**The Report Committee for Joshua Blake Brewster**
**Certifies that this is the approved version of the following report:**

# Dependency Based CCG Derivation and Application

**APPROVED BY**
**SUPERVISING COMMITTEE:**

**Supervisor:**

Jason Baldridge

Katrin Erk

# Dependency Based CCG Derivation and Application

by

**Joshua Blake Brewster, B.A.**

**Report**

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

**Master of Arts**

**The University of Texas at Austin**

**December 2010**

## Abstract

## Dependency Based CCG Derivation and Application

Joshua Blake Brewster, M.A.

The University of Texas at Austin, 2010

Supervisor: Jason Baldridge

This paper presents and evaluates an algorithm to translate a dependency treebank into a Combinatory Categorial Grammar (CCG) lexicon. The dependency relations between a head and a child in a dependency tree are exploited to determine how CCG categories should be derived by making a functional distinction between adjunct and argument relations. Derivations for an English (CoNLL08 shared task treebank) and for an Italian (Turin University Treebank) dependency treebank are performed, each requiring a number of preprocessing steps.

In order to determine the adequacy of the lexicons, dubbed DepEngCCG and DepItCCG, they are compared via two methods to preexisting CCG lexicons derived from similar or equivalent sources (CCGbank and TutCCG). First, a number of metrics are used to compare the state of the lexicon, including category complexity and category growth. Second, to measures the potential applicability of the lexicons in NLP tasks, the derived English CCG lexicon and CCGbank are compared in a sentiment analysis task. While the numeric measurements show promising results for the quality of the lexicons, the sentiment analysis task fails to generate a usable comparison.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1: Introduction

The ability to understand a document and extract meaningful data from text requires the assignment of an underlying structure. Current methods use parsing models trained on various grammar formalisms to attempt to tease apart the relationships between words, determine constituency structure, and assign head/argument relations.

The creation of parsers usually requires the use of a large annotated corpus; the Penn Treebank (Marcus, Santorini and Marcinkiewicz 1993; Marcus et al. 1994) has become the de facto corpus for training and testing parsing models for English. Early statistical methods of parsing on the Penn Treebank have met some success by use of part-of-speech (POS) tags, yet these methods often times ignored the intrinsic syntactic information offered the tags; for example, a word with the POS of *JJ* (adjective) will usually modified a noun and therefore can infer the existence of one in the sentence. Movement to capture the inherent syntactic structure of words resulted in a number of formalisms that allow the underlying argument structure to be recovered; among these are Lexical-Functional-Grammar (LFG) (Kaplan and Bresnan 1982), Tree-Adjoining-Grammar (TAG) (Joshi and Schabes 1992), Head-driven Phrase-Structure Grammar (HPSG) (Pollard and Sag 1994) and Combinatory Categorial Grammar (CCG) (Steedman 1996, 2000; Steedman and Baldridge (to appear)). As with any manual annotation task, creating formalism-specific lexicons by hand is both labor and time intensive. A number of approaches have been introduced to extract lexicons automatically from the Penn Treebank into various formalisms: LFG ( Cahill et al. 2002; O'Donovan et al. 2005; Shen and Joshi 2005), HPSG (Miyao, Ninomiya and Tsujii 2004), TAG (Xia 1999, 2001; Xia, Palmer and Joshi 2000; Chen and Vijay-Shanker 2000; Chen, Bangalore and Vijay-

Shanker 2006) and CCG (Hockenmaier and Steedman 2003). In this paper, CCG extractions will be the focus.

Combinatory Categorial Grammar is a lexicalized formalism that excels in dealing with long and short-range dependencies. Due to the ability to do supertagging as “almost parsing” (Bangalore and Joshi 1999), rapid parsing speeds have been achieved (Clark and Curran 2004), making CCG lexicons particularly attractive for further parsing applications, as well as other tasks that can take advantage of CCG supertags.[1] In this paper, I present a method to generate a CCG lexicon from a dependency treebank. I will revisit and expand upon the implementation in Ponvert (2008), drawing motivation from Hockenmaier & Steedman (2003) and Cakici (2005) to automatically induce and evaluate CCG lexicons from two dependency treebanks: the English dependency treebank extracted from the Penn Treebank and other sources that was used for the CoNLL-2008 shared task (Surdeanu et al. 2008) and the Italian Turin University Treebank (TUT). These two treebanks have been selected as there are preexisting CCG lexicons from the same or similar source that can be used as comparison.

Dependency trees encode both the linear ordering and dependencies that are needed to induce a CCG lexicon. The goal of this paper is to determine the effectiveness of dependency relations and dependency links in generating CCG categories without assuming or creating any intermediate structure in the tree.

## 1.1 DEPENDENCY GRAMMARS

Dependency grammars give hierarchical structure between headwords and dependents via dependency relationships. These relationships are usually based on some notion of semantic or syntactic roles. A dependency tree is formed as an acyclic graph

[1] For the purposes of this paper, I will use *supertag* and *categories* interchangeably. For origins of supertags, see Bangalore and Joshi (1999).

where nodes can only have one parent; dependency relations connect parent nodes to daughter leaf nodes: see Figure 1. In this example, the word *jumped* is the root of the sentence as there is no node that has *jumped* as its dependent. The node *jumped* has two dependents, *fox* and *over*, that are connected via the relations *NP-SUBJ* and *PP* respectively. These nodes, in turn, have their own dependents. This pattern continues until the leaf nodes of the tree are reached. While a node can have more than one dependent, only one parent node is permitted. See Nivre (2005) for a discussion on theoretical dependency grammars and analyses.

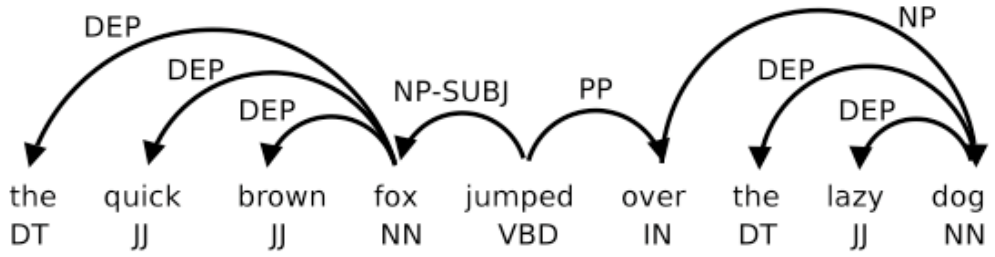


Figure 1: Sample Dependency Relations within Sentence

Dependency parsing involves an algorithm to predict dependencies from a unique headword to a dependent word, without having the benefit of syntactic structure. Two of the more successful dependency parsers are MaltParser (Nivre and Hall, 2007) and MSTParser (R. McDonald, 2005). MaltParser employs a left-to-right shift-reduce algorithm to determine dependency links, whereas MSTParser measures the strength of a link between words in conjunction with a search algorithm. With the induction of supertags from dependency relations, a much more rich feature structure would replace the simple POS tag, giving a better indication of syntactic structure and argument relation/placement. The supertag could help predict the kind and amount of dependent relations a given word may have.

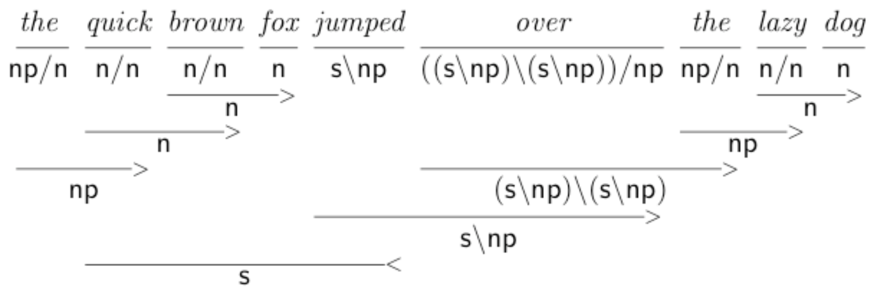
## 1.2 Combinatory Categorial Grammars

Combinatory Category Grammar (CCG) has its roots in the classical lexicalist theory of Categorial Grammar (Bar-Hillel 1953; Ajdukiewicz 1935; Lambek 1958). It is a lexical grammar formalism in which the categories of words combine together by means of a small number of defined rules. The combinatory nature of the categories removes the necessity of movement and deletion rules found in other syntactic theories. At the same time, the categories encode syntactic information following the Principle of Compositionality of Frege (1923), in that syntax and interpretation are directly related.

These categories are derived of a series of atomic categories as well as the functional slash operators \ and /. A category is given as an atomic category alone (e.g np, n, s ) or in the functional form of X "slash" Y, where the word looks in the direction of the slash for something of the category Y and, if found, returns the category X. Naturally, a category of the form X/Y expects a word with category Y to the right; a category of the form X\Y expects a word with category Y to the left. The combinatory rules are as follows:

1. Application:
    a. $A/B\ B \Rightarrow > A$
    b. $B\ A\backslash B \Rightarrow < A$
2. Composition:
    a. $A/B\ B/C \Rightarrow >_B A/C$
    b. $B\backslash C\ A\backslash B \Rightarrow <_B A\backslash C$
3. Cross Composition:
    a. $A/B\ B\backslash C \Rightarrow >_{Bx} A\backslash C$
    b. $B/C\ A\backslash B \Rightarrow <_{Bx} A/C$

Other rules, such as type-raising and substitution will not be discussed here. As an example, the sentence from Figure 1 is shown below with CCG categories assigned to each word. A derivation is given, showing that the sentence returns an s when all the categories are successfully combined. For more details and extensions, see Steedman, (2000).



the quick brown fox jumped over the lazy dog
np/n n/n n/n n s\np ((s\np)\(s\np))/np np/n n/n n
n
n
np
n
np
(s\np)\(s\np)
s\np
s

Figure 2: CCG Derivation of a Sample Sentence

# Chapter 2: Previous Work

## 2.1 CCGbank

Hockenmaier & Steedman (2003) converted phrase structure trees from the Penn Treebank and derived CCG categories, using both the syntactic ordering and hierarchy inherent in the sentence trees. Due to the fact that the Penn Treebank encodes trace and null elements within the trees, this treebank was ideal to allow resolution of long-range dependencies in relative clauses and extraction. Two wide-coverage parsers have been created using CCGbank that are able to recover the long-range dependencies quickly and efficiently (Clark, Hockenmaier and Steedman 2002; Hockenmaier and Steedman 2002; Clark and Curran 2004, 2007).

At a high level, the derivational algorithm used to create the CCGbank translates constituent-based sentence trees into binary-branching trees that model the combinatory aspect of CCG. Categories are then derived for each word and word-to-word dependencies were assigned in the end. While many details of Hockenmaier's algorithm are similar to the algorithm presented in this article, the fundamental difference is that the presented algorithm requires as input a dependency tree; thus, it is dependency relations and hierarchy rather than constituent structure that will drive category creation.

## 2.2 TutCCG

An Italian CCGbank was similarly created using the Turin University Treebank (TUT) by Bos, Bosco & Mazzei (2009). For this derivation, the TUT dependency treebank (Bosco 2003) is converted to a constituency treebank, then follows the method given by Hockenmaier for the English CCGbank.

This method resulted in an intermediary treebank called ConsTUT, for Constituent Turin University Treebank. Each terminal node in ConsTUT can be mapped back to a node in the original TUT format; the resulting non-terminal structure represents the projections of the terminal node in XBar theory. In this format, distinctions between arguments and modifiers are determined at structural level. As with the English CCGbank, each node is classified as a head, complement or adjunct and the derivation process uses these distinctions to determine how to assign categories.

## 2.3 TURKISH CCG

Cakici (2005) further explores the derivation process, deriving a CCG lexicon from a Turkish dependency treebank by assigning parts of speech corresponding to the relationship to the parent node. The process recursively progresss through each word of the sentence, using the dependency relation to derive an atomic category. For example, if the dependency relation from the head to the dependent is OBJECT, then the dependent is labeled as NP. Cakici mentions the possibility that dependency relations are too few to make correct (sub)category choices automatically, in that words with vastly different functions were bound under the same dependency relation. This potential set-back may be remedied by use of a larger treebank, as the Turkish treebank used by Cakici was an order of magnitude smaller than the Penn Treebank used by Hockenmaier, or perhaps by using a more explicit relation set.

Cakici's approach induces a lexicon that manages to overcome obstacles of pro-drop and subordination/relativization; results after the publishing of her paper showed that parsing in Turkish did in fact improve.

## 2.4 BENEFIT OF THIS APPROACH

There are a few inherent characteristics that suggest that an approach employing dependency trees rather than constituent structures may offer a better lexicon. As CCG is a binary branching grammar, a conversion process is needed to convert Penn Treebank sentences into pseudo-binary branching trees. The automated conversion process allows room for error. Using an inside-out combinatory aspect in conjunction with the one-to-one relationship between dependents and heads, one can traverse a dependency tree in a simulated binary fashion without creating an intermediate structure. Koller and Kuhlman (2009) show this conversion is not one-way, creating a process to generate a dependency tree by taking advantage of CCG sentence derivations' natural dependency structure.

The constituent Penn Treebank used to create the CCGbank has ambiguity at NP structure. As example, the NP *the hot pink roof* is ambiguous as it is unclear if this NP refers to a pink roof that is hot in temperature, or a roof that is the color of hot pink. As no time was available for manual re-annotation, these NPs were converted into right branching structure which were sometimes accurate and sometimes not (Hockenmaier, 2003). However, in Honnibal et al. (2010), the ambiguous noun phrases were corrected in the CCGbank using the manual annotation from Vadas and Curran (2007). This adaptation corrected 1.95% of the dependencies in CCGbank (Vadas and Curran 2008). For dependency treebanks, this flat structure is not an issue, thus NPs and their respective modifier structure will be accurate every time.

The use of dependency relations may offer better results in generating a lexicon, as the relation shows the association between parent and daughter more explicitly than a constituent structure approach. Furthermore, having various sets of dependency relations allows greater potential in modification/customization of how the derivational approach handles each relation.

This approach differs from Cakici's approach in the manner that atomic categories are assigned. Cakici used the dependency relations themselves to determine the atomic categories of the dependent nodes. In contrast, this approach uses the part of speech tags assigned to the dependent nodes; the dependency relations will determine the method in which atomic categories are combined to create factor categories.

# Chapter 3: Extracting Categories from Dependency Trees

A dependency tree can easily be converted to a CCG derivation. Given Figure 1, we can redraw this graph in a more tree-like arrangement, given in Figure 3. In this format, it is impossible to determine the original sentence order; however, many treebanks supply indices so that the linear sentential order can be preserved.

A broad description of the induction principle will be given below, with a more in depth algorithm given at the end of the chapter. As a simple example, in this sentence the root word *jumped* has two children: *fox* and *over*. Between the parent *fox* and the dependent *fox*, there is a NP-SUBJ relationship. As this is an argument relation, *jumped* can be viewed as something that is looking to the left for an NP-SUBJ argument: using POS tags, it is a VBD that is looking to the left for an NN. We can thus assign the category in (4a). Between the parent *jumped* and the descendent *over*, there is a PP (Prepositional Phrase) relation. Since PP is an adjunct relation to the root, it will not be included in the lexical category for *jumped*. As another example, the word *over* is an adjunct or modifier of the root *jumped*; therefore *over* is something that is looking to its right for the argument prepositional-object and to the left for a verb to modify. Therefore the category of *jumped* is used as a subcategory in the category for *over* (4b). The same process can be performed on each word in the tree.

Once all words are given a category, the sentence derivation given in Figure 4 is possible..

4. Sample Derived Categories
    a. jumped:= VBD\NN
    b. over:= ((VBD\NN)\(VBD\NN))/NN

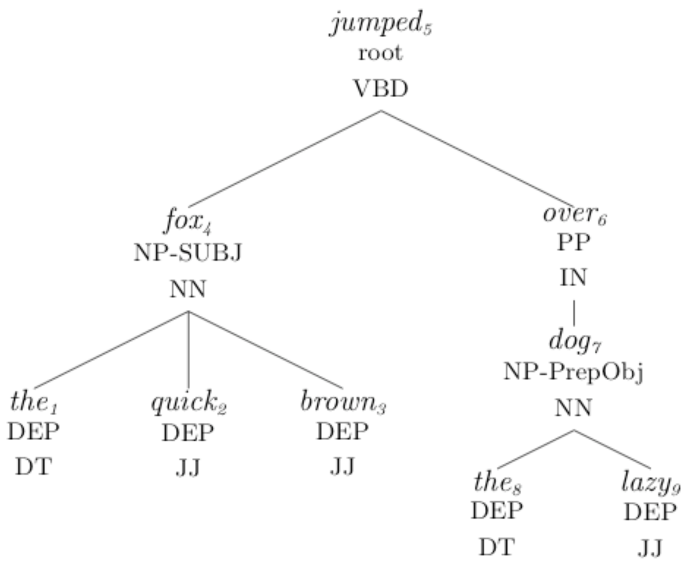


Figure 3: Dependency Tree Structure of a Sample Sentence

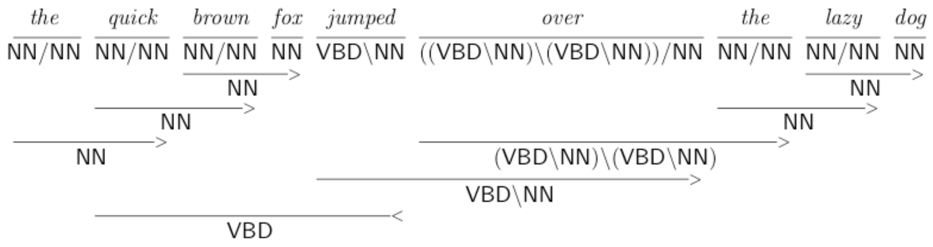


Figure 4: CCG Derivation Using Dependency Derived Categories

## 3.1 Current Data

The current work and analysis is being performed on two dependency treebanks. The first is supplied by the CoNLL-2008 shared task. This treebank was created from the Penn Treebank, BBNs named entity corpus, PropBank and NomBank (Surdeanu et al 2008). Each input corpus was translated from the constituent based formalism of the Penn Treebank to a dependency formalism (Johansson and Nugues 2007). There are a total of around 40,000 sentences.

The second treebank is the Italian TUT dependency treebank. It contains only around 2,400 sentences. This treebank comes in two different formats: the native TUT format and the adapted CoNLL format. This facilitates creating a single base algorithm for the derivation.

One significant distinction from other treebanks, namely the Penn Treebank, is that these treebanks do not encode trace or null elements that arise in movement scenarios such as wh-questions and relative clauses. This stymies the ability to easily resolve long-range dependencies and extraction via dependency structure and relations alone.

## 3.2 Derivation Algorithm

The algorithm traverses the dependency trees, creating categories as it passes from head to dependent much like was done above for the lexical entries in (7). In creating the CCG lexicon, the primary concerns are 1) whether a word is an argument or an adjunct to its parent and 2) how to treat said adjuncts and arguments during the conversion process. In very loose terms, an argument is a word that fulfills a certain pivotal syntactic-semantic role, such as SUBJECT, OBJECT or BENEFICIARY. Adjuncts play a slightly different role; their presence or lack thereof does not determine "grammaticality," but instead adds further information into the sentence; for example adjectives and adverbs. Ponvert (2008) determines if a dependency link is an argument relation or an adjunct relation in theCoNLL08 shared task treebank. For simplicity's sake, the previously derived sets of adjunct and argument links will be used. Dependency relations for the CoNLL08 treebank and the TUT treebank are given in Appendices A and B, respectively.

## 3.3 Basic Algorithm

The algorithm is as follows:

- Categorize parent-daughter relations as Adjunct or Argument.
- Assign each node's POS tag as its category.
- Calculate category of node using dependencies and their relations.

### *3.3.1 Categorize Argument/Adjunct Relations*

First, the relationship of each node to its parent node is determined. Quite simply, for a given node, if the relationship to its parent is in the predefined set of Arguments, then it is assigned an argument role. Conversely (and just as simply), if the relationship to its parent is in the predefined set of Adjuncts, then it is assigned an adjunct role. This hard-coded approach seems to work well.

### *3.3.2 Assign Each Node's POS Tag as Category*

This step is a brute force and computationally cheap way to ensures that argument nodes without their own argument daughters will have a category. As an example, see the words *fox* and *dog* in Figure 3.

### *3.3.3 Calculate Node Categories*

The category of each node is assigned. The category assignment can be viewed as a function of adjunct/argument relations, parenthood and POS tags. For each node that has an adjunct daughter, calculate the current element's category before calculating any of the dependent's categories. This will ensure that no adjuncts are induced using a parent node that has not yet been fully defined. Category assignment is accomplished using the following heuristic.

Starting at the root element and working down the tree:

- Adjunct: Given the current word $w_1$ with category $c_1$ and its dependent $w_2$, if R($w_1$, $w_2$) $\in$ AdjunctRoles, then $w_2$'s category is $c_1\backslash c_1$ or $c_1/c_1$
- Argument: Given the current word $w_1$ with category $c_1$ and its dependent $w_2$ with category $c_2$, if R($w_1$, $w_2$) $\in$ ArgumentRoles, then $w_1$'s category is $c_1\backslash c_2$ or $c_1/c_2$.

If $w_1$ has multiple argument relations that precede it, first calculate the left-most/first argument as determined by linear order, then each proceeding argument until $w_1$ is reached in linear order of the sentence. If $w_1$ has multiple argument relations that follow it, calculate first the right-most/last argument in linear order, the each preceding argument until $w_1$ is reached. If there are arguments that both precede and follow $w_1$, first calculate using those that preceded, then those that follow. This ordering will ensure the same category structure used in CCGbank and TutCCG., and will be quite necessary once logical forms can be involved in the derivational process.

## 3.4 Applying Compositional Rules to the Derivation

This simple algorithm does not extract an optimal lexicon; for example the algorithm assumes that adjuncts relations can only modify their heads by using the application rules of CCG. Thus, for the example adverb *quickly*, it would require two different categories to be able to modify the intransitive and transitive verbs in Figure 5 and Figure 6.

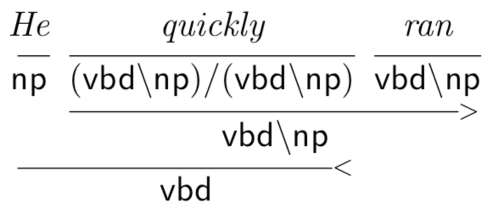


Figure 5: Example of the adverb ‘quickly’ in an intransitive sentence

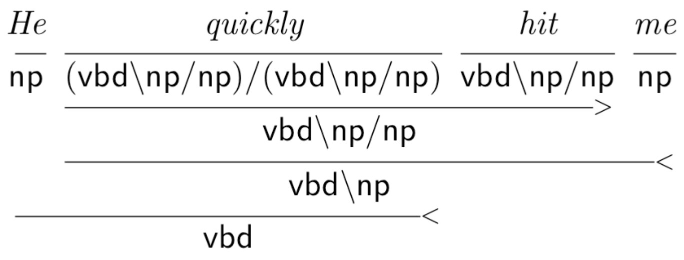


Figure 6: Example of the adverb ‘quickly’ in a transitive sentence

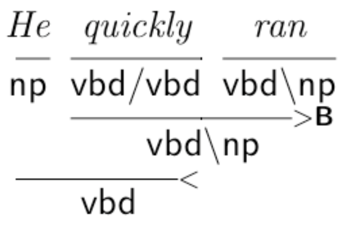


Figure 7: The adverb ‘quickly’ in a intransitive sentence with composition rules

Hockenmaier & Steedman (2003) found that allowing/assuming the CCG composition rules in the derivation process greatly reduced the number of categories in the lexicon while working on CCG Bank. This would allow a single category for the word *quickly*, for example, to modify verbs in general, as in Figure 7 and Figure 8.

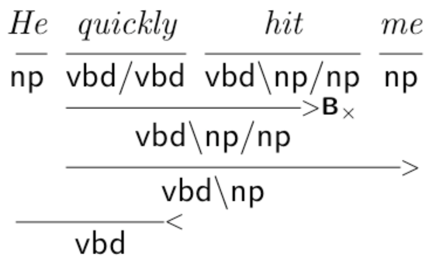


Figure 8: The word ‘quickly’ in a transitive sentence with composition rules

The algorithm needs to be quite selective when deriving categories assuming the composition rules, otherwise words might be able to combine in ways that would not be desired: an adverb combining with a noun, for example. The rule, therefore, is that composition rules can only be assumed when the following conditions are met:

5. Conditions for Assuming Composition
    a. The current word has an adjunct relation with its head.
    b. The current word modifies an argument or the root of a sentence.

To derive such a category: if the current word modifies an argument/root, then the category root is the modified argument's part of speech tag + "slash" + modified arguments part of speech tag. If the current word has an argument dependent, calculate the remainder of the category; otherwise, the category is complete.

As an example, take the word *over* from the sentence found in Figure 1.[2] *Over* meets the conditions of (5); its category root is therefore *jump*'s POS tag + \ + *jump*'s POS tag, thus rendering VBD\VBD. It also has an argument relation with *dog*, thus it looks to the right for an NN. The final category would then be (VBD\VBD)/NN. Compared to the original category of ((VBD\NN)\(VBD\NN))/NN, the new category is much simpler and concise, fulfilling the intuition that the preposition *over* tends to modify verbs in general, as opposed to strictly verbs of the intransitive or transitive variety.

[2] The quick brown fox jumped over the lazy dog.

## 3.5 Reducing Categories

There is an inherent issue of redundancy across categories, due to the simple existence of tense and plurality. For the basic intransitive verb like *sleep*, the following categories will be potentially derived: VBG\NN, VBZ\NN and VBD\NN. One simple and easy method to reduce a number of excess categories is to devise a mapping that collapses similar tags to a single tag group. For example:

6. VB = { VDZ,VBD, ... }
7. NN = { NNS, NN, ... }

In theory, the agreement data lost due to this step should not affect performance in statistical parsing models, as sequences that disagree in tense/plurality/gender will be inhibited due to their low probability.

## 3.6 Preprocessing Steps

For each input corpus, a number of individualized preprocessing steps were required.[3] Ideally this would not be necessary, however annotation conventions and inconsistencies argued otherwise.

### *3.6.1 CoNLL08 - Handling Coordination*

The dependency treebank encodes coordination by use of two dependency relations; coordinators are assigned the dependency relation COORD, while the conjuncts are given the category CONJ. For simple constructions like *Mary kissed John and Bill*, the algorithm is quite sufficient. However, the addition of more elements in the coordination required a number of preprocessing steps. Extending Mary's dubious behavior, for the elements in *Mary kissed Sam, John and Bill* that are not directly adjacent to the coordinator *and*, dependencies for the conjuncts needed to be reassigned

[3] All changes were verified with CCGbank and TutCCG lexicons.

and the commas needed to be reinterpreted as coordinating elements. Not only did this step prevent incorrect categories,[4] but it also helped model the underlying syntax of the constructions.

### *3.6.2 CoNLL08 – Money Issues*

Monetary constructions in the treebank such as "$ 100" were annotated with the dollar sign as the head and *6* and *billion* as dependents. This was done to maintain the same dependencies in the two trees in Figure 9. In each tree, the lemma for *dollar* is the head and *100* is the dependent.

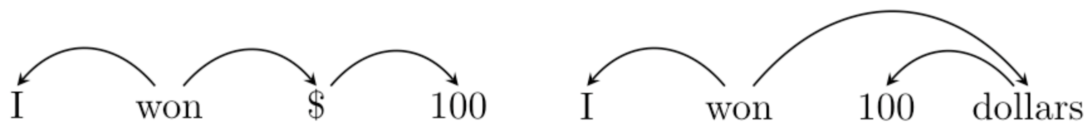


Figure 9: Dependency trees showing analysis of dollar/$

However, this would result in categories like $\$ for *100*. Perhaps more egregiously, the verb *won* in "I won $ 100" would be assigned a category like (VB\NN)/$. A potential solution would have been to map the dollar sign to the category NN to avoid the odd categories above.

The decision was made to reanalyze the last numeric item in the string to be the root and have the dollar sign be a dependent; cases with the word *dollar* did not change. While this loses the dependency similarities across similar constructions, it produces an equivalent derivation to similar constructions in CCGbank.

[4] Previously, elements in a list were all children of the first element. Thus *John* and *Bill* would be dependents of *Sam*. This would result in *John* and *Bill* acting in an adjectival capacity modifying *Sam*.

### *3.6.3 CoNLL08 - Hyphenated Constructions*

Multiword compounds joined by hyphens are given special treatment in the treebank. When annotating the treebank, information concerning the constituent structure of the compounds was included as nodes in the dependency tree. These extraneous nodes had to be removed, while reassigning correct dependency relations, POS tags and parenthood links. See Figure 9 for an example of the NP "190-point plunge." The tree on the left is as annotated in the treebank. The tree on the right is the tree post processing

### *3.6.4 CoNLL08 - Tagging Errors*

As the derivational algorithm takes advantage of the POS tags of words, the accuracy of the derived CCG categories is affected by misannotations. While checking POS tags manually is not within the scope of this project, errors are corrected when found.

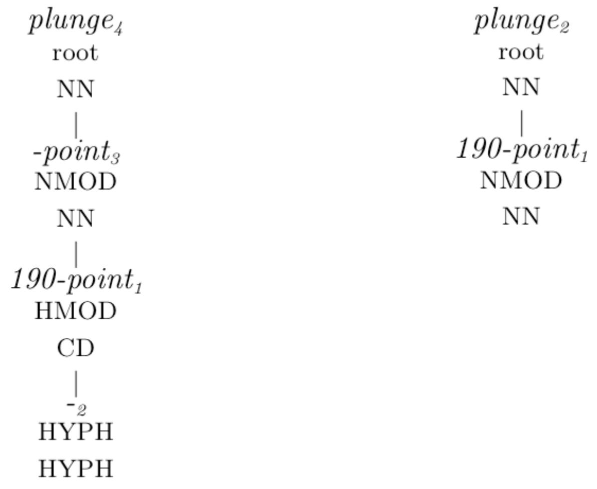


Figure 10: Example of a Hyphenated Construction.

### *3.6.5 TUT – Articles/Determiners*

Articles/determiners in the treebank tended to be analyzed as the head of an Article-Noun combination. This results in nouns receiving categories like ART\ART. This analysis is fixed to reflect the conventional linguistic theory that nouns are the heads of noun phrases.

### *3.6.6 TUT – Compressing Prepositions + Articles*

As in other Romance languages, certain sequences of preposition and article combine to form a single word. In the treebank, the resulting combination is entered as two individual sequential nodes in the dependency tree: the first to represent the preposition and the second to represent the article. This is not representative of how the language is actually used, nor does it aid in the development of a lexicon. The duplicate entries are compressed and the correct head-dependent relations are established.

### *3.6.7 TUT – Pre-Noun Adjectives*

In certain environments, adjectives that preceded nouns were labeled as the head while the nouns were labeled the dependent. As adjectives are traditionally thought to modify nouns, the head-dependent relationships were reversed when found.

## 3.7 Final Algorithm

The algorithm is as follows:

- Preprocesses
- Categorize parent-daughter relations as Adjunct or Argument.
- Assign each node's POS tag as its category.
- Calculate category of node using dependencies and their relations.
    - Allow application rules and composition rules

Figure 11 shows a sample category assignment for the one of the sentences common to Penn Treebank and the CoNLL08 shared task. The top sentence shows the results from the dependency based approach and the bottom shows the results from the constituent-based derivation of CCGbank. Note the differences found in the category assignments for the word *Section*. The dependency categories show that *Section* modifies *89* whereas the constituent-based category shows that it is a noun modifier of the word *rules*. Also note the lack of long-range dependencies in the dependency-based category for the word *lobbied*; they are present in the constituent-based approach.

Similarly, Figure 12 shows a derivation of an Italian sentence, with the top sentence being derived via dependency relations and the bottom derivation from the constituent-based derivation of TutCCG. Once again, the long-range dependencies are missing in the dependency-based approach. Note the method in which TutCCG assigns categories to punctuation, creating an overlaying category of *t*(ext). This is unique among the approaches referenced in this paper.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Small-business* | *interests* | *have* | *lobbied* | *against* | *the* | *so-called* | *Section* | *89* | *tax* | *rules* | . |
| NN/NN | NN | (VB\NN)/VB | VB | (VB\VB)/NN | NN/NN | NN/NN | (NN/NN)/(NN/NN) | NN/NN | NN/NN | NN | . |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| *Small-business* | *interests* | *have* | *lobbied* | *against* | *the* | *so-called* | *Section* | *89* | *tax* | *rules* | . |
| n/n | n | (s[dcl]\np)/(s[pt]\np) | (s[pt]\np)/pp | pp/np | np[nb]/n | n/n | n/n | n/n | n/n | n | . |

Figure 11: English sentence from dependency and constituent-based derivational processes

$$\frac{\mathit{Slitta}}{\mathsf{VB/NN}} \quad \frac{\mathit{a}}{\mathsf{(VB\backslash VB)/NN}} \quad \frac{\mathit{Tirana}}{\mathsf{NN}} \quad \frac{\mathit{la}}{\mathsf{(NN/NN}} \quad \frac{\mathit{decisione}}{\mathsf{NN}} \quad \frac{\mathit{sullo}}{\mathsf{(NN\backslash NN)/NN}} \quad \frac{\mathit{stato}}{\mathsf{NN}} \quad \frac{\mathit{di}}{\mathsf{(NN\backslash NN)/NN}} \quad \frac{\mathit{emergenza}}{\mathsf{NN}} \quad \frac{.}{.}$$

$$\frac{\mathit{Slitta}}{\mathsf{s\colon dcl/np}} \quad \frac{\mathit{a}}{\mathsf{((s\colon dcl/np)\backslash(s\colon dcl/np))/n}} \quad \frac{\mathit{Tirana}}{\mathsf{n}} \quad \frac{\mathit{la}}{\mathsf{(np/n}} \quad \frac{\mathit{decisione}}{\mathsf{n}} \quad \frac{\mathit{sullo}}{\mathsf{(n\backslash n)/n}} \quad \frac{\mathit{stato}}{\mathsf{n}} \quad \frac{\mathit{di}}{\mathsf{(n\backslash n)/n}} \quad \frac{\mathit{emergenza}}{\mathsf{n}} \quad \frac{.}{\mathsf{t\backslash s\colon dcl}}$$

Figure 12: Italian sentence from dependency and constituent-based derivational processes

# Chapter 4: Evaluation

Difficulty will arise when evaluation is required and no gold standard is present. Such is the case for this task; no gold standard set of lexical categories exists that can validate this approach. Instead, the resulting lexicons from this approach are compared with their previously created counterparts. For ease in the evaluation, the lexicon derived from the CoNLL08 treebank shall be referred to as DepEngCCG and the lexicon derived from the TUT treebank shall be referred to as DepItCCG.

The comparison follows two different approaches. The first is purely metric based, showing characteristics of the lexical datasets, such as category complexity and category growth. The second compares the effectiveness of using DepEngCCG supertags and CCGbank supertags as features in another task, namely sentiment detection.

## 4.1 Comparison with Category Statistics

Without context, a statistic is simply a number. The goal of these statistics is to help answer a question, namely *what makes a good lexicon*? Following tradition, Ockham's razor, which rewards simplicity, is assumed. For this application, this means that a small lexicon would rank higher than a large lexicon (with the assumption that the smaller lexicon would adequately model the language), and simpler categories would rank higher than more complex categories.

### 4.1.2 Category Counts and Growth

The resulting lexicon of DepEngCCG contains a total of 95,584 entries for 43,999 word types; there are a total of 1,952 different categories. As is expected, the majority of words only have a small number of categories assigned to them, yet a small number of functional words contained a high number of categories. This resultant lexicon is significantly larger than that of the CCGbank, with 74,669 entries and a total of 1,286 categories.

For DepItCCG, a count of 854 categories is derived across 8813 words, for a total of 14925 entries. The number of categories derived for DepItCCG is a degree less than the number found in TUTCCG, which was 1152 categories.

The ten words with the highest category count for each derivation are given in Table 1. One can generalize that the dependency derivation for English is typically more (perhaps overly) explicit, as counts for the same words are typically higher in DepEngCCG. It appears DepItCCG and TutCCG are more on par in terms of category counts, with values more tightly centralized. Tables 2 and 3 show the top ten most frequent categories in terms of type. Token counts are also given. In each derivation, it is not surprising that nouns and noun modifiers are the most frequent of categories. For these categories, type counts from DepEngCCG to CCGbank and from DepItCCG to TutCCG are quite close. This would hint that dependency and constituent-based derivation processes have quite similar results for the classes of nouns and noun modifiers. Conversely, the Tables 2 and 3 also show that the derivations for more functional categories, like verbs, differ across methods. These differences can be seen as limitations or constraints of the approach due to input source; as previously stated, the dependency trees did not syntactically encode null or trace elements that were used in the CCGbank and TutCCG derivations.

Figures 10-13 show the growth of the total number categories as the derivational processes generate categories from the input sentences. All graphs give a similar shaped plot, showing an initial period of rapid category growth whose rapidity slows over time.

As can be inferred from the category count differences for DepEngCCG and CCGbank, DepEngCCG shows a trend of generating more categories given the same amount of input. At this point, it is unclear if this should signify a better approach (constituent trees vs. dependency trees) or if the dependency approach simply has not yet had as much focus on more idiosyncratic grammar constructions. However, both approaches show a general trending plateau for categories that appear more than five times. This lends credence to the idea of a Zipfian distribution: that after a certain amount of input, only novel and unique categories would be generated; all the common and frequent categories are generated in the initial amount of input.

The growth for the Italian categories (see Figures 12 and 13), on the other hand, shows that while the dependency approach shows the category plateau, the trend for TutCCG shows a constant slope; this could be alleviated by the addition of combinatory rules for specific but common linguistic structures as well as abstracting over modifier and conjunction categories (Bos, Bosco and Mazzei 2009).

| DepEngCCG | | CCGbank | | DepItCCG | | TutCCG | |
|---|---|---|---|---|---|---|---|
| Word | # Cats | Word | # Cats | Word | # Cats | Word | # Cats |
| and | 252 | as | 130 | e | 136 | e | 176 |
| is | 226 | is | 109 | in | 90 | , | 93 |
| to | 203 | to | 98 | di | 87 | in | 82 |
| in | 196 | than | 90 | per | 77 | per | 70 |
| as | 168 | in | 79 | a | 77 | o | 62 |
| of | 156 | - | 67 | da | 62 | sono | 58 |
| are | 151 | ‘s | 67 | sono | 49 | ha | 51 |
| was | 150 | for | 66 | che | 46 | da | 45 |
| for | 137 | at | 63 | come | 45 | a | 45 |
| at | 134 | was | 61 | del | 43 | con | 42 |

Table 1: Ten tokens with highest category counts

| DepEngCCG | | | CCGbank | | |
|---|---|---|---|---|---|
| Cat | Type Count | Token Count | Cat | Type Count | Token Count |
| NN/NN | 20141 | 216630 | N/N | 21485 | 152508 |
| NN | 20126 | 208413 | N | 20544 | 206312 |
| (NN/NN)/(NN/NN) | 4514 | 22842 | (S[dcl]\NP)/NP | 2360 | 16055 |
| NN\NN | 3968 | 17929 | S[adj]\NP | 1873 | 7974 |
| VB/NN | 3166 | 18490 | (S[b]\NP)/NP | 1530 | 13033 |
| (NN\NN)/(NN\NN) | 3138 | 12870 | (N/N)/(N/N) | 1414 | 5830 |
| VB | 3052 | 12377 | S[pss]\NP | 1293 | 6988 |
| (VB\NN)/NN | 1800 | 11312 | (S[ng]\NP)/NP | 1247 | 5838 |
| VB/VB | 1757 | 18009 | N[num] | 1144 | 8547 |
| JJ | 1660 | 8284 | S[dcl]\NP | 1092 | 5564 |

Table 2: Ten Most Frequent Categories in the DepEngCCG and CCGbank

| DepItCCG | | | TutCCG | | |
|---|---|---|---|---|---|
| Cat | Type Count | Token Count | Cat | Type Count | Token Count |
| NN | 3466 | 11833 | n | 3528 | 9977 |
| NN\NN | 1520 | 3065 | n\n | 1245 | 2559 |
| NN/NN | 696 | 5158 | n/n | 521 | 1212 |
| VB/NN | 557 | 875 | n/pp | 399 | 781 |
| (NN\NN)\(NN\NN) | 514 | 892 | s:inf/np | 237 | 328 |
| VB\VB | 359 | 1020 | s:adj\np | 219 | 319 |
| NN/IN | 357 | 647 | (n\n)/pp | 200 | 326 |
| VB/VB | 338 | 1566 | (s:dcl\np)/np | 186 | 302 |
| VB | 266 | 373 | np | 175 | 1306 |
| JJ | 1660 | 8284 | S[dcl]\NP | 1092 | 5564 |

Table 3: Ten Most Frequent Categories in DepItCCG and TutCCG

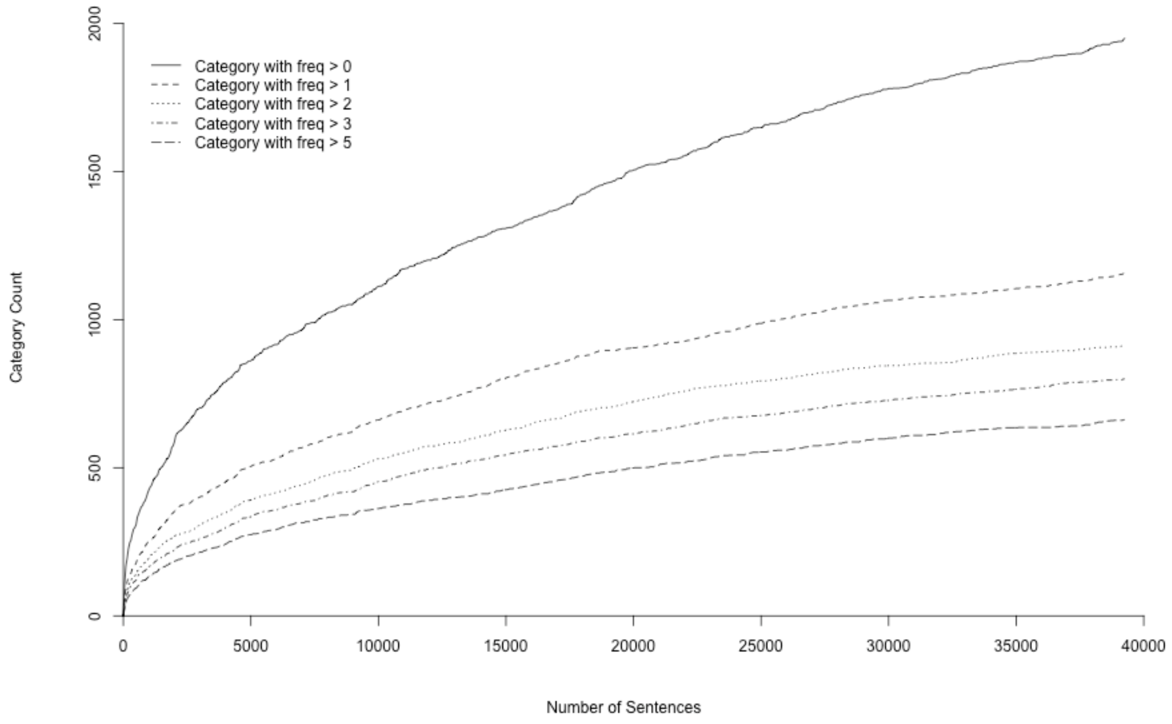


Figure 13: Category growth for DepEngCCG

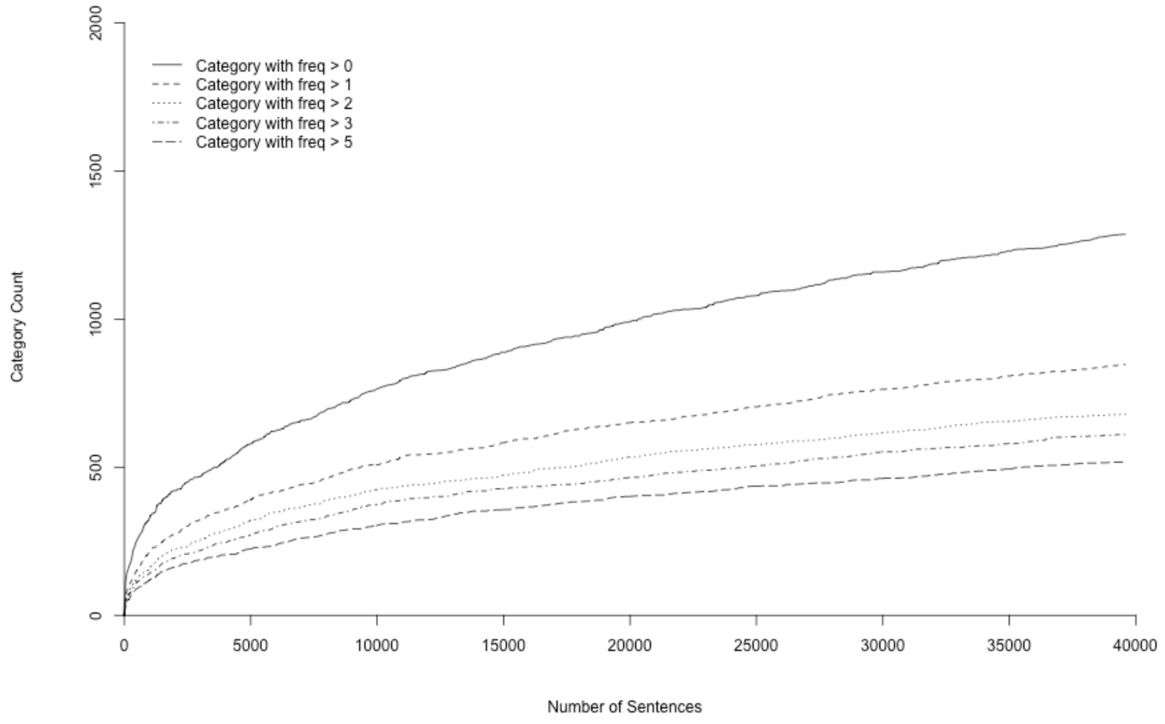


Figure 14: Category growth for CCGbank

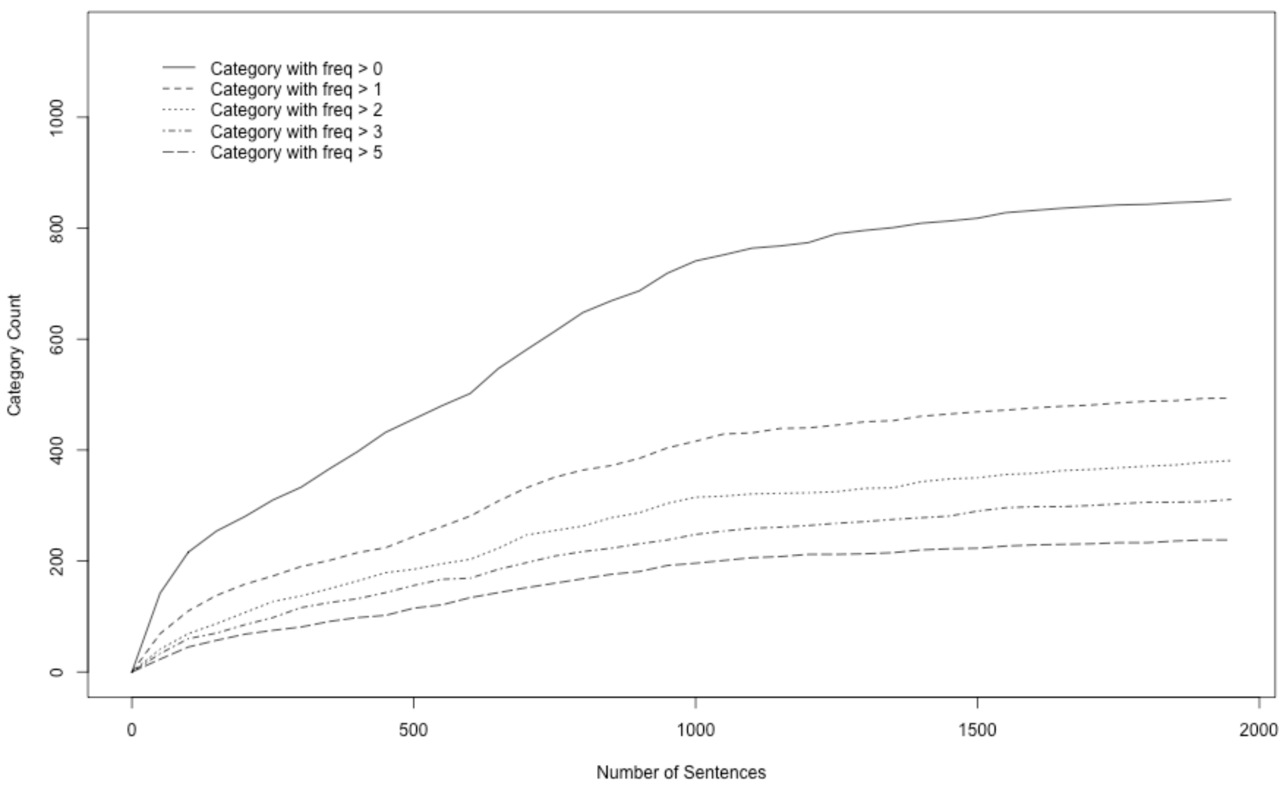


Figure 15: Category growth for DepItCCG

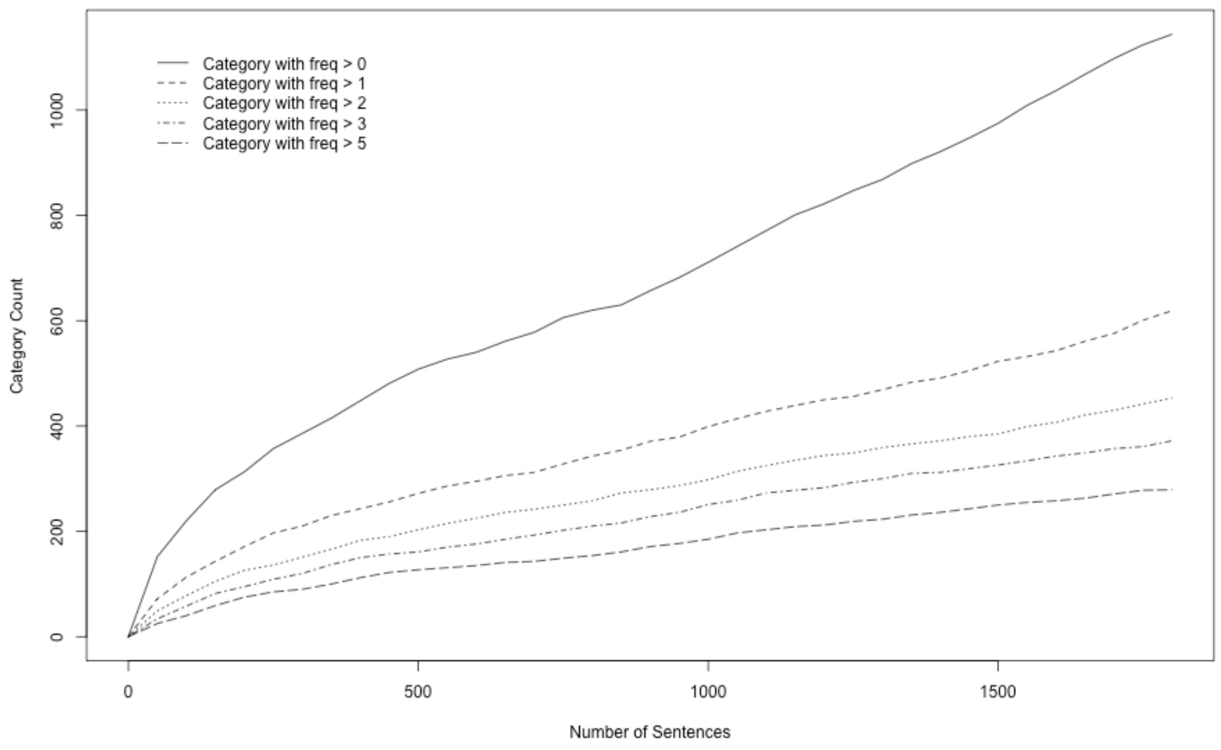


Figure 16: Category growth for TutCCG

### 4.1.2 Category-Category Mapping

To compare the lexical coverage given by the different derivation methods, a category mapping attempts to align equivalent categories. For example, perhaps the word *kicked* has the CCGbank category of (s\np)/np and the DepEngCCG category of (VB\NN)/(NN). These categories would be considered equivalent as each atomic category within the full functional category can be mapped to create an equivalent. The CCGbank category np\np and the DepEngCCG category NN/NN would not considered equivalent (due to the directionality of the slash operator). The mapping chart is given in

Appendix C. This mapping does not take into account that a single atomic category may be able to translate into a functional category. The similarity measure is calculated on a word-category basis. The table below gives the similarity measures of the two derivations, using CCGbank and TutCCG as gold standards for comparison.

| **Dataset** | **Precision**[5] | **Recall**[6] | **F-Score**[7] | **# Unique words** |
|---|---|---|---|---|
| DepEngCCG | 0.367 | 0.467 | 0.411 | 176 |
| DepItCCG | 0.460 | 0.469 | 0.464 | 1863 |

Table 4: Similarity Measures of Lexicons

Table 4 shows that nearly half of all word-categories derived by the process given in this paper have an equivalent category in its sister lexicon. Through a sampled manual comparison, a number of the non-matches are cases that could have been matches had the dependency trees encoded null and trace elements to aid in the derivation categories that predict long-range dependencies. While both approaches were able to encode that relative clauses modify nouns, as an example, the encoding method was quite different.

Table 4 also shows that there were a number of unique words that were found in the dependency-based derivations that were not found in the constituent-based derivations. For DepEngCCG, this is not unexpected, as the Penn Treebank is but one of a number of sources used to make the CoNLL08 shared task corpus. For DepItCCG, however, the same input source of the TUT was used in both approaches. The discrepancy lies in the fact that 23% of the sentences in TUT did not make it into the final

[5] Precisions = $(WC_{Dep} \cap WC_{Cons})/WC_{Dep}$ "Number of equivalent word-categories divided by the total number of dependency generated word-categories.

[6] Recall = $(WC_{Dep} \cap WC_{Cons})/WC_{Cons}$ "Number of equivalent word-categories divided by the total number of constituency generated word-categories.

[7] F-score (harmonic) = 2*(Recall*Precision)/(Recall+Precision)

version of TutCCG (Bos, Bosco and Mazzei 2009). For DepItCCG, only a small number of sentences were not included.

### 4.1.3 Category Complexity

In Baldridge 2008, a probability distribution that states the probability of a category is inversely proportional to its complexity is used to improve super-tagging performance on highly ambiguous words. The complexity measure used is employed here as well. The complexity measure simply counts the number of subcategories or tokens contained in a category.

For example, the category for the word *over* (4b) has the category *((VBD\NN)\(VBD\NN))/NN*. This category would have the complexity measure of 9: it has *VBD* twice, *NN* three times, *(VBD\NN)* twice, *(VBD\NN)/ (VBD\NN)* once and the full *((VBD\NN)\(VBD\NN))/NN* once.

Histograms showing the complexity distribution over categories for DepEngCCG and CCGbank are given below in Figures 17 and 18 and for DepItCCG and TutCCG in Figure 19 and 20.

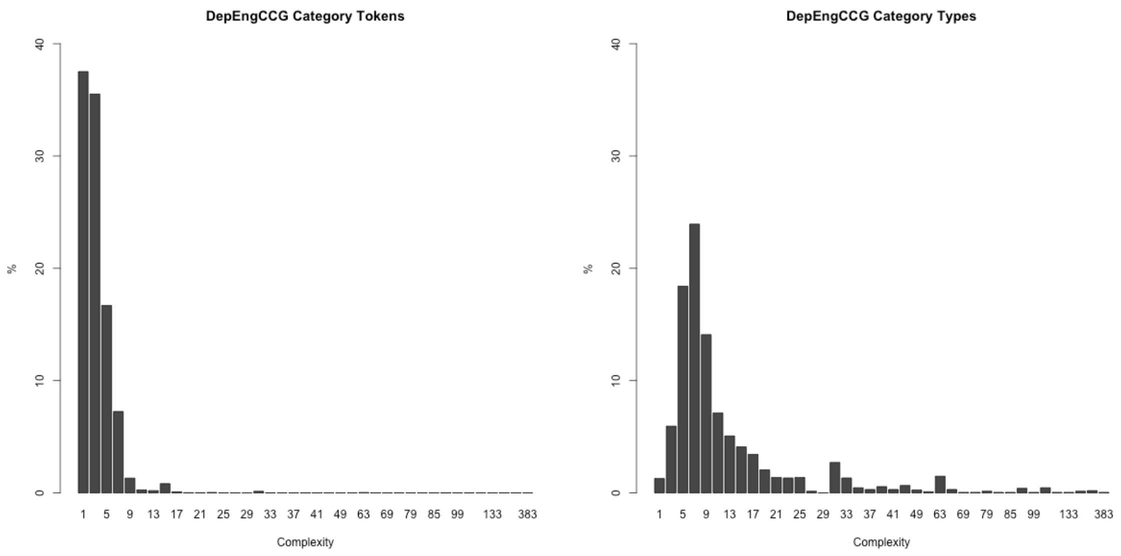


Figure 17: Category Complexity Distribution for DepEngCCG Derivations

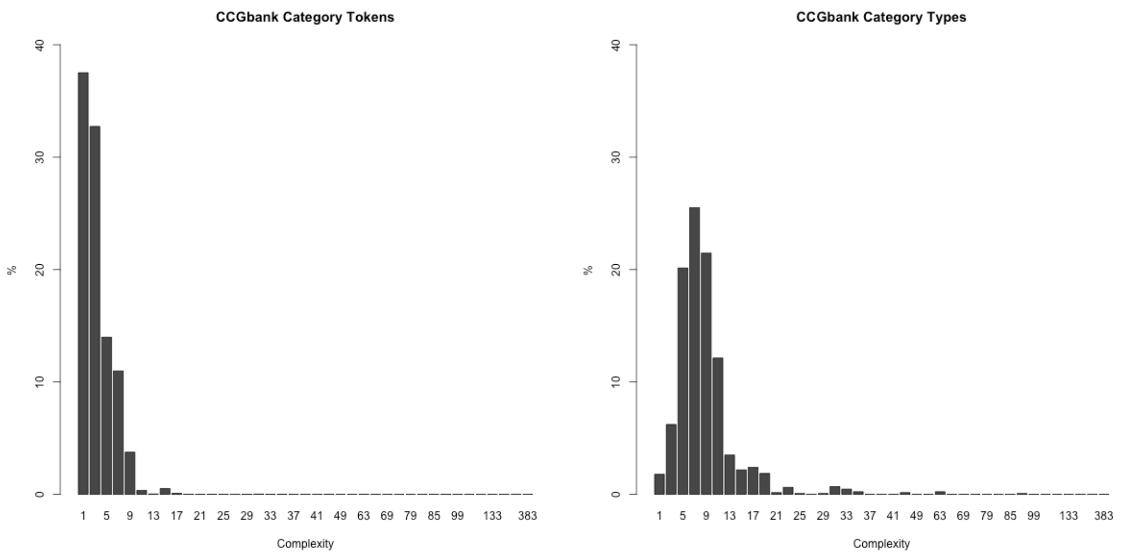


Figure 18: Category Complexity Distribution for CCGbank Derivations

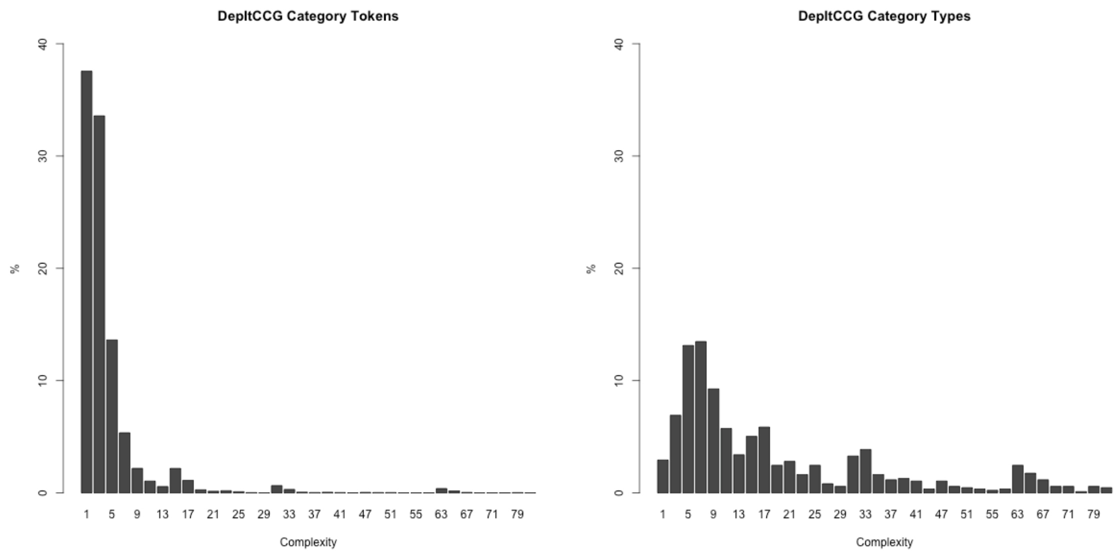


Figure 19: Category Complexity Distribution for DepItCCG Derivations

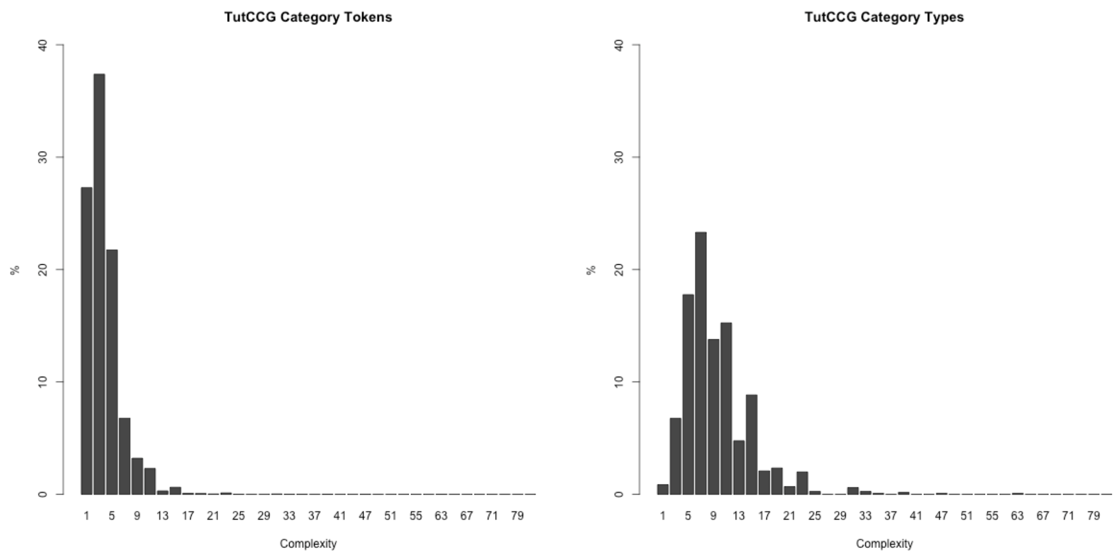


Figure 20: Category Complexity Distribution for TutCCG Derivations

All of the histograms demonstrate what is expected: the simpler the category, the more often it occurs. Higher complexity categories are not only few in type, but also few in occurrence. These categories could be due to very specialize linguistic constructions or due to inadequacies in the derivational process.

The small hump in the tail end of the DepItCCG category types distribution in Figure 19 can be attributed to the difficult sentence constructions that were dropped from TUT. Note the high token count of categories with complexity of 3 for TutCCG. While the other histograms show that 1-complexity categories appear more frequently than 3-complexity categories, the opposite is true of TutCCG. This is due to the fact that TutCCG is the only lexicon of the group that attempted to assign CCG functional categories to punctuation.

Aside from the nuances mentioned above, the histograms model an ideal lexicon well, showing similar shape for each language.

## 4.2 Statistical Predictive Task – Sentiment Analysis

With the use of the web as a social and marketable business medium, sentiment analysis is a growing area of research. The aim is to easily and automatically glean consumer opinion on products, movies, restaurants etc. Consumers' reviews can be classified as positive or negative (Pang and Lee (2004), Pang et al. (2002), Turney (2002)) or can be classified on a 1 to 5-star rating scale (Pang and Lee (2005), Snyder and Barzilay (2007)).

In a typical sentiment classification task, a supervised learning method will read a sentence/document as a bag of N-grams. In theory, the learning model would associate "positive" words like *like* and *good* with a positive review, while associating words like *hate* and *bad* with a negative review. This assumption works well for canonical displays of sentiment, however it is painfully obvious how easily words that predict one opinion can be used to express the exact opposite.

8. Word meaning expressing sentiment
   a. I **love** anything with a **good** plot.
   b. This was a really **bad** movie.
9. Word meaning expressing opposite sentiment
   a. I have no **love** for chick-flicks.
   b. There was nothing **bad** about the movie.

Various methods have found suitable methods to attempt to cope with this issue. For the sake of comparing the CCGbank to the lexicon derived in this paper, the CCG categories for each word in the document are added as features to the statistical model with the hypothesis that negative and positive usages of the same word have different categories assigned to them.

Testing on the same datasets with two different category sets as features allows a relatively easy and objective measure to differentiate the two category sets. Classification of the documents without the categories as features allows comparison with a baseline. It is important to stress that this will not be an effort to compete with current sentiment analysis work, but rather an attempt to gauge the quality of two lexicons via a modern and relevant task.

### 4.2.1 Data

For this experiment, a combined dataset from three different sources is assembled: from Pang and Lee (2004),[8] a collection of 2000 movie reviews; from Pang and Lee (2005),[9] a collection of 10,310 movie reviews in sentence/snippet form; and a set of 2000

[8] Polarity Dataset v2.0

[9] Sentence Polarity Dataset v1.0

video reviews and 2000 DVD reviews from Blitzer et al. (2007).[10] In total, this summed to a sizable collection of over 16,000 documents with an even split between the negative and positive classifications. The full dataset is then split into a development set of 2000 documents and a test set of the remaining 14310 documents. Any testing and parameter setting is done using the development set.

### 4.2.3 Process

A Hidden Markov Model (HMM) supertagger[11] is created for each dataset, using as training data the CCG derivations from the CCGbank and the CCG derivations presented in this paper. For an introduction/discussion to HMMs, see Rabiner (1989). Such sequencing prediction modes have proven quite effective for the supertagging task (Bangalore and Joshi, 1999; Nielsen, 2002). This HMM supertagger is then used to assign CCG lexical categories to the movie review documents. Future improvements on this tagging step could include verifying that the resulting supertags would fully combine for each sentence.

Positive/negative classes are assigned to each document by use of a Maximum Entropy classifier from the OpenNLP software package.[12] For more information on maximum entropy uses in natural language processing, see Berger et al. (1996) and Ratnaparkhi (1998). A Gaussian prior distribution is assumed, using a sigma value of 0.01, which was calculated using the development set.

As features in the maximum entropy models, the following are tested: words only, words with supertags, words with POS tags, supertags only and POS tags only. Words with tags are created using the following format: *word_tag*.

[10] Multi-Domain Sentiment Dataset v2.0

[11] Thanks to Baldridge (2008) for use of the HMM.

[12] http://maxent.sourceforge.net/

### 4.2.4 Results

Ideally, a sentiment classifier should be able to accurately predict a polarity regardless of the type of input. However, not all documents are created equally; a review may contain a general plot summary with only a single concluding sentence containing any opinion data or the entire review may be an opinionated rant or rave with no filler.

The full combined dataset is tested using ten-fold cross-validation to obtain a general baseline on how well the classifier performs on a set of varying data. The dataset is then broken up into their original subsets and tested again, allowing comparison with the full dataset as well as with previous work on the subsets of data. The sigma value of 0.01 is also used on the subsets.

#### *4.2.4.1 Full Dataset*

The results of a ten-fold cross-validation over the test dataset with word-categories as features are given below in Table 5. It is clear from the results that adding supertags to the words in the documents does not aid in the classification tasks. This is verified in Table 6, which shows that using only supertags as features in the maximum entropy model only outperforms chance by a small margin. At the same time, it does not appear that added supertags particularly hurt the accuracy to a large degree, losing only 0.3-0.5% accuracy; however, this small percentage is not statistically significant and should most likely be attributed to noise.

| Feature Lexicon | Accuracy | Positive Classification | | | Negative Classification | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F-Score | Precision | Recall | F-Score |
| DepEngCCG | 78.7% | 0.761 | 0.828 | 0.793 | 0.811 | 0.740 | 0.774 |
| CCGbank | 78.4% | 0.763 | 0.831 | 0.796 | 0.815 | 0.743 | 0.777 |
| POS | 78.2% | 0.760 | 0.823 | 0.791 | 0.807 | 0.741 | 0.773 |
| Token only | 78.9% | 0.771 | 0.824 | 0.796 | 0.811 | 0.755 | 0.782 |

Table 5: Results of Sentiment Analysis Using Word-Categories as Features

| Feature Lexicon | Accuracy | Positive Classification | | | Negative Classification | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F-Score | Precision | Recall | F-Score |
| DepEngCCG | 56.3% | 0.543 | 0.792 | 0.644 | 0.617 | 0.336 | 0.434 |
| CCGbank | 57.7% | 0.557 | 0.750 | 0.639 | 0.618 | 0.404 | 0.488 |
| POS | 56.6% | 0.547 | 0.769 | 0.639 | 0.611 | 0.364 | 0.346 |

Table 6: Results of Sentiment Analysis Using Categories Only as Features

Table 7 shows the breakdown of the errors made for each classification analysis. There is a clear trend that the majority of errors in classifications occur on negative polarity documents. For the classifications that took word-categories or word tokens only as input, this could be attributed to a number of documents using positive polarity items in a sarcastic tone or by explaining what a good movie *should* be. Table 8 demonstrates that this disparity occurs at the category/POS-tag level as well. The reasons for this are not immediately clear and merit future investigation.

| Dataset | % Error Pos | % Error Neg |
|---|---|---|
| DepEngCCG | 39.6% | 60.4% |
| CCGbank | 39.9% | 60.1% |
| POS | 40.6% | 59.6% |
| Untagged | 41.8% | 58.2% |

Table 7: Error Breakdown for Classification Using Word-Categories

| Dataset | % Error Pos | % Error Neg |
|---|---|---|
| DepEngCCG | 23.8% | 76.2% |
| CCGbank | 29.5% | 69.5% |
| POS | 26.5% | 73.5% |

Table 8: Error Breakdown for Classification Using Categories Only

#### *4.2.4.2 Movie Review Polarity Dataset Results*

Previous work using this body of documents has shown success in accurately identifying the sentiment of a given review. In Pang and Lee (2004), a classifier labels each sentence in a document as either subjective or objective; the objective data is removed and the resulting extract of subjective text is classified using Naïve Bayes (NB) and Support Vector Machines (SVM) at the document level. This approach aims to remove distracting text such as plot summaries that could otherwise mislead a classification. Accuracies for classifiers trained on the subjective extracts achieved a statically significant gain (from 82.8% to 86.4% for NB) or were able to maintain accuracy (SVMs) when using an average of 60% of the original document

Work done by Boiy et al. (2007) expands on Pang and Lee (2004) to include a maximum entropy classifier for the subjective extracts and attempts classifications using bigrams and just the adjectives found within the documents. Three different classification models are tested (support vector machine, naïve Bayes and maximum entropy) with four different feature sets (unigrams, unigrams with subjectivity extracts, bigrams, adjectives). A maximum accuracy of 87.4% was reported using a unigram word model of the subjectivity extracts as feature in a maximum entropy classifier. Interestingly, the classification using only adjectives as features achieved a relatively high accuracy of 82.0% for the Naïve Bayes classifier.

Results for the classification using the supertags as features is given in Table 9. When compared to the results found in Table 6, it is clear maintaining a homogeneous distribution of documents will improve classification accuracies; in this case, we see around a 5% increase in accuracy. The results of the maximum entropy classifier trained only on word tokens are consistent with the results found for the equivalent classifier in Boiy et al (2007).

As before, the addition of supertags as features has not shown successful in aiding the classification task.

| Feature Lexicon | Accuracy | Positive Classification | | | Negative Classification | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F-Score | Precision | Recall | F-Score |
| DepEngCCG | 83.8% | 0.827 | 0.055 | 0.841 | 0.850 | 0.821 | 0.832 |
| CCGbank | 83.7% | 0.827 | 0.853 | 0.840 | 0.848 | 0.821 | 0.834 |
| POS | 83.4% | 0.825 | 0.846 | 0.836 | 0.842 | 0.821 | 0.831 |
| Token only | 84.2% | 0832 | .0856 | 0.839 | 0.852 | 0.827 | 0.839 |
| Pang/Lee 2004 NB | 82.8% | | | | | | |
| Pang/Lee 2004 NB+subj extract | 86.4% | | | | | | |
| Pang/Lee 2004 SVM | 87.2% | | | | | | |
| Boiy et al 2007 MaxEnt | 84.8% | | | | | | |
| Boiy et al 2007 MaxEnt+subj extract | 87.4% | | | | | | |
| Boiy et al 2007 Maxent Bigrams | 85.4% | | | | | | |
| Boiy et al 2007 NB Adj | 82.0% | | | | | | |

Table 9: Classification Results on Movie Review Polarity Dataset

### *4.2.4.3 Sentence Polarity Dataset Results*

In comparison to many other datasets that contain multi-sentence documents, this corpus is a collection of single sentence documents, annotated to show positive or negative sentiment polarity. Created by Bo and Pang (2005), the sentences/snippets were described to be "striking," perhaps with the goal that a more striking sentence would expand the distinctions between positive and negative polarity. Bo and Pang (2005) used this dataset to develop a *positive-sentence percentage* (PSP) similarity measure that increased accuracy when attempting to assign 3-4 star ratings to a movie reviews. As the accuracy of sentence-level classification was not their primary focus, no information was given on the performance of the naïve Bayes classifier. However, it was shown that an author's rating for a movie and the review's PSP of said movie were directly proportional.

In Radovanoci and Ivanoci (2008), a number dimensionality reduction techniques are compared. Unigrams, bigrams and trigrams are all used together as features. Using the SIMPLS[13] (de Jong 1993) supervised linear feature extraction algorithm to garner features and employing the 25-nearest neighbor classification algorithm, the sentence polarity dataset was classified with about[14] 76% accuracy using only seven features. Naïve Bayes classification was found to have approximately 77% accuracy, while a support vector machine achieved approximately 73%.

A Naïve Bayes classifier was tested in Andreevskaia and Bergler (2008). Separate models were trained for unigrams, bigrams and trigrams. Results with ten-fold cross-validation show that the unigram-trained classifier performed best with 77.4% accuracy; bigram accuracy was second with 73.9% accuracy and trigrams came last with 65.4% accuracy. These results show a special property of sentence-level annotation: greater sensitivity to sparseness. With higher order n-grams, there is a higher probability of uniqueness, thus also increasing the chance of missing a sentiment marker in the sentence.

The results from the current tests are given in Table 10. Despite the "striking" quality of the sentence snippets, the results show that classifying sentence/snippets is less accurate than a full-bodied review. The best results are achieved by the classifier that trained only on words, yet still this is significantly less accurate than the results for the

[13] An efficient Least Partial Squares (LPS) algorithm.

[14] Values for accuracies are not expressly given in Radovanoci and Ivanoci (2008). Values are best estimates given a results graph.

naïve Bayes classifiers of Radovanovic and Ivanovic (2008) and Andeevskaia and Bergler (2008). This difference may be attributed to the use of unigrams as opposed to bigrams and trigrams. It is also possible that the sigma value determined on the developments set is not optimized for these shorter length documents. However, due to the greater sensitivity to sparseness, it is possible that recalculating sigma using only this dataset would improve accuracy better than would the addition of bigrams or trigrams to the model.

#### *4.2.4.4 Multi-Domain Sentiment Dataset Results*

Research using the Multi-Domain Sentiment dataset focuses on how to mitigate the negative effects of training on one domain (movies) and testing on another (kitchen appliances). Blitzer et al. (2007) calculate a measure of domain similarity that correlates to how well one domain may be adapted for classifying a different domain. To calculate this measure, a baseline is determined for each domain. For the DVD domain, a baseline of 82.4% accuracy is reported. Dredze et al. (2008) follows the approach of Blitzer et al. (2007) and applies it to confidence-weighted online learning methods. It is shown that the faster and simpler online linear classifier algorithms (passive-aggressive algorithm of Crammer et al. 2006) perform nearly as well as batch classifiers (MaxEnt, SVMs).

Classification in this paper joined the video and DVD domain datasets under the assumption that the domains were nearly indistinguishable. While this could introduce error, the classification accuracies from this paper match or exceed the classification accuracies found in other papers. These results are given in Table 11. As with the other tests in this paper, the addition of supertags has not increased the classification accuracy.

| Feature Lexicon | Accuracy | Positive Classification | | | Negative Classification | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F-Score | Precision | Recall | F-Score |
| DepEngCCG | 70.2% | 0.689 | 0.738 | 0.712 | 0.718 | 0.666 | 0.691 |
| CCGbank | 70.6% | 0.691 | 0.745 | 0.717 | 0.724 | 0.667 | 0.695 |
| POS | 71.5% | 0.701 | 0.750 | 0.725 | 0.732 | 0.680 | 0.705 |
| Token only | 72.5% | 0.711 | 0.760 | 0.734 | 0.742 | 0.691 | 0.715 |
| R&I 2008 – NB (1,2,3)-grams | ~77% | | | | | | |
| R&I 2008 – SIMPLS+25NN | ~76% | | | | | | |
| R&I 2008 – SVM (1,2,3)-grams | ~72% | | | | | | |
| A&B 2008 – NB unigram | 77.4% | | | | | | |
| A&B 2008 – NB bigram | 73.9% | | | | | | |
| A&B 2008 – NB trigram | 65.4% | | | | | | |

Table 10: Classification Results on Sentence Polarity Dataset

| Feature Lexicon | Accuracy | Positive Classification | | | Negative Classification | | |
|---|---|---|---|---|---|---|---|
| | | Precision | Recall | F-Score | Precision | Recall | F-Score |
| DepEngCCG | 83.3% | 0.832 | 0.836 | 0.834 | 0.835 | 0.832 | 0.833 |
| CCGbank | 82.9% | 0.831 | 0.825 | 0.828 | 0.826 | 0.833 | 0.829 |
| POS | 82.9% | 0.836 | 0.819 | 0.827 | 0.823 | 0.839 | 0.831 |
| Token only | 82.5% | 0.834 | 0.811 | 0.822 | 0.816 | 0.839 | 0.827 |
| Blitzer et al. 2007 (DVD) | 82.4% | | | | | | |
| Dredze et al. 2008 (DVD) – PA | 80.4% | | | | | | |
| Dredze et al. 2008 (Video) - PA | 80.1% | | | | | | |
| Dredze et al. 2008 (DVD) – MaxEnt | 80.7% | | | | | | |
| Dredze et al. 2008 (Video) - MaxEnt | 80.5% | | | | | | |

Table 11: Classification Results on Multi-Domain Dataset

### 4.2.5 Summary

While adding supertags as features can often reap benefits in statistical tasks, the results in this paper exhaustively show that such is not the case for sentiment analysis. At the same time, the supertags do not show much sign of causing a significant increase in error, showing that the sentiment classification in this study is occurring purely at the word level. Results from other studies showing increased accuracy from use of bigrams suggest that future investigation may be warranted. Under a bag-of-words approach, the use of bigrams or trigrams could realize the benefit of supertags, potentially being able to differentiate between phrases such as *nothing bad* and *very bad*. Conversely, this may also introduce a higher degree of sparseness and hurt results.

However, accuracies and improvements upon sentiment analysis were not the end goal of these tests. The attempt to measure the quality of the two CCG lexicons has met inconclusive results: at most, the differences in accuracies between DepEngCCG and CCGbank vary only by a fraction of a percentage. At best, it can only be said that DepEngCCG and CCGbank were equally inadequate for the given task.

# Chapter 5: Conclusions

This paper has given a method to generate a CCG lexicon from a dependency treebank, using only the dependency hierarchy and relations. At the heart of this process is the distinction between argument and adjunct relations that drives category derivation. As previously discussed, a dependency tree may better represent the relations between heads and children in comparison to a constituent tree. Unlike a number of other similar projects, this method does not project an intermediate syntactic level, decreasing the risk of error and noise, while remaining true to the binary branching nature of CCG. While the approach presented is meant to be universal to dependency treebanks, it is clear that each treebank will have its own set of idiosyncrasies that will inhibit a plug and play system; preprocessing and special cases will invariably be needed. More work is required to further reduce the category count and deal with specific linguistic structure; determining if each derived sentence can parse will aid in teasing out problems with the derivation processes as well as confirm working derivations.

To measure quality of the process, this derivation algorithm was compared to the lexicon given by Hockenmaier and Steedman (2002) for CCGbank, as well as the lexicon for the Italian TutCCG by Bos, Bosco and Mazzei (2009). Comparing category counts, growth and complexity trends, the dependency CCG lexicons have been shown to be quite similar to their previously derived counterparts. The ideas of lexical simplicity and Zipfian distributions drive these measures; both lexicons generated in this paper show trends of quality lexicons. However, despite a number of metrics being better for DepItCCG, it is clear that more development is required to account for special cases and other linguistic phenomenon.

To further analyze the quality of the derived lexicons, the DepEngCCG and CCGbank were directly compared on a sentiment analysis task. The DepEngCCG performed on par with CCGbank in the classification task, despite the fact that neither CCGbank nor DepEngCCG actually improved results overall. Results for the application study were inconclusive, with the supertags being overshadowed by the words themselves. A future comparison analysis could compare the effectiveness of adding the different category sets as features in a dependency parsing evaluation.

The basis of the algorithm need not be constrained to CCG style derivations; with some work, the algorithm may be modified to generate other lexical formalisms (TAG, HPSG) from dependency treebanks.

Future work could involve expanding the induction system to automatically derive logical forms for each word and its category(ies). This would be done in a similar fashion, but using the dependency relations to define the semantics. At this point, the lack of null and trace elements to encode long-range dependencies may require a reworking of the algorithm to capture the dependencies.

## Appendix A: CoNLL08 Dependency Relations

For more information on these relations, see Johansson and Nugues (2007).

| **Adjunct Relations** | | | | |
|---|---|---|---|---|
| ADV | ADV-GAP | AMOD | AMOD-GAP | APPO |
| DEP | DEP-GAR | GAP-MNR | GAP-NMOD | GAP-TMP |
| HMOD | HYPH | LOC | LOC-MNR | MNR |
| NMOD | POSTHON | PRN | SUFFIX | TITLE |
| TMP | VOC | PRP | | |

Table 12: CoNLL08 Adjunct Relations

| **Argument Relations** | | | | |
|---|---|---|---|---|
| SUBJ | OBJ | PMOD | ROOT | P |
| COORD | CONJ | OPRD | SUB | IM |
| VC | LGS | DIR | NAME | EXT |
| PRD | DTV | PRT | BNF | PUT |
| EXTR | | | | |

Table 13: CoNLL08 Argument Relations

## Appendix B: TUT Dependency Relations

For more information on these relations, see Bosco (2003).

| Adjunct Relations | | |
|---|---|---|
| MODIFIER | RMOD | RELCL |
| REDUC | RMOD+RELCL | RMOD+RELCL+REDUC |
| RMODPRED | RMODPRED+SUBJ | RMODPRED+OBJ |
| APPOSITION | NOFUNCTION | AUX |
| AUX+PASSIVE | AUX+PROGRESSIVE | AUX+TENSE |
| COORDINATOR | COORD | COORD+ADVERS |
| COORD+BASE | COORD+COMPAR | COORD+COND |
| COORD+CORRELAT | COORD+ESPLIC | COORD+RANGE |
| COORD+SYMMETRIC | COORD2ND | COORD2ND+ADVERS |
| COORD2ND+BASE | COORD2ND+COMPAR | COORD2ND+COND |
| COORD2ND+CORRELAT | COORD2ND+ESPLIC | COORD2ND+RANGE |
| COORD2ND+SYMMETRIC | COORDANTEC | COORDANTEC+COMPAR |
| COORDANTEC+CORRELAT | CONTIN | CONTIN+LOCUT |
| CONTIN+DENOM | CONTIN+PREP | EMPTYCOMPL |
| EMPTYLOC | INTERJECTION | VISITOR |
| VISITOR+ROBJ | | |

Table 14: TUT Adjunct Relations

| **Argument Relations** | | | |
|---|---|---|---|
| DEPENDENT | FUNCTION | ARG | SUBJ |
| OBJ | INDOBJ | INDCOMPL | PREDCOMPL+SUBJ |
| PREDCOMPLE+OBJ | EXTRAOBJ | | |

Table 15: TUT Argument Relations

# Appendix C:  Comparison Category Mapping

| DepEngCCG | CCGbank |
|---|---|
| TO | s[to] |
| VB, R | s  s[.+] |
| EX | np[thr] |
| IN | PP |
| NN | np, np[.+], n, n[.+] |

Table 16: English Category Mapping

| DepItCCG | TutCCG |
|---|---|
| TO | s[to] |
| VB, R | s  s:[.+] |
| EX | np[thr] |
| IN | PP |
| NN | np, np:[.+], n, n:[.+] |

Table 17: Italian Category Mapping

## References

Ajdukiewicz, Kazimierz. 1935. Die syntaktische Konnexität. In *Polish Logic 1920-1939*, ed. Storrs McCall. Oxford University Press, 207-231. translated from *Studia Philosophica*, 1, 1-27.

Andreevskaia, Alina and Sabine Bergler. 2008. When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging. In *Proceedings of ACL-08*. pages 290-298.

Baldridge, Jason. 2008. Weakly supervised supertagging with grammar-informed initialization. In *Proceedings of COLING-2008*. Manchester, UK.

Bangalore, Srivinas and Aravind Joshi. 1999. Supertagging: an approach to almost parsing. *Computational Linguistics*, 25(2): 237-265.

Berger, Adam, Stephen Della Pietra and Vincent Della Pietra. 1996. A Maximum Entropy approach to Natural Language Processing. In *Computational Linguistics*, vol. 22, pages 39-71.

Blitzer, John, Mark Dredze and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. Association of Computational Linguistics (ACL).

Boiy, Erik, Pieter Hens, Koen Deschacht, and Marie-Francine Moens. 2007. Automatic Sentiment Analysis in On-line Text. In *Proceedings of the 11th International Conference on Electronic Publishing*. Vienna, Austria.

Bosco, Cristina. 2003. A Grammatical Relation System for Treebank Annotation. PhD Thesis. University of Torino.

Bos J., Bosco, C., and Mazzei, A. 2009. Converting a Dependency-Based Treebank to a Categorial Grammar Treebank for Italian. In Proceedings of TLT8, Milan.

Cahill, Aoife, Mairead McCarthy, Josef van Genabith and Andy Way. 2002. Automatic annotation of the Penn Treebank with LFG F-structure information. In *LREC 2002 Workshop on Linguistic Knowledge Acquisition and Representation – Bootstrapping Annotated Language Data*. Las Palmas, Spain.

Chen, John, Srinivas Bangalore and K. Vijay-Shanker. 2006. Automated extraction of Tree-Adjoining Grammars from treebanks. *Natural Language Engineering*, 12(03):251-299.

Chen, John and K. Vijay-Shanker. 2004. Extraction of TAGs from Treebank. In H. Bunt, J. Caroll, and G.Satta, editors, *New Developments in Parsing Technology*.

Clark, Stephen and James R. Curran. 2004. The importance of supertagging for wide-coverage CCG parsing, In *Proceedings of the 20th international conference on Computational Linguistics*, p.282-es. Geneva, Switzerland.

Clark, Stephen and James R. Curran. 2007. Formalism-Independent Parser Evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics,* pages 328-334, Philadelphia, PA.

Clark, Stephen, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures using a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics,* pages 327-334, Philadelphia, PA.

Crammar, K, O. Dekel, J. Keshet, S. Shalev-Shwartz and Y. Singer. 2006. Online passive-aggressive algorithms. *JMLR*, 7.

Frege, G. (1923), Logische untersuchungen. dritter teil: Gedankenfuge, in 'Beitr¨age zur Philosophie des Deutschen Idealismus', Vol. III, pp. 36–51. Reprinted in I. Angelelli (ed.), Gottlob Frege. Kleine Schriften, Georg Olms, Hildeheim, 1967, pp. 378-394. Translated as Compound thoughts in P.T.Geach & R.H. Stoothoff (transl.), Logical investigations. Gottlob Frege, Basil Blackwell, Oxford, 1977, pp. 55-78.

Hockenmaier, Julia and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Annual Meetings of the Association for Computational Linguistics,* pages 335-342, Philadelphia, PA.

Hockenmaier, Julia. 2003a. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Hockenmaier, Julia. 2003b. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Annual Meetings of the Association for Computational Linguistics*, pages 359-366, Sapporo, Japan.

Honnibal, Matthew, James Curran and Johan BOs. 2010. Rebanking CCGbank for improved NP interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207-215. Uppsala, Sweden.

Johansson, Richard and Pierre Nugues. 2007. Extended Constituent-to-Dependency Conversion for English. In *Proc. of NODALIDA*.

de Jong, Sijmen. 1993. SIMPLS: An alternative approach to partial least squares regression. Chemometrics and Intelligent Laboratory Systems. 18(3), pages 251-263.

Joshi, Aravind and Yves Schabes. 1992. Tree Adjoining Grammars and lexicalized grammars. In M. Nivat and M. Podelski editors, *Tree Automata and Languages*. North-Holland, pages 409-432.

Kaplan, Ronald and Joan Bresnan. 1982. Lexical-Functional Grammar: A formal system for grammatical representation. In *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA, pages 173-291.

Koller, Alexander and Marco Kuhlman. 2009. Dependency trees and the strong generative capacity of CCG. In *Proceedings of the 12th Conference of the European Chapter of the ACL*, pages 460–468. Athens, Greece.

Lambek, Joachim. 1958. The Mathematics of Sentence Structure. *American Mathematical Monthly*, pages 65:154–170.

Miyao, Yusuke, Takashi Ninomiya and Jun'ichi Tsujii. 2004. Corpus-oriented grammar development for acquiring a Head-driven Phrase Structure Grammar from the Penn Treebank. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*, pages 684-693, Hainan Island, China.

Nieslsen, Leif. 2002. Supertagging with combinatory categorial grammar. In *Proceedings of the Seventh ESSLLI Student Session*, pages 209-220.

Nivre, Joakim. 2005. Dependency grammar and dependency parsing. Technical Report MSI report 05133, Vaxjo University: School of Mathematics and Systems Engineering.

O'Donovan, Ruth, Michael Burke, Aoife Cahill, Josef van Genabith and Andy Way. 2005. Large-scale induction and evaluation of lexical resources from the Penn-II and Penn-III Treebanks. *Computational Linguistics*, 31(3):329-365.

Pang, Bo and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proceedings of ACL*, pp. 271--278.

Pang, Bo and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Association for Computational Linguistics*. pp. 115–124.

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 79—86.

Ponvert, Elias. 2008, *Partial Induction of a Categorial Grammar from a Dependency Treebank*. Unpublished paper. University of Texas, Austin.

Pollard, Carl and Ivan Sag. 1994. *Head Driven Phrase Structure Grammar*. CSLI/Chicago University Press, Chicago, IL.

Rabiner, Lawrence. 1989. A tutorial on Hidden Markov Models and selected applications in speech recognition. In *Proceedings of the IEEE*, 77(2):2257-286.

Radovanovic, Milos and Mirjana Ivanovic. 2008. Text mining: approaches and applications. Nov Sad J Math 38(3).

Ratnaparkhi, Adwait. 1998. Maximum entropy models for natural language ambiguity resolution. Ph.D. Dissertation, University of Pennsylvania. IRCS Tech Report IRCS-98-15.

Shen, Libin and Aravind Joshi. 2005. Building an LTAG Treebank. Technical Report MS-CIS-05-15, CIS, University of Pennsylvania, Philadelphia, PA.

Snyder, Benjamin and Regina Barzilay. 2007. Multiple Aspect Ranking using the Good Grief Algorithm. In *Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference*. pp 300–307.

Steedman, Mark. 1996. *Surface Structure and Interpretation*. MIT Press, Cambridge, MA.

Steedman, Mark. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA.

Steedman, Mark and Jason Baldridge. Combinatory Categorial Grammar. To appear in Robert Borsley and Kersti Borjars (eds.) *Constraint-based approaches to grammar: alternatives to transformational syntax*. Oxford: Blackwell.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluis Marquez and Joakim Nivre. 2008. The CoNLL-2008 shared task on join parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*. Manchester, United Kingdom.

Turney, Peter. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. *In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. pp. 417–424.

Vadas, David and James Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 240-247. ACL, Prague, Czech Republic.

Vadas David and James Curran. 2008. Parsing noun phrase structure with CCG. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics,* pages 335-343. ACL, Columbus, Ohio, USA.

Xia, Fei. 1999. Extracting Tree Adjoining Grammars from bracketed corpora. In *Proceedings of the 5th Natural Language Processing Pacific Rim Symposium (NLPRS-99)*, pages 398-403, Beijing, China.

Xia, Fei. 2001. *Automatic Grammar Generation from two different perspectives*. Ph.D. thesis, University of Pennsylvania.

Xia, Fei, Martha Palmer and Aravind Joshi. 2000. A uniform method of grammar extraction and its applications. In *Proceedings of the 2000 Conference on Empirical Methods in Natural Language Processing*, pages 53-62, Hong Kong.