett, 1999). The two dimensions that define the circumplex model are *valence* and *activation*. *Valence* (also referred to as the *pleasure* dimension) refers to whether an emotion is positive or negative. *Activation* (or *arousal*) refers to the intensity with which the emotion is experienced or expressed. Both dimensions are independent, in that the valence of an emotion does not affect its activation and vice versa. A positive emotion can be highly activated, or it can be low on the activation axis ("deactivated"), and deactivated emotions can be positive or negative. The valence and activation dimensions in themselves are bipolar, meaning that if an emotion has a highly positive valence, it cannot have a high negative valence (Russell, 1980; Russell & Carroll, 1999). Russell & Barrett (1999) argue that any "core" emotion can be described using a combination of these dimensions. Emotionally charged words can thus be assigned valence and activation values, and they naturally seem to form a circle or ring in the two-dimensional emotional space, hence the term *circumplex* model (see Figure 2.1).

The circumplex model is not the only dimensional model of emotion. Bradley et al. (1992) attempted to link the strength of an emotional response when first encountering an object to the long term memory recall of the object. Subjects were shown pictures which caused a certain emotional response, and were asked to recall these picture at a later date. The authors found that there was a significant correlation between recall and the *arousal/activation* dimension, but that the *pleasure/valence* dimension had no effect on long-term memory performance. The **vector model**

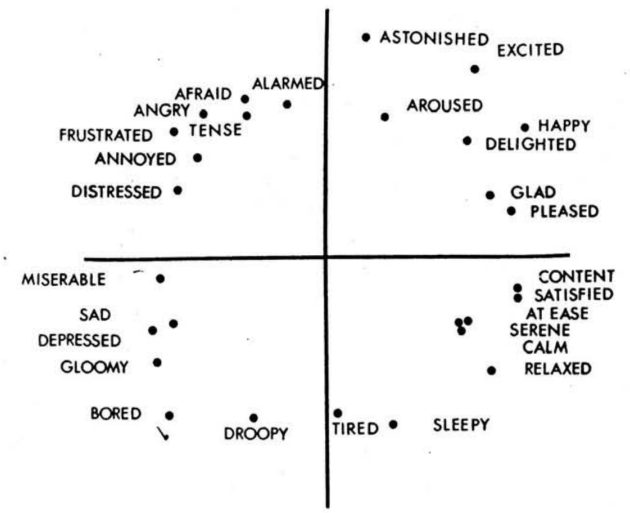

Figure 2.1: The positions of 28 emotion words in a two-dimensional space defined by the pleasure (horizontal axis) and arousal (vertical axis) dimensions. Reproduced from Russell (1980).

(see Figure 2.2) reflects these findings, in that the underlying dimension of *arousal* determines the strength of a vector, while *valence* simply determines its direction. An important practical difference between the vector and the circumplex model, is that the vector model does not allow for entities that are rated high on *arousal* but neutral in *valence*. The circumplex model on the other hand, needs these data points to complete the "circle" that gives the model its name. Rubin & Talarico (2009) had participants rate emotion related stimuli on scales for *valence* and *intensity* (or *arousal*), and attempted to fit the circumplex and vector models on this data. Interestingly, they were unable to find high-*intensity* neutral words, which gives credit to the vector model over the circumplex model when dealing with textual data. They further showed that the words in the ANEW lexicon (Bradley & Lang (1999b), also see section 2.3.2) which were labeled as having neutral *valence* but high *intensity*, were in fact not truly neutral, but their neutral values were an average over different, compensating positive and negative ratings.

Another well-respected model is the **PANA model** (Watson & Tellegen, 1985; Watson et al., 1988; Watson & Clark, 1999). PANA stands for "Positive Activation - Negative Activation", and it differs from the vector and circumplex models in the fact that the two defining dimensions (*positive* and *negative activation*) are unipolar, i.e. they can be measure independently, and a high PA value does not necessarily imply a low NA value. As was the case for the vector model, the PANA model suggests that emotions that are neutral in valence tend to be lowly activated, and that

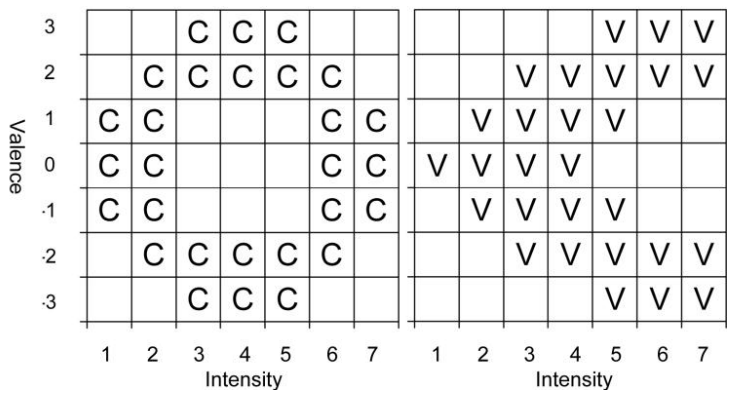

Figure 2.2: A graphical representation of where emotional words should be situated according to the circumplex model (left) and according to the vector model (right). Reproduced and adapted from Rubin & Talarico (2009).

highly activated neutral emotions are unlikely.

It is also not uncommon to represent emotions in terms of not two, but three dimensions. Often referred to as the **PAD model**, the *pleasure-arousal-dominance* model posits the existence of a third defining emotional dimension in addition to *pleasure* and *arousal*, namely the *dominance* dimension (Wundt, 1896; Osgood et al., 1957; Osgood, 1969; Russell & Mehrabian, 1977; Bradley & Lang, 1994; Scherer et al., 2006). This dimension refers to the controlling or submissive nature of an emotion, and is interesting for our purposes, as it will reoccur when we attempt to classify dominance and affinity in Chapter 3.

Despite the heated scientific discussions relating to dimensional emotion frameworks and the many different realizations proposed by psychologists, dimensional models are relatively unpopular with computational linguists. Intuitively, this is easy to explain. Text categorization, by definition, deals with categories. To be able to predict the emotionality of a text on a continuous scale (or on multiple continuous scales, as is the case with two- or three-dimensional frameworks) would require the use of significantly different techniques. As a consequence, a majority of the work in emotion classification prefers to focus on discrete frameworks.

It is also possible to discretize the space defined by a continuous model, thus reducing the more subtle dimensional problem to a "regular" discrete classification problem. Van Zaanen & Kanters (2010) use Thayer's model of emotion (1990), which, much like the circumplex model, maps emotions in a two-dimensional space defined by *valence* and *arousal*. The authors partition the space into several subsets of classes (16 fine-grained classes, 4 quadrants, 4 gradations on the *valence* or *arousal* axis).
Another interesting case is presented by Kim et al. (2010), who use dimensional in-

formation to classify sentences into four discrete emotion categories (*anger*, *fear*, *joy* and *sadness*). They use words from the ANEW lexicon (Bradley & Lang, 1999b), which are labeled with positions in the three-dimensional *valence-activation-dominance* space, to determine the position of an input sentence compared to the position of prototypical emotion words. The final evaluation, however, still happens on the basis of discrete classes.
Rubin et al. (2004) are also in favor of discretizing a continuous model. They argue that the PANA model (Watson & Tellegen, 1985), which is defined by the *positive affect* and *negative affect* dimensions, is a useful model for NLP applications when used in the form of eight distinct octants.

2.3.2 Emotion lexicons

Before we turn to a closer examination of some of the research we mentioned in passing in the previous paragraphs, we turn to a short description of the most important emotion lexicons. Much like sentiment lexicons (described in Section 2.2.1), these emotion lexicons are lists of words which have been labeled according to their emotional connotation. The label can simply be an emotion category the word is thought to belong to, or it can be a value representing the strength of a given emotional dimension reflected in the word. We will use emotion lexicons extensively in Chapters 5 and 6.

One of the most well-known and intensively used emotion lexicons is Bradley & Lang (1999b)'s Affective Norms for English Words (ANEW) word list. This lexicon was built to serve as a textual equivalent of the existing International Affective Picture System (IAPS) (Lang et al., 1999) and International Affective Digitized Sounds (IADS) (Bradley & Lang, 1999a) databases, which respectively link images and sounds to emotional values in a three-dimensional *valence-activation-dominance* framework (Osgood et al., 1957). ANEW does the same thing, but for a set of 1,034 English words. The words were manually annotated on the three dimensions of the framework.

Stevenson et al. (2007) added information regarding discrete emotional categories to the ANEW lexicon. They asked annotators to determine for each word in ANEW how strongly it belonged to one of five discrete emotion classes, using a scale from 1 (*not at all*) to 5 (*extremely*). These classes were *happiness*, *anger*, *sadness*, *fear*, *disgust*. Note that these are five out of the six Ekman emotions (Ekman, 1971), and that only *surprise* is missing.

Warriner et al. (2013) extended the ANEW lexicon to include close to 14,000 words, which were manually annotated through the Amazon Mechanical Turk service[2].

[2]See https://www.mturk.com/mturk/welcome - Last accessed on May 18th, 2014

Moors et al. (2012) generated an ANEW-like lexicon for Dutch words. A set of 4,300 words were annotated by Flemish and Dutch students on, again, *valence*, *activation* and *dominance* scales.

Not all lexicons are based on ANEW and its three-dimensional model. The NRC-10 lexicon, for instance, is a large corpus of 14,000 words labeled with 8 emotion labels (the "big six", plus *trust* and *anticipation*) and positive and negative sentiment. The corpus was built by Mohammad & Turney (2012) through crowdsourced manual annotation (using Amazon Mechanical Turk).

De Albornoz et al. (2012) constructed SentiSense, which is a "concept-based affective lexicon". SentiSense attaches one of 13 possible emotion labels to senses in WordNet. Using WordNet has a few important advantages. First of all, since emotion labels are attached to senses as opposed to words, it is possible to avoid the problem of homographs. It if for instance not plausible that the word "tear" in the sense of "There's a small *tear* in my shirt" would have the same emotional value as the same word in the sense of "He wiped a *tear* off his cheek", which is a distinction that cannot be made without the word sense information present in WordNet. Secondly, annotating emotions directly on WordNet makes it possible to use all WordNet relations, such as synonymy, antonymy, hyperonymy etc.

Finally, the WordNet Affect lexicon (Strapparava & Valitutti, 2004) is also an extended version of WordNet (more specifically, of WordNet Domains[3]). The construction of the lexicon was bootstrapped manually (by annotating a set seed of word senses with emotion labels), and expanded automatically through WordNet relations.

We will use the NRC lexicon (Mohammad & Turney, 2012) in our experiments in Chapters 5 and 6. We will also use a portion of the LIWC lexicon (Pennebaker et al., 2001), which is a lexicon on terms arranged by category, including categories relating to psychological –and emotional– processes.

2.3.3 Identifying emotion in text

We will now conclude this section on related research by reviewing some concrete examples of existing research into automatic emotion classification of text.

An important benchmark in emotion classification was the SemEval-2007 task on affective text organized by Strapparava et al. (2007). The task consisted of the classification of 1,000 headlines from various news sources (Google News[4], CNN, New York Times, BBC News) according to the six Ekman emotions (Ekman, 1971) or according to valence (positive vs. negative). The former task is of special interest to

[3]See http://wndomains.fbk.eu/wnaffect.html - Last accessed on March 22nd, 2013
[4]https://news.google.com/ - Last accessed on March 26th, 2013

us. Participants were asked to construct a system that would assign a score between 0 and 100 for each of the possible emotion classes to each test headline. The predicted scores were evaluated in a fine-grained and in a coarse-grained way. The fine-grained evaluation used Pearson correlation with the gold score, and the coarse-grained evaluation set a threshold for each emotion ($[0, 50)$ = absent, $[50, 100]$ = present), thus making it possible to use the more common accuracy, precision and recall measures. The gold scores were determined by taking the average of the scores assigned by six different annotators. The inter-annotator agreement was measured using the fine-grained Pearson correlation, taking the average of the correlations between each of the annotators and their five peers. The reported agreement scores range from 36.07 (*surprise*) to 68.19 (*sadness*). Five teams participated in the task at the time, three of which built a system to identify emotions (as opposed to only determining valence) (Chaumartin, 2007; Katz et al., 2007; Kozareva et al., 2007).

Chaumartin (2007)'s UPAR7 system was in essence rule based. In a first step, the capitalization of words in the news headlines was normalized, i.e. only named entities retained capitalization. This made it possible to correctly parse the headlines using an automated parser. The resulting combinations of words and part-of-speech tags allowed the authors to look up each word in WordNet, and to retrieve emotion ratings from an expanded version of WordNet Affect and SentiWordNet. Since word-sense disambiguation is almost impossible given the inexistent context of news headlines, the rating for each word was a weighted average of emotion ratings for each of the possible senses for the word. Since the agent or cause of an emotion can mean the difference between emotions such as *sadness* and *anger* (when the cause of your misfortune is another human being, *anger* is a more likely response, while *sadness* is more logical with natural phenomena), the authors also detected the presence of nouns relating to either human intention or natural factors, and adjusted emotion scores accordingly. The total scores for the complete news heading was calculated by attributing a significantly higher weight to what was identified to be the "main" word (determined using the dependency tree from the parser output) and by detecting verbs that signify an increase or a decrease ("reduce risk" is positive, despite the connotation of "risk"). Chaumartin's system achieves more than respectable results, finishing first on 4 out of 6 emotions according to fine-grained evaluation. Fine-grained scores range from 12.85 (*disgust*) to 44.92 (*fear*). Coarse-grained results seem more modest, with UPAR7 only winning in the *sadness* category.

Katz et al. (2007) used a supervised approach for their SWAT system. They expanded the initial development set of headlines (which contains only 250 instances) with an additional, manually annotated 1,000 headlines. They report on the agreement between the annotators of the original development data and their own annotators, which range from 0.19 (*surprise* and *fear*) to 0.81 (*sadness*). They also make two interesting points regarding the difficulty of the annotation task. First of all, the emotions experienced by the entities in the headline may not be the same emotion

experienced by the reader (e.g. "White House surprised at reaction to attorney firings"). Secondly, depending on the annotator's opinions and beliefs, their reaction may differ greatly from that of others (e.g. "Germany defeats Argentina in World Cup Final").
The extended training set was lemmatized, and the words in the training data were given emotion scores based on their co-occurrence with emotion labels. Synonyms and antonyms were added using a thesaurus. The resulting "emotion lexicon" was used to assign scores to the test headlines, averaging the emotion scores of the words that were present in the lexicon. SWAT took first place in the fine-grained evaluation only for the *disgust* class, and had the highest F-score only for *joy*.

Kozareva et al. (2007) built the UA system, which followed an approach similar to the one used by Hatzivassiloglou & McKeown (1997), who argued that words with similar valence occur together more often, making it possible to automatically expand lists of subjective terms. Kozareva et al. calculate Pointwise Mutual Information scores for each emotion label and the content words in a headline using the web as a reference corpus. Formally, the Pointwise Mutual Information score for emotion e is calculated as $log_2 \frac{hits(e,cw)}{hits(e)hits(cw)}$, where cw stands for the content words in the current headline, and $hits$ are the number of hits returned by a search engine. The resulting scores on the test set were normalized between 0 and 100 and evaluated using fine-grained Pearson correlation and accuracy, precision and recall, as was the case for the other participants. This corpus-based approach seems to work reasonably well given a coarse-grained evaluation, since the UA system has the best F-scores for 3 out of 6 emotions. They don't do so well when evaluated using Pearson correlation, where they're consistently outranked by their peers.

Another shared task that can serve as a useful benchmark for emotion classification was the 2011 Medical NLP Challenge, or more specifically, Track 2 of this challenge, which involved the identification of emotions in suicide notes (Pestian et al., 2012). The dataset from this task will form the basis of a case study in Chapter 4, and as such, related work on the 2011 Medical NLP Challenge will be described in Section 4.3.

Suicide notes and news headlines are not the only text types that can be analyzed for their emotional contents. Blogs (including Microblogs) are another sources of textual information that has garnered much attention in the field of emotion classification recently. Both blogs and microblogs are often used to share personal opinions — though the length of the message varies—, and as such they are valuable source of emotional text.

As we mentioned in Section 2.3.1, Mishne (2005) compiled a dataset of blog posts mined from LiveJournal[5]. LiveJournal is an especially interesting source of emotional

[5]http://www.livejournal.com – Last visited on April 2nd, 2013

text, since its very concept —writing an online journal— encourages the creation of emotionally-charged material. Equally interestingly, LiveJournal allows users to label their posts according to a predefined set of moods, which implies that a significant part of the posts on LiveJournal is pre-labeled. Strapparava & Mihalcea (2008), for instance, use a subset of the blog dataset by Mishne (2005) —its mood labels restricted to Ekman's "big six"— to train a Naïve Bayes classifier to categorize news headers from the SemEval 2007 shared task (Strapparava et al., 2007). The blog-based classifier is compared to a number of other methods (an approach based on WordNet Affect, and a number of Latent Semantic Analysis-based approaches), and the authors note that it is the best performer for those emotions that are strongly represented in the training data, whereas the other emotions were identified best using an Latent Semantic Analysis-based approach.

Aman & Szpakowicz (2007) constructed a smaller-scale emotion corpus of blog posts, mined from the web using a seed of emotional keywords. They then used this corpus to train and evaluate SVM classifiers trained with bag-of-words features, features from WordNet Affect (Strapparava & Valitutti, 2004), and emotional terms extracted from Roget's Thesaurus (1852). They conclude that the best performing system is a system that uses features from all of these sources.
Ghazi et al. (2010) use this same blog dataset to show that it is worthwhile to classify emotions hierarchically. They propose a two-level classification scheme (the first level separates emotional documents from neutral ones, whereas the second level determines the exact emotion of the emotional documents) and a three-level model (neutral vs. emotive; positive vs. negative emotion; specific emotion classes). They conclude that conceptualizing and implementing emotion classification as a hierarchical task can result in improvements in classification performance.
The NRC-10 lexicon (Mohammad (2012b), see Section 2.3.2) was also evaluated on Aman's blog dataset. Mohammad found that their emotion lexicon managed to produce respectable results across domains, whereas a classifier trained on n-grams from a given source domain will lose most of its relevance when applied to data from a different target domain. The corpus by Aman & Szpakowicz (2007) will be one of the corpora we use in Chapters 5 and 6, and we will describe the corpus in more detail when appropriate. In these chapters, we will also form our own opinion on the portability of emotion features in cross-domain experiments, and on the effectiveness of hierarchical emotion classification.

Microblogs are also an increasingly popular source of textual data for automated analysis. All over the world, people are generating a constant stream of short bits of text, often expressing thoughts and opinions, which can be highly emotionally charged. Twitter[6] is without a doubt the most popular example of the medium.

[6]http://www.twitter.com – Last accessed February 3rd, 2014

Twitter users generate 5,700 tweets per second, on average[7], so having a system that can analyze tweets automatically would be immensely valuable, as the medium has long outgrown the limits of what can be examined manually.

Mohammad (2012a) created a corpus of tweets (the Twitter Emotion Corpus) by relying on emotion hashtags assigned by Twitter users (a technique also followed by Wang et al. (2012)). Tweets with hashtags corresponding to the six Ekman emotions (*#anger*, *#disgust*, *#fear*, *#joy*, *#sadness*, and *#surprise*) were mined straight from Twitter. The reliability of a dataset where these hashtags are used as class labels is unknown, but the authors evaluate classifiers based on this Twitter data on a known dataset (the news headlines from the SemEval2007 Shared task by Strapparava et al. (2007)), positing that performance similar to previous work (Chaffar & Inkpen, 2011) indicates that the TEC dataset is sufficiently consistent. They also show that it is possible to improve on in-domain classification performance on the SemEval2007 dataset by adding the TEC data to the training dataset, and by performing a simple domain adaptation technique (Daume III, 2007)[8], which further supports the hypothesis that the user-chosen emotion hashtags are indeed reliable.

Where Mohammad used emotion hashtags as de facto labels for their Twitter data, Qadir & Riloff (2013) take things a step further, and attempt to bootstrap sets of emotion-related hashtags, which can then serve as the basis for new emotion classifiers. Starting from a seed set of manually chosen hashtags, they mine Twitter for data to train a logistic regression classifier. This classifier is then applied to new tweets, and the hashtags from the tweets that were classified with the highest confidence are added to the set of corresponding emotion hashtags. Through this iterative process, Qadir & Riloff end up with a relatively large set of tags for each emotion, and they show that combining conventional n-gram-based classifiers with list lookup in their bootstrapped hashtag lexicon significantly improves classification performance.

It is clear that the automatic emotion classification of text has many potential applications, and that the research community is still hard at work to find reliable tools, methods and datasets to work with. In the following two chapters, we will focus on two case studies. In Chapter 3, we will attempt to build a system that can classify Dutch conversations from a business context according to Leary's Rose, a model for interpersonal communication. Chapter 4 will focus on the 2011 Medical NLP Challenge task (Pestian et al., 2012), which involved the automatic identification of emotions in suicide notes.

[7] https://blog.twitter.com/2013/new-tweets-per-second-record-and-how – Last visited on July 17th, 2014

[8] The domain adaptation technique by Daume III (2007) will also be used in Chapter 6, though on different datasets.

Chapter 3

Case study: deLearyous

In this chapter and in Chapter 4, we will tackle two well-defined emotion classification tasks. We will walk the reader through the entire process, starting with a description of the data, following up with the instance creation steps, the training and optimization of the machine learner, and the evaluation of the resulting classification models. Both case studies have their unique features and eccentricities. The first case study, using the *deLearyous* dataset, is unique in that we attempt to classify sentences in a conversation between two parties according to a framework for interpersonal communication. The conversations in the deLearyous dataset are in Dutch, but the techniques we will describe in this chapter are applicable to any language. To our knowledge, this is the first time that anyone has ever tried to automatically classify text according to Leary's Rose (see Section 3.1.1), and we hope this chapter will provide inspiration and pointers for future, similar tasks.

3.1 Task description

The deLearyous dataset was created in the context of the eponymous deLearyous project[1], during which a serious game was developed that allowed users to converse with a virtual character using written natural language. Conversing with these virtual agents would help players learn about behavioral patterns and communicative roles. As a proof-of-concept, one scenario was fully implemented, meaning the the virtual agent (the AI) was capable of conversing with the player as long as the player stayed within the confines of this well-defined conversational topic. The deLearyous dataset

[1]See http://delearyous.groept.be (last accessed on July 10th, 2013) for further information concerning this IWT-TETRA project. deLearyous was a collaboration between Groep T's e-media Research Lab, the CLiPS Research Center at the University of Antwerp, and Opikanoba.

contains all training materials used to build the virtual agent, and to build the classifier that would interpret the player's natural language text input according to the emotional stance they adopt.

The proof-of-concept scenario was a conversation between an employer (played by the trainee) and his employee (the AI), after the company revoked free parking rights. There were several versions of this scenario, depending on the initial state of mind of the virtual character. The AI could be angry about this situation, in which case the player would be asked to try to calm their "employee" down. The AI could appear impassive, and the player would be instructed to get the virtual character to speak its mind. In another version, the AI would somewhat inappropriately try to impose its own solution to the problem, and it would be up to the player to firmly put it in its place.

Of course, the evolutions in the virtual character's state of mind in response to the player's input can't simply happen at random. There is an underlying framework that justifies these interactions, and it is about exactly this framework that the application is supposed to teach the trainees. The underlying model of interpersonal communication that we developed the application for —and the origin of the name deLearyous— is Leary's Rose (Leary, 1957).

3.1.1 Leary's Rose

Leary's Rose, also known as the Interpersonal Circumplex, takes its name from Timothy Leary, who devised it in 1957. The Rose shares many similarities to the dimensional emotion frameworks we described in Section 2.3.1. Much like Russell's circumplex model (Russell, 1980), the framework is two dimensional and takes on a circular shape (Figure 3.1). The *valence*-dimension is replaced with the closely-related *affinity* dimension, and instead of the *arousal*-dimension, the Interpersonal Circumplex's vertical dimension is defined by *dominance*, which brings it closer to Osgood et al.'s three-dimensional model of affect (Osgood et al., 1957). According to Leary, the space defined by the *affinity* and *dominance* dimensions is ideal to describe and analyze the emotions of participants in a conversation.

As figure 3.1 shows, the horizontal and vertical axes divide the Circumplex into four quadrants. Each quadrant is again divided into two octants. These octants should be sufficient to describe the dynamics at play in interpersonal communication.

Other than a descriptive function, however, Leary's Rose also has a predictive function. Based on one participants' position on the Rose, it is possible to anticipate the position the conversation partner will adopt in response.
There are two main mechanisms at play in the Interpersonal Circumplex: positions on the *affinity*-axis evoke a similar reaction, while positions on the *dominance*-axis

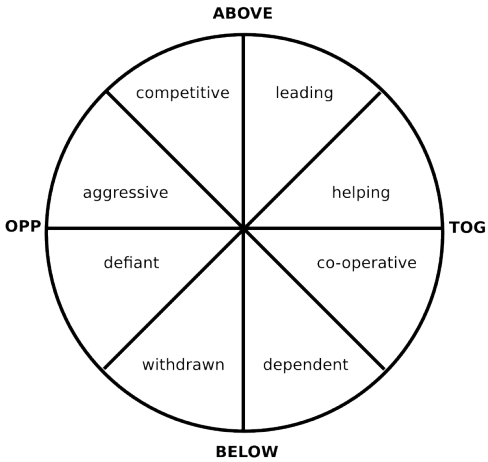

Figure 3.1: Leary's Rose (Leary, 1957)

evoke complementary reactions (see Figure 3.2). In practice, this means that someone who will act in a friendly way will most likely see their partner react in a similarly friendly manner, while rude or unfriendly statements will engender similarly unfriendly reactions. At the same time, dominant behavior solicits an instinctive submissive reaction from the conversation partner, while more docile behavior from the speaker will encourage the listener to take the lead. These dynamics make it possible to predict the most likely response of the listener to any way the speaker positions themselves in the Rose's two-dimensional space. We use this mechanic to steer the virtual character's state of mind.

Before we can steer the AI into the direction prescribed by Leary's Rose, we have to know the state of mind of the player, and this is where automatic emotion classification comes in. Given only the user's textual input and the conversation history, we attempt to position the player on the Interpersonal Circumplex.

The following section will describe the process behind this automatic classification task, starting with a description of the dataset (Section 3.2.1), followed by an explanation of the instance creation procedure (Section 3.2.2), and finally a description of the training and testing of the machine learning algorithms (Section 3.2.3).

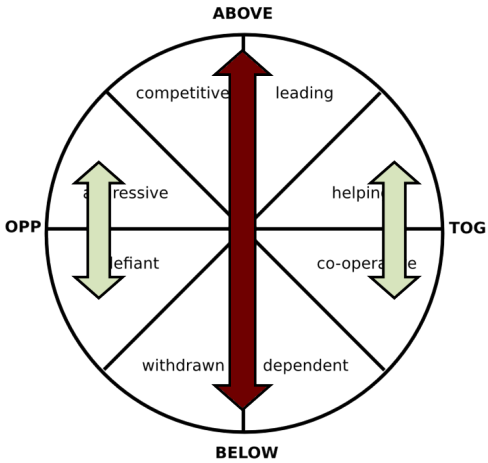

Figure 3.2: Leary's Rose and the two major dynamics at play in interpersonal communication: the complementarity of dominance, and the similarity of affinity reactions.

3.2 Classification procedure

3.2.1 Data description

The deLearyous dataset consists of 11 conversations between two parties on the topic agreed upon for the deLearyous proof-of-concept application. The scenario for the proof-of-concept was a conversation between employer (human player) and employee (virtual character) after the announcement that the company would revoke free parking rights. The data was collected in a Wizard-of-Oz setup, meaning the virtual character was replaced with a professional actor. Player and actor communicated with the same restrictions as those that would apply in the final application, i.e. the player could only send text, while the AI/actor could react to this textual input with both audio and video.

Eleven different conversations were played out, the employee starting from varying positions on Leary's Rose, and the conversations were entirely logged and transcribed. The result is a dataset of $1,143$ sentences (or $12,277$ words) in 812 turns, with conversations ranging from short 23-sentence conversations to longer conversations of up to 187 sentences (or in terms of turns, from 19 to 115 turns). The relatively large variance in conversation length can be explained by the fact that participants in the Wizard-of-Oz tests were not asked to respect a time limit, but were instead encouraged to take the conversation to its natural conclusion.

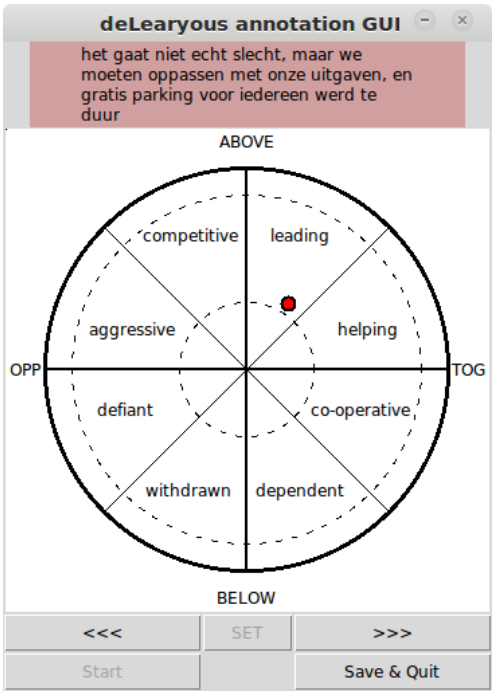

Figure 3.3: GUI for annotation of deLearyous data

The conversations were subsequently annotated. Initially, four annotators annotated non-overlapping groups of conversations, attributing a position on the Interpersonal Circumplex to each of the sentences in the conversation. To facilitate and streamline the annotation process, a tailor-made graphical interface was built (see Figure 3.3), which the annotators had to make use of. The sentences in a conversation appeared in order, and were marked for the current speaker (employer or employee). It was possible to look back at and correct previous annotations if necessary. To ensure agreement in the thought processes of the annotators, they were asked to determine the position of the sentence on Leary's Rose using the following series of questions:

1. Is the current speaker acting dominantly or submissively towards his interlocutor?
   Determines if the sentence is to be placed on the upper or the lower half of the Rose.

2. Is the speaker attempting to co-operate with the listener, or are they instead only trying to further their own goals?

Determines if the sentence is to be placed on the left or the right half of the Rose, and thus locks in the quadrant.

3. Which of the two aforementioned aspects is the strongest? Dominance or affinity?
   Determines the exact octant the sentence should be placed in.

4. How strongly is the emotion expressed?
   Determines the distance from the origin. Strongly expressed emotions will be placed further from the origin. Sentences with negligible emotional contents should be placed within the inner ring, while exceptionally strong expressions of emotion should be placed in the outer ring.

To evaluate inter-annotator agreement, one randomly chosen conversation was annotated by three annotators. The agreement was calculated on both the quadrant level (each quadrant of Leary's Rose is considered as one class) and on the octant level. The result of these calculations can be found in Table 3.1. The agreement is reported in terms of Fleiss' κ, which allows one to measure agreement scores with more than two annotators (Fleiss, 1971). Even though there are no firm guidelines as to how one should interpret κ scores, scores between 0.20 and 0.40 are usually taken to indicate "fair agreement" while lower scores indicate "low agreement". We also report on pairwise overlaps between annotators in Tables 3.2 and 3.3.

# of classes	κ
4	0.28
8	0.19

Table 3.1: Inter-annotator agreement in terms of Fleiss' κ on the Wizard-of-Oz dataset, 3 annotators

	annotator 1	annotator 2	annotator 3
annotator 1	1.00		
annotator 2	0.36	1.00	
annotator 3	0.47	0.46	1.00
average			
	0.43		

Table 3.2: Pairwise overlap between annotators on the Wizard-of-Oz dataset, quadrant level

It is immediately clear that the annotation of text on Leary's Rose is an extremely difficult task. Our three annotators, who were familiar with and had experience with Leary's Rose, find themselves disagreeing on the emotional contents of the sentences in the majority of cases. To give one more indication of the extent of the annotator-disagreement, the three annotators assigned the same quadrant in 26% of the cases,

	annotator 1	annotator 2	annotator 3
annotator 1	1.00		
annotator 2	0.23	1.00	
annotator 3	0.33	0.32	1.00
average	0.29		

Table 3.3: Pairwise overlap between annotators on the Wizard-of-Oz dataset, octant level

and they agreed on only 16% of the octant labels. Given the difficulty of manually annotating this dataset, we expect it to be similarly difficult to automatically classify text input in terms of Leary's Rose.

Table 3.4 shows the distribution of sentences across class labels as well as the resulting random baselines.

1143 sentences	together-above: 370 sentences	leading: 135 sentences
		helping: 235 sentences
	together-below: 227 sentences	co-operative: 161 sentences
		dependent: 66 sentences
	opposed-below: 192 sentences	withdrawn: 100 sentences
		defiant: 92 sentences
	opposed-above: 297 sentences	aggressive: 169 sentences
		competitive: 128 sentences
Random baseline	24.0%	12.8%

Table 3.4: Distribution of classes within the Wizard-of-Oz dataset

3.2.2 Feature extraction and instance creation

Looking at the sentences in the deLearyous dataset, it is not immediately obvious what aspects of a sentence indicate how it should be positioned on Leary's Rose. Consider, for instance, the following example:

dat is een beslissing die we met het bestuur hebben moeten nemen
that's a decision we had to take with management

This sentence does contain a few words which, semantically, have a clear link to leadership, namely "decision" and "management". It is however not because the utterance refers to concepts that fit under the topic of "leadership", that the player can automatically be said to be positioned in the *leading* octant on Leary's Rose. It is perfectly possible, for instance, that the player is on the defensive, attempting to justify management's decision (*defiant* behavior, in terms of the Interpersonal Circumplex), or that they are attempting to pass on responsibility to someone other

than themselves (in which case *withdrawn* would be a better fit). Clues to the "correct" octant might also be found in the use of verbs ("a decision we *had to take*" instead of "a decision we *took*"). Much depends on the context in which the sentence was written. The position of the previous sentence of both the speaker and their interlocutor will most likely be a very strong indicator of the position of the current sentence.

The above superficial analysis already points to three feature types we will attempt to extract from the deLearyous data. First, a simple bag-of-words approach will allow us to assert the presence or absence of certain words. This will capture any individual words that could provide a clue to the speaker's state of mind. If we wish to capture syntactic information, such as the frequency of verbs, we can parse the data and use part of speech tags as the basis of part of speech n-grams. Contextual information can also be extracted from the data, in the form of the previous positions of both speakers on the Interpersonal Circumplex.

Since we do not expect to benefit from differentiating between specific word forms ("decisions" vs. "decision") in the bag-of-words approach, we will also use n-grams of lemmas instead of word tokens, which allows us to map the occurrence of different forms of the same word to the same feature. We also take it one step further by using character n-grams. These sequences of n characters allow us to capture, among other things, the roots of semantically related words in one and the same feature ("*manag*ement", *manag*er, *manag*ing, etc.). Character n-grams also provide some resilience to spelling errors, as an incorrectly spelled word may still be mapped to the same feature as a correctly spelled word, provided that a cluster of characters still overlaps (Kapočiute-Dzikiene et al., 2012).

In order to extract lemmas and part of speech tags, the deLearyous dataset was first parsed using Frog, a parser for Dutch (van den Bosch et al., 2007). Frog tokenizes and parses the input text, returning a variety of syntactic information including a morphological breakdown of the words, the lemmas, fine-grained part of speech information, the chunks present in the sentences, dependency structures, and even named entity information. An example of the output is displayed in Table 3.5. The respective columns contain: the token index, the token, the lemma, the morphological analysis of the token, the part of speech tags, the chunks tags, the index of the parent token of the current word, and finally the syntactic function of the word in the sentence.

Note that in the case of part of speech n-grams, we use the base part of speech tag (i.e. the tag before the parentheses) as the basis for the n-grams. The fine-grained part of speech tags, while more informative, make for extremely sparse feature vectors, which makes it very hard for a machine learner to learn anything given the limited size of the dataset.

1	dat	dat	[dat]	VNW(aanw,pron,stan,vol,3o,ev)	B-NP	2	su
2	is	zijn	[zijn]	WW(pv,tgw,ev)	B-VP	0	ROOT
3	een	een	[een]	LID(onbep,stan,agr)	B-NP	4	det
4	beslissing	beslissing	[beslis][ing]	N(soort,ev,basis,zijd,stan)	I-NP	2	predc
5	die	die	[die]	VNW(betr,pron,stan,vol,persoon,getal)	B-SBAR	4	mod
6	we	we	[we]	VNW(pers,pron,nomin,red,1,mv)	B-NP	10	su
7	met	met	[met]	VZ(init)	B-PP	10	None
8	het	het	[het]	LID(bep,stan,evon)	B-NP	9	det
9	bestuur	bestuur	[be][stuur]/[bestuur]	N(soort,ev,basis,onz,stan)	I-NP	7	obj1
10	hebben	hebben	[heb][en]	WW(pv,tgw,mv)	B-VP	5	body
11	moeten	moeten	[moet][en]	WW(inf,vrij,zonder)	I-VP	10	vc
12	nemen	nemen	[neem][en]	WW(inf,vrij,zonder)	I-VP	11	vc

Table 3.5: An example of the output of the Frog parser (version 0.12.16)

N-grams of any of the above-mentioned feature types can be represented in several ways. We will compare four different feature representations: a binary representation (an n-gram either occurs in the document or it doesn't), absolute frequencies (how many times does the n-gram occur in the document?), relative frequencies (the absolute frequency normalized by the number of features in the document), and tf-idf scores (which put a number on the informativeness of a feature for the given document in a corpus).

In addition to the aforementioned context-independent feature types, we integrate features relating to the previous position of the current speaker and of the interlocutor on Leary's Rose. We will experiment with both the *gold* positions as provided by the annotators, and with the predicted class labels as returned by the classifier. The values we use for the gold labels are stored in the dataset as coordinates (normalized between -1.0 and 1.0) on the two defining axes (*affinity* and *dominance*). The *predicted* labels are represented using nine binary features: one for each octant, and one for the "neutral" label. (Only the feature corresponding to the previous predicted label is given a value of 1, whereas the others have 0 as their values.)
It is very important to realize that the results achieved with the *gold* positions will overestimate performance, as they use information which is not available when dealing with unseen, unannotated data. They will however give us an impression of how strong contextual features could be, given a perfect classifier. The *predicted* labels, on the other hand, allow us to accurately simulate how much information about the current position the classifier can extract from its previous predictions, even considering the fact that the previous predictions will often have been incorrect.

A final feature type is extracted from external sentiment lexicons. We used two sentiment lexicons for Dutch (see also Section 2.2.1 in Chapter 2), namely the DuOMAn sentiment lexicon and the Pattern sentiment lexicon. The DuOMAn lexicon contains $16,226$ lemmas with sentiment scores ranging from -1.0 (completely negative) and 0.31 (most positive). We refer to Jijkoun & Hofmann (2009); Jijkoun et al. (2010) for a more in-depth explanation of how these scores were calculated. The Pattern lexicon (De Smedt & Daelemans, 2012b) is a lexicon of $3,815$ adjectives with scores

on polarity, subjectivity and intensity.

We use these sentiment lexicons in two ways. The first approach is to treat the words in the lexicon as our feature set. To generate an instance vector for a given sentence, we check for each of the words in the lexicon how many times it occurs in the sentence. This is analogous to the bag-of-words approach with the important difference that the set of words is defined through an external dictionary. The same value representations (binary, absolute and relative frequency, tf-idf) can be used here.
The second way of using the lexicons involves calculating a single score for each sentence. The sentiment scores for each of the lemmas of the tokens in the sentence are retrieved from the lexicon (if they are present), and the individual scores are combined into a global score, optionally taking into account factors like modality, negation and the use of intensifiers. The Pattern Python library implements an algorithm which calculates the overall subjectivity score for a sentence taking into account modifiers and negation. For a detailed description of the Pattern package, we refer to De Smedt & Daelemans (2012a). For the DuOMAn lexicon, we take the sum of the individual word scores, flipping the sign of the score should the word be directly syntactically dependent on a negation term ("niet", "geen", "nooit").

To summarize, the feature types that were extracted from the deLearyous dataset are listed in Table 3.6. With the exception of the context features, the value of each feature can be represented as a binary value, an absolute frequency, a relative frequency, or a tf-idf score.

feature type	n-gram sizes
token	1, 2
lemma	1, 2
character	3, 4, 5, 6
part-of-speech	1, 2, 3
DuoMan lexicon	1
Pattern lexicon	1
previous positions (gold)	
previous positions (predicted)	

Table 3.6: The different feature types that have been tested

3.2.3 Machine learning

Now that feature extraction and feature vector creation process have been outlined, we can move on to the actual classification phase. Since our Wizard-of-Oz dataset can be naturally divided into 11 individual conversations, it makes sense to adopt a

leave-one-out evaluation strategy where each conversation occurs in the test partition once, and where the remaining 10 are used as training material. This avoids overfitting on author-specific features, or on topic markers that are specific to a single conversation.

In the context of this case study, we limit ourselves to Support Vector Machines. Our justification for this is that our focus in this case study is on finding the appropriate features to be able to identify emotions in text, and the specific learning algorithm used to do so is —at this stage— only of secondary concern. (See Chapter 6 for an analysis of more complex learner setups.) Additionally, we have shown SVMs to consistently perform significantly better than other popular classifiers on the deLearyous dataset in the past (Vaassen & Daelemans, 2011), which makes using Support Vector Machines a logical choice.

We used a modified version of LibShortText 1.0 (Yu et al., 2012) to generate our instances based on the bag-of-n-grams principle, and to train and test our models (see Appendix A.1 for a description of the modifications we carried out on LibShortText).

3.3 Results

All feature types described in Table 3.6 have been tested individually. Table 3.7 shows the micro-averaged F-scores[2] and macro-averaged F-scores for each feature type.

[2]Note that for single-label classification tasks (and deLearyous is one of them), the micro-averaged F-score is the same as the accuracy.

feature type	n-gram size	feature representation	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)
token	1	binary	0.168 (±0.022)	0.208 (±0.053)
		abs.freq.	0.172 (±0.027)	0.216 (±0.058)
		rel.freq.	0.173 (±0.024)	0.202 (±0.071)
		tf-idf	0.177 (±0.042)	0.201 (±0.046)
	2	binary	0.183 (±0.048)	0.212 (±0.066)
		abs.freq.	0.178 (±0.050)	0.208 (±0.068)
		rel.freq.	0.169 (±0.047)	0.207 (±0.065)
		tf-idf	0.183 (±0.061)	0.214 (±0.071)
lemma	1	binary	0.170 (±0.055)	0.206 (±0.066)
		abs.freq.	0.171 (±0.050)	0.196 (±0.052)
		rel.freq.	0.166 (±0.042)	0.197 (±0.053)
		tf-idf	0.172 (±0.065)	0.196 (±0.063)
	2	binary	0.176 (±0.037)	0.207 (±0.047)
		abs.freq.	0.181 (±0.045)	0.213 (±0.052)
		rel.freq.	0.177 (±0.049)	0.213 (±0.067)
		tf-idf	0.182 (±0.053)	0.211 (±0.061)
character	3	binary	0.186 (±0.027)	0.213 (±0.054)
		abs.freq.	0.179 (±0.034)	0.200 (±0.066)
		rel.freq.	0.168 (±0.053)	0.188 (±0.074)
		tf-idf	0.177 (±0.049)	0.210 (±0.066)
	4	binary	0.186 (±0.044)	0.222 (±0.061)
		abs.freq.	0.184 (±0.056)	0.218 (±0.063)
		rel.freq.	0.181 (±0.048)	0.209 (±0.073)
		tf-idf	0.177 (±0.061)	0.210 (±0.087)
	5	binary	0.181 (±0.042)	0.215 (±0.067)
		abs.freq.	0.183 (±0.037)	0.217 (±0.052)
		rel.freq.	0.185 (±0.066)	0.223 (±0.082)
		tf-idf	0.175 (±0.047)	0.223 (±0.093)
	6	binary	0.177 (±0.050)	0.214 (±0.082)
		abs.freq.	0.181 (±0.046)	0.212 (±0.074)
		rel.freq.	0.181 (±0.061)	0.212 (±0.070)
		tf-idf	0.174 (±0.040)	0.212 (±0.079)
part-of-speech	1	binary	0.125 (±0.013)	0.136 (±0.051)
		abs.freq.	0.131 (±0.028)	0.143 (±0.028)
		rel.freq.	0.121 (±0.032)	0.141 (±0.059)
		tf-idf	0.118 (±0.014)	0.128 (±0.027)
	2	binary	0.146 (±0.034)	0.160 (±0.044)
		abs.freq.	0.153 (±0.027)	0.178 (±0.037)
		rel.freq.	0.153 (±0.033)	0.177 (±0.045)
		tf-idf	0.163 (±0.043)	0.178 (±0.052)
	3	binary	0.138 (±0.046)	0.186 (±0.070)
		abs.freq.	0.137 (±0.042)	0.183 (±0.061)
		rel.freq.	0.130 (±0.031)	0.153 (±0.049)
		tf-idf	0.141 (±0.015)	0.159 (±0.032)
DuoMan lexicon	1	binary	0.167 (±0.024)	0.174 (±0.026)
		abs.freq.	0.161 (±0.040)	0.164 (±0.034)
		rel.freq.	0.164 (±0.036)	0.171 (±0.023)
		tf-idf	0.165 (±0.036)	0.167 (±0.023)
		sentiment scores	0.172 (±0.026)	0.172 (±0.026)
Pattern lexicon	1	binary	0.099 (±0.025)	0.121 (±0.038)
		abs.freq.	0.093 (±0.030)	0.111 (±0.035)
		rel.freq.	0.078 (±0.015)	0.099 (±0.031)
		tf-idf	0.080 (±0.020)	0.104 (±0.035)
		sentiment scores	0.102 (±0.024)	0.085 (±0.032)
previous positions (gold)			0.336 (±0.021)	0.379 (±0.024)
previous positions (predicted)			0.132 (±0.038)	0.138 (±0.066)

Table 3.7: Macro-averaged and micro-averaged F_1-scores for each of the possible feature types and representations.

As becomes clear from the results reported in Table 3.7, the system has a hard time identifying the correct octant on Leary's Rose, regardless of the feature type and feature representation chosen. It is, however, perfectly possible to improve on the random baseline of 12.8% (refer to Table 3.4 in Section 3.2.1 for a breakdown of the data distribution), and indeed most feature types do add some useful information to help in the classification of text according to the Interpersonal Circumplex.

Some feature types require special attention. Token n-grams, lemma n-grams and character n-grams attain similar scores, and given the small score differences, it is hard to paint a clear picture of which feature type would be "best" to use for this specific task. Using part of speech tags, however, does appear to score significantly lower than using the aforementioned feature types. This indicates that there does not seem to be a clear link between the emotion experienced by the speaker and their use of words in a given grammatical class (represented in the part of speech unigrams), or between the use of certain sentence structures (which are captured by the part of speech bigrams and trigrams) and a position on Leary's Rose. However, it would be inaccurate to say that the classifier learns nothing whatsoever from part of speech tags, since the achieved micro-averaged F-scores are still higher than the random baseline.

The results for the lexicon features are also remarkable. Using the Pattern lexicon entries as features does not seem to be a good approach. The lexicon in itself is relatively small, which means that for many of the sentences in the dataset, there are simply no words present that are also present in the lexicon. The only way in which a machine learner could learn from these "empty" instances would be if the absence of lexicon terms in a sentence were an indication of the class to which the sentence belongs. No classifier trained with Pattern features manages to beat the random baseline, however, which indicates that the lexicon simply covers too little of the deLearyous dataset.
The DuOMAn lexicon is significantly larger, and the sparseness problem is thus less severe. This also shows in the classification results, which on the surface are a few points lower than the lemma, character or token results, but which show no statistical difference from them[3].

Finally, the *gold* context features (previous positions of the speaker and the listener on the Interpersonal Circumplex as provided by the annotators) are clearly our top performers. The previous positions on the Rose are very telling of what the speaker's current position will be.
An important factor at play here is that in a conversation, there tends to be a natural and relatively slow movement across Leary's Rose. A speaker who starts a conversation in a certain position will tend to stay in that position unless their conversation partner gives them reason to change their stance. If we look at the

[3]Significance tested using approximate randomization testing (Noreen, 1989; Yeh, 2000). $p > 0.05$

transitions between the labels of two subsequent sentences by the same speaker in the deLearyous dataset, this becomes readily apparent. In 730 cases the speaker simply remains in the same octant on the circumplex, while there are only 413 transitions where a change in position occurs. In light of this pattern, it is clear how the "previous position" features will be highly informative for the machine learner. Given, of course, a perfect classifier that always makes accurate predictions.

Indeed, if we look at the *predicted* context features, we see that the positive effect vanishes. The output labels for each instance are determined using a classifier trained with character 4-grams, our best-performing feature type, and we use the predicted labels from previous instances to train a new context-based classifier. Since the character 4-gram classifier has an accuracy of only 22%, the vast majority of the predicted labels are in fact incorrect. As a result, our context-based classifer is trained on extremely noisy features, and fails to produce useful output.

We have looked at feature types in isolation, but it might be beneficial to train a classifier on a combination of different feature types. Table 3.8 shows the results for a selection of feature combinations. Since the differences between our initial token, lemma and character n-gram features were not statistically significant, we have chosen to take our highest-scoring feature type (in terms of macro-averaged F-score, in this case the character 4-grams in their binary form) as the basis for our combinations. We combine these character 4-grams with all of the other feature types (in their best performing sizes and representations) and note the influence of these added features on the classification results.

feature types	macro-avg. F_1	micro-avg. F_1
part-of-speech 2-grams (tf-idf)	+0.004	-0.007
DuOMAn lexicon (sentiment scores)	+0.009	-0.001
Pattern lexicon (sentiment scores)	+0.004	-0.008
previous positions (gold)	+0.015	-0.016
previous positions (predicted)	-0.008	-0.005

Table 3.8: Differences in macro-averaged and micro-averaged F_1-scores for combinations of character 4-grams with other feature types vs. the scores for character 4-grams only.

While part of speech tags did not perform well on their own, they do influence the classification performance when combined with character 4-grams. The micro-average F-score drops slightly, but the macro-averaged scores go up by a fraction of a percent. Since macro-averaged F-scores give equal weights to the F-scores of different classes, this implies that while this combination performs less well in the majority class, it does boost the performance in some of the smaller classes. Since in the context of the deLearyous project, every octant should be considered equally important, macro-averaged F-scores are the metric we would like to optimize.

Adding lexicon features to bag-of-words features would make little sense, as —at least in the training set— any word present in the lexicon would also be present in the

bag-of-words. In the case of bags of character 4-grams, however, the lexicon features are distinct from the n-gram features, since the lexicon contains full lemmas and not merely sequences of 4 characters. We see that while the addition of these extra lemmas does influence the performance, the effects remain limited. The DuOMAn features do appear to improve the macro-averaged F-scores somewhat, with only a very minimal hit to micro-averaged F-scores.

As expected, adding the *gold* context features to the character 4-grams provides a significant boost to classification performance. However, as we described in Section 3.1, the ultimate goal of the deLearyous classifier is for it to be integrated in a conversation system where a player interacts with a virtual agent. The correct classification of player input in terms of the Interpersonal Circumplex is vital in that it will serve as an important data point for determining the emotional response of the virtual agent. And while we know the position of the AI at all times, the previous position of the player is unknown and is determined by the classifier itself.
The cross-validation experiments including *gold* contextual features are thus inherently artificial. They make use of "gold" positions, which are not available to the classifier in a real-life situation. The results only serve as an illustration that given perfect predictions, contextual features would be very strong.

More interesting are the effects of adding predicted labels to the character 4-gram features. Our first intuition that predicted labels would be very noisy features appears to find confirmation in the results: the predicted context features do not change the classification results in any significant way, and in terms of raw performance, the scores even drop a little. While there is definite potential in contextual features (as evidenced by the higher performance of a classifier based on perfect predictions), the accuracy of our classifier is much too low to be able to leverage this potential.

In conclusion, character n-grams appear to be the best feature type for this specific task and dataset, and expanding the feature set with DuOMAn features may provide the system with a small performance boost. Capturing the conversational context of a sentence using previously predicted labels shows definite promise, but will only be truly effective when the base classifier can achieve high enough accuracy to make these predicted labels more reliable. On our current dataset, however, contextual features are too noisy to be of use.

3.4 Analysis

The scores in the previous section clearly show that classifying text input according to their interpersonal emotional stance is a very difficult task. However, the systems perform well above the random baseline, indicating something has clearly been

learned. In this section, we will have a look at the most informative features and their relation to the task at hand.

To determine which features were considered important by the SVM learner, we examine the weights the learner assigned to each feature in the training set. Features with higher weights should be strong pointers that the current sentence belongs to a certain class. Table 3.9 shows the ten most important lemmas from the DuOMAn lexicon, as well as the most important character 4-grams.

top DuOMAn entries	top char. 4-grams
gaan	.===
op	_ja_
optie	_ik_
kunnen	?===
laten	wat_
kosten	dat_
maken	vind
probleem	het_
zitten	natu
voor	snap

Table 3.9: Top DuOMAn lemmas and character tetragrams according to SVM weighting.

The strong DuOMAn features are very general lemmas that are hard to immediately associate with a certain position on Leary's Rose. "gaan" (*go*), "kunnen" (*can*), and "laten" (*let*) are often used as auxiliary verbs, which carry very little semantic content on their own. These auxiliaries might in themselves be interesting if we can show that the use of auxiliary verbs is linked to a certain position on the Interpersonal Circumplex. Instinctively, one might imagine that the use of auxiliaries gives an utterance a more tentative, careful feel, which would associate them with the less dominant positions on the Rose (octants on the bottom half of the Rose).
The character n-grams seem to form a complementary set of features, as the top-weighted features appear to originate from words different from the DuOMAn entries. Again, very few of the top features are content words. Interjections ("ja"/*yes*), personal pronouns ("ik"/*I*), interrogative or demonstrative pronouns ("dat", "wat" / *that, what*) and even punctuation marks (".", "?") are all weighted heavily. The few content words can be explained in the context of the Interpersonal Circumplex. The "natu" n-gram is found in "natuurlijk" (*naturally/of course*), which is a term that expresses certainty and the opinion that the speaker's words are self-evident. This is a state of mind one is unlikely to find in many of the submissive octants of the Rose. "Vind", from the verb "vinden" (*to find*), in the context of these conversations, is used in its meaning of *"holding an opinion"*. Finally, "snap" is a 4-gram extracted from the verb "snappen" (*to get, to understand*), which implies a certain empathy from the speaker towards the listener.

Of course, the above ranking of features does not tell us anything about which features contribute to which class. Instead they tell us about the importance of each feature in the model in general. We also have a look at the most important character 4-gram and DuOMAn features per class, based on the weights assigned to them by the SVM classifier. Table 3.10 shows the top 5 DuOMAn lemmas for each of the nine classes, while Table 3.11 shows the top 5 character 4-grams for each class.

aggressive	competitive	helping	leading	co-operative	NEUTRAL	defiant	withdrawn	dependent
niet	zijn	ik	een	ja	niet	ik	ja	ja
hebben	luisteren	zijn	zijn	ik	ik	niet	niet	ik
zijn	ik	zullen	ik	zijn	zijn	maar	ik	nee
ik	een	een	hebben	in	een	zijn	zijn	een
gaan	gaan	kunnen	willen	niet	dagen	hebben	hebben	zijn

Table 3.10: Top-weighted DuOMAn lemmas per class according to the SVM classifier.

aggressive	competitive	helping	leading	co-operative	NEUTRAL	defiant	withdrawn	dependent
niet	_je_	?===	een_	_ja_	.===	.===	.===	_ja_
?===	.===	_je_	_je_	.===	..._	_ik_	niet	.===
dat_	dat_	_ik_	_is_	_ik_	dag_	niet	..._	...=
_je_	iste	.===	_we_	dat_	wel_	dat_	_ja_	snap
_wat	uist	doen	_de_	_je_	daag	!===	_ik_	_ik_

Table 3.11: Top-weighted character tetragrams per class according the SVM classifier.

Judging by the top-weighted features per class according to the SVM classifier as shown in Tables 3.10 and 3.11, it is hardly surprising that the machine learner has a hard time making sense of the task. Many of the features are strong for many classes, and there appear to be very few strong features that unequivocally point to a single octant on the Interpersonal Circumplex. Some exceptions to the rule might be found in the character 4-grams, with the top *neutral* features being very easy to explain. "dag" and "daag" are both a form of a Dutch greeting, and since the initial phase of a conversation is often the only phase one could consider truly "neutral" (Leary's Rose does not acknowledge the existence of a neutral stance), greetings are a perfect fit. "Wel", much like the English *well*, is an interjection that in itself carries no emotion. And finally, "..." marks a hesitation, and could therefore rightly be said to not carry an emotion. Likewise, it is remarkable that the exclamation mark is so strong for *defiant* sentences, and that "snap" (as mentioned above, this 4-gram stems from the verb "snappen", which means *to understand*) is strong for *dependent*.
It is important to remark that a machine learner will not make decisions based on the presence of singular features, but on the basis of complex combinations of multiple features. The tables above, however, do give us an idea of the features that will most likely contribute to the classifiers' decisions.

In conclusion, while it is possible to beat the random baseline (12.8% in terms of accuracy) by a sizable margin, it appears to be extremely difficult to reach a performance that would be acceptable for practical use.

	annotator 1	annotator 2	annotator 3
annotator 1	1.00		
annotator 2	0.23	1.00	
annotator 3	0.33	0.32	1.00
average	0.29		

Table 3.12: Pairwise overlap between annotators on the Wizard-of-Oz dataset, octant level. (Repeated from Table 3.3)

However, when we look back at the performance of human annotators on the same task (Table 3.12 shows the pairwise overlap between annotators on the octant level), maybe this low performance is hardly surprising. On average, two annotators agreed on the position of only 29% of the sentences. Even if we managed to build a system with better performance, an accuracy over 30% would have little meaning.
We can however draw some general conclusions on the types of features that best serve this specific task. Simple character n-grams capture the bulk of the relevant information, and lexicon-based features may complement the feature set by identifying relevant key emotion terms. Since we are dealing with conversations, we can also make use of context features. In theory, the label of the previous sentence makes for a strong predictor for the current class. However, we have seen that the efficacy of this feature type is greatly dependent on the accuracy of the system. In this case study, the system's performance is too low for the predicted context features to be useful.

Chapter 4

Case study: Emotions in Suicide Notes

4.1 Introduction

This case study focuses on an entirely different facet of emotion classification. Where the goal of the deLearyous task (Chapter 3) was to identify the emotional stance of participants in a conversation based on their textual input, the goal of the current task is to identify emotions in suicide notes. The task was initially introduced in the 2011 Medical NLP Challenge (Pestian et al., 2012), where participants in the challenge were provided a training set of manually labeled suicide notes, and instructed to build a system that could accurately identify emotions in an unseen test set. The existence of such a system would greatly benefit the medical world, as it might serve as a tool to better understand the thought processes of individuals who choose to take their own lives, and it might thus indirectly help prevent suicides in the future. The system might also be applied in an online moderation context, where it would help moderators to more easily track down problematic interactions between users, so they can intervene before things escalate (also see Project AMiCA[1]).

This emotion classification task differs in a few significant aspects from the deLearyous case study, in ways that have an influence on the methodology.
First of all, the 2011 Medical NLP Challenge dataset contains suicide notes written in English. The data for deLearyous was in Dutch. While this is of no practical consequence to the way instances are created or classified, this does mean that it

[1] AMiCA: Automatic Monitoring for Cyberspace Applications - http://www.amicaproject.be (last accessed on November 15th, 2013)

is not possible to use the same Dutch lexicons that were used in the previous case study.
Secondly, the emotion framework adopted for classification is a discrete one, whereas deLearyous used a dimensional model. While we did not make use of the dimensional nature of Leary's Rose in Chapter 3, there is simply no possibility to see classes in the suicide note dataset as contiguous or otherwise spatially ordered.
Finally, and most importantly, it is possible to assign more than one emotion to a single sentence. This transforms a simple multi-class classification task into a multi-label one, which requires a different approach, as will become clear in section 4.4.2.

We will start with a description of the 2011 Medical NLP Challenge dataset in Section 4.2. In Section 4.3, we will briefly consider a few other systems that participated in the challenge, and we will mention some related research that is relevant to our own approach. Section 4.4 will describe our own emotion classification system, and its results will be described and analyzed in Sections 4.5 and 4.6. We will draw our conclusions in Section 4.7.

4.2 Data description

The 2011 Medical NLP Challenge dataset consists of 900 notes of variable length which were left behind by people who committed suicide. These notes were anonymized and annotated at the token level by at least three different annotators. The possible emotion categories are listed in Table 4.1. Token-level annotation implies that the boundary of an emotion-carrying fragment in a suicide note need not coincide with a sentence boundary. It is indeed possible that only a few words in a sentence express emotion, while the rest of the sentence remains unlabeled. It is also possible for the same token to carry more than one label. Token-level annotations were aggregated to the sentence level, and the final classification task was thus to classify sentences as either being free of emotion, or as belonging to one —or several— of the 15 possible emotion classes. Pestian et al. (2012) report an inter-annotator agreement of 0.546 (Krippendoff's α with Dice's coincidence index) at the sentence level. The dataset is split into a training set of 600 suicide notes and a test set of 300 documents.

abuse (0.18%)	anger (1.38%)	blame (2.15%)	fear (0.50%)
forgiveness (0.12%)	guilt (4.18%)	happiness/peacefulness (0.50%)	hopefulness (0.94%)
hopelessness (9.13%)	information (5.92%)	instructions (16.46%)	love (5.94%)
pride (0.30%)	sorrow (1.02%)	thankfulness (1.89%)	(no-annotation) (49.38%)

Table 4.1: Emotions present in the classification scheme, with an indication of their share in training

There are a few important points concerning the dataset that should be raised prior to discussing the classification methodology. First of all, the sentences that carry no emotion labels cannot simply be said to be entirely emotion-free or neutral. When the annotators were unable to reach an agreement concerning a certain fragment of text, the fragment was left unlabeled. This means that it is highly likely that if one interprets the sentences without labels to be part of a "neutral" class, that this class will be noisy, as it contains discarded sentences that were believed to contain an emotion by at least one of the annotators.
A second point is that some of the classes seem to correspond less well to what is generally considered an emotion. "Instructions" and "information" are the most blatant example of this mismatch. They are eminently appropriate for this specific classification task, but they do not fit the conventional emotion classification frameworks. We are also hard-pressed to see labels such as "abuse", "blame", and "forgiveness" as emotional labels, as they refer more to actions than to emotional states. This does not make the classification task any less interesting, of course, but it is important to mention that the 2011 Medical NLP Challenge was no typical emotion classification challenge. In Chapter 5, we will use the same suicide note dataset again, but we will only consider the emotion classes that can be mapped to one of Ekman's "Big Six" (1971). This chapter, however, will solve the task as it was presented in Pestian et al. (2012), and no changes will be made to the classification scheme.

4.3 Related research

24 teams submitted output for the 2011 Medical NLP Challenge. As it would be rather superfluous to describe each of the participating systems in detail, we will select a few approaches that scored particularly well. Our own submission to the shared task was described in Luyckx et al. (2012), and we will attempt to improve on our scores in section 4.4.

When looking at the top-scoring systems for the shared task (Yang et al., 2012; Xu et al., 2012; Sohn et al., 2012), a common factor seems to be the combination of machine learning methods with rule-based methods. A particularity of the task's data set was that some emotions were very rare, in that very few sentences (in both the training and the test sets) were labeled as belonging to these emotion classes. This makes these classes extremely difficult to learn for supervised machine learners, since they have very little information to go on. To remedy this problem, some have tried to use handcrafted rules for these specific emotions, which circumvents the need for extensive training data.

Yang et al.'s system (Yang et al., 2012) consists of several modules, some rule-based, others machine learning-based. In a first step, sentences are tagged and parsed in order to obtain part-of-speech information, lemmas, chunk information, and depen-

dency structures. A negation detection module then identifies negation terms using a handmade lexicon of negation words collected from the training data. The next module identifies the sentences that carry the six most common emotions. This module is both the most essential for the task, and the most complex, as it is in itself comprised of two components. The first component identifies emotions on a sentence level, and is an ensemble of an SVM classifier, an Naïve Bayes classifier, and a Maximum Entropy model. The second component identifies emotions on the token level, which it achieves through a combination of keyword-spotting based on an emotion lexicon, and a Conditional Random Fields model that recognizes emotion cues. The predictions from these modules are aggregated using several voting algorithms. A post-processing module makes a final pass over the test data and attempts to identify the sentences carrying *information* and *instructions*.

When evaluating their system, Yang et al. notice that the machine learners are responsible for the bulk of the correct predictions. The lexicon-based approach only adds very little to the the overall performance. Recall, however, that the machine learners only predict six out of the 15 possible emotion classes, and the lexicon-based approach is therefore solely responsible for identifying the sentences belonging to the remaining emotion classes. Performance on these classes is very low, indicating either that the emotion lexicon's coverage was too low, or that lexicon-based approaches simply cannot capture the complexity of emotions in text.

Xu et al. (2012) also note the data sparseness problem, and attempt to extend the shared task's limited training data by searching for similar data online. They retrieve similarly strongly emotional material from The Suicide Project[2] and annotate 1,814 sentences manually, adding the resulting documents to the training set. Even with this extra training data, some emotions are underrepresented and generally much more subtle and more difficult to detect. Two classes, *information* and *instructions*, seem to carry very little emotional information. Noticing these discrepancies, the authors decide to cluster the class labels into three subgroups, each requiring a different approach: "strong" emotions (*fear*, *guilt*, *hopelessness*, *love*, *sorrow*, *hopefulness*, *thankfulness*, *happiness_peacefulness*), "objective" classes (*information* and *instructions*), and "subtle" emotions (*abuse*, *anger*, *blame*, *pride*, *forgiveness*).

The "strong" emotions are handled by a one-versus-all ensemble of SVM classifiers. The innovation here is in the *spanning n-grams* the authors use, which are pairs of token n-grams which can skip over several words in a sentence. The spanning n-grams <take, any longer>, for instance, can be extracted from the sentence "I can't take it any longer".
The "objective" classes are also handled by an SVM classifier. The features, however, are entirely different. The authors posit that sentences from the *information* and *instructions* classes will contain more factual words relating to items and loca-

[2]http://www.suicideproject.org - Last accessed on April 2nd, 2013

tions. They thus compiled several dictionaries containing terms from the following categories: "daily items", "financial terms" "location prepositions" and "locations". Sentences are pre-processed by replacing terms with their term category. ("Please find <financial term> <location prep> <daily item>" would be the normalized version of "Please find my insurance in the briefcase".) From these normalized sentences, the authors extract n-grams of tokens, ranging from unigrams to tetragrams, which are then used to train the SVM classifier.
The "subtle" emotions are classified entirely using pattern matching, based on a list of strongly relevant patterns extracted from the training data, LiveJournal, and WordNet. If a pattern is matched on a test sentence, the corresponding emotion is added to its predicted labels.

Finally, Sohn et al. (2012) also combine machine learning and rule-based methods. They use a Multinomial Naïve Bayes classifier and a rule-induction algorithm trained on unigrams, bigrams and trigrams of tokens to assign the most common labels (*blame*, *guilt*, *hopelessness*, *information*, *instructions*, *love*, and *thankfulness*).
For the rule-based method, the authors compiled a set of regular expressions associated with an emotion label. If a sentence matches an expression, its label is assigned. The sentences were first parsed to make it possible for rules to generalize over part of speech patterns or lemmas, and WordNet was used to allow for synonyms.
The final system's output is the union of the output of these machine learning and rule-based methods, in that a sentence is assigned any label that is assigned by either subsystem. Much like Yang et al. (2012), however, Sohn et al. note that the machine learners do the bulk of the work, and that the rule-based methods only serve to boost the performance slightly. The machine learner performed best on those emotions with sufficient training data, while the rules managed to generalize better over the more uncommon emotions.

While the participating systems vary in implementation and complexity, there does seem to be a common thread. While it is possible to train a classifier that performs reasonably well on the well-represented emotions, the less common classes are harder to learn, and a rule-based approach may be a better solution for these sparse labels. However, authors note that the improvement from a rule-based approach over a pure machine learning system is minimal.
Our own system (Luyckx et al., 2012) consisted of a machine-learning module only, and this will also be the focus of the current chapter.

4.4 Classification procedure

4.4.1 Instance creation

The instance creation procedure is analogous to the procedure for the deLearyous case study (see Chapter 3), with the important difference that we do not parse the text prior to feature extraction. The reason for this is a practical one: the suicide notes contain many spelling and grammatical errors, and a parser would get confused by those errors and annotate the data incorrectly, only adding to the noise. We extracted n-grams of tokens and of characters from the dataset and again represented them using binary features, absolute counts, relative counts, and tf-idf scores.

4.4.2 Machine learning

The fact that it is possible to assign more than one label to a single instance makes the classifier setup slightly more complex. We have tried three different approaches to solve this problem.

Single-label classification using complex labels

Our first solution implies considering each combination of labels as a single complex label. An instance labeled with the emotions "blame" and "guilt" will thus be attributed the single label "blame+guilt". This approach has the advantage that one can simply apply any standard multi-class classification algorithm to the dataset without worrying about multi-label instances. The disadvantage, however, is that since any combination of labels forms a new class, in many cases these complex classes will be extremely sparse. It is quite common, even, that a specific combination of labels that occurs in the test set will never have occurred during training, making instances in that class impossible to predict. One can also not rely on our SVM-based classifier to understand that the complex label "blame+guilt" is more similar to "blame" than it is to, for instance, "thankfulness+forgiveness", since it sees all labels as distinct and unrelated.

One vs. all ensembles with thresholding

Our second and third systems function in a similar way, but differ in one important aspect. Both systems are ensembles consisting of one classifier per possible emotion label. For each emotion, a classifier will be trained on all the instances in the training set, but all instances which are not labeled with the given emotion will be considered to be negative, while only the instances labeled with the emotion will be considered

positive. During testing, each test instance will be classified by each of the 15 (the number of possible emotions in the dataset) one vs. all classifiers. Each classifier will return a probability score reflecting how likely it is that the current instance contains the given emotion.

Where our two one-vs.-all systems differ, is in the way they determine which emotion labels to assign to a test instance. Both use a thresholding approach, i.e. if a class probability exceeds a predefined threshold, it will be included as one of the predicted class labels. However, one system uses a single threshold, while the other uses two thresholds. The **single-threshold** system straightforwardly assigns any of the 15 emotion labels that have a probability that exceeds a given value. The **dual-threshold system**, however, also takes into account the predictions of a classifier trained to recognize sentences which carry *no* emotions, and thus needs a separate threshold value for the probability of the document containing no emotion. Only if the probability of a document not containing any emotions is low enough (below the *no-emotion* threshold) do we proceed to assigning those emotion labels whose probability exceed the *emotion* threshold.

4.5 Results

To be able to better compare our current results to the results achieved during the 2011 Medical NLP Challenge, we describe them in two steps. In a first phase, the feature types, feature representations and learner parameters are determined and evaluated on training data. In a second phase, the best performing parameters from the first step are applied to an independent test set. We did not make any modification to the system we selected during the training step, and no parameters or settings were altered during the test step.

Results on training data: Single-label classification using complex labels

Table 4.2 shows the results for the training experiments where combinations of labels were interpreted as a single, complex label. We used a modified[3] build of LibShortText 1.0 (Yu et al., 2012) to build instances, and used an SVM classifier in a 6-fold cross-validation setup (6 partitions of 100 training documents) to classify the training data. We report on micro-averaged F-scores and macro-averaged F-scores and their standard deviations[4].

[3] See Appendix A.1 for an overview of the modifications we made to LibShortText.

[4] Not attributing any label to a document is not considered as equivalent to attributing a "neutral" or "none" label. Macro-averaged F-scores and micro-averaged F-scores thus also do not assume the

It is important to note that even though the classifiers in these experiments return complex labels, that these complex labels are decomposed into simple labels prior to evaluation. Concretely, this means that if our classifier predicts the "anger + hopelessness" label, we interpret this as the classifier voting for two simple labels, i.e. "anger" and "hopelessness". This seems to us to be the only fair approach, as this is how the systems were evaluated during the 2011 Medical NLP Challenge, and as it also allows us to compare the results from these experiments with those of the one vs. all experiments in Tables 4.3 and 4.4 below.

feature type	n-gram size	feature representation	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)
token	1	binary	0.172 (±0.011)	0.418 (±0.014)
		abs.freq.	0.173 (±0.027)	0.406 (±0.031)
		rel.freq.	0.165 (±0.020)	0.394 (±0.017)
		tf-idf	0.129 (±0.015)	0.357 (±0.024)
	2	binary	0.170 (±0.015)	0.392 (±0.023)
		abs.freq.	0.164 (±0.013)	0.387 (±0.028)
		rel.freq.	0.152 (±0.026)	0.370 (±0.044)
		tf-idf	0.156 (±0.021)	0.394 (±0.022)
character	3	binary	0.183 (±0.009)	0.419 (±0.017)
		abs.freq.	0.182 (±0.016)	0.422 (±0.017)
		rel.freq.	0.147 (±0.013)	0.378 (±0.014)
		tf-idf	0.141 (±0.019)	0.377 (±0.047)
	4	binary	0.189 (±0.024)	0.440 (±0.011)
		abs.freq.	0.206 (±0.008)	0.437 (±0.017)
		rel.freq.	0.155 (±0.008)	0.398 (±0.018)
		tf-idf	0.146 (±0.019)	0.391 (±0.019)
	5	binary	0.196 (±0.015)	0.448 (±0.012)
		abs.freq.	0.191 (±0.021)	0.438 (±0.013)
		rel.freq.	0.170 (±0.009)	0.421 (±0.032)
		tf-idf	0.162 (±0.009)	0.415 (±0.014)
	6	binary	0.190 (±0.009)	0.439 (±0.021)
		abs.freq.	0.183 (±0.008)	0.432 (±0.016)
		rel.freq.	0.173 (±0.017)	0.417 (±0.027)
		tf-idf	0.169 (±0.011)	0.420 (±0.014)

Table 4.2: Results for the complex-label cross-validation experiments on the training data. Macro-averaged and micro-averaged F_1-scores and their standard deviations for each of the possible feature types and representations.

Results on training data: One vs. all ensembles with a single threshold

Table 4.3 shows the results on training data for the one vs. all experiments using a single probability threshold. As explained above, if the probability of a class exceeds a given threshold, it is assigned to the current document. As above, we used a 6-fold cross-validation setup, with each fold having 100 of the 600 documents in test. The probability threshold set for the experiments in Table 4.3 was 0.50. It should be noted that at this stage, this threshold is not an optimal threshold. It was merely

existence of such an empty label. For a short overview of the different possible ways of scoring multi-label tasks, see Appendix A.2.

chosen as a reasonable baseline to first determine the most interesting feature types and feature representations. A threshold of 0.50 seemed reasonable for this purpose, as it is very close to the threshold we found in Luyckx et al. (2012). However, the experiments we performed here are not exact matches to those performed during the shared task, and we can expect the optimal threshold to differ.

feature type	n-gram size	feature representation	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)
token	1	binary	0.084 (±0.014)	0.244 (±0.012)
		abs. freq.	0.086 (±0.021)	0.240 (±0.009)
		rel. freq.	0.179 (±0.025)	0.428 (±0.011)
		tf-idf	0.085 (±0.010)	0.227 (±0.016)
	2	binary	0.149 (±0.034)	0.355 (±0.017)
		abs. freq.	0.145 (±0.033)	0.346 (±0.014)
		rel. freq.	0.184 (±0.028)	0.427 (±0.008)
		tf-idf	0.167 (±0.034)	0.411 (±0.012)
character	3	binary	0.095 (±0.020)	0.236 (±0.012)
		abs. freq.	0.085 (±0.019)	0.224 (±0.008)
		rel. freq.	0.161 (±0.017)	0.405 (±0.010)
		tf-idf	0.175 (±0.020)	0.415 (±0.009)
	4	binary	0.139 (±0.016)	0.345 (±0.013)
		abs. freq.	0.134 (±0.015)	0.339 (±0.012)
		rel. freq.	0.185 (±0.019)	0.434 (±0.006)
		tf-idf	0.168 (±0.027)	0.385 (±0.006)
	5	binary	0.154 (±0.022)	0.377 (±0.015)
		abs. freq.	0.157 (±0.030)	0.375 (±0.016)
		rel. freq.	0.188 (±0.025)	0.445 (±0.009)
		tf-idf	0.175 (±0.024)	0.408 (±0.023)
	6	binary	0.166 (±0.021)	0.389 (±0.014)
		abs. freq.	0.159 (±0.030)	0.392 (±0.014)
		rel. freq.	0.193 (±0.020)	0.455 (±0.008)
		tf-idf	0.178 (±0.018)	0.433 (±0.014)

Table 4.3: Results for the one vs. all cross-validation experiments on training data using a probability threshold of 0.50. Macro-averaged and micro-averaged F_1-scores and their standard deviations for each of the possible feature types and representations.

From the results in Table 4.3, we selected the most promising set of parameters (character 6-grams represented as relative frequencies) and performed a search for the optimal probability threshold. Figure 4.1 shows the effects of shifting this threshold. It is clear (and quite logical) that as the threshold goes up, the micro-averaged precision of the system goes up as well, but the micro-averaged recall suffers greatly. A balance between precision and recall is struck around a threshold of 0.23. We therefore select this threshold to apply to test set in the second phase.

Results on training data: One vs. all ensembles with dual thresholds

The previous results (Table 4.3) showed the performance of an ensemble which did not include a classifier which separates the emotionally-charged sentences from the unlabeled sentences (as we noted in Section 4.2, calling these instances "neutral" would be questionable). We also constructed a dual-threshold ensemble, which does

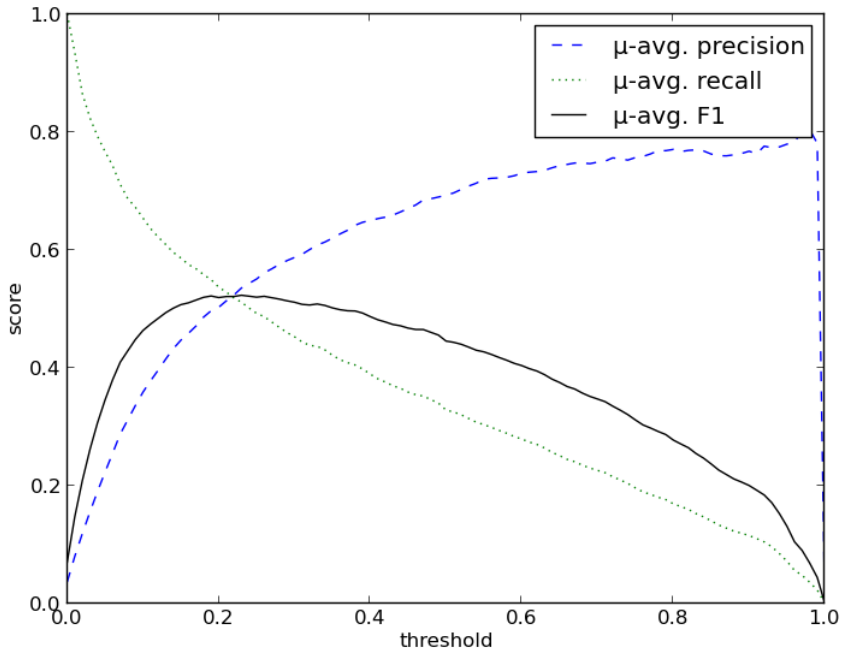

Figure 4.1: micro-averaged precision, recall and F-score as a function of the probability threshold set on emotion labels

include a classifier for unlabeled instances. Assigning an emotion label requires that two conditions be met: firstly, the probability of the document being devoid of emotion (as returned by this extra classifier) should be low enough, and secondly, the probability of the emotion being present (according to the remaining one vs. all classifiers) must be sufficiently high. Following a similar train of thought as for the single-threshold experiments, the no-emotion threshold was set to 0.80 while the emotion threshold was set to 0.20. These thresholds were shown to be effective in Luyckx et al. (2012), and thus seemed like a reasonable starting point.

Again we selected the best-performing system from Table 4.4 (the system trained with character 4-grams in their relative frequencies) and performed a grid search on the best probability thresholds. Figure 4.2 shows micro-averaged F-scores as a function of the "emotion" and "no-emotion" thresholds. For clarity, let us re-iterate,

feature type	n-gram size	feature representation	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)
token	1	binary	0.156 (±0.013)	0.404 (±0.007)
		abs. freq.	0.156 (±0.011)	0.409 (±0.009)
		rel. freq.	0.204 (±0.018)	0.470 (±0.012)
		tf-idf	0.155 (±0.021)	0.394 (±0.013)
	2	binary	0.188 (±0.027)	0.451 (±0.016)
		abs. freq.	0.185 (±0.027)	0.449 (±0.016)
		rel. freq.	0.196 (±0.025)	0.453 (±0.013)
		tf-idf	0.170 (±0.023)	0.410 (±0.010)
character	3	binary	0.168 (±0.010)	0.410 (±0.014)
		abs. freq.	0.173 (±0.018)	0.401 (±0.017)
		rel. freq.	0.201 (±0.029)	0.473 (±0.017)
		tf-idf	0.211 (±0.022)	0.483 (±0.013)
	4	binary	0.190 (±0.022)	0.459 (±0.017)
		abs. freq.	0.218 (±0.019)	0.454 (±0.023)
		rel. freq.	0.227 (±0.016)	0.487 (±0.010)
		tf-idf	0.211 (±0.022)	0.461 (±0.012)
	5	binary	0.222 (±0.020)	0.476 (±0.008)
		abs. freq.	0.213 (±0.015)	0.472 (±0.025)
		rel. freq.	0.227 (±0.017)	0.479 (±0.015)
		tf-idf	0.207 (±0.018)	0.462 (±0.011)
	6	binary	0.209 (±0.021)	0.482 (±0.020)
		abs. freq.	0.201 (±0.017)	0.477 (±0.015)
		rel. freq.	0.218 (±0.023)	0.476 (±0.019)
		tf-idf	0.201 (±0.021)	0.458 (±0.015)

Table 4.4: Results for the one vs. all cross-validation experiments on training data using two probability thresholds: one for no-emotion (0.80), one for emotions (0.20). Macro-averaged and micro-averaged F_1-scores and their standard deviations for each of the possible feature types and representations.

in pseudo code, how the "emotion" and "no-emotion" thresholds interact:

```
for instance in testset do
    if P(no-emotion) > no-emotion threshold then
        DO NOT assign any labels to instance;
    else
        for emotion in emotions do
            if P(emotion) > emotion threshold then
                ASSIGN emotion label to instance;
            end
        end
    end
end
```

The best results are reached when the "no-emotion" threshold is set to 0.98, and when the "emotion" threshold is set to 0.22. In other words, we only very rarely block an instance from receiving emotion labels based on the predictions of the "no-emotion" classifier, and we already assign a specific emotion label when its probability crosses 22%.

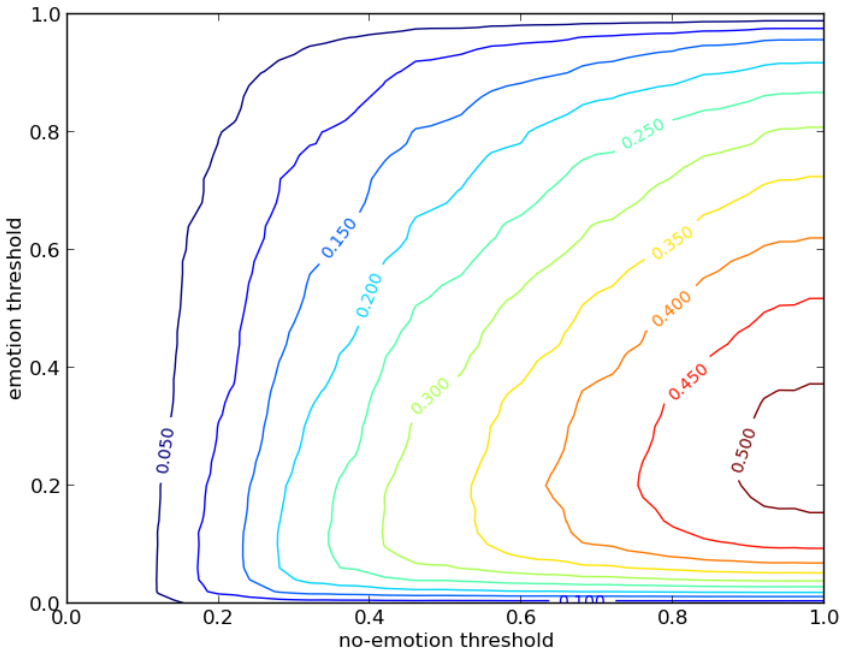

Figure 4.2: micro-averaged F-scores as a function of the thresholds set on the probability of a document not containing any emotion (*no-emotion*) and on the probability of an emotion being present (*emotion*).

Results on independent test data

We now have three systems that were fully calibrated on training data. In a second phase, we evaluate these systems on unseen test data. Table 4.5 shows the performance of the systems and compares them to a baseline which naively assigns all labels to each document, to the results we achieved in Luyckx et al. (2012), and to the results of the top-performing system in the 2011 Medical NLP Challenge (Yang et al., 2012). Note that since the systems that participated in the 2011 Medical NLP Challenge were evaluated on micro-averaged F-scores, we cannot report on macro-averaged F-scores in all cases.

system	macro-avg. F_1	micro-avg. F_1
naïve baseline	0.014	0.017
complex labels	0.208	0.436
OVA (single threshold)	0.235	0.538
OVA (dual threshold)	0.240	0.540
system from Luyckx et al. (2012) (single threshold)		0.473
system from Luyckx et al. (2012) (dual threshold)		0.502
system by Yang et al. (2012) (best performer)		0.614

Table 4.5: Results on test data. Macro-averaged (where available) and micro-averaged F_1-scores.

4.6 Analysis

We first look at what our three systems (the complex-label system, the single-threshold one vs. all system, and the dual-threshold one vs. all system) have in common before focusing on to how they are different.
It is might not be surprising that all three systems perform best using the same types of features. After all, the features carry all the information on which the classifiers base their decision. The exact setup of the classifier ensemble only helps in better leveraging this information to determine the correct labels for the test documents, it will not be able to create information where none exists. What is remarkable, however, is that the best-performing features are not the more commonly used bag-of-words features (which we also used in our systems in Luyckx et al. (2012)), but that character n-grams consistently take the crown. The exact size of these n-grams appears to matter less, as windows of 4, 5 or 6 characters all seem to be viable. Table 4.6 shows the training set's top character n-grams of different sizes according to information gain.

Whether they are represented in the form of 4-, 5- or 6-grams, the top features appear to capture the same information, which explains why model's performance remains largely unaffected. The top features capture words and phrases that can immediately be connected to the emotion classification task at hand. As "love" is one of the possible labels, it is little surprising to see n-grams containing parts of the word "love" or the phrase "I love you" being weighted as strong predictors. The same can be said for the relation between "forgive" and the *forgiveness* label, for instance. These top features also show us some of the advantages of using character n-grams as opposed to bag-of-words features: the fact that character n-grams can cross word boundaries means we can capture highly informative phrases in a single, compact feature. Character n-grams also provide some robustness against spelling errors and alternative spellings, which would be harder to capture using token n-grams. "love you" and the alternative, more colloquial "love ya" would for instance be mapped to the same character 6-gram "love_y", whereas it would have to be represented by two

top char. 4-grams	top char. 5-grams	top char. 6-grams
love	_love	_love_
_lov	love_	i_love
ove_	_you_	love_y
_you	i_lov	ove_yo
you_	ove_y	ve_you
i_lo	ve_yo	e_you_
ve_y	e_you	_i_lov
e_yo	_i_lo	please
_to_	ease_	lease_
_for	lease	forgiv
_my_	_john	sorry_
ase_	pleas	orgive
_the	_for_	_forgi
_joh	_the_	_jane_
plea	thank	rgive_
ease	_and_	a_n't_
_i_l	forgi	_sorry
and_	orgiv	ca_n't
the_	sorry	_pleas
john	rgive	_ca_n'

Table 4.6: Top character tetra-, penta-, and hexagrams according to information gain weighting.

different features when using word bigrams. All these factors contribute to giving character n-grams a slight edge over other feature types.

While our three systems prefer similar feature types, there are still important performance differences. The complex-label system and the two one vs. all ensembles especially perform significantly differently. For reference, Table 4.7 repeats the performance of our three systems.

system	macro-avg. F_1	micro-avg. F_1
complex labels	0.208	0.436
OVA (single threshold)	0.235	0.538
OVA (dual threshold)	0.240	0.540

Table 4.7: Results on test data (repeated from Table 4.5). Macro-averaged and micro-averaged F_1-scores.

The complex label system, which was effectively trained as if every combination of emotion labels was a single emotion label, performs significantly worse[5] than the one vs. all systems. Table 4.8 shows precision, recall and F-score for each of the 15 possible emotions.

[5]The statistical significance of the difference between the systems was measured using approximate randomization testing (Noreen, 1989; Yeh, 2000). ($p < 0.05$)

class	precision	recall	F_1
abuse	0.000	0.000	0.000
anger	0.077	0.038	0.051
blame	0.107	0.067	0.082
fear	0.000	0.000	0.000
forgiveness	0.000	0.000	0.000
guilt	0.383	0.265	0.313
happiness/peacefulness	0.500	0.062	0.111
hopefulness	0.182	0.053	0.082
hopelessness	0.523	0.454	0.486
information	0.384	0.269	0.316
instructions	0.617	0.476	0.538
love	0.630	0.483	0.546
pride	0.000	0.000	0.000
sorrow	0.143	0.029	0.049
thankfulness	0.543	0.556	0.549

Table 4.8: Results on test data for the complex-label system. Precision, recall and F_1-scores per class.

There are quite a few classes for which the classifier fails to produce any useful predictions. “abuse”, “fear”, “forgiveness” and “pride” all remain empty, while “anger”, “blame”, “hopefulness” and “sorrow” also do not manage to cross the 0.10 F-score barrier. This behavior, however, is not entirely unexpected, as these are all classes for which we have very little data (we refer to Table 4.1 for the distribution of documents across emotions). The training for “abuse”, “fear”, “forgiveness” and “pride” all make up less than 0.50% of the training data, while “anger” (1.38%), “blame” (2.15%), “hopefulness” (0.94%) and “sorrow” (1.02%) are also strongly underrepresented.

The one vs. all classifiers do not perform any better in this respect, as can be seen in Tables 4.9 and 4.10. It is on the more strongly represented classes that the differences become obvious.

Indeed, “hopelessness”, “information”, “instructions”, “love”, and “thankfulness” all get a strong boost when we use a one vs. all ensemble instead of a non-multi-label classifier. These are the best represented labels in the training set, which points to a confirmation of our hypothesis that performance on a given emotion class is strongly linked to its presence in the training set. The complex-label system evaluated in Table 4.8 exacerbates the data sparseness problem, since documents that belong to more than one class will be considered part of a new, combined class, and removed from the pool of training instances for each of the component emotions. Whereas a sentence labeled with “anger+blame” will be part of the training set for both the “anger” and “blame” classes in a one vs. all setup, it will only serve as training material for the (very rare) “anger+blame” class in the complex-label system. Given the sparse nature of the 2011 Medical NLP Challenge dataset, using a one vs. all

class	precision	recall	F_1
abuse	+0.000	+0.000	+0.000
anger	+0.256	+0.000	+0.018
blame	+0.004	−0.045	−0.045
fear	+0.000	+0.000	+0.000
forgiveness	+0.000	+0.000	+0.000
guilt	+0.151	+0.000	+0.041
happiness/peacefulness	+0.500	+0.000	+0.007
hopefulness	+0.151	−0.027	−0.033
hopelessness	+0.031	+0.127	+0.081
information	−0.015	+0.193	+0.094
instructions	−0.025	+0.254	+0.116
love	+0.142	+0.139	+0.143
pride	+0.000	+0.000	+0.000
sorrow	−0.143	−0.029	−0.049
thankfulness	+0.061	+0.155	+0.104

Table 4.9: Results on test data for the single-threshold system. Differences in precision, recall and F_1-scores per class compared to the complex-label results.

ensemble for multi-label classification is clearly the better option.

The differences between the single-threshold ensemble and the dual-threshold one are minimal, and we suggest that given a properly calibrated ensemble with a single probability threshold, separately classifying and assigning a "no-emotion" label may not be very useful. This is also apparent from the probability thresholds found for the dual-threshold system. The threshold for any emotion probability was extremely close to the threshold found for the single-threshold system (0.23 vs. 0.22), and the certainty with which the "no-emotion" classifier had to predict the "no-emotion" label before the instance would be prohibited from gaining any emotion labels was very high (0.98). The number of instances affected by this second threshold is therefore minimal.

4.7 Conclusions

In conclusion, we have re-affirmed the power of character n-grams as features for emotion classification (as character n-grams also appeared to be useful in the previous shared task, see Chapter 3). We have also shown that it is possible to recognize emotions in suicide notes with respectable accuracy, on the condition that there is enough training data available for the given emotion. As soon as an emotion gets too specific and too rare in the training set, it is almost impossible for an SVM classifier to pick up on it. Finally, we have demonstrated that for this multi-label classification task, it is better to build an ensemble of classifiers that are specialized in a single emotion, as opposed to a multi-class classifier which considers combinations of more than

class	precision	recall	F_1
abuse	+0.000	+0.000	+0.000
anger	+0.173	+0.000	+0.016
blame	−0.007	−0.045	−0.046
fear	+0.000	+0.000	+0.000
forgiveness	+0.000	+0.000	+0.000
guilt	+0.125	+0.009	+0.043
happiness/peacefulness	+0.500	+0.000	+0.007
hopefulness	+0.068	−0.027	−0.034
hopelessness	+0.013	+0.136	+0.075
information	−0.023	+0.193	+0.089
instructions	−0.034	+0.262	+0.113
love	+0.138	+0.144	+0.144
pride	+0.000	+0.000	+0.000
sorrow	−0.143	−0.029	−0.049
thankfulness	+0.068	+0.177	+0.118

Table 4.10: Results on test data for the dual-threshold system. Differences in precision, recall and F_1-scores per class compared to the complex-label results.

one emotion as a single class label. A relatively straightforward way of building this ensemble is to use a single calibrated probability threshold which determines whether an emotion is probable enough to be included in the set of predicted labels. Adding an extra threshold for non-emotional sentences shows very little benefit.

Chapter 5

Model Portability

In Chapters 3 and 4 we examined two very different emotion datasets and showed that with the right techniques and the right features, it is possible to build classifiers that, at the very least, achieve significantly higher than baseline performance. We wouldn't go so far as to call their performance convincing, and we would only integrate them in the most undemanding of applications, but it is undeniable that, given enough training data, a classifier can learn to discriminate between the emotions contained in text.

A lingering question, however, is to what extent a model that was trained on a given dataset can be properly termed "an emotion classifier", with the implication that it can be applied to emotional text from any source. While evaluating the models trained on the datasets from deLearyous (Chapter 3) and from the Medical NLP Challenge (Chapter 4), we took great care not to overfit the models (and consequently, not to overestimate their performance) either by using a completely separate test set, or by cross-validating across the dataset, and by keeping the instances originating from a single document contained within either the training or test partition. We are confident that the reported scores are good reflections of the real-life performance of the classifiers, on the condition that the sentences to be classified are from the same domain as the sentences the classifier was trained on. What happens, however, if we try to classify a sentence that originates from, for instance, a blog corpus, with the classifier we trained on suicide notes? Will a classifier be able to generalize across different domains, or will performance plummet, indicating it is only sensible to apply a classifier to data from its own domain? Might it be possible to improve performance by training on data from a different domain when the available data from the target domain is especially skewed or noisy?

There has been significant research into cross-domain classification, and various techniques have been proposed to adapt models trained on their source domain to better fit a target domain. *Domain adaptation* is a known and much-studied subject in natural language processing (Blitzer et al., 2006; Daumé III & Marcu, 2006; McClosky, 2010), and while it is not our ambition to improve upon the state of the art, we will apply some existing domain adaptation techniques to see to what extent our emotion classifiers hold up when applied to data from different domains.

Our experiments will be limited to two English-language datasets[1], which will be described in Section 5.1. Section 5.2 will outline the different experiments and form hypotheses as to the expected outcome. Section 5.3 will list the results of the experiments. We will wrap up this chapter on cross-domain classification in Section 5.4, where we will analyze the experimental data in some detail, and draw conclusions.

5.1 Data description

To evaluate the generalizing power of our emotion classifiers and the portability of features for emotion classification across domains, we need two datasets originating from different sources. One of the datasets we will experiment on is the 2011 Medical NLP Challenge dataset, which formed the focus of our investigations in Chapter 4. As we described in greater detail in Section 4.1, this dataset contains written and transcribed suicide notes, which are labeled on the sentence level following a framework of 15 emotions.

To represent a different domain, we chose Aman & Szpakowicz's dataset of emotion-labeled blog posts (Aman & Szpakowicz, 2007). This dataset contains blog posts which were also annotated on the sentence level, using Ekman's six "basic" emotions (Ekman, 1971). The blog posts were collected by querying the web for posts containing one or more keywords from a set of seed words (a limited number of these seed words were selected per emotion. For *happiness*, for instance, the seed words were "happy", "enjoy", and "pleased"). Four annotators participated in the annotation of the resulting dataset. The data was annotated at the sentence level, and annotators were asked to assign one of the six Ekman emotion to the sentence (or, if no emotion was present, the *no emotion* label), and to mark those words that contributed to this decision. The possibility that a sentence contains more than one emotion was also taken into account, and such sentences were assigned the "mixed emotion" tag. Every sentence was annotated by two different annotators.
We chose to use only the so-called "gold-standard" subset of this dataset. This

[1]No Dutch experiments were carried out, since cross-domain classification requires two datasets labeled according to the same or comparable frameworks. To our knowledge, no second Dutch dataset that uses a framework comparable to Leary's Rose is publicly available.

gold standard dataset contains only the sentences on which all annotators agreed. Table 5.1 shows the distribution of sentences within this dataset.

label	count (percentage)
anger	179 (4%)
disgust	172 (4%)
fear	115 (3%)
happiness	536 (13%)
sadness	173 (4%)
surprise	115 (3%)
no emotion	2800 (68%)

Table 5.1: Distribution of sentences across the possible emotion labels in the gold standard subset of the blog dataset by Aman & Szpakowicz (2007).

The blog and suicide note datasets represent a similar task realized in strongly different domains. The highly emotionally charged suicide notes from the Medical NLP Challenge are very different from the more casual, often less intense blog posts.

To evaluate the performance of an emotion classifier across the two domains represented by these datasets, it is essential that the annotation scheme used in the datasets be the same, or at least equivalent. However, the 2011 Medical NLP Challenge dataset is annotated with 15 emotions, which do not entirely overlap with Ekman's "big six" emotions. We therefore make a selection of only those emotions that do overlap, or that have an equivalent in both annotation frameworks. Table 5.2 illustrates this procedure. The emotions we will consider in the experiments are written in bold.

Medical NLP labels	"big six" (blog data)
abuse	–
anger	**anger**
blame	–
–	disgust
fear	**fear**
forgiveness	–
guilt	–
happiness/peacefulness	**happiness**
hopefulness	–
hopelessness	–
information	–
instructions	–
love	–
pride	–
sorrow	**sadness**
–	surprise
thankfulness	–

Table 5.2: Comparison between the annotation schemes for the 2011 Medical NLP Challenge and the six "basic" emotions (Ekman, 1971). Overlapping emotions are bolded.

emotion label	suicide notes	blog posts
anger	69 (1.5%)	179 (4%)
fear	25 (0.5%)	115 (3%)
happiness	25 (0.5%)	536 (13%)
sadness	51 (1.1%)	173 (4%)
"other"	2012 (42.8%)	287 (7%)
no emotion	2513 (53.5%)	2806 (69%)

Table 5.3: Distribution of sentences across the 4 overlapping emotion labels in both datasets.

The end result of this matching procedure is illustrated in Table 5.3. We now have two datasets from different domains, structured for the same emotion classification task.

The fact that we are dealing with extremely skewed datasets in favor of sentences that have not been assigned an emotion, is even more obvious when we restrict the number of emotion classes to the four classes the datasets have in common. For both datasets, the *no emotion* sentences are in the majority. The "other" class, which contains all instances containing one of the emotions we do not consider for our cross-domain experiments, also makes up a sizable portion of the suicide notes dataset. The four emotion classes, which for our intents and purposes are the most relevant, have relatively few instances, which we expect will complicate the classification task significantly.

5.2 Experimental setup

To investigate the effect of out-of-domain sentences on our SVM-based emotion classifiers, we carry out four distinct types of experiments. The first type of experiment does no cross-domain testing whatsoever, simply training a classifier on a single domain, and testing it on sentences from the same domain. These experiment will provide us with a baseline we can use to estimate how much of a drop in performance accepting sentences from a different domain results in. We will use the same 6-fold cross-validation setup for the 2011 Medical NLP Challenge dataset as we did in Chapter 4 (with the important difference that we only consider the 4 emotions mentioned in Section 5.1), and the blog dataset will also be split into 6 randomized folds.

In a second set of experiments, we train a classifier on sentences from a source domain, and test on sentences from a different target domain. This means that in effect, we

report on the performance of a single classifier that was trained on the entirety of the source dataset, and tested on the entirety of the target dataset.

The third set of experiments evaluates the effect of adding a small sample of target-domain sentences to the training data. Just like in the previous experiment, we use the full source dataset to train the classifier, but we augment this data with one fold from the target dataset. Each of these augmented classifiers is thus trained on the entirety of the source domain, plus one fold from the target domain, and tested on the remaining 5 folds from the target domain.

In the final experiments, we apply a simple domain adaptation technique devised by Daume III (2007). This domain adaptation technique is easily understood, and starts from the principle that an instance which works well in the source domain may be a bad predictor in the target domain, and vice versa. Every feature in the feature space is tripled: one version of the feature will serve as a general cross-domain version, the second will denote the presence of the feature in the source domain, and the third will symbolize the presence in the target domain. The idea is that a classifier will be able to weight the same feature differently depending on their origins, reflecting its differing usage in different domains.

To illustrate this approach, imagine that we would like to build an emotion classifier that can handle data from different internet forums, including a forum on weight loss, and another on gaming. Consider, then, the problematic role of a verb like "to lose". It is clear that this feature will have a very different distribution across emotions in both forums. Losing a game is generally considered a negative, while losing weight is more a cause for joy. Of course, the exact interplay between the word and the author's emotion is strongly dependent on the context, but in general terms, we would not expect this feature to be given the same weight in both datasets.

The technique adopted by Daume III is to augment the feature space with domain-specific versions of each feature. The "general", original version of the feature will be realized in instances from both domains, but a "source" version will only be assigned a value for instances in the source domain, while a "target" version will only receive a value for instances from the target domain.

Imagine that our feature space contains only "lose", then the augmented feature space would contain:

$$lose,\ lose_{source},\ lose_{target}$$

And this is how sentences from the weight loss domain (the "source") will be transformed into instances:

"I managed to lose weight."
$lose$: 1, $lose_{source}$: 1, $lose_{target}$: 0

while sentences from the gaming forum (the "target" domain) will be constructed as follows:

"I always lose at the very last level."
lose: 1, $lose_{source}$: 0, $lose_{target}$: 1

The general version of the feature will allow the classifier to generalize over the two domains, but the presence of the domain-specific versions also allows a classifier to assign more importance to a given feature in a specific domain.

Note that this approach presupposes two things: 1) a part of the target data was annotated and included while training, and 2) it is known which domain each test sentence originates from.

In the context of our experiments, we apply this technique by training a classifier on the augmented instances from the full source dataset in addition to instances from a single fold from the target dataset, and we test on the augmented instances from the remaining 5 folds of the target dataset.

We will limit ourselves to three types of features, namely bag-of-words features, character n-grams and emotion lexicon entries —specifically, the NRC word-emotion association lexicon (Mohammad, 2012b) and the "Affect" category in the LIWC lexicon (Pennebaker et al., 2001).

5.3 Results

5.3.1 In-domain classification

We first list the results for the in-domain experiments. Table 5.4 shows the results for the 2011 Medical NLP dataset (Pestian et al., 2012), while Table 5.5 shows the results for the blog dataset (Aman & Szpakowicz, 2007). Note that the former results will not match the results from Chapter 4, for the simple reason that we are only taking 4 out of the possible 15 emotions into account (as we described above, we restrict our experiments to the emotions that overlap across datasets). We show only the top performing (in terms of macro-averaged F-scores) settings and feature representation for each feature type.

Performance on blog data is very respectable (Table 5.5), with character n-grams and especially LIWC lexicon features being the top performing feature types. Note that given the way the dataset was compiled (documents were queried using a limited set of seed keywords and subsequently annotated), it is possible that this high performance is in part due to the presence of some unnaturally predictive features. We will investigate this thoroughly in Section 5.4. A glance at Table 5.4 tells us that the in-domain performance on the Medical NLP dataset using this limited subset of

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (rel. freq.)	0.128 (±0.011)	0.620 (±0.011)	0.668 (±0.015)
character 5-grams (rel. freq.)	0.132 (±0.012)	0.641 (±0.017)	0.693 (±0.016)
LIWC 1-grams (rel. freq.)	0.135 (±0.027)	0.558 (±0.028)	0.479 (±0.028)
NRC 1-grams (tf-idf)	0.127 (±0.032)	0.545 (±0.032)	0.501 (±0.031)

Table 5.4: Results for the in-domain experiments on the suicide notes dataset. Macro-averaged and micro-averaged F_1-scores and their standard deviations for the top scoring feature representations only.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (binary)	0.490 (±0.027)	0.545 (±0.011)	0.778 (±0.008)
character 4-grams (tf-idf)	0.509 (±0.049)	0.561 (±0.040)	0.789 (±0.016)
LIWC 1-grams (tf-idf)	0.538 (±0.052)	0.565 (±0.039)	0.797 (±0.013)
NRC 1-grams (rel. freq.)	0.401 (±0.045)	0.456 (±0.035)	0.750 (±0.015)

Table 5.5: Results for the in-domain experiments on blog data. Macro-averaged and micro-averaged F_1-scores and their standard deviations for the top scoring feature representations only.

classes is less convincing. The negative (*no-emotion*) and "other" sentences are in the overwhelming majority, which accounts for the high accuracy and micro-averaged F-scores (many '*no emotion* and "other" guesses are indeed correct). The low macro-averaged F-scores show that the classifiers struggle to correctly identify instances in the smaller classes, which, unfortunately, are the classes we're most interested in. Character 5-grams and LIWC lexemes perform best, but in general the emotions we want to identify are much too sparse in the 2011 Medical NLP Challenge dataset, and we cannot expect a classifier to learn much useful information from it without some form of balancing.

5.3.2 Cross-domain classification

We will apply several techniques to improve the quality of the dataset, or to lessen the impact of the problems inherent in the dataset, in the next chapter (Chapter 6). However, for now, the low in-domain performance for this dataset makes for an interesting subject of study. In-domain performance is extremely low because of the sparse nature of the dataset, but what would happen if we attempted to classify the sentences in this 2011 Medical NLP Challenge dataset using a model trained on a less sparse dataset, like the blog data by Aman & Szpakowicz (2007)?

Table 5.6 shows the results of exactly this experiment. The blog data was used as training data, and the resulting model was applied to the suicide notes. No domain adaptation techniques were applied.

Even training a classifier on the more stable blog dataset does not help us classify the emotional instances in the suicide notes dataset. In fact, performance is worse

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (abs. freq.)	0.076 (±0.043)	0.044 (±0.008)	0.467 (±0.027)
character 6-grams (binary)	0.084 (±0.059)	0.028 (±0.010)	0.498 (±0.026)
LIWC 1-grams (tf-idf)	0.072 (±0.065)	0.039 (±0.017)	0.479 (±0.026)
NRC 1-grams (tf-idf)	0.054 (±0.031)	0.047 (±0.010)	0.453 (±0.022)

Table 5.6: Results for the cross-domain experiments with classifiers trained on the blog dataset and tested on the suicide note dataset. Macro-averaged and micro-averaged F_1-scores and their standard deviations for the top scoring feature representations only.

than when doing in-domain classification on the 2011 Medical NLP Challenge data directly. Performance on the four emotion classes only drops slightly, but most of the instances now find themselves put in the *no-emotion* class, resulting in a drop in micro-averaged F-scores and accuracy.

For completeness, we also show the results for the inverse experiment, where the classifier is trained on the suicide notes data, and applied to the blog data (Table 5.7).

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (tf-idf)	0.033 (±0.012)	0.056 (±0.027)	0.510 (±0.014)
character 3-grams (absolute frequency)	0.046 (±0.007)	0.102 (±0.009)	0.207 (±0.013)
LIWC 1-grams (binary)	0.080 (±0.026)	0.010 (±0.009)	0.213 (±0.016)
NRC 1-grams (absolute frequency)	0.057 (±0.016)	0.085 (±0.014)	0.320 (±0.015)

Table 5.7: Results for the cross-domain experiments with classifiers trained on the suicide note dataset and tested on the blog dataset. Macro-averaged and micro-averaged F_1-scores and their standard deviations for the top scoring feature representations only.

As expected, cross-domain performance using the suicide dataset as source data is nowhere near the performance of the in-domain experiments. Simply building a model on a source domain and expecting it to work on a different domain without modification is clearly too optimistic. The next set of experiments will therefore slightly adapt the model to the target data before attempting to classify target instances.

5.3.3 Cross-domain classification with partial target training data

The previous two experiments assumed a single annotated dataset from the source domain, and no annotated data from the target domain. But it is not unrealistic that someone who wanted to adapt an emotion classifier to a different domain would have some time to spend on annotating part of the target dataset. This is what is being simulated in the experiments summarized in Tables 5.8 and 5.9. The former table shows the performance of a classifier trained on the blog data with a single fold

of the suicide note data, and tested on the rest of the suicide data. The latter shows the performance of classifiers trained on the suicide note data supplemented with a fold of blog data, applied to the rest of the blog data.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (abs. freq.)	0.160 (±0.010)	0.407 (±0.023)	0.567 (±0.005)
character 5-grams (binary)	0.180 (±0.008)	0.465 (±0.020)	0.612 (±0.009)
LIWC 1-grams (binary)	0.113 (±0.013)	0.226 (±0.026)	0.518 (±0.009)
NRC 1-grams (binary)	0.105 (±0.011)	0.246 (±0.022)	0.513 (±0.007)

Table 5.8: Results for the cross-domain experiments with classifiers trained on the blog data supplemented with a single fold of the suicide note data, and tested on the remainder of the suicide notes. Macro-averaged and micro-averaged F_1-scores and their standard deviations for the top scoring feature representations only.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (tf-idf)	0.130 (±0.008)	0.157 (±0.009)	0.611 (±0.006)
character 5-grams (tf-idf)	0.108 (±0.025)	0.142 (±0.025)	0.651 (±0.015)
LIWC 1-grams (tf-idf)	0.213 (±0.015)	0.171 (±0.007)	0.299 (±0.016)
NRC 1-grams (tf-idf)	0.127 (±0.006)	0.129 (±0.007)	0.375 (±0.005)

Table 5.9: Results for the cross-domain experiments with classifiers trained on the suicide note data supplemented by a single fold of blog data, and tested on the remainder of the blog dataset. Macro-averaged and micro-averaged F_1-scores and their standard deviations for the top scoring feature representations only.

For cross-domain classification, adding part of the suicide note data to the blog data appears to lead to an improvement compared to using only blog data for training. This is not unexpected, as the classifier gains information about the target dataset. More interesting, however, is the fact that the classifiers based on token unigrams and character n-grams outperform all in-domain classifiers on the suicide note dataset. In other words, a model trained on blog data and a small part of the suicide note data performs better on suicide note data than a model that was trained exclusively on suicide note data. This is a surprising finding, in that it goes against the intuition that training a classifier on data from the same domain as the test data, will make for a better performing system. What we see in our experiments seems to be an exception to this rule. In order to create a "general-purpose" emotion classifier (should such a thing be possible), one is better off training a model on a general and better-balanced dataset (even if this data originates from a different domain) and to supplement it with a part of the target data, than to simply train a classifier on a target dataset that is noisy and imbalanced.

When we look at the inverse experiment, we see that adding part of the blog posts to the suicide note data does improve on the pure cross-domain results, but we are nowhere near in-domain performance for the blog dataset. The 2011 Medical NLP Challenge data (at least for the classification task we're attempting to solve) is clearly not an ideal dataset on which to build a general purpose emotion classifier, as

it appears to be very difficult to learn from, and any information a classifier extracts from the dataset will be too noisy to be of use.

5.3.4 Cross-domain classification with domain adaptation

The final pair of experiments mirrors the previous pair (we also supplement the source dataset with a fold from the target dataset before classifying the rest of the target data), but with the important difference that we modify the feature vectors to facilitate domain adaptation. As we explained in Section 5.1, we create source-specific and target-specific versions of each feature, so the classifier can weight each version individually. Table 5.10 shows the results where the suicide notes are our target data, whereas Table 5.11 shows the results of a classifier adapted to handle the blog data.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (tf-idf)	0.134 (±0.009)	0.550 (±0.012)	0.592 (±0.012)
character 4-grams (tf-idf)	0.138 (±0.011)	0.567 (±0.015)	0.600 (±0.011)
LIWC 1-grams (binary)	0.146 (±0.005)	0.401 (±0.039)	0.580 (±0.017)
NRC 1-grams (binary)	0.159 (±0.032)	0.440 (±0.059)	0.592 (±0.007)

Table 5.10: Results for the cross-domain experiments applying the domain adaptation technique proposed in Daume III (2007). Classifiers are trained on the blog dataset supplemented with a single fold of the suicide notes data, and tested on the remainder of the suicide notes. Macro-averaged and micro-averaged F_1-scores and their standard deviations for the top scoring feature representations only.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (tf-idf)	0.160 (±0.020)	0.226 (±0.018)	0.703 (±0.004)
character 4-grams (binary)	0.216 (±0.025)	0.284 (±0.014)	0.698 (±0.007)
LIWC 1-grams (tf-idf)	0.369 (±0.019)	0.403 (±0.020)	0.697 (±0.007)
NRC 1-grams (abs. freq.)	0.208 (±0.021)	0.256 (±0.031)	0.717 (±0.010)

Table 5.11: Results for the cross-domain experiments applying the domain adaptation technique proposed in Daume III (2007). Classifiers are trained on all suicide notes supplemented by a single fold of blog data, and tested on the remainder of the blog dataset. Macro-averaged and micro-averaged F_1-scores and their standard deviations for the top scoring feature representations only.

When comparing the results above (Tables 5.10 and 5.11) to the cross-domain experiments in Tables 5.8 and 5.9, we see a significant improvement in almost all cases (the token unigrams and character n-grams in Table 5.10 seem to be the exceptions). The most striking improvements are to be found when we use the blog data as target data. This makes sense intuitively, since in-domain performance on the blog dataset was high, indicating that the dataset contains strongly predictive features. Creating source and target-specific versions of every feature allows our cross-domain classifiers

to put more emphasis on these salient keywords from the blog dataset, thus boosting cross-domain performance. We will look at how exactly the SVMs weights the domain-specific feature realizations in Section 5.4.2.

5.4 Analysis

5.4.1 Examining the unmodified feature space

anger	fear	happiness	sadness
mad	scared	fun	miss
angry	nervous	hahah	sad
bitch	afraid	lol	hurt
annoyed	mistake	laughing	awful
annoying	dangerous	lovely	guilty
shut	scares	haha	longing
hell	worry	exciting	missed
frickin	terror	happy	cried
yelling	worried	enjoy	depression
fuck	shudder	proud	sorry

Table 5.12: Highest-weighted features per class in the blog data (Aman & Szpakowicz, 2007)

The first result that needs to be looked into is the sharp contrast between the high in-domain scores for the blog dataset and the low in-domain scores for the suicide note dataset. The former scores show the classifiers performing admirably, with macro- and micro-averaged F-scores sitting comfortably around the 50% mark[2]. Table 5.12 shows the most heavily weighted token unigram features as determined by the classifier.

The selected features seem eminently sensible for the corresponding emotion. So sensible, even, that we have to wonder if these features being rated so highly isn't a result of the data collection process followed by Aman & Szpakowicz (2007). Indeed, and we quote, they "(...) took words commonly used in the context of a particular emotion. Thus, [they] chose "happy", "enjoy", "pleased" as seed words for the *happiness* category, "afraid", "scared", "panic" for the *fear* category, and so on. Next, using the seed words for each category, [they] retrieved blog posts containing one or more of those words." While there will not be an exact match between the retrieved

[2]We reiterate here that the averaged f-scores do not take the *no-emotion* class into account. (See also Appendix A.2) Since our focus is on accurately identifying the emotional sentences, we believe that this method of calculating the scores is best suited to the task at hand.

blog posts and the annotated emotions, the heavily weighted features in Table 5.12 may very well be these seed words. This means the results are overestimating the performance of the classifier on real-life data, which will of course not always contain these seed terms. Regardless of this problem, we have seen that the blog dataset tends to make for classifiers that generalize better in cross-domain experiments than the suicide note dataset. The strongest features do tend to be very general and representative of the corresponding emotion, irrespective of the domain.

anger	fear	happiness	sadness
single	treatments	pleasure	thieves
sucker	scared	loveliest	punished
haunts	ashame	handkerchiefs	stinker
bitterly	craze	rich	needy
neglect	severe	loveliness	numbetal
truthful	afraid	dwelling	perfection
lousy	determinated	finer	succeed
thrilled	happend	mourn	failure
easier	arguments	peaceful	sadness
badge	terror	contentment	wone

Table 5.13: Highest-weighted features per class in the 2011 Medical NLP Challenge dataset (Pestian et al., 2012)

Table 5.13 shows the top LIWC features (the best performing feature type for the task) in the 2011 Medical NLP Challenge dataset[3]. The link between the features and the emotions is much less obvious than it was for the blog data. We see that quite a few top features seem very domain-specific, such as "treatments" which is highly associated with *fear*. This association seems plausible considering the fact that many of the authors of the suicide notes in the dataset were afflicted with incurable diseases, where going through treatment is a thoroughly unpleasant experience. However, this strong link with fear is much less plausible in any other context. Phenomena like this go a long way towards explaining why the suicide dataset is not of much help in cross-domain experiments. The data sparseness problem (remember that less than 4% of the sentences are associated with one of the four selected emotion labels) is exacerbated by the fact that those features that do help identifying the few emotional instances in the dataset, are themselves very domain-specific. Any model trained on the 2011 Medical NLP Challenge dataset will thus not generalize well over data from other domains.

[3]Note that even though the features in this experiment originate from a fixed lexicon, it is possible for them to contain errors and typos, as is apparent in the features "ashame", "determinated", or "happend". The reason for this is that the LIWC lexicon contains word *patterns* and not just fully realized words. The pattern "determin*" will thus match "determined" or "determining", but it will also match the grammatically incorrect "determinated".

5.4.2 Examining domain-specific features in the modified feature space

We will now turn to the feature sets that were supplemented using the domain adaptation technique proposed in Daume III (2007). As we explained in Section 5.2, each feature is tripled, meaning that the feature space contains one general version of each feature, one version for the source domain, and one version for the target domain. This allows our SVM classifier to weight each version differently, thus accentuating or downplaying the importance of a given feature in each domain.

Tables 5.14 and 5.15 focus on the domain adaptation task where the blog posts are used as source data, and the suicide notes are seen as the target dataset. They show the NRC features (the top performing feature type for this task, see Table 5.10) for which the weights for the source and target versions were most different. This means that the classifier determined that these features should have a very different effect depending on the domain the instance originates from.
Tables 5.16 and 5.17, then, show the most divergent LIWC features (the top performing feature type, see Table 5.11) in a classifier trained on a source corpus of suicide notes, and supplemented with a part of the target blog data.

anger				**fear**			
feature	gen wgt	src wgt	tgt wgt	feature	gen wgt	src wgt	tgt wgt
mad	2.281	2.563	-0.283	nervous	2.511	2.928	-0.417
worse	1.136	1.716	-0.580	afraid	2.071	2.635	-0.564
feelings	0.772	-0.740	1.512	fear	1.292	1.889	-0.597
found	-1.789	-1.789	0.000	worried	1.916	2.072	-0.155
show	0.556	-0.555	1.111	children	-1.789	-1.789	0.000
miserable	0.332	-0.615	0.946	child	-1.789	-1.789	0.000
children	0.554	1.047	-0.493	subject	-1.232	-1.232	0.000
clean	-1.512	-1.512	0.000	fire	0.688	0.885	-0.197
fall	-1.494	-1.494	0.000	faith	0.662	0.870	-0.208
rule	0.768	-0.301	1.070	hospital	0.596	0.755	-0.159

Table 5.14: Features with the highest difference in weight between their general, source, and target version in the "anger" and "fear" classes. The source is the blog dataset, the target is the suicide notes dataset.

Extending the feature set with source- and target-specific features had a significant positive effect when the blog dataset was the source and the suicide notes were the target. The features in Tables 5.14 and 5.15 correspond to this experiment. We see the classifier attempting to attribute different weights to features used in the target dataset, but it is hard to make sense of some of these changes. The term "mad", which at first glance is a good predictor of the *anger* class, originally has a relatively high weight, which is downplayed considerably for the target realization. The "mad" feature in the target domain thus acquires a negative weight for the *anger* class, which seems to make little sense. When we look at the presence of the word in both datasets, though, we see that the classifier is effectively capturing some oddities

happiness				sadness			
feature	gen wgt	src wgt	tgt wgt	feature	gen wgt	src wgt	tgt wgt
love	1.233	2.885	-1.652	cry	1.958	2.905	-0.947
enjoy	1.475	2.835	-1.360	guilty	1.499	2.267	-0.768
loving	1.518	2.847	-1.329	hurt	1.627	2.142	-0.515
wonderful	1.452	2.790	-1.338	awful	2.134	2.393	-0.259
happy	2.092	3.061	-0.969	upset	0.650	1.628	-0.978
glad	1.364	2.608	-1.244	longing	1.692	2.076	-0.384
proud	1.124	2.459	-1.335	depression	1.726	2.042	-0.316
loved	1.137	2.447	-1.310	crying	2.090	2.090	0.000
happiness	1.156	2.431	-1.275	pain	1.239	1.634	-0.396
loves	1.443	2.439	-0.996	hurts	0.562	1.267	-0.706

Table 5.15: Features with the highest difference in weight between their general, source, and target version in the "happiness" and "sadness" classes. The source is the blog dataset, the target is the suicide notes dataset.

inherent to the data. In the blog dataset, all 5 sentences that contain the word "mad" have been assigned the *anger* label. Strangely, this includes an instance where the word is not used in its most common sense of "angry", and where we believe the *anger* label has been assigned erroneously:

"But I, with my mad Photoshop skills, was able to convert it to the corect color space while faithfully maintaining the appearence and detail of the image, including the gradations."

More of such examples can be found throughout the feature set. The word "happiness", which gets a high rating in the blog data but which is significantly downgraded in the suicide note dataset, is similarly over-associated with the *happiness* label in the blog dataset. In the 2011 Medical NLP Challenge dataset, the presence of the word "happiness" is not nearly as strongly correlated with the *happiness* class, as it is often used while wishing happiness to other people, or to express gratitude for happiness caused by others:

"Mary Smith has never let me down I love them all and wish them happiness and joy ."

These examples show that using source and target-specific versions of each feature does successfully help adapt the classifier to the target dataset. The adaptations may not always be intuitive, but they do capture the quirks of the datasets very well.

Tables 5.16 and 5.17 show the most ambiguous features for the experiment where the suicide data is the source, and where the blog data is the target. This specific experiment was highly successful, as we saw a big boost in performance compared to a cross-domain classifier where the domain adaptation technique was not applied. We see the classifier "rectifying" feature weights to better fit the target dataset, and the target scores do seem more in-line with our intuition than the source scores.

anger				fear			
feature	gen wgt	src wgt	tgt wgt	feature	gen wgt	src wgt	tgt wgt
well	-0.385	-1.852	1.682	have	-4.692	-4.572	-0.667
love	-3.816	-3.616	-0.514	hate	-0.401	-1.478	1.757
mad	0.897	-0.976	2.080	love	-3.722	-3.520	-0.515
fucking	1.566	-0.696	2.200	good	-2.764	-2.681	-0.265
played	0.212	1.558	-1.209	afraid	1.084	-0.223	1.705
worse	0.148	-1.180	1.565	care	-2.233	-2.050	-0.364
like	-1.881	-2.282	0.325	best	-2.063	-1.932	-0.277
blame	-0.992	-1.566	0.843	nervous	2.814	0.786	2.419
lost	-0.665	-1.382	0.919	made	-1.869	-1.785	-0.206
innocent	0.984	-0.523	1.774	funeral	-1.616	-1.603	-0.037

Table 5.16: Features with the highest difference in weight between their general, source, and target version in the "anger" and "fear" classes. The source is the suicide notes dataset, the target is the blog dataset.

happiness				sadness			
feature	gen wgt	src wgt	tgt wgt	feature	gen wgt	src wgt	tgt wgt
love	-2.323	-3.971	3.642	good	-1.973	-3.193	1.957
good	-1.363	-3.170	2.970	bad	-0.759	-1.842	1.603
loved	-0.829	-2.145	2.181	pain	-0.128	-1.457	1.891
loves	0.418	-1.598	2.622	lost	-0.267	-1.566	1.698
nice	-0.011	-1.853	2.278	worry	-0.163	-1.311	1.675
have	-3.983	-4.005	-0.283	hand	-0.481	-1.471	1.298
had	-1.888	-2.726	0.944	grand	0.642	-0.839	1.845
wonderful	-0.515	-1.605	1.801	lose	0.174	-1.108	1.533
like	-1.778	-2.564	0.830	nervous	0.921	1.713	-0.770
better	-1.126	-2.033	1.320	best	-2.562	-2.476	-0.214

Table 5.17: Features with the highest difference in weight between their general, source, and target version in the "happiness" and "sadness" classes. The source is the suicide notes dataset, the target is the blog dataset.

This can be seen as a reflection of the fact that the blog dataset contains more representative data, and that the emotion categories are less sparse, but it is also in part a consequence of the fact that it was compiled based on a number of emotion keywords which turn out to be both intuitively pleasing and highly predictive.

5.5 Conclusions

What can we conclude from these results? For one, that building a classifier that identifies "emotion" in general, domain-independently, is extremely difficult. Good in-domain performance for an emotion classifier is probably due to properties of the data which will likely not exist in a dataset from a different domain. In our case, the solid performance on the blog data by Aman & Szpakowicz (2007) was due to the way the data was gathered. The classifiers seemed to pick up on the keywords that were used to query blogs for suitable posts. When applying our model to a different

dataset, the performance dropped dramatically, since these cues were not present in the data from the target domain.

Secondly, there seems to be no "killer" feature type that assures good performance across different domains. Lexicon-based features seem to fare relatively well, but this may in part be a consequence of the aforementioned use of emotion keywords during data construction. LIWC features specifically seemed to hold their own, despite the fact that the "affect" portion of the LIWC lexicon is relatively small. The NRC lexicon is much larger, and should theoretically capture more of the sentences, but the results are less convincing. The small, finely-honed set of emotional keywords from the LIWC lexicon ensures that the model trained on one domain will generalize to some extent to the target domain.

Perhaps the most interesting result originating from the series of experiments carried out in this chapter, is that it is possible to build a cross-domain classifier that outperforms an in-domain classifier built on target data. These findings are in line with the research described in Mohammad (2012a), who used a classifier trained on their Twitter Emotion Corpus to —after applying the same domain adaptation technique described in Daume III (2007)— improve the classification performance of documents from a different domain. When a dataset from one domain is excessively sparse or skewed, it may be better to train a classifier on a better-balanced and more general and representative dataset. The blog dataset was a good example of this. The fact that it was constructed based on a few emotion keywords (which in in-domain experiments favors overfitting) might actually work in its favor when we use it as training data for cross-domain experiments. The documents contained in the dataset are very typical of the emotions they are labeled with, which makes it easier for a classifier trained with these documents to generalize over other domains.

Applying domain adaptation techniques such as the technique proposed by Daume III (2007) can serve two purposes. First, it improves cross-domain classification performance by allowing the classifier to set source and target-specific weights for each feature, which makes it possible for the model to preempt the specificities of the target data. Secondly, examining the features for which the source and target weights diverge the most is also an excellent way to discover quirks and oddities in either of the datasets. In our case, highly diverging feature weights appeared to not always be a consequence of the different use of certain terms depending on their contexts (medical "treatments" being a source of fear in the suicide note dataset, while there was no such association in the blog data), but they sometimes unveiled inconsistencies in the annotation of the data.

To sum up, if one sets out to build an emotion classifier that will "work" on a wide range of data, it is recommended to use source data that is as prototypical as possible, and features that can capture prototypical emotion features. While this may not always be possible, it is recommended to add a set of documents from the

target dataset to the training set, as this will significantly improve cross-domain classification performance. It is also recommended to apply domain adaptation prior to classification, as this procedure (which can be relatively simple) will have a positive impact in most cases.

Chapter 6

Noise Reduction

In Chapter 5, we concluded that building a general-purpose "emotion classifier" is extremely difficult, as a classifier that is built to function in a given domain, on a specific dataset, is likely to perform poorly on data from a different domain or dataset. We argued that the generalizing power of an emotion classification model is likely to improve when the documents are represented using general, domain-independent features —in our case, the affective keywords included in the LIWC lexicon. We also saw better performance when the source dataset itself was more stable and contained less noise.

It is this latter factor —the interplay between noise in the dataset and classifier performance— that we are concerned with in this chapter. We attempt to filter out noise from the training dataset prior to model construction, in the hope of building more stable and reliable classifiers.

6.1 Problem description

The 2011 Medical NLP Challenge dataset (Pestian et al., 2012) is an especially difficult dataset to work with, certainly when we only consider those four emotions which overlap with the "Big Six" (Ekman, 1971), as we did in Chapter 5. Its class distribution is highly skewed, with the unlabeled sentences making up 50% of the dataset. Even then, only a small part of the remaining sentences (3.6% of the dataset) belong to retained emotion classes ("anger", "fear", "happiness", and "sadness"), with the other being relegated to a catch-all "other" class for the purposes of our experiments (Table 6.1). To compound the problem, the majority set of unlabeled instances cannot be said to belong to a single "no emotion" class, as it also contains

sentences on which annotators failed to agree. These unlabeled and "other" instances seem like they might be a source of confusion for the classifiers, as even intuitively, the "no emotion" and "other" classes are not clearly delineated, and instead form a cloud of mostly emotionally unrelated text fragments.

anger	69 (1.5%)
fear	25 (0.5%)
happiness	25 (0.5%)
sadness	51 (1.1%)
other	2012 (42.8%)
no label	2513 (53.4%)

Table 6.1: Distribution of sentences in the 2011 Medical NLP dataset (Pestian et al., 2012).

The blog dataset by Aman & Szpakowicz (2007) (Table 6.2) also has quite a few unlabeled instances, but the problem is less acute. The four emotion classes are better represented (taking up a quarter of the instances), and the "no-emotion" class is clearly agreed upon. However, the subjective nature of an emotion classification task makes noise virtually unavoidable, and we might see some improvement using noise-reduction techniques on this dataset as well.

anger	179 (4%)
fear	115 (3%)
happiness	536 (13%)
sadness	173 (4%)
other	287 (7%)
no-emotion	2800 (69%)

Table 6.2: Distribution of sentences in the blog dataset by Aman & Szpakowicz (2007).

In the next section (Section 6.2), we will try several approaches to minimize the influence of noise on the classifiers, or to minimize the amount of noise in the dataset prior to classification.

6.2 Experiments

We will compare three different techniques to automatically reduce the (effect of) noise in the training dataset, hopefully yielding more stable, generalizable models. The first technique (Section 6.2.1) is the least invasive approach, as it does not alter the training data in any way. Instead, it simply weights up or down individual classes,

boosting the small emotion classes, and downplaying the large, noisy classes. The second technique (Section 6.2.2) is to apply hierarchical classification. A first binary classifier determines which instances contain emotion, and a second one —trained on only the emotion-labeled instances— determines the exact emotion of the instances that were flagged by the binary classifier. The third approach (Section 6.2.3) makes a selection of training instances prior to model construction, keeping only those instances that are highly typical of their class, thereby eliminating instances that are not representative and –so we hypothesize– only add noise.

6.2.1 Class weighting

Before we start complicating matters by building classifier ensembles or resampling training instances, we resort to the options that are available to us without further modification. The package we use for instance creation, LibShortText (Yu et al., 2012) (which we modified slightly, as described in Appendix A.1), is built around the LibLinear package for classification (Fan et al., 2008). It is thus possible to use most of LibLinear's options, one of which is to modify the cost parameter for individual classes. The cost parameter C determines the "hardness" of the boundary between classes. A lower value of C will make for a softer separation between instances of opposing classes, even if this means misclassifying more training instances. A higher C will attempt to classify more training items correctly, resulting in a harder separation between items of differing class.
LibLinear's default behavior is to use the same value for the cost parameter C, irrespective of class, but for unbalanced data it may be interesting to increase the value of the C parameter for underrepresented classes. This will force the classifier to try and fit the model to classify more minority instances correctly, which in the case of a highly skewed data distribution increases the likelihood of those smaller classes being predicted.

Experimental setup

Unfortunately, there is no ready-made answer to the question how strongly one should weight up the minority classes. We therefore incorporate a rough search for acceptable class weights in an optimization step. We (somewhat arbitrarily) weight the C values of the minority classes with exponents of 10, i.e. 10, 100, 1000[1]. Instead of simply doing a grid search for an acceptable value of C that will apply to every class, we thus additionally search for a weight factor to apply to the cost parameter of the smaller emotion classes.

[1]Note that since we already have results for unmodified C values, the default weight is 1 was not included in our search space.

Results and analysis

Tables 6.3 and 6.4 show the classification results after applying extra weight to the four emotion classes. Note that these results are entirely comparable to the in-domain results in Chapter 5 (Tables 5.4 and 5.5), as we use the same data and investigate the same feature types and feature representations. For convenience, we compare the weighted results to the unweighted results in Figures 6.1 and 6.2.

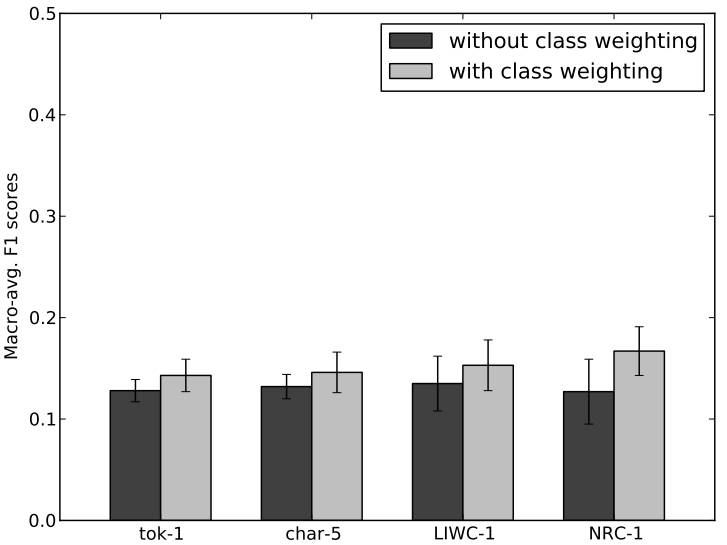

Figure 6.1: Results for in-domain experiments on the suicide note dataset, with and without class weighting.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (rel. freq.)	0.143 (±0.016)	0.571 (±0.011)	0.632 (±0.019)
character 5-grams (rel. freq.)	0.146 (±0.020)	0.622 (±0.016)	0.684 (±0.021)
LIWC 1-grams (rel. freq.)	0.153 (±0.025)	0.542 (±0.022)	0.463 (±0.023)
NRC 1-grams (tf-idf)	0.167 (±0.024)	0.538 (±0.030)	0.495 (±0.031)

Table 6.3: Results for the in-domain experiments on the suicide notes dataset after applying higher weights to the emotion classes.

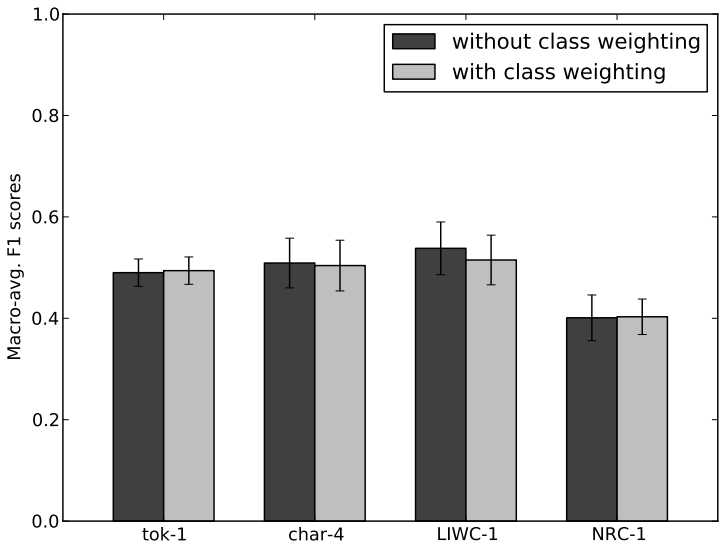

Figure 6.2: Results for in-domain experiments on the blog data, with and without class weighting.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (binary)	0.494 (±0.027)	0.550 (±0.019)	0.783 (±0.007)
character 4-grams (tf-idf)	0.504 (±0.050)	0.558 (±0.038)	0.792 (±0.016)
LIWC 1-grams (tf-idf)	0.515 (±0.049)	0.550 (±0.030)	0.778 (±0.014)
NRC 1-grams (rel. freq.)	0.403 (±0.035)	0.440 (±0.033)	0.715 (±0.008)

Table 6.4: Results for the in-domain experiments on blog data after applying higher weights to the emotion classes.

The difference in performance is hardly convincing, on either dataset. We see some improvement where the class imbalance is the greatest (on the 2011 Medical NLP Challenge Dataset), but the difference is not significant. On the blog data, where the class imbalance is less severe, the scales sometimes even tilt towards the negative. Fitting the model too tightly on training data results —not unexpectedly— in lower performance on the test data.

More interesting for us is to see in what way adapting misclassification costs per class can influence the generalizing power of a classifier. Tables 6.5 and 6.6 show how the cross-domain classifiers from Chapter 5 (Tables 5.6 and 5.7) fare when we apply individual class weights. Their performance are compared to the unweighted results in Figures 6.3 and 6.4.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (abs. freq.)	0.072 (±0.047)	0.044 (±0.012)	0.448 (±0.022)
character 6-grams (binary)	0.078 (±0.060)	0.013 (±0.005)	0.497 (±0.025)
LIWC 1-grams (tf-idf)	0.057 (±0.039)	0.037 (±0.014)	0.440 (±0.032)
NRC 1-grams (tf-idf)	0.047 (±0.035)	0.028 (±0.010)	0.415 (±0.021)

Table 6.5: Results for cross-domain experiments trained on blog data and tested on suicide notes after applying higher weights to the emotion classes.

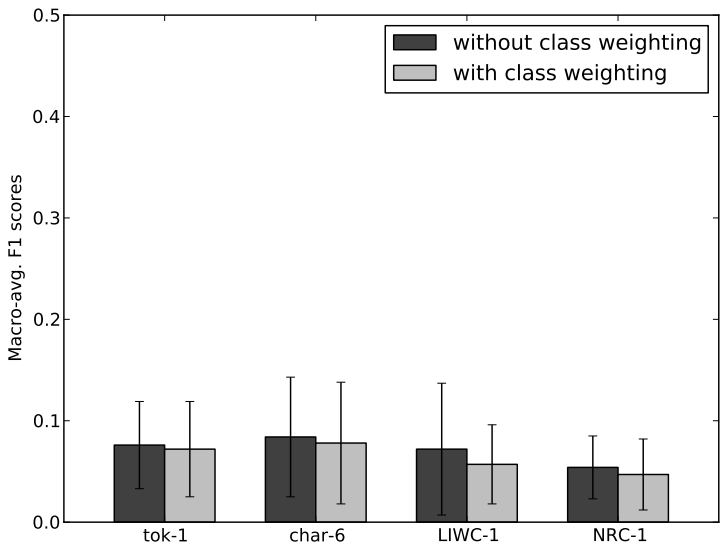

Figure 6.3: Results for cross-domain experiments trained on blog data and tested on suicide notes, with and without class weighting.

For the cross-domain experiments, we see a similar pattern emerge. Setting individual class weights does not benefit classifiers trained on the blog data, but there is a slight

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (tf-idf)	0.092 (±0.015)	0.089 (±0.017)	0.475 (±0.018)
character 3-grams (abs. freq.)	0.066 (±0.022)	0.066 (±0.020)	0.557 (±0.021)
LIWC 1-grams (binary)	0.080 (±0.027)	0.093 (±0.008)	0.217 (±0.016)
NRC 1-grams (abs. freq.)	0.057 (±0.016)	0.085 (±0.014)	0.320 (±0.015)

Table 6.6: Results for cross-domain experiments trained on suicide notes and tested on blog data after applying higher weights to the emotion classes.

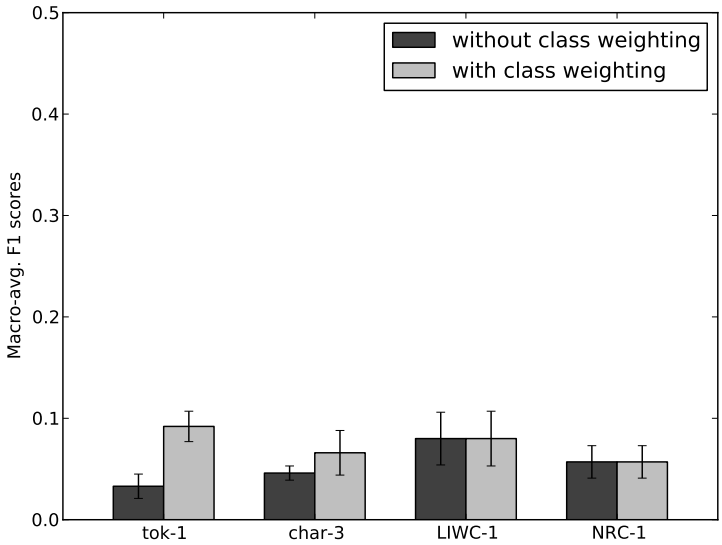

Figure 6.4: Results for cross-domain experiments trained on suicide notes and tested on blog data, with and without class weighting.

improvement (and in the case of token unigrams, a significant improvement) when we apply higher weights to the underrepresented classes in the suicide notes corpus. Class weighting is clearly not a surefire way of improving the generalizing power of a classifier. This is not entirely unexpected, as what works best on the source dataset (increasing the weights of the underrepresented classes) inevitably results in harder class boundaries, and thus in a more tightly fitted model. This model, while it might work well on similar data, is less likely to work on a dataset with different properties.

6.2.2 Hierarchical classification

Conceptually speaking, identifying the emotions as represented in discrete emotion models such as Ekman's (1971) is a two-step process. Not every utterance contains an emotion[2], so the first step is to determine *if* an utterance is emotional, after which we try to determine *which* emotion is reflected in the utterance.

This conceptual hierarchy is however not reflected in the way we build our classifiers: the task for the classifier is simply to assign an instance to one of several classes, either to one of the emotions, or to one of the larger and more loosely defined "other" or "no-emotion"-classes. As we described in Section 6.2.1, this often results in these larger classes soaking up instances from the underrepresented classes. Another approach to avoiding this problem is to follow the conceptual idea of a hierarchical classifier, and to determine the class of a document in two steps: a top-level classifier will determine whether the document contains an emotion, and the bottom-level classifier will determine which emotion is present in the document. Figure 6.5 gives a schematic representation of this setup.

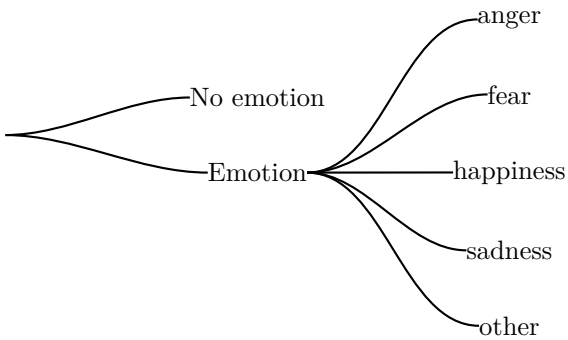

Figure 6.5: Structure of the the hierarchical classifier ensemble.

Since "hierarchical classification" is a term that has many interpretations, and because even given a single interpretation, there are many possible implementations (see Silla & Freitas (2011) for a survey of hierarchical classification across different application domains), we feel it is necessary to explicitly describe what it is we do when we say we build a "hierarchical classifier".

Our system consists of two classifiers, one for each level in the hierarchy. The first classifier is a binary one, and is trained to separate emotional from non-emotional sentences. This "top-level" classifier is trained using the entirety of the training set, but the four emotion classes and the "other" class are collapsed into one large "emotion"

[2]Note that this cannot be said when adopting a dimensional model such as Leary's Interpersonal Circumplex (Leary, 1957), since following these models, no utterances are truly ever emotion-free.

category. The second classifier is a multi-class classifier[3], and it is trained on *part of* the training set, as the instances that were left unlabeled (the "no-emotion" instances) are not included. The training set for this "bottom-level" classifier thus only contains instances that were annotated as belonging to the "anger", "fear", "happiness", "sadness", or "other" classes. Both classifiers are trained independently.

During the test phase, an instance is first classified using the top-level classifier, which determines if the sentence is emotional. Instances that are classified as being non-emotional do not get assigned a label, and it is only when this first classifier predicts "emotion" that we run the sentence through the bottom-level classifier to determine the final emotional label.

Experimental setup

There are several ways of calibrating our hierarchical classifiers. In the first classification step, we can either focus on the precision or on the recall for the "emotion" class. If we build a high-precision binary classifier for the top level, the advantage is that we can be reasonably confident that the sentences that get sent to the bottom-level classifier are indeed emotional. The downside of this approach is that we might discard quite a few emotional instances in the first step, which can never be retrieved by the bottom-level classifier. If we instead build a high-recall top-level classifier, we might label *too many* documents as emotional, which will result in the over-generation of emotion labels in the second step. In the following section (Section 6.2.2), we experiment with both of these options.

Results and analysis

Table 6.7 shows the performance of the top-level binary classifiers ("emotion" vs. "no-emotion") that were optimized for precision on the "emotion" class[4]. Table 6.8 shows the performance of the bottom-level classifier, which determines whether the emotion a sentence contains is one of "anger", "fear", "happiness", or "sadness", or if it belongs to the catchall "other" class. This classifier is tuned for recall on the four emotion classes[5], as the predictions from the top-level classifier should serve as an adequate initial filter for emotional instances. For this classifier, on the 2011 Medical NLP Challenge dataset, we were forced to apply the class weighting techniques as outlined in Section 6.2.1, since without these cost corrections, all instances would be

[3]Note that internally, libLinear builds n one vs. all classifiers for a n-way multi-class problem. A test instance is classified by every binary classifier, and the label that has the highest likelihood is returned.

[4]The classifier was calibrated to have a high $F_{0.5}$ score for the "emotion" class, which favors precision while still taking recall into account.

[5]High macro-averaged F_2 score for the four emotion classes.

attributed to the "other" class, which is by far the most common category. The results in Table 6.9 show the performance of the full hierarchical setup: only the instances that were classified as belonging to the "emotion" class by the top-level classifier are run through the bottom-level classifier to determine their exact emotion.

	suicide notes	blog posts
feature type (representation)	character 5-grams (binary)	character 3-grams (tfidf)
"emotion" precision	0.704	0.791
"emotion" recall	0.575	0.526
"emotion" $F_{0.5}$	0.674	0.719

Table 6.7: High-precision results for the top-level classifier on both datasets.

	suicide notes	blog posts
feature type (representation)	character 3-grams (rel. freq.)	token 1-grams (tfidf)
"anger" precision	0.025	0.181
"anger" recall	0.085	0.531
"anger" F_2	0.043	0.383
"fear" precision	0.000	0.276
"fear" recall	0.000	0.513
"fear" F_2	0.000	0.438
"happiness" precision	0.000	0.245
"happiness" recall	0.000	0.909
"happiness" F_2	0.000	0.589
"sadness" precision	0.051	0.221
"sadness" recall	0.125	0.509
"sadness" F_2	0.073	0.404
"other" precision	0.442	0.169
"other" recall	0.959	0.571
"other" F_2	0.653	0.387

Table 6.8: High-recall results for the bottom-level classifier on both datasets.

	suicide notes	blog posts
"anger" precision	0.000	0.494
"anger" recall	0.000	0.229
"anger" F_1	0.000	0.313
"fear" precision	0.000	0.809
"fear" recall	0.000	0.330
"fear" F_1	0.000	0.469
"happiness" precision	0.000	0.669
"happiness" recall	0.000	0.595
"happiness" F_1	0.000	0.630
"sadness" precision	0.125	0.624
"sadness" recall	0.094	0.335
"sadness" F_1	0.107	0.436
"other" precision	0.687	0.541
"other" recall	0.576	0.296
"other" F_1	0.626	0.383
micro-avg. F_1 (±stdev.)	0.600 (±0.010)	0.504 (±0.026)
macro-avg. F_1 (±stdev.)	0.147 (±0.021)	0.446 (±0.046)

Table 6.9: Results for hierarchical classifier ensembles with a high-precision top-level classifier, and a high-recall bottom-level classifier.

The performance of top-level classifiers optimized for emotion recall[6] is shown in Table 6.10. The bottom-level classifiers in Table 6.11 focus on precision for the four emotion classes[7], to compensate for the non-emotional sentences that will inevitably slip through the mazes of the top-level classifier. The results of combining both levels can be seen in Table 6.12.

	suicide notes	blog posts
feature type (representation)	character 5-grams (tf-idf)	character 3-grams (abs. freq.)
"emotion" precision	0.654	0.703
"emotion" recall	0.665	0.623
"emotion" F_2	0.663	0.638

Table 6.10: High-recall results for the top-level classifier on both datasets.

	suicide notes	blog posts
feature type (representation)	character 3-grams (binary)	character 4-grams (rel. freq.)
"anger" precision	0.043	0.306
"anger" recall	0.043	0.207
"anger" $F_{0.5}$	0.043	0.279
"fear" precision	0.000	0.667
"fear" recall	0.000	0.157
"fear" $F_{0.5}$	0.000	0.404
"happiness" precision	0.059	0.157
"happiness" recall	0.045	0.974
"happiness" $F_{0.5}$	0.050	0.189
"sadness" precision	0.000	0.317
"sadness" recall	0.000	0.225
"sadness" $F_{0.5}$	0.000	0.293
"other" precision	0.438	0.242
"other" recall	0.985	0.411
"other" $F_{0.5}$	0.492	0.263

Table 6.11: High-precision results for the bottom-level classifier on both datasets.

Analysis

Table 6.13 compares the averaged scores of the hierarchical classifiers to the best "flat" classifiers described in Chapter 5.

The results of the hierarchical classification experiments are interesting in that they amplify the phenomenon we observed when we used class weighting. The suicide note dataset seems to benefit from the hierarchical setup, while the classifiers trained on blog data take a significant hit. The explanation for this is again to be found in class imbalance and data sparseness. The suicide dataset has few examples from the four considered emotion categories, and the "flat" classifier would assign fewer emotion

[6] Based on F_2 scores for the "emotion" class.

[7] Classifiers were ranked on the basis of macro-averaged $F_{0.5}$ scores on the four basic emotion classes.

	suicide notes	blog posts
"anger" precision	0.062	0.651
"anger" recall	0.021	0.156
"anger" F_1	0.032	0.252
"fear" precision	0.000	1.000
"fear" recall	0.000	0.148
"fear" F_1	0.000	0.258
"happiness" precision	0.200	0.407
"happiness" recall	0.045	0.688
"happiness" F_1	0.074	0.512
"sadness" precision	0.000	0.686
"sadness" recall	0.000	0.139
"sadness" F_1	0.000	0.231
"other" precision	0.638	0.573
"other" recall	0.675	0.286
"other" F_1	0.656	0.381
micro-avg. F_1 (±stdev.)	0.633 (±0.018)	0.427 (±0.030)
macro-avg. F_1 (±stdev.)	0.152 (±0.029)	0.327 (±0.041)

Table 6.12: Results for hierarchical classifier ensembles with a high-recall top-level classifier, and a high-precision bottom-level classifier.

	suicide notes		blog posts	
	macro-avg. F_1	micro-avg. F_1	macro-avg. F_1	micro-avg. F_1
"flat" classifier	0.143	0.558	0.538	0.565
hierarchy (high rec. top-level)	0.152	0.633	0.327	0.427
hierarchy (high prec. top-level)	0.147	0.600	0.446	0.504

Table 6.13: In-domain performance of the hierarchical classifiers compared to the top-scoring flat in-domain classifier.

labels still. By forcing the bottom-level classifier to assign labels to all instances that make it through the top-level classifier, this problem is somewhat mitigated. Sadly, most of the correctly classified instances are from the "other" emotion class, which is of lesser interest to us, and the classifier still scores badly on the four emotion classes. The blog data, however, does not benefit from the hierarchical classification setup. "Flat" performance was respectable, and no improvement is to be gained from classifying instances in two steps. Our results are therefore somewhat contradicting findings by Ghazi et al. (2010), who did find a significant improvement when using a hierarchical classification setup on the blog dataset. Results are however not entirely comparable, as we use a subset of 4 emotions of the classes present in the data.

We also apply the above hierarchical classifier ensembles to cross-domain data. The results are summarized in Table 6.14.

As expected, cross-domain results are quite a bit lower than the in-domain results. Hierarchical classification results are also not better than flat classification results. When using the blog data as target data, the micro-averaged F-scores are higher, while macro-averaged F-scores are significantly lower. This would indicate that our hierarchical classifier assigns more instances to the "other" class, which is the largest

target dataset	suicide notes		blog posts	
	macro-avg. F_1	micro-avg. F_1	macro-avg. F_1	micro-avg. F_1
best "flat" classifier	0.084	0.028	0.080	0.010
hierarchy (high rec. top-level)	0.043	0.021	0.013	0.021
hierarchy (high prec. top-level)	0.074	0.028	0.017	0.032

Table 6.14: Cross-domain performance of the hierarchical classifiers compared to the top-scoring "flat" cross-domain classifier.

class in our dataset (the "no-emotion" category, containing instances without labels, is not technically a class label, and is also not considered in the score calculations). As the instances in the "other" class are less interesting in the context of emotion classification (we would, ideally, get as many of the sentences belonging to four basic emotions correct as possible), we have to rule out hierarchical classification as a solution for cross-domain classification. It does however seem to be a valid approach for in-domain classification, when skewed data results in a flat classifier performing badly and assigning too few emotion labels.

6.2.3 Typicality-based resampling

In a final attempt to limit the influence of noise on our emotion classifiers, we turn to the idea of *typicality*. Simply put, the typicality of an instance is a measure of how representative the instance is for its class. By filtering out the training instances that are not "typical" for their emotion, we hope to obtain a dataset that is less noisy, and that will result in a more stable model.

Instance typicality is most commonly used in conjunction with lazy learners, such as memory-based learners (Daelemans & van den Bosch, 2005), and other classifiers based on the nearest-neighbors algorithm (Brighton & Mellish, 2002). Unlike "eager" learners (which include SVMs), these classifiers do not abstract away from the set of training instances, but instead keep all items in memory. A k-Nearest Neighbors classifier for instance, will determine the class of an incoming test instance by comparing it to the training instances that are located nearest in the vector space. Of course, to do this, all training instances have to be present in memory. Instance typicality is most commonly used in conjunction with lazy learners, such as memory-based learners (Daelemans & van den Bosch, 2005), and other classifiers based on the nearest-neighbors algorithm (Brighton & Mellish, 2002). Unlike "eager" learners (which include SVMs), these classifiers do not abstract away from the set of training instances, but instead keep all items in memory. A k-Nearest Neighbors classifier for instance, will determine the class of an incoming test instance by comparing it to the training instances that are located nearest in the vector space. Of course, to do this, all training instances have to be present in memory.

The disadvantage of using lazy learners, is that memory requirements quickly spiral out of control. As the number of training instances (and the number of features) increases, so too does the size of the model, and for very large datasets keeping all training instances in memory may be out of the question. Secondly, whereas an SVM classifier simply draws a boundary between instances belonging to different classes —which may be minimally affected by a few instances lying on the "wrong side" of the instance space—, a k-Nearest Neighbors classifier will determine the class of an instance by looking in the instance's immediate vicinity. If the incoming instance's neighbors happen to be outliers, the predicted label is likely to be incorrect. This is where typicality comes in. First of all, limiting the number of training instances will reduce the size of the lazy learner's model. Secondly, it will first filter out the outliers, which will avoid problems related to noise.

While typicality-based filtering of the training set has been investigated (with varying results) for lazy learners (Daelemans et al., 1999; Brighton & Mellish, 2002), its efficacy in combination with eager learners (and with SVMs specifically) is uncertain. While limiting the presence of atypical instances might indeed result in a more general SVM model, it is possible that for smaller datasets, reducing the number of training instances will only serve to further destabilize the classifier.

Experimental setup

The idea behind typicality measures for feature vectors is relatively straightforward: if an instance of a given class is more similar to instances from other classes than it is to other instances from the same class, then its typicality can be said to be low. Conversely, if an instance is more similar to other instances carrying the same label than it is to instances carrying different labels, then its typicality score will be higher. It is also possible to put typicality in terms of distances: a highly typical point in the instance space will, on average, be closer to points of its own class than it will be to points in other classes, whereas an atypical instance will be located further from its own classmates than from points in other classes.

Formally, we define typicality as follows:

$$Typ(x) = \frac{Intra(x)}{Inter(x)}$$

with

$$Intra(x) = \frac{1}{|same(x)|} \sum_{i=1}^{|same(x)|} sim(x, same(x)_i)$$

and

$$Inter(x) = \frac{1}{|diff(x)|} \sum_{i=1}^{|diff(x)|} sim(x, diff(x)_i)$$

There are many metrics that can be used to calculate the similarity (or the distance) between instances. For the purposes of our experiments, however, we choose to keep it simple. Since we're working with vectors of numeric values, we simply use Euclidean distances to evaluate how closely pairs of instances are related.

A typicality score of 1.0 would imply that the instance is just as close to other instances in the same class as it is to instances in other classes. The instance is neither typical nor atypical. An atypical instance would have a typicality score in the range of $[0.0, 1.0)$, with lower scores indicating that the instance is more like instances from different classes than it is like instances from its own class. Typical instances have scores in the range of $(1.0, +\infty)$.

The resulting typicality scores can be used in several ways to trim the set of training instances. One way is to set a hard threshold on the typicality scores, only allowing those instances which have a score of more than 1.0. The rationale behind this approach is that instances with a typicality of 1.0 or below are atypical, and that they will only serve to confuse the classifier. The disadvantage of this approach is that for very noisy classes, not many instances will score above the threshold, leaving the class with very few training instances. Given the fact that we are already dealing with small datasets, with some very sparse classes, this may be a problem.

A second approach is to set a threshold on the number of instances per class based on the typicality scores. Only the n most typical instances per class will get included. Since this technique does not set a lower bound to the typicality score of an instance, it is still possible that the training data will contain atypical instances. Conversely, it is also possible for typical instances to get filtered out of the training set.
For our experiments, we need to be very careful where we set our n. In the suicide notes especially, emotional sentences are rare, so limiting the training instances per class to a number that is too high will not trim any potential noise from these smaller classes. On the flipside, setting our n lower than the number of instances in the smallest class, will severely limit the size of the training corpus. We therefore choose to uniformly filter out 50% of the data. For each class, half of the instances —the most atypical ones— are eliminated. Some classes contain a large amount of atypical instances, and 50% seems to be a reasonable compromise between eliminating atypical instances and retaining typical ones without upsetting the class balance.

Results and analysis

Tables 6.15 and 6.16 show how well our classifiers work when trained with only those instances that have a typicality score greater than 1.0, on the 2011 Medical NLP Challenge data an on the blog dataset respectively. We again work with the same feature types and feature representations as those used in Chapter 5, Section 5.3.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (rel. freq.)	0.000 (±0.000)	0.000 (±0.000)	0.542 (±0.021)
character 5-grams (rel. freq.)	0.000 (±0.000)	0.000 (±0.000)	0.542 (±0.021)
LIWC 1-grams (rel. freq.)	0.000 (±0.000)	0.000 (±0.000)	0.542 (±0.021)
NRC 1-grams (tf-idf)	0.000 (±0.000)	0.000 (±0.000)	0.542 (±0.021)

Table 6.15: Results for the in-domain typicality experiments on the suicide notes dataset, using only strictly typical instances.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (binary)	0.134 (±0.021)	0.331 (±0.040)	0.729 (±0.017)
character 4-grams (tf-idf)	0.000 (±0.000)	0.000 (±0.000)	0.685 (±0.018)
LIWC 1-grams (tf-idf)	0.000 (±0.000)	0.000 (±0.000)	0.685 (±0.018)
NRC 1-grams (rel. freq.)	0.015 (±0.008)	0.048 (±0.031)	0.414 (±0.265)

Table 6.16: Results for the in-domain typicality experiments on blog data, using only strictly typical instances.

It is hardly necessary to draw up a table comparing the typicality results with the original in-domain results using the full training set. Filtering out all training instances that are atypical has a dramatic effect on classification performance, as the classifier based on the suicide notes dataset fails to identify even a single emotional instance. The classifier trained on the blog data doesn't fare much better, with only the token unigram classifier and the classifier based on the NRC lexicon picking up a few emotional sentences.

	blog data	suicide notes
anger	1	18
fear	4	16
happiness	478	8
sadness	34	22
other	7	33
no-emotion	2799	2495

Table 6.17: Distribution of strictly typical token unigram instances in both datasets.

When we look at the distribution of typical instances in each dataset (Table 6.17), this behavior hardly seems surprising. It would seem both datasets are very noisy, with instances of the same class forming disparate clouds rather than clear clusters. As a result, if we set the typicality threshold to 1.0, some classes are left with very few training instances. This of course amplifies the sparseness problem presented by our already very small datasets. This distribution does shed light on the quality of our training corpus, however. The "other" class especially has many atypical instances, which intuitively makes sense, since this class is a grouping of instances with many different possible labels. Surprisingly, maybe, is that the no-emotion class is relatively stable. For the suicide notes especially, one would have expected the no-emotion class

to contain many atypical examples, since this class also contains instances which were emotional, but on which the annotators could not reach an agreement. However, the instances in this class seem to be relatively well clustered. Another remarkable fact is that the "happiness" class in the blog dataset holds its own. The vast majority of the "happy" instances are typical for their class.

Despite the insights this distribution of typical instances gives us, it is clear that using only strictly typical instances as training material is not workable –at least for our datasets. We therefore turn to an alternative approach, which is to use 50% of the instances in each class, keeping only those with the highest typicality scores. The distribution of instances across classes thus stays the same, with the only difference being that each class only contains half the instances. We settled on half of the instances, since the distribution of typical instances shows that some classes may contain as few as a single typical instance, while others almost contain nothing but typical examples. Table 6.18 shows this new distribution, and Tables 6.19 and 6.20 show the classification results.[8]

[8]Additional tables have been left out for clarity, but the interested reader can find them in Appendix A.3.

emotion label	suicide notes	blog posts
anger	34	89
fear	12	57
happiness	12	268
sadness	25	86
"other"	1006	143
no emotion	1256	1403

Table 6.18: Distribution of instances across classes in both datasets after eliminating the 50% least typical instances per class.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (rel. freq.)	0.115 (±0.045)	0.564 (±0.223)	0.457 (±0.051)
character 5-grams (rel. freq.)	0.115 (±0.045)	0.564 (±0.223)	0.457 (±0.051)
LIWC 1-grams (rel. freq.)	0.134 (±0.004)	0.509 (±0.162)	0.452 (±0.055)
NRC 1-grams (tf-idf)	0.153 (±0.023)	0.548 (±0.151)	0.448 (±0.053)

Table 6.19: Results for the in-domain typicality experiments on the suicide notes dataset using the 50% most typical instances per class.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (binary)	0.270 (±0.049)	0.303 (±0.045)	0.643 (±0.016)
character 4-grams (tf-idf)	0.000 (±0.000)	0.000 (±0.000)	0.685 (±0.018)
LIWC 1-grams (tf-idf)	0.000 (±0.000)	0.000 (±0.000)	0.685 (±0.018)
NRC 1-grams (rel. freq.)	0.022 (±0.010)	0.024 (±0.010)	0.653 (±0.019)

Table 6.20: Results for the in-domain typicality experiments on blog data using the 50% most typical instances per class.

The loss of performance on the blog data is dramatic, as is clearly visible in Figure 6.6. Character n-grams and LIWC-based unigrams lose their full capacity (indicating that the resulting classifier systematically put instances in the *no-emotion* bin), whereas token unigrams and NRC-based unigrams see a very significant drop in performance. Figure 6.7 shows the effect of typicality-based resampling on the 2011 Medical NLP Challenge data. There's a slight improvement in performance when using NRC-based unigrams, but even considering this, it seems obvious that keeping all instances in training is the best option for in-domain classification.

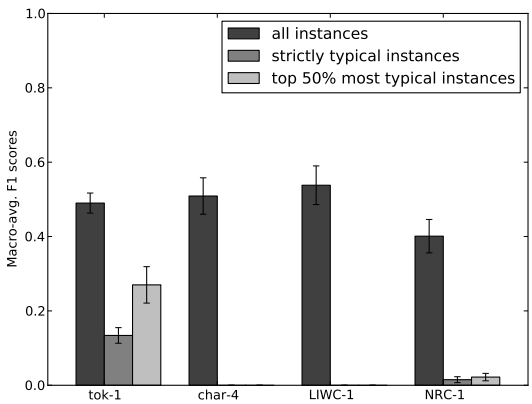

Figure 6.6: Results for in-domain experiments on blog data, with and without typicality-based resampling.

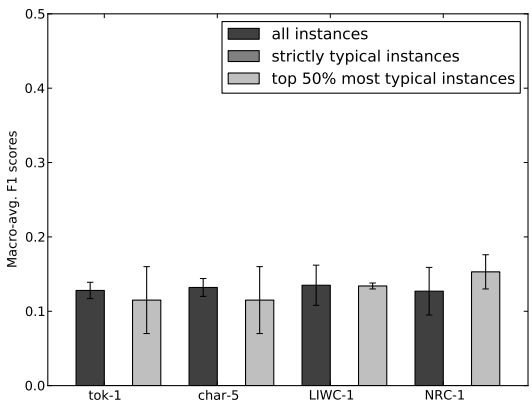

Figure 6.7: Results for in-domain experiments on suicide note data, with and without typicality-based resampling.

The interesting point, of course, is whether or not distilling the most typical training instances makes for a better generalizing model. Tables 6.21 and 6.22 show the cross-domain results for the approach based on a fixed score threshold.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (binary)	0.028 (±0.016)	0.030 (±0.017)	0.624 (±0.021)
character 3-grams (abs. freq.)	0.023 (±0.020)	0.020 (±0.017)	0.657 (±0.019)
LIWC 1-grams (binary)	0.036 (±0.018)	0.019 (±0.010)	0.680 (±0.019)
NRC 1-grams (abs. freq.)	0.010 (±0.010)	0.003 (±0.004)	0.677 (±0.019)

Table 6.21: Results for cross-domain typicality experiments trained on suicide notes and tested on blog data, using only strictly typical instances.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (abs. freq.)	0.007 (±0.005)	0.044 (±0.008)	0.504 (±0.024)
character 6-grams (binary)	0.002 (±0.002)	0.028 (±0.010)	0.303 (±0.018)
LIWC 1-grams (tf-idf)	0.000 (±0.000)	0.000 (±0.000)	0.542 (±0.021)
NRC 1-grams (tf-idf)	0.003 (±0.002)	0.010 (±0.005)	0.099 (±0.011)

Table 6.22: Results for cross-domain typicality experiments on trained on the blog dataset and tested on the suicide notes dataset, using only strictly typical instances.

Tables 6.23 and 6.24 show the results for cross-domain experiments using the training sets comprising of the 50% most typical instances per class.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (binary)	0.024 (±0.007)	0.066 (±0.019)	0.462 (±0.013)
character 3-grams (abs. freq.)	0.024 (±0.007)	0.069 (±0.016)	0.428 (±0.010)
LIWC 1-grams (binary)	0.027 (±0.007)	0.080 (±0.011)	0.283 (±0.016)
NRC 1-grams (abs. freq.)	0.025 (±0.005)	0.078 (±0.011)	0.408 (±0.018)

Table 6.23: Results for cross-domain typicality experiments trained on suicide notes and tested on blog data, using only the 50% most typical instances per class.

feature type (representation)	macro-avg. F_1 (±stdev.)	micro-avg. F_1 (±stdev.)	accuracy (±stdev.)
token 1-grams (abs. freq.)	0.056 (±0.040)	0.069 (±0.014)	0.417 (±0.021)
character 6-grams (binary)	0.001 (±0.008)	0.025 (±0.011)	0.371 (±0.022)
LIWC 1-grams (tf-idf)	0.057 (±0.069)	0.010 (±0.012)	0.525 (±0.024)
NRC 1-grams (tf-idf)	0.036 (±0.037)	0.039 (±0.021)	0.517 (±0.023)

Table 6.24: Results for cross-domain typicality experiments on trained on the blog dataset and tested on the suicide notes dataset, using only the 50% most typical instances per class.

Figures 6.8 and 6.9 graphically summarize the performance of typicalilty-based resampling techniques in cross-domain experiments. Unfortunately, the classifiers trained with typical instances (either using only strictly typical instances or using the 50% most typical instances per class) all perform worse in a cross-domain setup than classifiers where no such instance selection was performed. The datasets we work with are small, and filtering out a large part of the instances is far more destructive to the classification model than the noise that is naturally present in the data. The experiments we performed so far do not allow us to state whether or not instance selection based on typicality is always harmful when using SVM classifiers. However, our results do seem conform to previous research showing that forgetting atypical examples may do more harm than good in automated language learning (Daelemans et al., 1999).

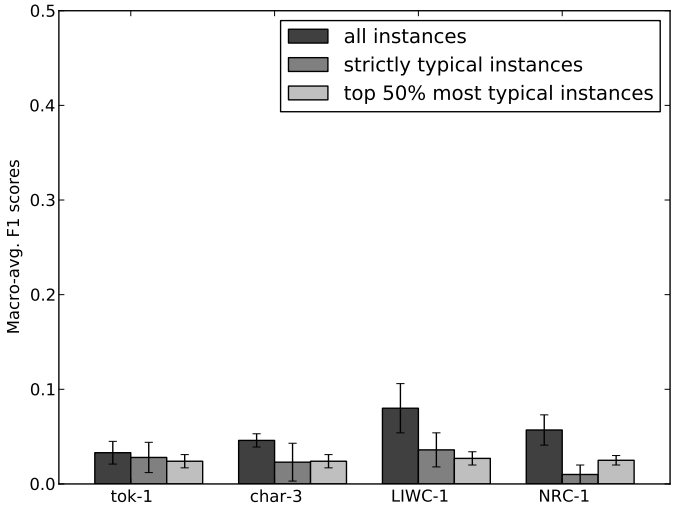

Figure 6.8: Results for cross-domain experiments trained on suicide notes and tested on blog data, with and without typicality-based resampling.

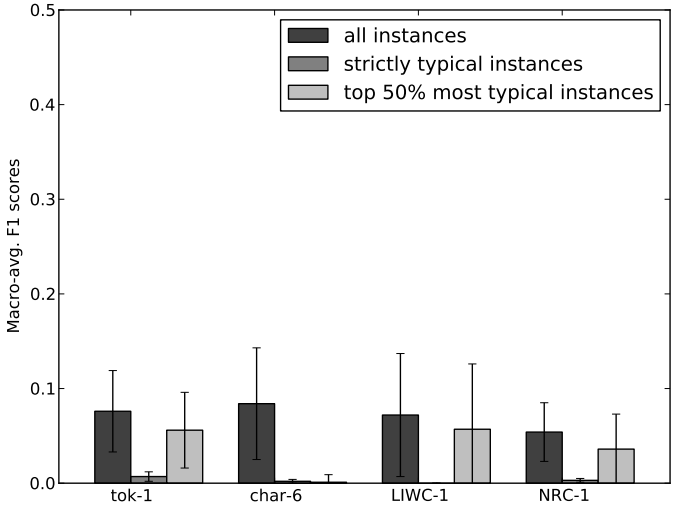

Figure 6.9: Results for cross-domain experiments trained on blog data and tested on suicide note data, with and without typicality-based resampling.

6.3 Conclusions

We have attempted to limit the influence of noise in the training data using three very different approaches. We have evaluated whether these approaches improved in-domain classification performance, but we have also evaluated cross-domain performance in the hope that these techniques make for better generalizing models.

We first attempted to limit the influence of noise in the majority classes (the unlabeled instances and the "other" class) by strengthening the influence of smaller classes by weighting up their misclassification cost parameter. This approach proves successful when the class distribution is highly skewed. If a classifier using unmodified class weights classifies most instances as belonging to the majority class (as was the case on the 2011 Medical NLP Challenge data), then boosting the cost parameter for the smaller classes forces the classifier to predict more minority labels, which has a positive impact on performance. The difference is small, however, and when the class imbalance is less problematic (as it was for the blog dataset), then weighting up the minority classes might result in overfitting.
That overfitting is a problem, is immediately obvious when we look at cross-domain results. Modifying the misclassification cost for the smaller classes has a negative effect on cross-domain classification. This is logical, since the model is fitted more tightly on the source data, which makes it less likely that it will be adequate for data from a different domain.

Our second approach was to build a hierarchical classifier ensemble. A top-level classifier determines if an instance contains an emotion –any emotion–, and only if an emotion is detected, does the bottom-level classifier determine the exact nature of the emotion. As was the case for the previous technique, this approach is helpful to make sure enough emotion labels are predicted when a flat classifier mostly predicts the majority class. The approach is not effective on the blog dataset, and neither does it work on the cross-domain classification tasks. Instead of mitigating misclassifications, the hierarchical setup multiplies the classification errors. An instance that is erroneously classified as not containing any emotions by the top-level classifier, simply cannot be retrieved by the bottom-level classifier.

The third and final attempt at reducing the noise in the training data, was to filter out atypical training instances. We tried eliminating all atypical instances from the training set, but this results in an accentuation of the data sparseness problem, since some classes contain a large majority of atypical instances. This means that the instances in these classes are not positioned in a cluster, but that they are positioned in a disparate cloud, which explains why separating instances into classes is so difficult. Using only typical instances as training material is detrimental to the classification performance. We also attempted to eliminate atypical instances in a stratified way: we eliminated the least typical 50% of instances in each class. This way a large part of the atypical instances were filtered out, but the class balance was not disturbed.

This approach also proved fruitless both on in-domain and on cross-domain experiments, leaving us with a solid first indication that it is not beneficial to eliminate atypical training instances when dealing with eager learners.

Chapter 7

Conclusions

"What if your computer could understand how you feel?" We started this thesis by placing the thought of a perfect, practically workable emotion classifier firmly in the realm of science-fiction. After having researched the problem of automatic emotion categorization in detail, we do not feel tempted to change our views in this respect. However, we have become more aware of what exactly holds emotion classification back, and we have offered some pointers on how to work *around* —or at the very least *with*— the task's difficulties.

The largest hurdle for automatic emotion classification is simply the subjective nature of the task. Other problems —noisy datasets, the small size of existing annotated corpora— are a natural consequence of this primary difficulty. Emotions are subjective, and this makes it problematic to determine gold standard datasets on which to base supervised classification systems. Smaller, low-agreement datasets are more the norm than the exception in emotion classification, and we saw how this influenced classification performance in two different case studies.

The deLearyous task (Chapter 3) involved detecting the emotional stance of participants in a conversation. This "stance" was defined in terms of positions on the Interpersonal Circumplex, a dimensional emotion framework defined by *affinity* and *dominance* dimensions. Although the conversations in our dataset originated from a controlled environment, and although they were concerned with a single, well-defined conversational topic, we noticed that the ways one can express an emotion are numerous and diverse, and that finding the defining features that point to a given emotion is no simple matter. Classification performance on the task was modest, and while we learned which features were most useful given the task, we were far from reaching a level of performance that was high enough to be of practical use.
We also remarked that given the low agreement of human annotators on the same

emotion classification task, it was perhaps unrealistic to expect an automated system to do much better. The interpretation of the emotional charge of an utterance is highly dependent on extra-textual context, body language, intonation..., which is information we do not have access to when we classify text.

Our second case study, the 2011 Medical NLP Challenge (Chapter 4), showed another facet of emotion classification. The challenge dataset contained documents from a very different domain compared to the aforementioned business conversations. The goal of the challenge was to classify sentences originating from suicide notes into 11 discrete classes, with the extra difficulty that classes were not mutually exclusive. Performance on this multi-label classification task was acceptable given our pure machine learning approach, but it became obvious that classification accuracy on a given emotion was highly dependent on the emotion's presence in the training set, which underlined the importance of a dataset where all emotions are sufficiently represented.

Despite the fact that the case studies concerned very different domains and emotion frameworks, we did see some common patterns emerge. In both cases, we saw that, given a restricted and well-defined domain, character n-grams are a good choice of feature. At their "worst", they perform just as well as the more common bag-of-words features. At their best, however, they are capable of capturing phenomena which are beyond the grasp of the standard word tokens. The way we implemented them, characters n-grams can cross word boundaries. This means that they can capture salient word sequences, whereas bag-of-words features end at the word boundary. They also give us a way to handle spelling errors, since they are able to capture correctly spelled parts of a word, while wrongly spelled n-grams can safely be discarded. Finally, since semantically related words (or variants of the same word) often share the same root, character n-grams can group related words under a single feature by capturing only this common root. Character n-grams are low-level features that are easy to implement, but they are really quite powerful in a machine learning setup.

A second point that rose from both case studies was —again— that a good dataset is of primordial importance. Both datasets were relatively small, and we clearly saw that performance suffered when a class was not adequately represented. Similarly, while the nature of the task makes a reliable gold standard (in terms of data with high inter-annotator agreement) a difficult concept, it is clear that a classifier will never meaningfully perform better than the humans who annotated the data. However, given the difficult and time-consuming nature of constructing a large, reliable eomtion dataset, it is unrealistic to expect to have a good, task-specific gold-standard available for every possible emotion classification task.

This is why we suggested an alternative solution: if the dataset for the task at hand is too small or does not contain enough information to be able to construct an accurate classification model, it may be best to make use of a different dataset which is larger

and less difficult to learn. Even if this larger dataset is from a different domain. As a general rule, one is usually better served by a model trained on a smaller set of in-domain documents, but we hypothesized that a model built on a larger dataset from a different domain would generalize better than one built on a small in-domain dataset, resulting in better predictions.

Through a variety of in-domain and cross-domain experiments (Chapter 5), we attempted to construct a model that would generalize well across emotion classification datasets. We showed that it is too optimistic to want to construct a true general-purpose emotion classifier, but that if optimizing the generalizing power of a model is one's goal, that the use of emotion lexicons is recommended. We showed that it may be more interesting to use a smaller, well-constructed lexicon, than it is to use a bigger lexicon with more coverage, but with potentially more noise.
We also confirmed our previous hypothesis by showing that when a target-domain dataset is especially small or skewed, it may be beneficial to train a classifier on a larger, better balanced dataset from a different domain. This is especially true if it is possible to integrate a small part of target data while training, and if one can apply some simple domain adaptation techniques, which will significantly boost cross-domain classification performance.

Finally, we attempted to boost our emotion classifiers' performance —in both in-domain and cross-domain setups— by applying a variety of techniques to limit (the influence of) noise or skewed distributions (Chapter 6). We saw that for in-domain classification, and in the case of a highly skewed data distribution, it may be beneficial to boost the minority classes' misclassification costs in the SVM training algorithm. In all other cases, however, this had a negative effect on performance. This is especially true in cross-domain experiments, as the tighter fitting of the model on the source-domain data will be problematic on target-domain instances.
We have shown —in contrast to previous research by Ghazi et al. (2010)— that performing hierarchical classification is not always beneficial. Similarly, we have shown that there is no gain to be had from filtering unrepresentative instances from the training set. Filtering away all strictly atypical instances was destructive for our datasets, as instances of a single class are not always well-clustered in the instance space, meaning the majority of the instances were considered "atypical" and thus filtered away. As a consequence of the drastically reduced number of training instances for these "diffuse" classes, the resulting classifiers performed very badly. But even filtering away just a part of the atypical instances proved not to be effective, which corroborates previous research on the topic by Daelemans & van den Bosch (2005). We concluded that it is better to keep all training instances, as even the bad examples can be useful to the classifier.

In summary, we have illustrated the challenges involved in the classification of text according to its author's emotions. The challenge starts with data collection and manual annotation, which is always difficult and often problematic. As a result,

many tasks have very little annotated data available, which makes it very difficult to train an accurate emotion model. We have suggested ways to improve training datasets, either by drawing from data from other domains, or by filtering noise from the training data.

If we turn to the future, the logical recommendation would be to strive to build one large, reliable corpus of emotion-annotated texts. We have shown that one large, reliable dataset's information can be of great use, even in tasks concerning different domains. We believe a good candidate for an emotion-annotated text corpus would be a collection of movie subtitles. Movies contain emotionally charged dialogue in a variety of registers and contexts, which would make the resulting dataset very versatile. Using movie subtitles also has the added advantage that the audio and video material can be used in the case of annotator disagreement, which should make deciding on a gold standard somewhat easier.

However, we cannot help but ask ourselves if, for highly subjective tasks such as emotion classification, perhaps we should abandon —or at the very least rethink— the concept of a "gold standard" annotation entirely, and if instead we should not simply consider each annotator's output as potentially valid. The way we interpret an emotion is dependent on our personality, our life experiences, our mood, etc., and in this sense, there may not always be a true correct emotion class for a given emotional document. A classifier output would then no longer be seen as correct or incorrect, but it would be placed on a *scale* of acceptability.

This would of course have major implications for the rest of the classification and evaluation process[1], since there may be no single gold standard emotion label for a given text fragment. We do not expect that the concept of a gold standard for supervised categorization will disappear anytime soon, and we would rather expect to see a future where emotion classification has been "solved" through a multi-modal approach. Our facial expressions, the intonation in our voice, our body language, our pulse... They all provide information about how we feel. While raw text in itself may not always contain enough information to determine the emotion of its author, we are confident that by combining text with other sources of information, we will eventually reach a point where emotion classification will work, and where it will work reliably enough to change our lives.

[1]We refer to the works by Reidsma (Reidsma, 2008; Reidsma & Carletta, 2008; Reidsma & op den Akker, 2008; Reidsma et al., 2009) for thought-provoking research on the topic of subjective annotations and their impact on classification tasks.

Appendix A

Background information

A.1 Modifications to LibShortText 1.0

Libshorttext is a Python package that was designed especially for short text classification, and it is built around liblinear (Fan et al., 2008), a package for linear SVM classification. Libshorttext makes some assumptions about instance generation in its implementation, and while these assumptions are not likely to cause any large difference in the produced models and their accuracies, we have chosen to modify the software slightly where we disagreed with them. For transparency, the following is a list of modifications made on LibShortText 1.0:

- LibShortText only allows the construction of unigrams or bigrams. We have extended this functionality up to 6-grams.
- LibShortText assumes n-grams to start at the document boundary, i.e. the first element of the first n-gram of a document will be the first word (or character, or part of speech tag, depending on the feature type in the source files) of the document. We instead chose to allow n-grams start before the document boundary, i.e. a document-initial token trigram would start with two placeholders (because the trigram starts out-of-bounds) followed by the first token in the document. This allows us to capture these document-initial and document-final n-grams, whereas without these out-of-bounds placeholders, this is information the learner would simply not have access to.
- We have replaced LibShortText's tokenizer with a very simple "pass-through" tokenizer of our own. LibShortText's tokenizer strips all non ASCII characters from the input, which may be appropriate for most English texts, but which

is less interesting for other languages. Since in the deLearyous dataset, all tokenizing was done using Frog, we can safely bypass the internal tokenizer.

Fundamentally, Libshorttext's learner functionality remains untouched, and our modifications apply only to the instance generation stage.

A.2 Considerations on averaged precision, recall, and F-scores

Evaluating precision, recall and F-score of classifier predictions on a given class is relatively straightforward. We refer to the Glossary for more information on the calculations involved. Calculating the average of one of these measures across multiple classes is also relatively straightforward, though one will have to make a choice between micro-averaged F-scores and macro-averaged F-scores. The former option will give larger classes more weight in the calculation of the average, while the latter option will consider all classes equally important. Both metrics have their merits and disadvantages, which is why we report on both where appropriate.

Complications arise when one tries to calculate these same measure for multi-labeled classification tasks such as those we carried out in Chapter 4. The problematic element in the evaluation of multi-label classification does not so much arise when we predict more than one class for a document, but when we predict no class at all. In the threshold-based approaches we outlined in the case study, it was quite common that a sentence would not get assigned any emotions, because none of the emotions were probable enough to exceed the emotion threshold. The question, then, is whether or not we should consider the absence of a label as a noteworthy result in itself. More concretely, we could see the absence of a prediction as the presence of a special "None" label which excludes all other possible labels.

Whether or not we consider such a phantom label in our calculations has important repercussions for both the results and their interpretation. When calculating average scores, including the "None" label increases the number of classes to be considered by one, which in turn significantly alters the scores. In favor of including the "None" label, one could argue that a classifier should be rewarded for abstaining from assigning a class label when no labels are required. On the other hand, this information is implicit in the false positives of the other classes. The "None" label also does not behave like other labels (as it excludes any other predictions), which makes it hard to justify considering it a regular label in our calculations.

Given these considerations, we have decided to evaluate all multi-label classification tasks *without* taking the fictitious empty label into account. Our main argument is that we want to see how well *emotion* labels are predicted, and that the neutral

sentences are of lesser importance. Additionally, the scores on the non-empty labels give us enough data to infer further information from. Another argument is consistency: the empty label was not considered during the shared task in Chapter 4, and it makes sense to follow the same pattern in all subsequent experiments.

A.3 Appendix to Chapter 6

This appendix contains tables which were withheld from Chapter 6 to improve readability, and because they were not essential to understand the main points of the experiments. The tables give detailed results (both in-domain and cross-domain) for all class weighting experiments as well as for the experiments with typicality-based resampling.

A.3.1 Extra data for Class weighting

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using class weights
token 1-grams (rel. freq.)	0.128 (±0.011)	+0.015
character 5-grams (rel. freq.)	0.132 (±0.012)	+0.014
LIWC 1-grams (rel. freq.)	0.135 (±0.027)	+0.018
NRC 1-grams (tf-idf)	0.127 (±0.032)	+0.040

Table A.1: Results for the *unweighted* in-domain experiments on the suicide note data and the gain (or loss) caused by using individual class weights. (Results repeated from Table 5.4.)

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using class weights
token 1-grams (binary)	0.490 (±0.027)	+0.004
character 4-grams (tf-idf)	0.509 (±0.049)	-0.005
LIWC 1-grams (tf-idf)	0.538 (±0.052)	-0.023
NRC 1-grams (rel. freq.)	0.401 (±0.045)	+0.002

Table A.2: Results for the *unweighted* in-domain experiments on blog data and the gain (or loss) caused by using individual class weights. (Results repeated from Table 5.5.)

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using class weights
token 1-grams (abs. freq.)	0.076 (±0.043)	-0.004
character 6-grams (binary)	0.084 (±0.059)	-0.006
LIWC 1-grams (tf-idf)	0.072 (±0.065)	-0.015
NRC 1-grams (tf-idf)	0.054 (±0.031)	-0.007

Table A.3: Results for the *unweighted* cross-domain experiments trained on blog data and tested on suicide notes, and the gain (or loss) caused by using individual class weights. (Results repeated from Table 5.6.)

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using class weights
token 1-grams (tf-idf)	0.033 (±0.012)	+0.059
character 3-grams (abs. freq.)	0.046 (±0.007)	+0.020
LIWC 1-grams (binary)	0.080 (±0.026)	+0.000
NRC 1-grams (abs. freq.)	0.057 (±0.016)	+0.000

Table A.4: Results for the *unweighted* cross-domain experiments trained on the suicide note data and tested on blog data, and the gain (or loss) caused by using individual class weights. (Results repeated from Table 5.7.)

A.3.2 Extra data for Typicality-based resampling

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using typical data
token 1-grams (rel. freq.)	0.128 (±0.011)	-0.013
character 5-grams (rel. freq.)	0.132 (±0.012)	-0.017
LIWC 1-grams (rel. freq.)	0.135 (±0.027)	-0.001
NRC 1-grams (tf-idf)	0.127 (±0.032)	+0.026

Table A.5: Results for the regular in-domain experiments on suicide note data and the gain (or loss) caused by using only the 50% most typical instances.

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using typical data
token 1-grams (binary)	0.490 (±0.027)	-0.220
character 4-grams (tf-idf)	0.509 (±0.049)	-0.509
LIWC 1-grams (tf-idf)	0.538 (±0.052)	-0.538
NRC 1-grams (rel. freq.)	0.401 (±0.045)	-0.379

Table A.6: Results for the regular in-domain experiments on blog data and the gain (or loss) caused by using only the 50% most typical instances.

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using typical data
token 1-grams (tf-idf)	0.033 (±0.012)	-0.005
character 3-grams (abs. freq.)	0.046 (±0.007)	-0.023
LIWC 1-grams (binary)	0.080 (±0.026)	-0.044
NRC 1-grams (abs. freq.)	0.057 (±0.016)	-0.047

Table A.7: Results for the original cross-domain experiments trained on the suicide note data and tested on blog data, and the gain (or loss) caused by using only strictly typical instances.

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using typical data
token 1-grams (abs. freq.)	0.076 (±0.043)	-0.069
character 6-grams (binary)	0.084 (±0.059)	-0.082
LIWC 1-grams (tf-idf)	0.072 (±0.065)	-0.072
NRC 1-grams (tf-idf)	0.054 (±0.031)	-0.051

Table A.8: Results for the original cross-domain experiments trained on blog data and tested on suicide notes, and the gain (or loss) caused by using only the 50% most typical instances.

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using typical data
token 1-grams (tf-idf)	0.033 (±0.012)	-0.011
character 3-grams (abs. freq.)	0.046 (±0.007)	-0.022
LIWC 1-grams (binary)	0.080 (±0.026)	-0.053
NRC 1-grams (abs. freq.)	0.057 (±0.016)	-0.032

Table A.9: Results for the original cross-domain experiments trained on the suicide note data and tested on blog data, and the gain (or loss) caused by using only the 50% most typical instances.

feature type (representation)	macro-avg. F_1 (±stdev.)	improvement using typical data
token 1-grams (abs. freq.)	0.076 (±0.043)	-0.020
character 6-grams (binary)	0.084 (±0.059)	-0.083
LIWC 1-grams (tf-idf)	0.072 (±0.065)	-0.015
NRC 1-grams (tf-idf)	0.054 (±0.031)	-0.018

Table A.10: Results for the original cross-domain experiments trained on blog data and tested on suicide notes, and the gain (or loss) caused by using only the 50% most typical instances.

Bibliography

Alm, C. O., Roth, D., & Sproat, R. (2005). Emotions from text: Machine learning for text-based emotion prediction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, (pp. 579–586). Stroudsburg, PA, USA: Association for Computational Linguistics.

Aman, S., & Szpakowicz, S. (2007). Identifying expressions of emotion in text. In *Text, Speech and Dialogue*, vol. 4629/2007, (pp. 196–205). Springer Berlin.

Arnold, M. (1960). *Emotion and Personality: Psychological aspects*. Emotion and Personality. Columbia University Press.

Baccianella, S., Esuli, A., & Sebastiani, F. (2009). Multi-facet rating of product reviews. In M. Boughanem, C. Berrut, J. Mothe, & C. Soule-Dupuy (Eds.) *Advances in Information Retrieval*, vol. 5478 of *Lecture Notes in Computer Science*, (pp. 461–472). Springer Berlin Heidelberg.

Baccianella, S., Esuli, A., Sebastiani, F., & Baccianella, A. E. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, (pp. 2200–2204). Valletta, Malta: European Language Resources Association (ELRA).

Balahur, A. (2013). Sentiment analysis in social media texts. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (pp. 120–128). Atlanta, Georgia: Association for Computational Linguistics.

Balahur, A., Steinberger, R., Kabadjov, M., Zavarella, V., van der Goot, E., Halkia, M., Pouliquen, B., & Belyaeva, J. (2010). Sentiment analysis in the news. In N. C. Chair, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, & D. Tapias (Eds.) *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA).

Bellegarda, J. (2010). Emotion Analysis Using Latent Affective Folding and Embedding. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, (pp. 1–9). Los Angeles, CA: Association for Computational Linguistics.

Bennett, E. M., Alpert, R., & Goldstein, A. (1954). Communications through limited-response questioning. *Public Opinion Quarterly*, *18*(3), 303–308.

Bermingham, A., & Smeaton, A. F. (2010). Classifying Sentiment in Microblogs: Is Brevity an Advantage? In *CIKM 2010 - 19th International Conference on Information and Knowledge Management*. Toronto, Canada.

Bermingham, A., & Smeaton, A. F. (2011). On using twitter to monitor political sentiment and predict election results. *Sentiment Analysis where AI meets Psychology (SAAIP)*, (p. 2).

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). Association for Computational Linguistics.

Bradley, M., & Lang, P. J. (1999a). *The International affective digitized sounds (IADS): stimuli, instruction manual and affective ratings*. NIMH Center for the Study of Emotion and Attention.

Bradley, M. M., Greenwald, M. K., Petry, M. C., & Lang, P. J. (1992). Remembering pictures: pleasure and arousal in memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition; Journal of Experimental Psychology: Learning, Memory, and Cognition*, *18*(2), 379.

Bradley, M. M., & Lang, P. J. (1994). Measuring emotion: The self-assessment manikin and the semantic differential. *Journal of Behavior Therapy and Experimental Psychiatry*, *25*(1), 49 – 59.

Bradley, M. M., & Lang, P. J. (1999b). Affective norms for english words (anew): Instruction manual and affective ratings. Tech. rep., Technical Report C-1, The Center for Research in Psychophysiology, University of Florida.

Brighton, H., & Mellish, C. (2002). Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, *6*(2), 153–172.

Cerini, S., Compagnoni, V., Demontis, A., Formentelli, M., & Gandini, G. (2007). *Language resources and linguistic theory: Typology, second language acquisition, English linguistics.*, chap. Micro-WNOp: A gold standard for the evaluation of automatically compiled lexical resources for opinion mining. Milano, IT: Franco Angeli Editore.

Chaffar, S., & Inkpen, D. (2011). Using a heterogeneous dataset for emotion analysis in text. In *Advances in Artificial Intelligence*, (pp. 62–67). Springer.

Chaumartin, F.-R. (2007). UPAR7: A knowledge-based system for headline sentiment tagging. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, (pp. 422–425). Prague, Czech Republic: Association for Computational Linguistics.

Chesley, P., Vincent, B., Xu, L., & Srihari, R. K. (2006). Using verbs and adjectives to automatically classify blog sentiment. *Training*, *580*(263), 233.

Councill, I. G., McDonald, R., & Velikovich, L. (2010). What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the workshop on negation and speculation in natural language processing*, (pp. 51–59). Association for Computational Linguistics.

Daelemans, W., & van den Bosch, A. (2005). *Memory-Based Language Processing*. Cambridge, UK: Cambridge University Press.

Daelemans, W., Van Den Bosch, A., & Zavrel, J. (1999). Forgetting exceptions is harmful in language learning. *Machine Learning*, *34*(1-3), 11–41.

Daume III, H. (2007). Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, (pp. 256–263). Prague, Czech Republic: Association for Computational Linguistics.

Daumé III, H., & Marcu, D. (2006). Domain adaptation for statistical classifiers. *J. Artif. Intell. Res.(JAIR)*, *26*, 101–126.

Dave, K., Lawrence, S., & Pennock, D. (2003). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. *Proceedings of the 12th international conference on World Wide Web*, (pp. 519–528).

de Albornoz, J. C., Plaza, L., & Gervás, P. (2010). A hybrid approach to emotional sentence polarity and intensity classification. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, July, (pp. 153–161). Uppsala, Sweden: Association for Computational Linguistics.

de Albornoz, J. C., Plaza, L., & Gervás, P. (2012). SentiSense: An easily scalable concept-based affective lexicon for sentiment analysis. In N. Calzolari, K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, J. Odijk, & S. Piperidis (Eds.) *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA).

De Choudhury, M., Gamon, M., & Counts, S. (2012). Happy, nervous or surprised? classification of human affective states in social media. In *Sixth International AAAI Conference on Weblogs and Social Media*.

De Smedt, T., & Daelemans, W. (2012a). Pattern for python. *Journal of Machine Learning Research*, *13*, 2063–2067.

De Smedt, T., & Daelemans, W. (2012b). "vreselijk mooi!" (terribly beautiful): A subjectivity lexicon for dutch adjectives. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, (pp. 3568–3572).

Ekman, P. (1971). Universals and cultural differences in facial expressions of emotion. In *Nebraska Symposium on Motivation*. University of Nebraska Press.

Ekman, P. (1999). Facial expressions. *Handbook of cognition and emotion*, (pp. 301–320).

Esuli, A., & Sebastiani, F. (2007). Random-walk models of term semantics: An application to opinion-related properties. In *Proceedings of LTC-07, the 3rd Language & Technology Conference, Poznan, PL*.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, *9*, 1871–1874.

Feldman Barrett, L., & Russell, J. A. (1998). Independence and bipolarity in the structure of current affect. *Journal of personality and social psychology*, *74*(4), 967.

Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, *76*(5), 378.

Francisco, V., Hervás, R., Peinado, F., & Gervás, P. (2012). Emotales: creating a corpus of folk tales with emotional annotations. *Language Resources and Evaluation*, *46*(3), 341–381.

Gendron, M., & Barrett, L. F. (2009). Reconstructing the Past: A Century of Ideas About Emotion in Psychology. *Emotion review : journal of the International Society for Research on Emotion*, *1*(4), 316–339.

Ghazi, D., Inkpen, D., & Szpakowicz, S. (2010). Hierarchical versus Flat Classification of Emotions in Text. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, (pp. 140–146). Los Angeles, CA: Association for Computational Linguistics.

Ghazi, D., Inkpen, D., & Szpakowicz, S. (2012). Prior versus contextual emotion of a word in a sentence. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis*, (pp. 70–78). Association for Computational Linguistics.

Hatzivassiloglou, V., & McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, (pp. 174–181). Association for Computational Linguistics.

Inkpen, D., Keshtkar, F., & Ghazi, D. (2009). Analysis and Generation of Emotion in Texts. In P. U. Clujeana (Ed.) *KEPT 2009 Knowledge Engineering-Principles and Techniques, Selected Papers*, (pp. 3–13). Militon Frentiu and Horia F. Pop.

Jijkoun, V., de Rijke, M., & Weerkamp, W. (2010). Generating focused topic-specific sentiment lexicons. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, (pp. 585–594). Uppsala, Sweden: Association for Computational Linguistics.

Jijkoun, V., & Hofmann, K. (2009). Generating a non-english subjectivity lexicon: Relations that matter. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, (pp. 398–405). The Association for Computer Linguistics.

Jung, Y., Park, H., & Myaeng, S. H. (2006). A hybrid mood classification approach for blog text. In *PRICAI 2006: Trends in Artificial Intelligence*, (pp. 1099–1103). Springer.

Kapočiute-Dzikiene, J., Vaassen, F., Daelemans, W., & Krupavičius, A. (2012). Improving topic classification for highly inflective languages. In *Proceedings of COLING 2012, the 24th International Conference on Computational Linguistics*. Mumbai, India.

Katz, P., Singleton, M., & Wicentowski, R. (2007). SWAT-MP:the semeval-2007 systems for task 5 and task 14. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, (pp. 308–313). Prague, Czech Republic: Association for Computational Linguistics.

Keshtkar, F., & Inkpen, D. (2009). Using Sentiment Orientation Features for Mood Classification in Blogs. In *IEEE International Conference on Natural Language Processing and Knowledge Engineering (IEEE NLP-KE 2009), Dalian, China*.

Kim, S.-M., & Hovy, E. (2004). Determining the sentiment of opinions. In *Proceedings of the COLING conference, Geneva, 2004*.

Kim, S. M., Valitutti, A., & Calvo, R. A. (2010). Evaluation of unsupervised emotion models to textual affect recognition. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, (pp. 62–70). Los Angeles, CA: Association for Computational Linguistics.

Kozareva, Z., Navarro, B., Vazquez, S., & Montoyo, A. (2007). UA-ZBSA: A headline emotion classification through web information. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, (pp. 334–337). Prague, Czech Republic: Association for Computational Linguistics.

Kunneman, F., Liebrecht, C., & van den Bosch, A. (2014). The (un)predictability of emotional hashtags in twitter. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, (pp. 26–34). Gothenburg, Sweden: Association for Computational Linguistics.

Lang, P. J., Bradley, M. M., & Cuthbert, B. N. (1999). International affective picture system (IAPS): Technical manual and affective ratings.

Leary, T. (1957). *Interpersonal Diagnosis of Personality: Functional Theory and Methodology for Personality Evaluation*. Ronald Press Company: New York.

Lin, K. H.-Y., & Chen, H.-H. (2008). Ranking reader emotions using pairwise loss minimization and emotional distribution regression. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (pp. 136–144). Association for Computational Linguistics.

Luyckx, K., Vaassen, F., Peersman, C., & Daelemans, W. (2012). Fine-grained emotion detection in suicide notes: a thresholding approach to multi-label classification. *Biomed Inform Insights*, *5*(Suppl. 1), 61–69. [PubMed Central:PMC3409486] [DOI:10.4137/BII.S8966] [PubMed:22879761].

Maks, I., & Vossen, P. (2012). Building a fine-grained subjectivity lexicon from a web corpus. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, 3, (pp. 3070–3076). Istanbul, Turkey.

Maron, M. (1961). Automatic indexing: an experimental inquiry. *Journal of the ACM*, *8*(3), 404–417.

McClosky, D. (2010). *Any Domain Parsing: Automatic Domain Adaptation for Natural Language Parsing*. Ph.D. thesis, Brown University, Providence, RI, USA. AAI3430199.

Mihalcea, R., & Strapparava, C. (2012). Lyrics, music, and emotions. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, (pp. 590–599). Association for Computational Linguistics.

Mishne, G. (2005). Experiments with Mood Classification in Blog Posts. In *Proceedings of the 1st Workshop on Stylistic Analysis Of Text For Information Access*.

Mohammad, S. (2012a). #emotional tweets. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, (pp. 246–255). Montréal, Canada: Association for Computational Linguistics.

Mohammad, S. (2012b). Portable features for classifying emotional text. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (pp. 587–591). Association for Computational Linguistics.

Mohammad, S., Dunne, C., & Dorr, B. (2009). Generating High-Coverage Semantic Orientation Lexicons From Overtly Marked Words and a Thesaurus. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2009)*.

Mohammad, S. M., & Turney, P. D. (2012). Crowdsourcing a word–emotion association lexicon. *Computational Intelligence*.

Moors, A., De Houwer, J., Hermans, D., Wanmaker, S., van Schie, K., Van Harmelen, A.-L., De Schryver, M., De Winne, J., & Brysbaert, M. (2012). Norms of valence, arousal, dominance, and age of acquisition for 4,300 dutch words. *Behavior research methods*, (pp. 1–9).

Morinaga, S., Yamanishi, K., Tateishi, K., & Fukushima, T. (2002). Mining product reputations on the web. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge*

discovery and data mining, (pp. 341–349).

Noreen, E. W. (1989). *Computer-Intensive Methods for Testing Hypothesis- An Introduction.*. New York [etc.]: Wiley.

Novielli, N., & Strapparava, C. (2013). The role of affect analysis in dialogue act identification. *IEEE Transactions on Affective Computing*, *4*(4), 439–451.

op den Akker, H., Bruijnes, M., Peters, R., & Krikke, T. (2013). Interpersonal stance in police interviews: content analysis. *Computational Linguistics in the Netherlands Journal*, *3*, 193–216. Impact=0.000, ISSN=2211-4009.

Osgood, C. E. (1969). On the whys and wherefores of e, p, and a. *Journal of Personality and Social Psychology*, *12*(3), 194.

Osgood, C. E., Suci, G., & Tannenbaum, P. (1957). The measurement of meaning. *Urbana: University of Illinois Press*.

Pak, A., & Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA).

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, *2*(1-2), 1–135.

Parrott, W. (2001). *Emotions in social psychology: Essential readings.*. Psychology Press.

Pennebaker, J. W., Francis, M. E., & Booth, R. J. (2001). Linguistic inquiry and word count: Liwc 2001. *Mahway: Lawrence Erlbaum Associates*, (p. 71).

Pestian, J. P., Matykiewicz, P., Linn-Gust, M., South, B., Uzuner, O., Wiebe, J., Cohen, K. B., Hurdle, J., & Brew, C. (2012). Sentiment analysis of suicide notes: A shared task. *Biomedical Informatics Insights*, *5*, 3–16.

Plutchik, R. (1980). A general psychoevolutionary theory of emotion. *Emotion: Theory, research, and experience*, *1*(3), 3–33.

Prinz, J. (2004). Which emotions are basic. *Emotion, evolution, and rationality*, *69*, 88.

Purver, M., & Battersby, S. (2012). Experimenting with distant supervision for emotion classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, (pp. 482–491). Association for Computational Linguistics.

Qadir, A., & Riloff, E. (2013). Bootstrapped learning of emotion hashtags #hashtags4you. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (pp. 2–11). Atlanta, Georgia: Association for Computational Linguistics.

Reidsma, D. (2008). *Annotations and Subjective Machines of Annotators, Embodied Agents, Users, and Other Humans*. Ph.D. thesis, University of Twente, Enschede.

Reidsma, D., & Carletta, J. (2008). Reliability measurement without limits. *Computational Linguistics*, *34*(3), 319–326.

Reidsma, D., Heylen, D. K. J., & op den Akker, H. J. A. (2009). On the contextual analysis of agreement scores. In J. C. Martin, P. Paggio, M. Kipp, & D. K. J. Heylen (Eds.) *Multimodal Corpora – From Models of Natural Interaction to Systems and Applications, Morocco*, vol. 5509 of *Lecture Notes in Computer Science*, (pp. 122–137). Berlin: Springer Verlag.

Reidsma, D., & op den Akker, H. J. A. (2008). Exploiting "subjective" annotations. In R. Artstein, G. Boleda, F. Keller, & S. Schulte im Walde (Eds.) *Coling 2008: Proceedings of the workshop on Human Judgements in Computational Linguistics, Manchester, UK*, (pp. 8–16). Coling 2008 Organizing Committee.

Rentoumi, V., Petrakis, S., Klenner, M., Vouros, G. A., & Karkaletsis, V. (2010). United we stand: Improving sentiment analysis by joining machine learning and rule based methods. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odjik, S. Piperidis, M. Rosner, & D. Tapias (Eds.) *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, (pp. 1089–1094). Valletta, Malta: European Language Resources Association (ELRA).

Reyes, A., & Rosso, P. (2011). Mining subjective knowledge from customer reviews: A specific case of irony detection. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011)*, (pp. 118–124). Portland, Oregon: Association

for Computational Linguistics.

Roberts, K., Roach, M. A., Johnson, J., Guthrie, J., & Harabagiu, S. M. (2012). Empatweet: Annotating and detecting emotions on twitter. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*, (pp. 3806–3813).

Rubin, D., & Talarico, J. (2009). A comparison of dimensional models of emotion: Evidence from emotions, prototypical events, autobiographical memories, and words. *Memory*, *17*(8), 802–808.

Rubin, V. L., Stanton, J. M., & Liddy, E. D. (2004). Discerning Emotions in Texts. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*, 1985. Stanford, US.

Russell, J. A. (1978). Evidence of convergent validity on the dimensions of affect. *Journal of personality and social psychology*, *36*(10), 1152.

Russell, J. A. (1980). A circumplex model of affect. *Journal of Personality and Social Psychology*, *39*, 1161–1178.

Russell, J. A., & Barrett, L. F. (1999). Core affect, prototypical emotional episodes, and other things called emotion: Dissecting the elephant. *Journal of personality and social psychology*, *76*(5), 805.

Russell, J. A., & Carroll, J. M. (1999). On the bipolarity of positive and negative affect. *Psychological bulletin*, *125*(1), 3.

Russell, J. A., & Mehrabian, A. (1977). Evidence for a three-factor theory of emotions. *Journal of research in Personality*, *11*(3), 273–294.

Scherer, K., Dan, E., & Flykt, A. (2006). What determines a feeling's position in affective space? a case for appraisal. *Cognition & Emotion*, *20*(1), 92–113.

Scherer, K. R. (2005). What are emotions? and how can they be measured? *Social science information*, *44*(4), 695–729.

Scott, W. A. (1955). Reliability of content analysis: The case of nominal scale coding. *Public opinion quarterly*.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, *34*(1), 1–47.

Silla, J., CarlosN., & Freitas, A. (2011). A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, *22*(1-2), 31–72.

Sintsova, V., Musat, C., & Pu, P. (2013). Fine-grained emotion recognition in olympic tweets based on human computation. In *Proceedings of the 4th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, (pp. 12–20). Atlanta, Georgia: Association for Computational Linguistics.

Sohn, S., Torii, M., Li, D., Wagholikar, K., Wu, S., & Liu, H. (2012). A hybrid approach to sentiment sentence classification in suicide notes. *Biomed Inform Insights*, *5*(Suppl. 1), 43–50. [PubMed Central:PMC3409488] [DOI:10.4137/BII.S8961] [PubMed:22879759].

Spitters, S., Sanders, G., op den Akker, H., & Bruijnes, M. (2013). The recognition of acted interpersonal stance in police interrogations. In *Proceedings of the 4th IEEE International Conference on Cognitive Infocommunications, CogInfoCom 2013*, (p. 6). Hungary: IEEE Hungary Section, IEEE Joint Chapter of IES and RAS, Hungary.

Stevenson, R. a., Mikels, J. a., & James, T. W. (2007). Characterization of the affective norms for English words by discrete emotional categories. *Behavior research methods*, *39*(4), 1020–4.

Stone, P. J., Kirsch, J., & Associates, C. C. (1966). *The general inquirer: a computer approach to content analysis*. M.I.T. Press.

Strapparava, C., & Mihalcea, R. (2007). Semeval-2007 task 14: Affective text. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, (pp. 70–74). Prague, Czech Republic: Association for Computational Linguistics.

Strapparava, C., & Mihalcea, R. (2008). Learning to identify emotions in text. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, (pp. 1556–1560). New York, NY, USA: ACM.

Strapparava, C., Mihalcea, R., & Battocchi, A. (2007). A parallel corpus of music and lyrics annotated with emotions. In *Proceedings of the 8th international conference on Language Resources*

and Evaluation (LREC-2012), Istanbul, Turkey, May 2012., (pp. 2343–2346).

Strapparava, C., & Valitutti, A. (2004). WordNet-Affect: an affective extension of WordNet. In *Proceedings of LREC*, vol. 4, (pp. 1083–1086).

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistics*, *37*(2), 267–307.

Thayer, R. E. (1990). *The biopsychology of mood and arousal*. Oxford University Press, USA.

Tokuhisa, R., Inui, K., & Matsumoto, Y. (2008). Emotion classification using massive examples extracted from the web. In *COLING '08: Proceedings of the 22nd International Conference on Computational Linguistics*, (pp. 881–888). Morristown, NJ, USA: Association for Computational Linguistics.

Tsur, O., Davidov, D., & Rappoport, A. (2010). ICWSM — a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In M. Hearst, W. Cohen, & S. Gosling (Eds.) *Proceedings of the Fourth International Conference on Weblogs and Social Media (ICWSM-2010)*. The AAAI Press, Menlo Park, California.

Turney, P. D., & Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *CoRR*, *cs.CL/0309*.

Vaassen, F., & Daelemans, W. (2011). Automatic Emotion Classification for Interpersonal Communication. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011)*, (pp. 104–110). Portland, Oregon: Association for Computational Linguistics.

van Atteveldt, W., Kleinnijenhuis, J., Ruigrok, N., & Schlobach, S. (2008). Good news or bad news: Conducting sentiment analysis on Dutch texts to distinguish between positive and negative relations. *Journal of Information Technology & Politics*, *5*(1), 73–94.

van den Bosch, A., Busser, B., Canisius, S., & Daelemans, W. (2007). An efficient memory-based morphosyntactic tagger and parser for Dutch. In F. V. Eynde, P. Dirix, I. Schuurman, & V. Vandeghinste (Eds.) *Selected Papers of the 17th Computational Linguistics in the Netherlands Meeting*, (pp. 99–114). Leuven, Belgium.

van Dijk, B. (2000). *Beïnvloed anderen, begin bij jezelf. Over gedrag en de Roos van Leary*. Thema, 4th ed.

van Dijk, B., & Moes, F. (2005). Het grote beïnvloedingsspel.

van Zaanen, M., & Kanters, P. (2010). Automatic Mood Classification Using TF*IDF Based on Lyrics. In J. S. Downie, & R. C. Veltkamp (Eds.) *ISMIR*, Ismir, (pp. 75–80). International Society for Music Information Retrieval.

Vossen, P., Hofmann, K., de Rijke, M., Sang, E. T., & Deschacht, K. (2007). The Cornetto database: Architecture and user-scenarios. In M. F. Moens, T. Tuytelaars, & A. P. de Vries (Eds.) *Proceedings DIR 2007*, (pp. 89–96).

Wang, W., Chen, L., Thirunarayan, K., & Sheth, A. P. (2012). Harnessing twitter "big data" for automatic emotion identification. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, (pp. 587–592). IEEE.

Warriner, A., Kuperman, V., & Brysbaert, M. (2013). Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behavior Research Methods*, (pp. 1–17).

Watson, D., & Clark, L. A. (1999). The PANAS-X: Manual for the positive and negative affect schedule-expanded form.

Watson, D., Clark, L. A., & Tellegen, A. (1988). Development and validation of brief measures of positive and negative affect: the PANAS scales. *Journal of personality and social psychology*, *54*(6), 1063.

Watson, D., & Tellegen, A. (1985). Toward a consensual structure of mood. *Psychological bulletin*, *98*(2), 219.

Wawer, A. (2010). Is sentiment a property of synsets? evaluating resources for sentiment classification using machine learning. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*. Valletta, Malta: European Language Resources Association (ELRA).

Whissell, C. (1989). The dictionary of affect in language. *Emotion: Theory, research, and experience*, *4*(113-131), 94.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, (pp. 347–354).

Wundt, W. (1896). Grundriss der psychologie. *Leipzig: Engelmann*.

Xu, Y., Wang, Y., Liu, J., Tu, Z., Sun, J. T., Tsujii, J., & Chang, E. (2012). Suicide note sentiment classification: a supervised approach augmented by web data. *Biomed Inform Insights*, *5*(Suppl. 1), 31–41. [PubMed Central:PMC3409493] [DOI:10.4137/BII.S8956] [PubMed:22879758].

Yang, H., Willis, A., de Roeck, A., & Nuseibeh, B. (2012). A hybrid model for automatic emotion recognition in suicide notes. *Biomed Inform Insights*, *5*(Suppl. 1), 17–30. [PubMed Central:PMC3409477] [DOI:10.4137/BII.S8948] [PubMed:22879757].

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, (pp. 947–953). Association for Computational Linguistics.

Yu, H.-F., Ho, C.-H., Juan, Y.-C., & Lin, C.-J. (2012). Libshorttext: A library for short-text classification and analysis. Tech. rep., CSIE, NTU.

Zachar, P., & Ellis, R. D. (2012). *Categorical Versus Dimensional Models of Affect: A Seminar on the Theories of Panksepp and Russell*, vol. 7. John Benjamins Publishing Company.

Glossary

accuracy In text classification, accuracy refers to the ratio of documents that have been correctly attributed to their class vs. the total number of documents.
Formally:

$$acc = \frac{\text{correctly classified documents}}{\text{all documents}}$$

or

$$acc = \frac{tp+tn}{tp+fp+tn+fn}$$

(where tp = true positives, tn = true negatives, fp = false positives, fn = false negatives). 19, 35

Amazon Mechanical Turk A web service which allows requesters to create human intelligence tasks, which will then be solved by workers, for a fee. Within the field of computational linguistics, the Mechanical Turk platform is mostly used for annotation, as it provides a cost-efficient way to get data annotated by a large number of people (or indeed to get a lot of data annotated quickly). Almost anyone can be a worker for a Mechanical Turk task, and it is therefore essential for requesters to build in sanity checks to weed out random or willfully incorrect input. Amazon also provides ratings for workers, so it is possible to limit your workers to experienced and trusted individuals. 18

bag-of-words Bag-of-words is a popular approach to constructing feature vectors from text documents. An index (a "bag of words") of all words occurring in the training set is compiled. For each document, a feature vector is created by counting the number of times each word occurs in the current document. 11, 21, 32, 34, 38, 39, 55, 66, 104

boosting Boosting is an ensemble method wherein the instances of smaller, hard to detect classes are given higher weights so as to boost their classification performance. 11

chunk A chunk —in the context of linguistics— is a sequence of uninterrupted words that carry similar syntactic function. Possible chunk types include noun phrase chunks, verb phrase chunks, etc.
e.g. *[NP The black cat] [VP drank] [NP all the milk].* 32

content word A content word is a word which carries lexical meaning, as opposed to function words, which serve to indicate syntactic function. Content words are usually nouns, adjectives, verbs, and adverbs. 20

cross-validation To better evaluate the performance of a classifier on a dataset, one can perform n-fold cross-validation. Instead of using a single static training set to train the model and

a single static test set to evaluate it, for n-fold cross-validation, the dataset is split into n parts, and n experiments are carried out where a single part of the dataset is held out as the test set. 50

eager learner An "eager learner" (as opposed to a "lazy learner") is a learner which, given the set of training instances, builds a model that abstracts away from the instances on which it was trained. A Naïve Bayes classifier, for instance, will build a probability model which can be used to classify new instances. The original training instances are no longer required. 92

F-score The F-score is the harmonic mean of precision and recall.
Formally:

$F_\beta = (1 + \beta^2).\frac{\text{precision.recall}}{(\beta^2.\text{precision})+\text{recall}}$

or in terms of true positives (tp), false negatives (fn) and false positives (fp):

$R = \frac{tp}{tp+fn}$
(where tp = true positives, fn = false negatives) . 37–39, 54, 56, 66, 71, 108

hashtag On a variety of social networks, a hashtag is a keyword or a phrase (without spaces) preceded by a hash (#) that serves to indicate the topic or another salient property of the current post. A tweet can for instance be labeled with the hashtag #2014Olympics to indicate that it is about the 2014 Olympics. #WhenIWasAKid may group tweets containing anecdotes from users' childhoods. Social networks rarely impose restrictions on hashtags, and they take on many forms and serve many purposes. 22

homograph Homographs are words that have the same written form but have different meaning. A popular example is the difference between "bank" (river bank) and "bank" (financial institution). 18

information gain Information gain is a measure of how much a feature can contribute to determining the correct class label of an instance. Mathematically, it measures the difference between the entropy of a system which has no knowledge from the feature and a system that does. The larger this difference, the more information is said to be contained by the feature. Formally:

$IG_i = H(C) - \sum_{v \in V_i} P(v)xH(C|v)$

where $H(C)$ is the entropy of the class labels C ($- \sum_{c \in C} P(c)log_2P(c)$) and V_i is the set of values for feature i.
See TiMBL Reference Guide: http://ilk.uvt.nl/timbl/ (Last accessed: July 3rd, 2014). 55

inter-annotator agreement The extent to which annotators agree with each other on the annotation of a given dataset. Inter-annotator agreement is commonly expressed through κ scores (Cohen's or Fleiss' κ), but alternative measures of agreement exist (Bennett, 1954; Scott, 1955). 30, 104

k-Nearest Neighbors classifier A k-NN classifier is a lazy learner which keeps all training instances in memory, and which determines the label of a new instance by determining the labels of the n training instances that are nearest to it (according to some distance metric). 91, 92

Latent Semantic Analysis Latent Semantic Analysis is a technique used to model the meaning of words by analyzing their occurrence in a representative collection of documents. A word will be represented in function of its occurrence in every document (or, possibly, in every text fragment) in the dataset. The dimensionality of this term-document matrix is then reduced using Singular Value Decomposition, which results in a lower-dimensional matrix that retains the semantic relationships between terms. This matrix can then be used to calculate semantic similarity between terms, between documents and terms, and between documents. 8, 21

lazy learner A "lazy learner" (as opposed to an "eager learner") is a learner which does not attempt to build a generalizing model from the training data. The typical example of a lazy learner is a k-NN classifier, which keeps all training instances, unchanged, in memory. When a new instance comes in, its class is determined by comparing it with the training instances in memory. 91, 92

lemma A lemma is a word in its canonical form. The words "manages", "managing" and "managed", for instance, all have the same lemma, namely "manage". 32

macro-averaged F-score F-scores are a measure of the classification performance for a given class. To get a global view of the performance of a classifier on a multi-class task, one can average F-scores. Macro-averaging is on possible technique to calculate the average of multiple scores. For macro-averaging, the F-score is calculated for each possible class, and the average of these scores is taken. Formally:

$$F_{micro} = \frac{\sum_{c=1}^{n} F(tp_{c_i}, fp_{c_i}, fn_{c_i}, tn_{c_i})}{n}$$

where c_i represents class i in the set of possible classes $C = c_1, c_2 ... c_i ... c_n$ with size n. 35, 49, 54, 67, 90, 108

Maximum Entropy A Maximum Entropy classifier builds a model given a set of training instances. The model should satisfy the contraints of the training data, but should make no decisions whatsoever about constraints that are not reflected in the data. In fact, a ME classifier attempts to keep uncertainty/entropy at its highest at all times, so the model will be able to generalize over new data. A ME classifier does not presuppose independence between features, as is the case for a Naïve Bayes classifier. 46

micro-averaged F-score F-scores are a measure of the classification performance for a given class. To get a global view of the performance of a classifier on a multi-class task, one can average F-scores. Micro-averaging is one possible technique to calculate the average of multiple scores. For micro-averaging, true positives, false positives, false negatives and true negatives for each class are summed prior to the calculation of the F-score. Formally:

$$F_{micro} = F(\sum_{i=1}^{n} tp_{c_i}, \sum_{i=1}^{n} fp_{c_i}, \sum_{i=1}^{n} fn_{c_i}, \sum_{i=1}^{n} tn_{c_i})$$

where c_i represents class i in the set of possible classes $C = c_1, c_2 ... c_i ... c_n$ with size n. 35, 49, 52, 54, 67, 68, 90, 108

Naïve Bayes A Naïve Bayes classifier is a probabilistic classifier which uses conditional probability to determine the most probable label of an instance given its (independent) features. 11, 21, 46, 47

n-gram In the context of feature extraction from text, an n-gram is a sequence of n adjacent textual units. These units can be realized in many different ways. Common n-gram types are token n-grams (n sequential word tokens) and character n-grams (n sequential characters), but it is theoretically possible to construct n-grams from any type of sequential data. 21, 32, 35, 37–39, 42, 48, 55, 66, 67, 69, 70, 97, 104

one vs. all A one vs. all classifier is a classifier which, given a multi-class dataset, pits the instances of a single class against the instances from all other classes. The term "one vs. all classifier" is often used to indicate an ensemble of multiple one vs. all classifiers –one for each class in the dataset–, where the final label predictions are an aggregate of the individual classifier predictions. 48, 50, 51, 55–57, 87

part of speech The part of speech of a word (also known as "word class") is the word's lexical category based on its syntactic function. Parts of speech include *noun*, *verb*, etc. 5, 12, 32, 37, 38, 47, 107

Pearson correlation Formally known as the "Pearson product-moment correlation coefficient" (PPMCC), the Pearson coefficient is a measure of correlation between two variables. The coefficient can range between -1 and 1, with 0 indicating no correlation between the variables, -1 indicating perfect negative correlation (i.e. there is a linear relationship between variables X and Y; as X goes up, Y goes down by a proportionate amount), and 1 perfect positive correlation (as X goes up, Y goes up by a proportionate amount).
Formally, the Pearson coefficient is calculated as:

$$r = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2 \sum_{i=1}^{n}(Y_i-\bar{Y})^2}}$$

with X and Y as the variables between which correlation is to be measured. 19, 20

Pointwise Mutual Information Pointwise Mutual Information is a measure of association between two terms. It weighs the probability of the two terms occurring together against the probabilities of each term occurring separately:

$PMI(x, y) = log\frac{p(x,y)}{p(x)p(y)}$. 8, 20

precision In text classification, precision refers to the ratio of documents that have been correctly attributed to a given class vs. the total number of documents attributed to that class.
Formally:

$$P = \frac{\text{documents correctly classified as belonging to } C_i}{\text{all documents classified as belonging to } C_i}$$

or

$P = \frac{tp}{tp+fp}$
(where tp = true positives, fp = false positives) . 19, 51, 56, 108

recall In text classification, recall refers to the ratio of documents that have been correctly attributed to a given class vs. the total number of documents attributed to that class.
Formally:

$$R = \frac{\text{documents correctly classified as belonging to } C_i}{\text{all documents truly belonging to } C_i}$$

or

$R = \frac{tp}{tp+fn}$
(where tp = true positives, fn = false negatives) . 19, 51, 56, 108

Support Vector Machines Put simply, an SVM classifier is a binary classification algorithm which places all training instances in a multidimensional space in such a way that there is a

maximally large gap between the instances in the positive class and those in the negative class. The hyperplane drawn in the middle of this gap serves as a boundary for future instances: if a test instance falls on the positive side of the boundary, it will be labeled as positive, and an instance mapped on the negative side will be labeled as negative. 11, 35

SVM Support Vector Machines. 10, 11, 21, 35, 40, 41, 46–49, 58, 70, 73, 91, 92, 99, 105, 107, *Glossary:* Support Vector Machines

synset Words that are semantically related, i.e. a word and its synonyms. 8, 11

tf-idf The tf-idf score, or "term frequency–inverse document frequency" score, is a measure of importance of word for a given document in a dataset. It takes into account the number of times the word occurs in said document, but normalizes it for the frequency of the word in the entire corpus. Words which occur often in the document, but which are otherwise relatively rare in the dataset, will thus get a higher tf-idf score. 33, 34, 48

turn In the context of conversations, a *turn* is a sequence of utterances produced by the same speaker without interruption by their interlocutor. Interjections and short interruptions by the listener are not considered as separate turns. It is only when the interlocutor takes the floor that we indicate a change of turn. 28

Wizard-of-Oz A Wizard-of-Oz experiment is an experiment in which parts of the tasks of a software application are taken over by a human intelligence. In most cases, the subjects interacting with the "wizard" do not know they are in fact interacting with a human being. In the case of the deLearyous-project, the virtual character was entirely replaced by a human actor. Due to the nature of the interaction between the player and the virtual character (the player can see and hear the AI), the player was aware that the AI had been replaced. All other parameters within the experiment remained constant, however: players could only talk to the actor using text, while they could see and hear the actor respond directly. The actor, on the other hand, was transmitting audio and video, but had no access to any other communication channels. 28, 34

WordNet WordNet is a large network of words which are clustered by meaning into synsets. These synsets are connected among themselves by relations such as synonymy, hypernymy, hyponymy... 8, 9, 18, 19, 47

word-sense disambiguation Word-sense disambiguation is concerned with identifying the specific meaning of a word which may have several homographs. Classic examples include "tear", which, depending on the context, can refer to a hole (e.g. in a piece of clothing) or to a "teardrop". 6, 19

Index