In: *Proceedings of CoNLL-2000 and LLL-2000*, pages 43–48, Lisbon, Portugal, 2000.

# Memory-Based Learning for Article Generation

**Guido Minnen***
Cognitive and Computing Sciences
University of Sussex
Falmer BN1 9QH, Brighton, UK
Guido.Minnen@cogs.susx.ac.uk

**Francis Bond**†
MT Research Group
NTT Communication Science Labs
2-4 Hikari-dai, Kyoto 619-0237, JAPAN
bond@cslab.kecl.ntt.co.jp

**Ann Copestake**
CSLI
Stanford University
Stanford CA 94305-2150, USA
aac@csli.stanford.edu

## Abstract

Article choice can pose difficult problems in applications such as machine translation and automated summarization. In this paper, we investigate the use of corpus data to collect statistical generalizations about article use in English in order to be able to generate articles automatically to supplement a symbolic generator. We use data from the Penn Treebank as input to a memory-based learner (TiMBL 3.0; Daelemans et al., 2000) which predicts whether to generate an article with respect to an English base noun phrase. We discuss competitive results obtained using a variety of lexical, syntactic and semantic features that play an important role in automated article generation.

## 1 Introduction

Article choice can pose difficult problems in natural language applications. Machine translation (MT) is an example of such an application. When translating from a source language that lacks articles, such as Japanese or Russian, to one that requires them, such as English or German, the system must somehow generate the source language articles (Bond and Ogura, 1998). Similarly in automated summarization: when sentences or fragments are combined or reduced, it is possible that the form of a noun phrase (NP) is changed such that a change of the article associated with the NP's head becomes necessary. For example, consider the sentences *A talk will be given on Friday about NLP; The talk will last for one hour* which might get summarized as *Friday's NLP talk will last one hour*. However, given the input sentences, it is not clear how to decide not to generate an article for the subject NP in the output sentence.

Another important application is in the field known as augmentative and alternative communication (AAC). In particular, people who have lost the ability to speak sometimes use a text-to-speech generator as a prosthetic device. But most disabilities which affect speech, such as stroke or amyotrophic lateral sclerosis (ALS or Lou Gehrig's disease), also cause some more general motor impairment, which means that prosthesis users cannot achieve a text input rate comparable to normal typing speeds even if they are able to use a keyboard. Many have to rely on a slower physical interface (headstick, head-pointer, eye-tracker etc). We are attempting to use a range of NLP technology to improve text input speed for such users. Article choice is particularly important for this application: many AAC users drop articles and resort to a sort of telegraphese, but this causes degradation in comprehension of synthetic speech and contributes to its perception as unnatural and robot-like. Our particular goal is to be able to use an article generator in conjunction with a symbolic generator for AAC (Copestake, 1997; Carroll et al., 1999).

In this paper we investigate the use of corpus data to collect statistical generalizations about article use in English so as to be able to generate them automatically. We use data from the Penn Treebank as input to a memory-based learner (TiMBL 3.0; Daelemans et al., 2000) that is used to predict whether to generate *the* or *a/an* or no article.[1] We discuss a variety of lexical, syntactic and semantic features that

* Visiting CSLI, Stanford University (2000).
† Visiting CSLI, Stanford University (1999-2000).

[1] We assume a postprocessor to determine whether to generate *a* or *an* as described in Minnen et al. (2000).

play an important role in automated article generation, and compare our results with other researchers'.

The paper is structured as follows. Section 2 relates our work to that of others. Section 3 introduces the features we use. Section 4 introduces the learning method we use. We discuss our results in Section 5 and suggest some directions for future research, then conclude with some final remarks in Section 6.

## 2 Related Work

There has been considerable research on generating articles in machine translation systems (Gawrońska, 1990; Murata and Nagao, 1993; Bond and Ogura, 1998; Heine, 1998). These systems use hand-written rules and lexical information to generate articles. The best cited results, 88% accuracy, are quoted by Heine (1998) which were obtained with respect to a very small corpus of 1,000 sentences in a restricted domain.

Knight and Chander (1994) present an approach that uses decision trees to determine whether to generate *the* or *a/an*. They do not consider the possibility that no article should be generated. On the basis of a corpus of 400K NP instances derived from the Wall Street Journal, they construct decision trees for the 1,600 most frequent nouns by considering over 30,000 lexical, syntactic and semantic features. They achieve an accuracy of 81% with respect to these nouns. By guessing *the* for the remainder of the nouns, they achieve an overall accuracy of 78%.

## 3 Features Determining Automated Article Generation

We have extracted 300K base noun phrases (NPs) from the Penn Treebank Wall Street Journal data (Bies et al., 1995) using the tgrep tool. The distribution of these NP instances with respect to articles is as follows: *the* 20.6%, *a/an* 9.4% and 70.0% with no article.

We experimented with a range of features:

1. Head of the NP: We consider as the head of the NP the rightmost noun in the NP. If an NP does not contain a noun, we take the last word in the NP as its head.

2. Part-of-speech (PoS) tag of the head of the NP: PoS labels were taken from the Penn Treebank. We list the tags that occurred with the heads of the NPs in Table 1.

```
(PP-DIR to/TO
    (NP the/DT problem/NN))
```

Figure 1: An example of a prepositional phrase annotated with a functional tag

| PoS Tag | *the* | *a/an* | no |
|---|---|---|---|
| NN | 42,806 | 27,160 | 53,855 |
| NNS | 10,705 | 446 | 58,118 |
| NNP | 6,938 | 271 | 47,721 |
| NNPS | 536 | 2 | 1,329 |
| CD | 382 | 180 | 13,368 |
| DT | 18 | 0 | 3,045 |
| PRP | 0 | 0 | 21,214 |
| PRP$ | 0 | 0 | 25 |
| EX | 0 | 0 | 1,073 |
| IN | 0 | 1 | 502 |
| JJ | 388 | 143 | 931 |
| JJR | 11 | 1 | 310 |
| JJS | 184 | 0 | 282 |
| RB | 15 | 41 | 498 |
| VBG | 43 | 12 | 210 |
| VB | 0 | 1 | 89 |
| WDT | 2 | 0 | 4,812 |
| WP | 0 | 0 | 2,759 |
| Misc. | 40 | 8 | 269 |
| Total: | 62,068 | 28,266 | 210,410 |

Table 1: Distribution of NP instances in Wall Street Journal data (300,744 NPs in all)

3. Functional tag of the head of the NP: In the Penn Treebank each syntactic category can be associated with up to four functional tags as listed in Table 2. We consider the sequence of functional tags associated with the category of the NP as a feature; if a constituent has no functional tag, we give the feature the value `NONE`.

4. Category of the constituent embedding the NP: We looked at the category of the embedding constituent. See Figure 1: The category of the constituent embedding the NP *the problem* is `PP`.

5. Functional tag of the constituent embedding the NP: If the category of the constituent embedding the NP is associated with one or more functional tags, they are used as features. The functional tag of the constituent embedding *the problem* in Figure 1 is `DIR`.

6. Other determiners of the NP: We looked at the presence of a determiner in the NP. By definition, an NP in the Penn Treebank can only

| Functional Tag (ft) | Marks: |
|---|---|
| | **Text categories** |
| HLN | headlines and datelines |
| LST | list markers |
| TTL | titles |
| | **Grammatical functions** |
| CLF | true clefts |
| NOM | non NPs that function as NPs |
| ADV | clausal and NP adverbials |
| LGS | logical subjects in passives |
| PRD | nonVP predicates |
| SUBJ | surface subject |
| TPC | topicalized/fronted constituents |
| CLR | closely related |
| BNF | beneficiary of action |
| DTV | dative object |
| | **Semantic roles** |
| VOC | vocatives |
| DIR | direction and trajectory |
| LOC | location |
| MNR | manner |
| PRP | purpose and reason |
| TMP | temporal phrases |
| PUT | locative complement of *put* |
| EXT | spatial extent of activity |

Table 2: Functional tags and their meaning (Santorini, 1990)

have one determiner (Bies et al., 1995), so we expect it to be a good predictor of situations where we should not generate an article.

7. Head countability preferences of the head of the NP: In case the head of an NP is a noun we also use its countability as a feature. We anticipate that this is a useful feature because singular indefinite countable nouns normally take the article *a/n*, whereas singular indefinite uncountable nouns normally take no article: *a dog* vs *water*. We looked up the countability from the transfer lexicon used in the Japanese-to-English machine translation system ALT-J/E (Ikehara et al., 1991). We used six values for the countability feature: `FC` (fully countable) for nouns that have both singular and plural forms and can be directly modified by numerals and modifiers such as *many*; `UC` (uncountable) for nouns that have no plural form and can be modified by *much*; `SC` (strongly countable) for nouns that are more often countable than uncountable; `WC` (weakly countable) for nouns that are more often uncountable than countable; and `PT` (pluralia tantum) for nouns that only have plural forms, such as for example, *scissors* (Bond et al., 1994). Finally, we used the value `UNKNOWN` if the lexicon did not provide countability information for a noun or if the head of the NP was not a noun. 41.4% of the NP instances received the value `UNKNOWN` for this feature.

8. Semantic classes of the head of the NP: If the head of the NP is a noun we also take into account its semantic classification in a large semantic hierarchy. The underlying idea is that the semantic class of the noun can be used as a way to back off in case of unknown head nouns. The 2,710 node semantic hierarchy we used was also developed in the context of the ALT-J/E system (Ikehara et al., 1991). Edges in this hierarchy represent IS-A or HAS-A relationships. In case the semantic classes associated with two nodes stand in the IS-A relation, the semantic class associated with the node highest in the hierarchy subsumes the semantic class associated with the other node.

Each of the nodes in this part of the hierarchy is represented by a boolean feature which is set to `1` if that node lies on the path from the root of the hierarchy to a particular semantic class. Thus, for example, the semantic features of a noun in the semantic class `organization` consists of a vector of 30 features where the features corresponding to the nodes `noun`, `concrete`, `agent` and `organization` are set to `1` and all other features are set to `0`.[2]

## 4 Memory-based learning

We used the Tilburg memory based learner TiMBL 3.0.1 (Daelemans et al., 2000) to learn from examples for generating articles using the features discussed above. Memory-based learning reads all training instances into memory and classifies test instances by extrapolating a class from the most similar instance(s) in memory.

Daelemans et al. (1999) have shown that for typical natural language tasks, this approach has the advantage that it also extrapolates from exceptional and low-frequency instances. In addition, as a result of automatically weighing features in the similarity function used to determine the class of a test instance, it allows the user to incorporate large

[2]If a noun has multiple senses, we collapse them by taking the semantic classes of a noun to be the union of the semantic classes of all its senses.

numbers of features from heterogeneous sources: When data is sparse, feature weighing embodies a smoothing-by-similarity effect (Zavrel and Daelemans, 1997).

## 5 Evaluation and Discussion

We tested the features discussed in section 3 with respect to a number of different memory-based learning methods as implemented in the TiMBL system (Daelemans et al., 2000).

We considered two different learning algorithms. The first, IB1 is a $k$-nearest neighbour algorithm.[3]. This can be used with two different metrics to judge the distance between the examples: overlap and modified value difference metric (MVDM). TiMBL automatically learns weights for the features, using one of five different weighting methods: no weighting, gain ratio, information gain, chi-squared and shared variance. The second algorithm, IGTREE, stores examples in a tree which is pruned according to the weightings. This makes it much faster and of comparable accuracy. The results for these different methods, for $k = 1, 4, 16$ are displayed in Table 3. IB1 is tested with leave-one-out cross-validation, IGTREE with ten-fold cross validation.

The best results were (82.6%) for IB1 with the MVDM metric, and either no weighting or weighting by gain ratio. IGTREE did not perform as well. We investigated more values of $k$, from 1 to 200, and found they had little influence on the accuracy results with $k = 4$ or 5 performing slightly better.

We also tested each of the features described in Section 3 in isolation and then all together. We used the best performing algorithm from our earlier experiment: IB1 with MVDM, gain ratio and $k = 4$. The results of this are given in Table 4.

When interpreting these results it is important to recall the figures provided in Table 1. The most common article, for any PoS, was `no` and for many PoS, including pronouns, generating no article is always correct. There is more variation in NPs headed by common nouns and adjectives, and a little in NPs headed by proper nouns. Our baseline therefore consists of never generating an article: this will be right in 70.0% of all cases.

[3]Strictly speaking, it is a $k$ nearest distance algorithm, which looks at all examples in the nearest $k$ distances, the number of which may be greater than $k$.

| **Feature** | **Accuracy** |
|---|---|
| head | 80.3% |
| head's part-of-speech | 70.0% |
| NP's functional tag | 70.5% |
| embedding category | 70.0% |
| embedding functional tag | 70.0% |
| determiner present or not | 70.0% |
| head's countability | 70.0% |
| head's semantic classes | 72.9% |
| hline | |

Table 4: Accuracy results by feature

Looking at the figures in Table 4, we see that many of the features investigated did not improve results above the baseline. Using the head of the NP itself to predict the article gave the best results of any single feature, raising the accuracy to 79.4%. The functional tag of the head of the NP itself improved results slightly. The use of the semantic classes (72.1%) clearly improves the results over the baseline thereby indicating that they capture useful generalizations.

The results from testing the features in combination are shown in Table 5. Interestingly, features which were not useful on their own, proved useful in combination with the head noun. The most useful features appear to be the category of the embedding constituent (81.1%) and the presence or absence of a determiner (80.9%). Combining all the features gave an accuracy of 82.9%.

| **Feature** | **Accuracy** |
|---|---|
| head+its part-of-speech | 80.8% |
| head+functional tag of NP | 81.1% |
| head+embedding category | 80.8% |
| head+embedding functional tag | 81.4% |
| head+determiner present or not | 81.7% |
| head+countability | 80.8% |
| head+semantic classes | 80.8% |
| hline all features | 83.6% |
| all features-semantic classes | 83.6% |

Table 5: Accuracy with combined features

Our best results (82.6%), which used all features are significantly better than the baseline of generating no articles (70.0%) or using only the head of the NP for training (79.4%). We

| **Algorithm** | $k$ | None | Gain ratio | Information gain | $\chi^2$ | Shared variance |
|---|---|---|---|---|---|---|
| IB1 | 1 | 83.5% | 83.5% | 83.3% | 83.2% | 83.3% |
| (MVDM) | 4 | 83.5% | **83.6%** | 83.3% | 83.3% | 83.3% |
| | 16 | 83.6% | 83.5% | 83.2% | 83.2% | 83.2% |
| IB1 | 1 | 83.1% | 83.5% | 83.3% | 83.2% | 83.3% |
| (overlap) | 4 | 82.9% | 83.1% | 83.1% | 83.1% | 83.1% |
| | 16 | 82.9% | 83.0% | 82.9% | 82.9% | 82.9% |
| IGTREE | — | — | 82.9% | 82.5% | 82.4% | 82.6% |

Table 3: Accuracy results broken down with respect to memory-based learning methods used

also improve significantly upon earlier results of 78% as reported by Knight and Chander (1994), which in any case is a simpler task since it only involved choice between *the* and *a/an*. Further, our results are competitive with state of the art rule-based systems. Because different corpora are used to obtain the various results reported in the literature and the problem is often defined differently, detailed comparison is difficult. However, the accuracy achieved appears to approach the accuracy results achieved with handwritten rules.

In order to test the effect of the size of the training data, we tested used the best performing algorithm from our earlier experiment (IB1 with MVDM, gain ratio and $k = 4$) on various subsets of the corpus: the first 10%, the first 20%, the first 30% and so on to the whole corpus. The results are given in Table 6.

| **Size** | **Accuracy** |
|---|---|
| 10% | 80.95% |
| 20% | 81.67% |
| 30% | 82.14% |
| 40% | 82.45% |
| 50% | 82.69% |
| 60% | 83.04% |
| 70% | 83.17% |
| 80% | 83.24% |
| 90% | 83.45% |
| 100% | 83.58% |

(100% is 300,744 NPs)

Table 6: Accuracy versus Size of Training Data

The accuracy is still improving even with 300,744 NPs, an even larger corpus should give even better results. It is important to keep in mind that we, like most other researchers, have been training and testing on a relatively homogeneous corpus. Furthermore, we took as given information about the number of the NP. In many applications we will have neither a large amount of homogeneous training data nor information about number.

### 5.1 Future Work

In the near future we intend to further extend our approach in various directions. First, we plan to investigate other lexical and syntactic features that might further improve our results, such as the existence of pre-modifiers like superlative and comparative adjectives, and post-modifiers like prepositional phrases, relative clauses, and so on. We would also like to investigate the effect of additional discourse-based features such as one that incorporates information about whether the referent of a noun phrase has been mentioned before.

Second, we intend to make sure that the features we are using in training and testing will be available in the applications we consider. For example, in machine translation, the input noun phrase may be *all dogs*, whereas the output could be either *all dogs* or *all the dogs*. At present, words such as *all*, *both*, *half* in our input are tagged as pre-determiners if there is a following determiner (it can only be *the* or a possessive), and determiners if there is no article. To train for a realistic application we need to collapse the determiner and pre-determiner inputs together in our training data.

Furthermore, we are interested in training on corpora with less markup, like the British National Corpus (Burnard, 1995) or even no markup at all. By running a PoS tagger and then an NP chunker, we should be able to get a lot more training data, and thus significantly improve our coverage. If we can use plain text to train on, then it will be easier to adapt our tool quickly to new domains, for which there are unlikely to be fully marked up corpora.

## 6 Concluding remarks

We described a memory-based approach to automated article generation that uses a variety of lexical, syntactic and semantic features as provided by the Penn Treebank Wall Street Journal data and a large hand-encoded MT dictionary. With this approach we achieve an accuracy of 82.6%. We believe that this approach is an encouraging first step towards a statistical device for automated article generation that can be used in a range of applications such as speech prosthesis, machine translation and automated summarization.

### Acknowledgments

The authors would like to thank the Stanford NLP reading group, the LinGO project at CSLI, Timothy Baldwin, Kevin Knight, Chris Manning, Walter Daelemans and two anonymous reviewers for their helpful comments. This project is in part supported by the National Science Foundation under grant number IRI-9612682.

## References

Ann Bies, Mark Fergusona, Karen Katz, and Robert MacIntyre, 1995. *Bracketing Guidelines for Treebank II Style*. Penn Treebank Project, University of Pennsylvania.

Francis Bond and Kentaro Ogura. 1998. Reference in Japanese-to-English machine translation. *Machine Translation*, 13(2–3):107–134.

Francis Bond, Kentaro Ogura, and Satoru Ikehara. 1994. Countability and number in Japanese-to-English machine translation. In *15th International Conference on Computational Linguistics: COLING-94*, pages 32–38, Kyoto. (`http://xxx.lanl.gov/abs/cmp-lg/9511001`).

Lou Burnard. 1995. User reference guide for the British National Corpus. Technical report, Oxford University Computing Services.

John Carroll, Ann Copestake, Dan Flickinger, and Victor Poznanski. 1999. An efficient chart generator for (semi-)lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG'99)*, pages 86–95, Toulouse, France.

Ann Copestake. 1997. Augmented and alternative NLP techniques for augmentative and alternative communication. In *Proceedings of the ACL workshop on Natural Language Processing for Communication Aids*, pages 37–42, Madrid.

Walter Daelemans, Antal van den Bosch, and Jakub Zavrel. 1999. Forgetting exceptions is harmful in language learning. *Machine Learning*, 34.

Walter Daelemans, Jakub Zavrel, Ko van der Sloot, and Antal van den Bosch. 2000. TiMBL: Tilburg memory based learner, version 3.0, reference guide. ILK Technical Report 00-01, ILK, Tilburg, The Netherlands. (ILK-0001; `http://ilk.kub.nl`).

Barbara Gawrońska. 1990. "Translation Great Problem" on the problem of inserting articles when translating from Russian into Swedish. In *13th International Conference on Computational Linguistics: COLING-90*, Helsinki.

Julia E. Heine. 1998. Definiteness predictions for Japanese noun phrases. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics: COLING/ACL-98*, pages 519–525, Montreal, Canada.

Satoru Ikehara, Satoshi Shirai, Akio Yokoo, and Hiromi Nakaiwa. 1991. Toward an MT system without pre-editing – effects of new methods in **ALT-J/E**–. In *Third Machine Translation Summit: MT Summit III*, pages 101–106, Washington DC. (`http://xxx.lanl.gov/abs/cmp-lg/9510008`).

Kevin Knight and Ishwar Chander. 1994. Automated postediting of documents. In *Proceedings of the 12th National Conference on Artificial Intelligence: AAAI-94*, pages 779–784, Seattle. (`http://xxx.lanl.gov/abs/cmp-lg/9407028`).

Guido Minnen, John Carroll, and Darren Pearce. 2000. Robust, applied morphological generation. In *Proceedings of the first International Natural Language Genration Conference*, Mitzpe Ramon, Israel.

Masaki Murata and Makoto Nagao. 1993. Determination of referential property and number of nouns in Japanese sentences for machine translation into English. In *Fifth International Conference on Theoretical and Methodological Issues in Machine Translation: TMI-93*, pages 218–25, Kyoto, July. (`http://xxx.lanl.gov/abs/cmp-lg/9405019`).

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the Penn Treebank Project. Technical Report MS-CIS-90-47, Department of Computer and Information Science, University of Pennsylvania.

Jakub Zavrel and Walter Daelemans. 1997. Memory-based learning: Using similarity for smoothing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid, Spain.