All-Topology, Semi-Abstract Syntactic Features for Text Categorisation

Ari Chanen
School of Information Technologies
University of Sydney
Sydney, Australia, 2006
ari@it.usyd.edu.au

Jon Patrick
School of Information Technologies
University of Sydney
Sydney, Australia, 2006
jonpat@it.usyd.edu.au

Abstract

Good performance on Text Classification (TC) tasks depends on effective and statistically significant features. Typically, the simple bag-of-words representation is widely used because unigram counts are more likely to be significant compared to more compound features. This research explores the idea that the major cause of poor performance of some complex features is sparsity. Syntactic features are usually complex being made up of both lexical and syntactic information. This paper introduces the use of a class of automatically extractable, syntactic features to the TC task. These features are based on subtrees of parse trees. As such, a large number of these features are generated. Our results suggest that generating a diverse set of these features may help in increasing performance. Partial abstraction of the features also seems to boost performance by counteracting sparsity. We will show that various subsets of our syntactic features do outperform the bag-of-words representation alone.

1 Introduction

One of the keys to obtaining good performance out of an automatic TC system is choosing appropriate types of features. Humans can often do better than automatic classification systems on difficult-to-categorise corpora, probably by using the human ability to extract the **meaning** of documents from the document words. Many researchers have tried to use features that are closer to a more semantic representation of documents. Unfortunately, many of these attempts do not succeed in doing better than bag-of-words e.g. (Moschitti and Basili, 2004). A major problem seems to be that the more complex a feature type is, the less frequently it occurs. A complex feature's occurance rate in a training set will often not be significant in an unseen test set. A simple example is how a purely bigram representation of documents is expected to do worse than a representation of the same documents using unigrams, as we will demonstrate below in the Experiments section.

Attempting to use syntactic features can be appealing because the syntax of document sentences holds clues into how an author encodes the semantics into the sequence of document words. Syntax can be thought of as the middle ground between the lexical and semantic levels of representation. An apt quote on this subject is "... the aim of syntactical representation is to constitute a bridgehead to semantics"(Schneider, 1998). The syntax of a document is usually represented in a hyper-linear representation such as trees. Syntactic trees are rich in information where the nodes encode information about sentence terms and the links encode information about the relationships between terms.

It is tempting to construct complex features by extracting them from parse trees. However, there

is always the problem of feature sparsity. The right type of complex, syntactic feature may be closer to the semantic level and may seem to hold out the promise of improving performance on a TC task, yet more complex phrasal or syntactic features suffer from "inferior statistical qualities" (Lewis, 1992).

It is the aim of this research to try to use syntactic information in TC features in such a way that the problems of sparsity might be overcome. In this effort, two main lines of attack are used. First, we automatically generate a large number (possibly hundreds of thousands or more) of features out of dependency parse trees, in the hope that some of these many features will turn out to be highly discriminative TC features with a significant number of occurrences. Second, we use partial abstraction of these syntactic features to bring separate groups of features together, thus increasing the feature count and alleviating sparsity.

It should be noted that sparsity may be only one reason that complex features may not be as effective as simple features given a particular task. Other reasons might be that some types of complex features are simply noisy with respect to a learning task. Another possible reason is that some simple features subsume more complex features with respect to a task. These two reasons beyond sparsity will not be explored in this research.

The TC task that we do our experiments on is that of scam detection i.e. separating scam websites from non-scam websites in the financial domain. The dataset comes from the ScamSeek project (Patrick, 2006b).

2 Syntactic Features

This section first describes related work on syntactic features and than describes the type of syntactic feature developed for this research.

2.1 Related Work

A Word Sense Disambiguation (WSD)[1] system using syntactic features alone is described in (Lin, 2000). This system inspired the current research. This system also adhered to the principle of only using syntactic features **discovered automatically** in the corpus. In Lin's WSD system, one linked features were formed by starting at the target-word[2] and jumping to surrounding lexical nodes that could be reached from there. Two-link features were formed by jumping from the endpoint of one-link features to any non-target word nodes.

The precursor to this research, (Chanen and Patrick, 2004), details the use of ATSAS features for WSD. ATSAS features were based upon Lin's syntactic features but were designed to be more general. Lin's features were formed by expanding a feature one link at a time in a figurative line. The differences between Lin's syntactic features and those described in (Chanen and Patrick, 2004) are: 1) Lin's features are restricted to the linear (e.g. under his syntactic feature definition, a feature like Figure 1(e), where three links come together in one node, would not be possible); and 2) Lin's do not have abstraction.

A similar use of dependency parse subtrees is applied for producing state-of-the-art machine translation results in (Quirk et al., 2005), where the syntactic features are called *treelets*. After parsing a sentence in the source language, parallel treelets in the target language are then extracted to aid in the translation task. There is a very limited use of wildcards, only at the root of the tree.

For the experiments of this research, the syntactic features were extracted from dependency parse trees rather than from constituency parse trees. (Quirk et al., 2005) also used dependency parsers over constituency parsers citing the compactness of the sentence structure representation; for example, a verb is adjacent to its arguments in a dependency parse tree, whereas in a constituency parse tree the path from the verb to its arguments would first need to go up the tree before going down to the arguments.

Another WSD system using syntactic features is described in (Férnandez-Amorós, 2004). The system also uses parse subtrees as well as some wildcard abstraction for increasing recall, although his wildcards could only replace pronouns and content words. Much manual labour went into identifying the base syntactic features and the ways of abstracting the features. This research also uses

[1] See (Kilgarriff, 1998) for a description of the WSD task and the SENSEVAL competition.

[2] The word whose sense is being disambiguated

transformations on these sub-tree patterns in a further attempt to increase recall. This research did not achieve the same amount of automation in either identifying syntactic tree topologies or in generating wildcard features.

Not all syntactic features are directly related to the properties of a specific syntax tree. Some researchers have developed syntactic features that are aggregate, ratio, or statistical properties of all the sentences in a document or group of related documents. In one example of such syntactic features (Uzuner, 2005), the aim of the research is to differentiate between the works of different authors mainly by use of clever aggregate measures of parse tree properties. For instance, one set of features suggested in this research is the number of left-heavy, balanced, and right-heavy sentences that can be found in a particular author's work. By comparing averages and ratios of such features, powerful methods can be developed for differentiating between authors. These kinds of features have a different focus from ATSAS features, differentiating between authors. It might be interesting, though, to see if such aggregate features might be useful in a TC task like scam detection.

2.2 ATSAS Features

The type of syntactic feature that is developed here has been named the "All Topology, Semi-Abstract Syntactic" (ATSAS) feature type. This type of feature is extracted from a dependency parse tree. Dependency parse trees are preferred to constituency parse trees because they are more compact and arguably more directly related to document semantics (Schneider, 1998).

Each extracted feature is a subtree of a complete sentence parse tree. Theoretically, such features could consist of any number of links, but practically, a limit must be placed on the maximum number of links that are allowed to be in the extracted features. This is because problems with speed and memory limitations can result from the massive number of features that tree features have the potential to generate. With a maximum link level of only two, the experiments for this research generated over 10 million features. This massive number of features, though, was reduced by discarding features that occurred in less than three documents from a training corpus.

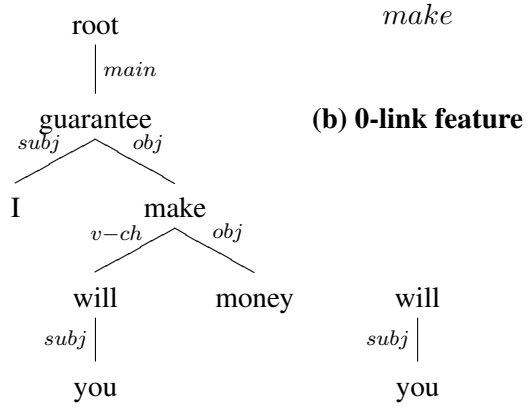
(Chanen and Patrick, 2004) succeeded in generalizing Lin's syntactic features to produce a greater variety of feature topologies, not just the linear features that Lin's system could extract. The current research had as a main goal of expanding the use of ATSAS features beyond their use in WSD to the TC task. In that main goal, this research seems the largely successful. However, the experiments in this research were not able to demonstrate the use of non-linear syntactic features such as the example seen in Figure 1(e) because of the memory limitations that occurred at feature extraction time. Since a maximum of two link features could be formed, the features extracted for this research is experiments were linear like those in Lin's system.

One of our important desideratum for syntactic features is that they are automatically extractable. Some syntactic features that were reviewed eariler in the Related Work section use manually-identified syntactic features. It is better, if possible, to eliminate human input in identifying the features both for greater efficiency and for the possibility that novel new syntactic features may be identified in places where human experts may not think to look or have time to look.

Figure 1 shows a small parse tree and several ATSAS features extracted from the complete parse tree with zero through three link example features.

Each feature link is labelled with a dependency link label e.g. *subj*, *obj*, *cc*. The links are joined by nodes that consist of both a lemma and part of speech (POS) tag. When the initial set of features is generated, all of the lemmas referred to literal sentence content. In a second phase of feature generation, all combinations of a feature's lemmas are abstracted to a "*". That is, each lemma in an ATSAS can be either literal or abstract. So, for a two-link, three-node feature, a total of seven abstract features can be generated from the one literal feature for a total of eight features. By such abstraction the hope is that some abstract features will join groups of documents in a way that is useful for the TC task. The nodes in Figure 1 do not show any abstraction and they also do not show the POS elements of the nodes.

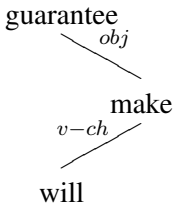
One of the advantages of syntactic features like



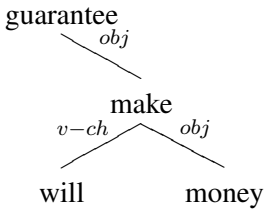
(a) Whole sentence DP tree

(b) 0-link feature

(c)1-link



(d) 2-link feature



(e) 3-link feature

Figure 1: **(a)** The complete dependency parse tree of the sentence "I guarantee you will make money." **(b-e)** These figures show examples of syntactic features with zero (just a node) through 3 links.

ATSAS features is that it is possible to increase recall over features like n-grams because certain syntactic features will allow for matching parts of documents by skipping over some words. For instance, the syntactic pattern in Figure 1(c) can match, in a test document, the phrase "you will" as well as "you definitely will" even though "definitely" is not part of the syntactic pattern.

3 Methods

3.1 Data Parsing

Before extracting the syntactic features, the data was parsed using the Connexor dependency parser (Järvinen and Tapanainen, 1997).

3.2 Classifier

The maximum entropy classifier was selected to run the TC experiments, partly because publications such as (Ratnaparkhi, 1998) demonstrate that the ME classifier turned in state-of-the-art or near state-of-the-art results for a wide range of NLP tasks.

An important aspect of the ME algorithm is that it can robustly deal with a large number of features, which is important in this research since hundreds of thousands of features are possible (see (Ratnaparkhi, 1998).)

The software package used to perform the ME experiments of this research is the Megam (MEGA Model Optimisation)[3] package, which uses an optimised version of limited memory BFGS (Daumé, 2004).

3.3 Training and Testing

Ten-fold cross validation was used for the training and testing evaluation process. All tests used the same set of folds.

For any given training fold, after all the given feature types were extracted from the train documents, all token types were removed from consideration if that token type was observed in less than three documents.

Classification accuracy was chosen as the performance metric in these experiments.

Hypothesis testing was used to determine if the TC performance between a pair of feature types was significantly different. The paired t-test was the specific type of hypothesis test used. A standard significant cutoff value of $\alpha = 0.05$ was utilised. The null hypothesis is that the 10-fold cross validation determined accuracies of any two feature tests do not differ significantly from each other.

4 Experiments

This section describes the series of experiments performed to determine the usefulness of the ATSAS family of syntactic features.

[3]This package is described and can be downloaded at the website: `http://www.cs.utah.edu/~hal/megam/`

Feature or feature combination	
Title	**Description**
SAA	Only the ATSAS features that have only abstract lemmas. "AA" stands for **all abstract**. Every lemma in this feature set is abstacted to a "*".
2g	The corpus is represented only in terms of bigrams.
1g	Representation only in unigrams (bag-of-words). The baseline feature.
1g2g	A combination of unigrams and bigrams
1gS-A	A combination of ATSAS and unigrams features except for any syntactic features that contain any abstract lemmas. Here "-A" stands for **minus abstract**.
S-A	All ATSAS features except for any syntactic features that containing any abstract lemmas. In other words, this is the set of literal syntactic features.
S	All ATSAS features Including abstract and non-abstract features.
S-NA	All ATSAS features except for any syntactic features that do not have abstract lemmas at all. Here "-NA" stands for **minus non-abstract** where non-abstract means every feature in which no lemmas are a "*". In other words, this is the S set minus the totally literal features.
1gS-NA	All ATSAS and unigram features except for any syntactic features that do not have any abstract lemmas at all.
1gS	A combination of unigram features and all ATSAS features

Table 1: Descriptions of the feature types used in this paper's experiments.

4.1 Data

A subset of the original ScamSeek (Patrick, 2006a) project's dataset was used in this research. This subset consists of 2130 documents. This means that during 10-fold cross validation, each training fold consists of 1917 documents and each testing fold consists of 213 test documents. Out of the total corpus, 27% of the documents are scams and 73% are non-scams.

4.2 Features

A variety of different feature types are compared in the below experiments. Table 1 gives the abbreviated name and feature description for each feature type (or a combination of feature types)associated with one of the experiments.

Table 2 shows the number of token types in the **S** and **S-A** feature types, broken down by the number of links in each feature type. This table illustrates how the abstraction of some elements of literal syntactic features helps multiply the number of statistically significant ATSAS features. At first glance, the feature counts for the **S-A** (literal or non-abstract) feature type seem to be counterintuitive. One might expect an explosion of features as more links are added. However, since features observed in less than three documents in the training set are removed, this causes the number of significant 2-link **S-A** features to be less than the number of 1-link **S-A** features. When one or more of the lemmas in an ATSAS feature are abstracted, two or more literal features that were distinct may map to the same abstract feature. Literal features that are not significant on their own, may produce a significant feature through this abstraction process. The count of an abstract feature is the sum of the counts of the literal features that map to it. This is the reason for the much higher number of statistically viable features in the 2-link **S** feature type, where partial or full abstraction of lemma elements is allowed.

Producing abstract features can be quite memory intensive because all literal features found in any document must be stored even if a feature occurs in just a single document. This is necessary since only after all documents have been processed can the abstraction process join low count literal features. The full **S** feature type includes more than 10 million features after all abstract features have been added to the literal features. This exponential explosion of features is the reason for not being able to perform any TC experiments using 3-link features. See the Future Work section for ideas for including 3-link features.

	Non-abstract (S-A)		**Abs. and Non-abs. (S)**	
	count	**%**	**count**	**%**
0-links	7873	10.6%	7873	2.2%
1-link	34466	46.4%	70610	19.8%
2-link	31895	43.0%	277301	77.9%
Total	**74234**		**355787**	

Table 2: Feature counts in both the **S-A** experimental set (which includes only features with no abstraction) and the full set of ATSAS features in **S** (which includes both abstract and non-abstract syntactic features) as the number of links are inceased. A 0-link feature is just a lemma/POS node.

5 Results

The results of the experiments are measured in terms of the classification accuracies of each type of feature, as well as the pairwise t-test results for each pair of feature experiments. If the t-test results for a given pair is at or below the $\alpha = 0.05$ level, then it is assumed that results for such a pair of experiments will differ significantly. Table 3 displays the results. This table presents the number of features in each feature set and also gives the performance of each feature type in terms of classification accuracy. The table also gives the paired t-test score for each pair of experimental results. The t-test pairwise significance results are displayed in the half-triangle part of the table where significant results are shown in bold face.

6 Analysis

The baseline for these tests is the Results using the bag-of-words feature, **1g**, which shows an accuracy of 86.9%. In order to judge the ATSAS feature type (or some derivative thereof) as a successful TC feature, we need to show a significant t-test score between one of the ATSAS feature types and the bag-of-words features.

Experiment one used ATSAS features that eliminated all features that had literal lemmas (i.e. not abstract). There are only 4,291 instances of this subtype of ATSAS features. Not surprisingly, this feature type did the worst with an accuracy of 82%, which was significantly worse than every other type of feature tested. One hypothesis as to why this feature type did not perform well is because there are too few features to perform good classification alone. Another possibility is that the literal lemmas in many of the ATSAS features are important and help the best of these features two separate the classes. In other words, having some literal content may be helpful for forming good features for separating classes.

Another non-syntactic feature that was experimented with, was a pure bigram (2g) feature representation. See experiment #2 in Table 3 for the results. One can observe that the bigram representation scores significantly below the unigrams representation of the corpus in the TC task. Even though bigrams often carry more meaning than a

Experiment		**Corpus features**	**Score**	SAA	2g	1g	1g2g	1gS-A	S-A	S	S-NA	1gS-NA
Title	#			1	2	3	4	5	6	7	8	9
SAA	1	4,291	82.0									
2g	2	141,346	85.5	**0.0024**								
1g	3	20,402	86.9	**0.0001**	**0.0372**							
1g2g	4	161,748	87.0	**0.0001**	**0.0341**	0.8456						
1gS-A	5	94,576	88.6	**0.0013**	**0.0239**	0.1830	0.1497					
S-A	6	74,234	88.8	**0.0000**	**0.0004**	**0.0003**	**0.0048**	0.8845				
S	7	355,787	89.1	**0.0004**	**0.0078**	0.0709	0.0689	0.5797	0.8501			
S-NA	8	281,553	89.2	**0.0000**	**0.0008**	**0.0171**	**0.0051**	0.6334	0.6249	0.9341		
1gS-NA	9	301,895	89.3	**0.0002**	**0.0043**	**0.0419**	**0.0365**	0.3859	0.6899	0.1382	0.8921	
1gS	10	376,129	89.4	**0.0002**	**0.0020**	**0.0288**	**0.0240**	0.2695	0.6135	0.1530	0.8138	0.5911

Table 3: Experimental results with significance of paired t-tests

single word, because of sparsity problems with word combinations, bigrams usually fall short of unigrams. A combination of unigram and bigram feature types scores slightly higher than unigrams alone, but not significantly so. The bigrams feature is one of the simplest complex features. This experiment demonstrates how even simpler complex features suffer from sparsity.

For the **S** feature type (FT) (Feature type 7 in Table 3), although the accuracy is 2.2% higher then the bag-of-words features, the t-test is not significant (although it is close to being significant). However, if the **S** feature type is combined with the **1g** feature type then the results are a full 2.5% better in terms of accuracy and are significant (See row 10 in Table 3.)

Another positive result for the ATSAS family of syntactic features can be observed for FT **S-A** (#6 in the results table). This result is positive in that it is the most significant t-test score with the bag-of-words feature when compared with all the other feature types and their t-test score with the **1g** FT. This is surprising for at least two reasons: 1) one of the initial assumptions for including partial abstraction as a property of ATSAS features was that the abstraction would alleviate problems with sparsity and the syntactic features would not do well without it; and 2) the **S-A** FT has only 74,234 token types from the whole corpus compared to 355,787 token types for the **S** FT. One possibility is that some feature selection would be beneficial even though publications such as (Ratnaparkhi, 1998) claim that the maximum entropy classification method does not suffer much from the curse of dimensionality. Another thought is that, because tens of thousands of non-abstract features are generated as the sentence parse trees of documents are traversed across the entire training set, many good TC features are discovered even though they may not take advantage of the power of abstraction.

The results from experiment #6 suggest that abstraction is not necessarily needed for ATSAS features to realise gains over simple bag-of-words. However, we do have some evidence that abstraction in ATSAS features does help them achieve even better performance. Experiments 8 and 9 involving the feature types **S-NA** and **1gS-NA** respectively, and both have 10-fold cross validation accuracies that allow the rejection of the null hypothesis. Therefore, it can be concluded that these feature types are significantly better than bag-of-words alone. These two feature types do better then experiment #6 involving the **S-A** feature type, by an accuracy difference of 0.4% and 0.5% respectively. Unfortunately, the difference between the three feature types **S-A**, **S-NA**, and **1gS-NA** is not significant so further experimentation would be required to settle the question of whether abstraction is needed more satisfactorily.

7 Conclusion

More complex feature types may seem desirable because they may allow for features that are closer to the semantic level and thus possibly better for difficult TC tasks. However, experience has often shown that more complex features do not usually live up to their promise because of sparsity problems.

In this research we have proposed a family of syntactic features, the All-Topology, Semi-Abstract Syntactic feature family. We have experimentally shown that several variations of ATSAS feature types have significantly out-performed bag-of-words features. Specifically, the set of subtrees from a dependency parse tree with zero through two links when combined with bag-of-words gave the best performance, significantly better than bag-of-words alone. Surprisingly, variations on ATSAS that either eliminated abstract features or eliminated totally literal features also did significantly better than bag-of-words alone.

8 Future Work

A logical next direction would be to expand the ATSAS by pushing the maximum number of links per feature from two to three. With only two links, the syntactic features are linear just as in (Lin, 2000). With three or more links, the kinds of tree topologies and the different combinations of dependency link types would increase exponentially along with greater possibilities of finding an interesting syntactic feature for separating classes.

However, memory limitations prevented using three links as the maximum. This problem could

be addressed in several ways. As experiment #6 involving the **S-A** variation suggested, ATSAS features do not necessarily need abstraction to perform well. So a logical step is to simply see if there is enough memory if three link ATSAS are generated without any abstraction.

Another direction is to only generate ATSAS features for TC that involve certain lemmas. For instance, the top N ($N < 500$) words by information gain or some other measure could be used. If an ATSAS is generated but does not involve one of the selected lemmas, then that feature would be discarded. Another way to determine the top terms from which the syntactic features would be built would be to take the top terms by probability from each topic from a Latent Dirichlet Allocation (LDA) (Blei, 2004) model. The topical terms might differ from the information-gain terms in interesting ways.

It would also be desirable to see if similar, encouraging results can be derived from using the same type of features on public-domain and widely used benchmark datasets such as Reuters.

Finally, using a better feature selection strategy might be advantageous. (Ratnaparkhi, 1998) presents evidence that the maximum entropy learning algorithm can handle a large number of features with little degradation in performance. However, that evidence was based on a single dataset. It is possible that the ScamSeek dataset might benefit from a more sophisticated feature selection strategy. Feature selection might also help separate the truly useful ATSAS features from the background noise.

Acknowledgements

We would like to thank the Capital Markets CRC and the University of Sydney for financial support and everyone in the Sydney Language Technology Research Group for their support. Also, thanks to Sue Zeckendorf for editing help .

References

[Blei2004] David Blei. 2004. *Probabilistic models of text and images*. Ph.D. thesis, U.C. Berkeley.

[Chanen and Patrick2004] Ari Chanen and Jon Patrick. 2004. Complex, corpus-driven, syntactic features for word sense disambiguation. In *Proceedings of the Australasian Language Technology Workshop 2004*, pages 1–8, Sydney, Australia, December.

[Daumé2004] Hal Daumé. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper and implementation available at `http://www.cs.utah.edu/~hal/megam/`, August.

[Férnandez-Amorós2004] David Férnandez-Amorós. 2004. Wsd based on mutual information and syntactic patterns. In Rada Mihalcea and Phil Edmonds, editors, *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 117–120, Barcelona, Spain, July. Association for Computational Linguistics.

[Järvinen and Tapanainen1997] Timo Järvinen and Pasi Tapanainen. 1997. A dependency parser for English. Technical Report TR-1, Department of General Linguistics, University of Helsinki, Finland.

[Kilgarriff1998] Adam Kilgarriff. 1998. SENSEVAL: An exercise in evaluating word sense disambiguation programs. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 581–588, Granada, Spain.

[Lewis1992] David D. Lewis. 1992. An evaluation of phrasal and clustered representations on a text categorization task. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 37–50. ACM Press.

[Lin2000] Dekang Lin. 2000. Word sense disambiguation with a similarity-smoothed case library.

[Moschitti and Basili2004] Alessandro Moschitti and Roberto Basili. 2004. *Complex Linguistic Features for Text Classification: A Comprehensive Study*. Springer Verlag.

[Patrick2006a] Jon Patrick. 2006a. The scamseek project - text mining for financial scams on the internet. In *Selected Papers from AusDM*, pages 295–302.

[Patrick2006b] Jon Patrick. 2006b. The scamseek project: Text mining for financial scams on the internet. In Graham J. Williams and Simeon J. Simoff, editors, *Selected Papers from AusDM*, volume 3755 of *Lecture Notes in Computer Science*, pages 295–302. Springer.

[Quirk et al.2005] Christopher Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *ACL*. The Association for Computer Linguistics.

[Ratnaparkhi1998] A. Ratnaparkhi. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, University of Pennsylvania.

[Schneider1998] Gerold Schneider. 1998. A linguistic comparison of constituency, dependency and link grammar. Technical Report 1.1, Institut für Informatik der Universität Zürich.

[Uzuner2005] Özlem Uzuner. 2005. *Identifying Expression Fingerprints Using Linguistic Information*. Ph.D. thesis, MIT.