Functional Elements and POS Categories

Qiuye Zhao **Mitch Marcus**
Dept. of Computer & Information Science
University of Pennsylvania
qiuye, mitch@cis.upenn.edu

Abstract

We propose a bootstrapping algorithm which successfully resolves two fundamental tasks: morphology acquisition and the acquisition of a subset of functional words. Given the outputs of these fundamental tasks, we build a nearly state-of-art morphology analyzer performing with a F1-score of 80.94%; also, we can improve the baseline model for acquiring functional words by an absolute error reduction of 26%. Furthermore, with these acquisition outputs, a minimally supervised tagging system proposed before can be turned into a totally unsupervised one, achieving a tagging accuracy of 85.26% for open-class words.

1 Introduction

Studies of child language acquisition have shown that functional elements as distinctive categories are available to the child from very early on. These include both functional words (closed class items) such as determiners (Valian et al., 2009), and functional bound morphemes such as verbal inflections (Yang, 2002). Motivated by such studies, we propose that functional categories, including both functional words and functional morphemes, should be identified first in the process of acquiring syntactic categories automatically from language input. Further motivation comes from the experimental results of the minimally supervised tagging system in (Zhao and Marcus, 2009)[1], which, given only seven functional features, including four contextual features, whether modal verbs or determiners are left or right neighbours, and three specific morphological features, whether *'-ing', '-ed'*, or *'-s'* are observed as endings, performs clustering in order to generate the two fundamental open class categories, verbal vs. nominal. This work suggests that functional elements are highly useful in further classification of open class items.

This is quite different from most other POS induction systems in the Natural Language Processing (NLP) field for inducing Part-of-Speech (POS) tags, which, instead of generating clusters complying with the common understanding of 'syntactic' categories, such as distinct clusters of determiners, nouns or inflections of verbs, tend to output scattered clusters consisting words of interesting similarities on many different dimensions (Christodoulopoulos et al., 2010). This is because, in these clustering-based systems, a vector space over the words is spanned by lexical features and suffix/prefix features, so that the generated clusters mix semantic and syntactic similarities (Clark, 2000).

In this work, we explore an alternative 'top-down' view of deriving categories, as opposed to the 'bottom-up' view adopted by these earlier distributional clustering methods. Here, we report on experiments which acquire functional elements first and integrate the acquisition output into a full unsupervised POS tagging system later. Since we are not aware of previous work to acquire functional elements, we approach the problem by seeking answers for the following two questions: 1) What are the special properties of the distribution of functional elements that enables the child to distinguish them easily from other categories at a very early stage of acquisition? 2) What might the acquisition processes of the two forms of functional elements (bound vs. free) have in common, reflecting some deeper distributional property of functional elements in general?

[1]The tagging system in (Zhao and Marcus, 2009) was referred to as 'unsupervised' in their paper, because back then, a lexicon was a common input to so-called unsupervised POS tagging systems. We classify their system as minimally supervised here, so as to differentiate it from this work which requires only raw text as input.

We believe that answers to these questions begins to emerge from the success of a single bootstrapping algorithm proposed here to the acquisition of both bound and free functional elements, i.e. closed class words and bound morphemes. This bootstrapping algorithm explores a particular simple contextual property of functional elements, which we call the *diversity property*, and resolves both fundamental acquisition tasks without any input beyond raw written text:

> 1) **Bound morphemes** Separating functional elements in the form of bound morphemes from contentful elements. As for English, the two output sets are a list of productive endings such as *'-ed'*, and *'-s'*, and a list of base stems such as *'consist-'* and *'bootstrapp-'*.
>
> 2) **Free morphemes** Separating 'first-order' functional elements in form of free morphemes (words) from contentful elements. As for English, the two output sets are a list of modal verbs and determiners (in a general sense[2]), and a list of nominal elements as well as verbs in bare forms.

The bootstrapping algorithm we propose here operates by iteratively generating two complementary sets, (e.g. base stems and productive endings for the bound morpheme task); in these way, it reflects the intuition behind co-training (Abney, 2004), but in a greatly reduced form and requiring no seeds for initialization. For both tasks, the bootstrapping algorithm generates highly reliable outputs, with barely any errors and requires no task-specific parameter settings. For example, it discovers 26 modal verbs and determiners (in a general sense) from raw text of WSJ Treebank (Marcus et al., 1993) with only a single noisy term.

We validate these outputs on a range of useful applications. Given the two sets output by the bound morpheme task, we can straightforwardly build an unsupervised morphology analyzer which achieves an F1-score of 80.94 evaluated on the CELEX corpus, comparable to a state-of-art morph analyzer (Lignos, 2010) which achieves 82.21 F1 on the same task. Next, given this new morphology analyzer and the two output sets from the free morpheme task, we give a new very simple algorithm for acquiring the full set of functional/closed class words, improving on a reasonable baseline model for acquiring functional/closed-class words by an absolute error reduction of 26%, from 63% to 89% type accuracy as evaluated on the WSJ Treebank[3]. Finally, we plug this newly acquired closed-class lexicon into a minimally supervised tagging system, (Zhao and Marcus, 2009), which requires as input exactly such a lexicon. The resulting system, now completely unsupervised, achieves a tagging accuracy of 85.26% for tagging open-class words as evaluated on the whole WSJ Treebank. Although the evaluation is done over 6 open-class tags only, and thus is not directly comparable to related works, the tagging performance reported here is still satisfying given that recently reported unsupervised tagging accuracy vary among 50%-70% e.g. (Abend et al., 2010), (Reichart et al., 2010) and (Moon et al., 2010).

2 Acquiring Functional Elements

Functional elements are those elements that provide structural clues in expressions, which form a complementary set to the contentful elements that provide semantic content of the expressions. Two forms of elements in languages come into our attention, bound morphemes and free morphemes (words). As for English, functional elements of bound morphemes are inflectional endings or derivational endings/prefixes, for example, in a word *'runs'* the contentful part *'run-'* provides us with some contentful information, whereas, the functional part *'-s'* provide us with some grammatical specification which are specially important when the word is used in context. On the other hand, functional elements of free morphemes, generally refer to those closed-class words that don't fall into lexical categories, i.e. those words that are not nouns, verbs, adjectives or adverbs; however, the functional roles these closed-class words play in context are more complicated.

The algorithm described in section 2.2 solves two tasks on acquiring functional elements: 1) identifying productive endings/prefixes vs. base stems (the contentful parts of words), and 2) identifying determiners (possessive determiners and

[2]In this paper, we refer to determiners in a general sense which includes determiners, possessive determiners and demonstratives.

[3]Given the lack of previously reported results, the baseline model we compare against is to select the most frequent words in the corpus as closed-class items.

demonstratives as well) and modal verbs vs. nouns and verbs. Given these acquisition outputs, we can acquire a full set of closed-class words, falling in all functional categories, with a simple extra step. The subset of functional words that are acquirable by the bootstrapping algorithm is referred as 'first-order' functional words in this work, since they include determiners and modal verbs only which don't project to other functional words.

Before introducing the bootstrapping algorithm in section 2.2, we discuss the diversity property of functional elements first, in section 2.1, which is an important concept to be explored in the algorithm.

2.1 Diversity Property

Our algorithm is built upon a distributional property of functional elements well-known to linguists: they occur in diverse contexts. For example, determiner *'the'* in English is observed everywhere in text and inflectional ending *'-ed'* can be concatenated to most verbs to derive past forms. As discussed above, functional elements provide structural clues to compose expressions and are basically independent to the meanings conveyed by the expressions; therefore, they are not bound to co-occur with specific contentful elements and are natively expected to occur in diverse contexts.

In the NLP field, a more popular property known for distinguishing functional elements is that they occur more often. Although frequency can be used to approximate 'diversity', the high frequency of functional elements is only a reflection of their high degree of contextual diversity, which, following the definition, is more accurately approximated by the types of contexts that an element occurs in.

There are few previous works that quantitatively demonstrate that 'contextual' diversity is a better diversity measurement than frequency regarding experimental results. In this work, we are going to explicitly compare these two options for measuring diversity within the bootstrapping algorithm. As shown in all applications (section 3), measuring contextual diversity for diversity brought in a consistent improvement as compared to measuring frequency.

Proper Contexts

We haven't define in what occurrences two elements may form a contextual relationship, and let's consider the case for bound morphemes first. Given a type of word, e.g. *'laughing'*, several ways of dividing it can yield a pair of possible morphemes with either one being *as/in*-context of the other morpheme: for the division *'laugh-ing'*, *'laugh-'* can be considered *in*-context of *'-ing'* and *'-ing'* considered *as*-context of *'laugh-'* or vice verse; or for the division *'laughin-g'*, *'laughin-'* *in*-context of *'-g'* and *'-g'* *as*-context of *'laughin-'*, or vice verse.

For words, ideal scopes to define contextual relationships should be phrases/phases, from a linguistic point of view, such as N(ominal)P, V(erbal)P and so on. However, it is not easy to detect boundaries of phases in unannotated input; therefore, we only consider contextual relationships between two words in adjacency. For the sake of coherence, we consider the preceding word *as*-context of the following word and the following word *in*-context of the preceding one, however, it could also be vice verse, though with worse experimental results.

As one may have noticed, without any further constraints on the concept of contexts, the top 3 bound morphemes with the highest contextual diversity will be *'-d'*, *'-e'* and *'-t'*, which do not comply with our understanding of functional morphemes in English. In other words, for acquiring inflectional or derivational suffix/prefixes, we want to compute contextual diversity of bound morphemes according to properly justified contextual relations only, instead of from all arbitrary divisions of words. The most simple way of justifying one element as proper contexts for other elements is to check whether it can serve *as*-context of more than one type of element, by which criteria *'laughin-'* is not justified, since it cannot be concatenated by other suffixes than *'-g'* to form a legal word. It is first noticed in (Chan, 2008) that morphological transformations should be discovered with respect to 'base forms', i.e. properly justified contentful stems; and we generalize this idea for a better measurement of diversity to acquire both forms of functional elements in this work.

Suppose that we are given a set of properly justified contexts of bound morphemes including *'laugh-'* but not *'b-'*, the diversity measurement of *'-ing'* will increase by one given the existence of word *'laughing'* but not given word *'bing'*. For the case of words, if only nouns are justified to be proper contexts, then the top words of highest diversity should be determiners. More formally, given a set $\mathbb{B}$ of justified contexts, we can define two measurements of diversity for an element e, frequency or contextual diversity as discussed above:

$$tokenC(e, \mathbb{B}) = \sum_{e' \in \mathbb{B}} \text{\# occur. of } e' \textit{ in}\text{-context of } e$$

$$typeC(e, \mathbb{B}) = \sum_{e' \in \mathbb{B}} \text{\# occur. of } e' \textit{ in}\text{-context of } e > 0$$

2.2 The Bootstrapping Algorithm

The proposed bootstrapping algorithm in Algorithm 1 generates two complementary sets during the bootstrapping process, both of which justify proper contexts for each other. As the two sets updated during bootstrapping, the diversity measurement of the other set is expected to be more and more accurate. This strategy reflects the intuition behind co-training (Abney, 2004), but in a greatly reduced form and requiring no seeds for initialization.

Given a specific form of elements (bound or free morphemes), inputs to this algorithm are a dataset $\mathbb{S}$[4] containing all the elements of this form in some corpus and a choice of diversity measurement, $tokenC$ or $typeC$ as defined in section 2.1.

Assuming that functional elements can be distinguished by higher diversity degree, as discussed in section 2.1, we explicitly let a set $\mathbb{F}$ contain the most diverse elements as computed by the diversity measurement function with respect to its complementary set $\mathbb{B}$; and at each bootstrapping iteration, we increase the size of $\mathbb{F}$ by one.

On the other hand, set $\mathbb{B}$, which is generated to provide proper contexts for $\mathbb{F}$, contain those elements of a diversity degree greater than one with respect to $\mathbb{F}$, implementing our understanding of properly justified contexts in section 2.1. Since the order of diversity ranking of elements in $\mathbb{S}$ varies over iterations with respect to $\mathbb{B}$, which is also updated at each iteration according to current $\mathbb{F}$, an element classified into $\mathbb{F}$ at some iteration is not guaranteed to show up in $\mathbb{F}$ in the following iterations.

In addition to the update of $\mathbb{F}$ and $\mathbb{B}$, we also introduce a special 'filtering' step to prevent those elements that are ever seen *as/in*-context of elements in $\mathbb{F}$ from being classified into $\mathbb{F}$. This filtering idea is implemented through a set $\mathbb{R}$ containing elements to be excluded from $\mathbb{F}$, which is also updated at each iteration after the update of $\mathbb{F}$. This filtering step guarantees that there is no element in $\mathbb{F}$ ever occurring *as/in*-context of any other element in $\mathbb{F}$. We will explain more about the linguistic motivation behind this filtering step in section 4.

[4]For each element e in $\mathbb{S}$, the number of its occurrences *in/as*-context of other elements are also provided.

2.3 The Acquisition Output

We run the bootstrapping algorithm introduced in Section 2.2 for two acquisition tasks: acquiring functional morphemes and acquiring 'first-order' functional words. The outputs of both tasks, which are done separately, are depicted in Table 1 and 2 respectively. For each task, we experiment with two options of diversity measurement: $tokenC$ or $typeC$ as defined in section 2.1. If any element classified to be functional is of a diversity degree lower than a threshold, the bootstrapping process stops.

First of all, as clearly shown by the quality of acquisition outputs, this algorithm is sensitive to the choice of diversity measurement, and $typeC$, which measures the contextual diversity of an element in proper contexts, is always a better option compared to $tokenC$, which measures the frequency of an element in proper contexts. *tokenC* produces better outputs in the sense that fewer noisy elements are acquired and if acquired, they are acquired later in the bootstrapping process.

In both Table 1 and 2, functional elements are shown ordered by diversity in the second column, which order varies over the bootstrapping iterations, corresponding to the update of its complementary set, which is shown at the last column.

For the case of acquiring words, it is worth wondering about why, at the first few iterations, there were only determiners generated as functional, but at later iterations, modal verbs also show up. This fact seems to contradict our intuition that modal verbs are not of high diversity with respect to a complementary set containing nominal words only, which are generated according to determiners. However, we know that, at least in English, words are of ambiguous categories; therefore, those words of both nominal and verbal senses, such as *'consider'* and *'help'*, are classified to the complementary set by determiners according to their nominal sense, but their existence in the complementary set also enlarges the diversity of modal verbs due to their verbal sense.

While measuring the diversity by $typeC$, the

Algorithm 1 The bootstrapping algorithm for acquiring functional elements

```
Require: a data set $\mathbb{S}$ to be classified and a diversity measurement function $C$
  initialize set $\mathbb{F}$ and set $\mathbb{R}$ to be empty and initialize set $\mathbb{B}$ to be $\mathbb{S}$
  for $k$ iterations do
    let $\mathbb{F}$ be the top $k$ most diverse elements with respect to $C(e, \mathbb{B})$, for $e$ in $\mathbb{S} - \mathbb{R}$
    let $\mathbb{B}$ be those elements with a diversity greater than one, i.e. $C(e, \mathbb{F}) > 0$, for $e$ in $\mathbb{S}$
    let $\mathbb{R}$ be those elements ever as/in-context of any element in $\mathbb{F}$
  end for
  return $\mathbb{F}$ and $\mathbb{B}$
```

Measure the diversity by $typeC$		
*k*th iter.	the smaller set	the bigger set
5th	*the a its their his*	[2697] *protest, code, hats...*
10th	*the a its their his some this any no an*	[3203] *emerging, results, seemed...*
15th	*the a its their his some this an any no will these our another those*	[3525] *help, consider, follow...*
26th	*the a its their his some will an any would can could these those our another may her several* ***york*** *my might each whose your every*	[3653] *all, settlements, just, four, help, dollar, teaching, soon...*
Measure the diversity by $tokenC$		
16th	*the a its will an would their ... could may any* ***york*** *these* ***according***	[4228] *concept, consider, all...*

Table 1: The acquisition outputs at each iteration of the bootstrapping algorithm running for words.

Measure the diversity by $typeC$		
*k*th iter.	the smaller set	the bigger set
5th	*-$, -s, -ing, -ed, -e*	[2616] *degree-, cook-, topp-, excit-...*
10th	*-$, -s, -ing, -ed, -e, -er, -es, -ion, -ers, -ly*	[3343] *comply-, drink-, opt-, devot-, ...*
15th	*-$, -s, -ing, -ed, -e, -er, -ly, -ion, -es, -ers, -al, -y, -or, -ive, -ity*	[3496] *poor-, recept-, arriv-, mot- ...*
26th	*-$, -s, -ing, -ed, -e, -er, -ly, -ion, -es, -y, -ers, -al, -ies, -or, -ive, -ity, -ist, -man, -ic, -est, -on, -en, -ism, -ors, -ant, -ial*	[3772] *shell-, deliver-, comparabl-, juni- produc-, buri-, specifi-, impress-, good-...*
Measure the diversity by $tokenC$		
15th	*-$, -s, -e, -ed,* ***-t****, -ing, -ion, -y, -er, -al,* ***-n****, -ly, -ic, -or, -th*	[9576] *diploma-, conduc-, begin-, leg-...*

Table 2: The acquisition outputs at each iteration of the bootstrapping algorithm running for morphemes.

notable noisy item for acquiring functional words is *'york'*, which is classified as functional because 1) we lowercase all texts and 2) in WSJ articles, there are a lot complex NPs composed of *'York'*, i.e *'York'* is seen in many occurrences followed by nominal elements that may have already been classified as contentful during the bootstrapping process. On the other hand, with the diversity measurement $tokenC$, *'York'* has showed up as noisy much earlier in the bootstrapping process, as well as other noisy items.

It is even clearer in table 2 that the contextual diversity is a better measurement to distinguish functional elements than frequency. While measuring the diversity by $tokenC$, there are two noisy items show up as early as 15th iteration, *'-t'* and *'-n'*, which are successfully excluded from the output with diversity measurement $typeC$. While measuring the diversity by $typeC$, a possibly noisy item is *'-ers'*, which can be decomposed into *'-er'* and *'-s'*.

3 Applications

As shown above, the proposed bootstrapping algorithm generates outputs of high accuracy for both acquisition tasks: acquiring productive endings and acquiring 'first-order' functional words. In addition to the functional elements, this algorithm also generates complementary sets of contentful elements, which are base stems of words when acquiring morphemes and nouns plus verbs when acquiring words. Given these acquisition outputs, we can immediately accomplish the following two tasks, using very straightforward strategies only: 1) building a morph analyzer (section 3.1) ; and 2) acquiring the full set of functional categories, not just the first-order ones (section 3.2). Finally, we can plug the acquired list of closed-class words into a minimally supervised tagging system, (Zhao and Marcus, 2009), which requires the input of such a lexicon only. The resulting tagging system is then totally unsupervised and performs with satisfying accuracy as a totally unsupervised POS tagger for open-class words (section 3.3). The bootstrapping algorithm runs over the WSJ Treebank, which contains 1173766 words, with raw text input only for both tasks. And all experiments described below are trained over the same corpus.

3.1 Morphological Analysis

Those syntactic related morphological features, such as *'ended with -ed'* or *'ended with -ing'* have already been proven useful in syntactic related tasks such as POS tagging, parsing and so on; therefore, accomplishing such a morph analysis in an unsupervised way is meaningful for related unsupervised works.

Given an acquired set $\mathbb{B}$ of base stems, e.g. $\mathbb{B}$ = { *'laugh-'*, *'analyz-'* and *'-define'*}, and a set $\mathbb{F}$ of functional morphemes , e.g. $\mathbb{F}$ = {*'-ed'*, *'-s'* and *'pre-'*}, we can divide a word w into morphemes b- and -e where $b + e == w$, if b- in $\mathbb{B}$ and -e in $\mathbb{F}$; or if vice verse, i.e. b- in $\mathbb{F}$ and -e in $\mathbb{B}$. If such a division $b+e$ can be successfully performed on the word w with -e in $\mathbb{F}$, we specify that w has a morph feature *'ended with -e'*. For instance, given the above examples of $\mathbb{B}$ and $\mathbb{F}$, *'laughed'* has an ending *'-ed'* but not ending *'-ing'*, and *'analyzed'* has an ending *'-ed'*, but *'red'* doesn't have such an ending *'-ed'* as its morph feature. Overall, given two lists of morphemes, we can build a morph analyzer implementing this constrained stemming strategy for detecting morph features of interest.

For evaluating the morph analyzer composed by our acquisition outputs from WSJ Treebank, we use the CELEX corpus (Baayen et al., 1996) providing gold annotations of 6 syntactic related morphological features: {*'-s'*, *'-es'*, *'-ed'*, *'-ing'*, *'-er'*, and *'-est'*}. Among the 20058 types of words seen in both WSJ and CELEX corpus, 8789 types are annotated with these features. The precision number reported in Table 3 is the percentage of correct predictions out of all predictions the analyzer makes about the 6 morph features; and the recall is calculated by the percentage of correctly-predicted featured words out of all types of words with these features (8789).

We report the performance of two morph analyzers: one is composed by the acquired outputs with the diversity measurement $tokenC$; and the other one acquired with $typeC$. We also compare our results against a state-of-art morphology analyzer, (Lignos, 2010), which is adapted for this experiment. As shown in Table 3, our general bootstrapping algorithm, which is designed for various applications, is able to generate useful outputs for building a morph analyzer that performs comparably well as the stat-of-art achievements.

Algorithm	Precision	Recall	F-score
bootstrap-$tokenC$	77.89	82.38	80.07
bootstrap-$typeC$	78.71	83.31	80.94
(Lignos, 2010)	80.16	84.37	82.21

Table 3: The performance of morph analyzers evaluated for syntactic related morphological features.

3.2 Acquiring Closed-class Words

The bootstrapping algorithm is not designed to generate closed-class words falling in all functional categories, instead, it acquires those 'first-order' functional words, including only determiners and modal verbs. Given this acquired subset of closed-class words and the morph analyzer built in section 3.1, it is only a step away from acquiring all closed-class words falling in various functional categories.

We are not aware of previous work that acquires closed-class words in an automatic process, and in the NLP field, they are often approximated by the most frequent words. In other words, as a baseline model, we can acquire closed-class words by selecting k top words of the highest diversity measured by $tokenC$. Given discussion of diversity measurement in section 2.1, we also experiment with another baseline model, which acquires closed-class words by selecting k top words of the highest diversity measured by $typeC$. As shown in Table 4, with a better approximation of diversity, the type accuracy of predicting closed-class words is improved by an absolute error reduction of 11%, from 63% to 74% (k=100).

Moreover, as discussed in section 2, we know that those 'first-order' functional words, which can be generated by the bootstrapping algorithm with high accuracy, don't project onto other functional words. Therefore, after running the bootstrapping algorithm for words, we have already obtained a list of words that can be used for filtering the output of the baseline models. More specifically, given a set of 'first-order' functional words, those words that occur relatively often *in*-context of them should not be output as closed-class words. For example, two common noisy

top k *words*	20	40	80	100
baseline-$tokenC$	90.00	80.00	68.75	63.00
baseline-$typeC$	100.0	90.00	76.25	74.00
bootstrap-$tokenC$	95.00	95.00	90.00	86.00
bootstrap-$typeC$	**100.0**	**97.50**	**93.75**	**89.00**

Table 4: The percentage of correctly predicted closed-class word types out of k predictions

items in the output of the baseline models, *'billion'* and *'say'*, can be successfully filtered by the output set shown in Table 1.

In addition, since we have already built an unsupervised morph analyzer as described in section 3.1, which is acquired by the same bootstrapping algorithm, we would like to make use of them as well, in a straightforward way: as long as a word can be analyzed by the acquired morph analyzer, it should not be output as closed-class items.

Models 'bootstrap-$tokenC$' and 'bootstrap-$typeC$' implement the above idea using diversity measurement $tokenC$ and $typeC$ respectively. For example, the model 'bootstrap-$typeC$' uses the diversity measurement $typeC$ to both acquire the morph analyzer and to acquire the set of 'first-order' functional words, and it uses these acquisition outputs to filter the baseline model that uses $typeC$ as diversity measurement as well. The gold lexicon we use to evaluate the acquired list contains 288 closed-class words as described in (Zhao and Marcus, 2009).

With the help of acquisition outputs by a single algorithm running for different tasks, we can significantly improve the baseline model for acquiring closed-class words. As shown in Table 4, the performance of the baseline models drops rapidly as the size k grows; however, by taking acquired categories as constraints, we achieve quite reliable models. For example, 'bootstrap-$typeC$' improves 'baseline-$tokenC$' by an absolute error reduction of 26% (k=100), from 63% to 89%. So as to show how the models perform as k keeps growing, we provide more results in Figure 1, which plots the percentage of correctly predicted closed-class word types out of k predictions.

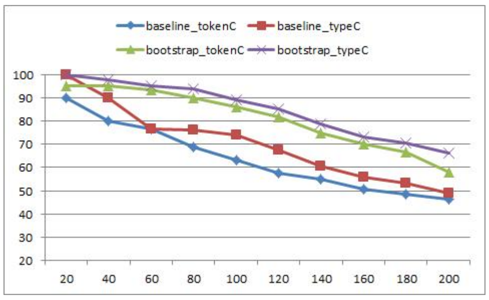

Figure 1: The percentage of correctly predicted closed-class word types out of k predictions

3.3 Unsupervised POS Tagging

Finally, to demonstrate the value of such a highly pure list of closed-class words, which previously could not be acquired through unsupervised learning, we plug the acquired list into a previously existing minimally supervised tagging system, (Zhao and Marcus, 2009), which only requires a closed-class lexicon for tagging all words. The resulting system now provides a totally unsupervised system for tagging open-class words, inducing both the categorical information itself and also the disambiguation rules for tagging open-class words.

Following (Zhao and Marcus, 2009), the six open-class POS categories to be predicted are verbs, nouns, past participles, present participles and numbers. The tagging predictions for open-class words (any word not in the closed-class list) are evaluated by the percentage of correct predictions on all tokens in the open-class set. Although all tagging experiments are done over the whole WSJ Treebank, the total number of tagging predictions may vary according to different input closed-class lists.

We experiment with four lists of closed-class words of the top 200 words acquired by the 'baseline-$typeC$', 'baseline-$tokenC$', 'bootstrap-$typeC$' and 'bootstrap-$tokenC$' respectively, as introduced in section 3.2. We evaluate against a gold-standard closed-class lexicon containing 288 words, also used for evaluation in section 3.2.

closed-class input	accuracy	totally tagged
baseline-$tokenC$	81.68	536535
baseline-$typeC$	75.51	554012
bootstrap-$tokenC$	77.30	627803
bootstrap-$typeC$	**85.26**	612997
gold lexicon	*91.03*	*611028*

Table 5: The percentage of correctly tagged tokens out of all predictions. The system tags open-class words only and distinguishes 6 POS categories.

As shown in Table 5, the tagging system using the closed-class set acquired by 'bootstrap-$typeC$' tags 612997 open-class tokens, which accounts for more than half of all the tokens in the WSJ Treebank, with an accuracy of 85.26%. In recent work on unsupervised tagging, tagging accuracys are reported in the range of 50-70%, e.g. (Abend et al., 2010), (Reichart et al., 2010), (Moon et al., 2010) and so on. Compared to these results, the tagging performance reported here, even though only on open-class words for six categories, is quite promising.

Because most previous work distinguishes between more categories than we do here, our results may be misleading, in that we appear to be reporting on a simpler task. However, these previous systems operate by clustering over a vector space of lexical features and suffix/prefix features, resulting in a large number of scattered clusters with similarities on different semantic and syntactic dimensions. As a result, it has proven difficult for them to use further agglomerative processing to induce simple distinct syntactic categories which map to POS tags naturally (Christodoulopoulos et al., 2010), and thus operating in a mode which achieves a higher tagging accuracy with coarser categories is not available to those systems at all. Therefore, achieving high accuracy with a smaller tag set is the harder, not easier, task for those systems.

4 Discussion

We propose a bootstrapping algorithm which successfully resolves two fundamental acquisition tasks: acquiring functional morphemes and acquiring 'first-order' functional words. We have shown that the outputs of these two fundamental acquisition tasks are very useful for more generalized tasks: they can be directly used to built a nearly state-of-art morph analyzer and they can be used to acquire a full set of closed-class words with high accuracy. Furthermore, the acquired list of closed-class words allows us to turn a minimally supervised tagging system into a totally unsupervised tagger for tagging open-class words. As a completely unsupervised tagger, the resulting system performs at a satisfying tagging accuracy above 85%.

The bootstrapping algorithm proposed here also gives us cause to think about the connection between functional categories in two forms: bound morphemes and words. As shown in this work, identically the same computational process that acquires functional morphemes also can be used to acquire the subset of functional words that includes modal verbs and determiners. All these elements share the property that, from the point of view of modern generative grammar, they only project locally, in other words, the elements that are acquired by this algorithm project to contentful elements *directly* most of the time but not through any other functional elements. Since all productive bound morphemes project mainly locally, they can all be acquired by the algorithm; in contrast, those functional words that project in larger scopes quite often, rather than in local contexts (i.e. not determiners and modal verbs), will not be acquired by this algorithm directly. This understanding of viewing local contexts as first-order functional projections motivates the filtering step in the bootstrapping algorithm. This overall understanding was motivated for us by the strict locality constraints assumed in phase theory (Chomsky, 2006).

References

Omri Abend, Roi Reichart, and Ari Rappoport. 2010. Improved unsupervised pos induction through prototype discovery. In *ACL*.

S. Abney. 2004. Understanding the yarowsky algorithm. *Computational Linguistics*, 30(3):365–395.

R.H. Baayen, R. Piepenbrock, and L. Gulikers. 1996. Celex2. Linguistic Data Consortium, Philadelphia.

E. Chan. 2008. *Structures and distributions in morphology learning*. Ph.D. thesis, University of Pennsylvania.

N. Chomsky. 2006. Approaching UG from below.

C. Christodoulopoulos, S. Goldwater, and M. Steedman. 2010. Two decades of unsupervised pos induction: How far have we come? In *EMNLP*.

A. Clark. 2000. Inducing syntactic categories by context distribution clustering. In *Proceedings of the Computational Natural Language Learning and the Second Learning Language in Logic Workshop (CoNLL-LLL)*.

C. Lignos. 2010. Learning from Unseen Data. In *Proceedings of Morpho Challenge 2010*, pages 35–38, Helsinki, Finland, September 2–3. Aalto School of Technology.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

T. Moon, K.Erk, and J. Baldridge. 2010. Crouching dirichlet, hidden markov model: unsupervised pos tagging with context local tag generation. In *EMNLP*.

R. Reichart, R. Fattal, and A.Rappoport. 2010. Improved unsupervised pos induction using intrinsic clustering quality and a zipfian constraint. In *CoNLL*.

V. Valian, S. Solt, and J. Stewart. 2009. Abstract categories or limited scope formulae: The case of children's determiners. *Journal of Child Language*, 36:743–778.

C. Yang. 2002. *"Knowledge and Learning in Natural Language"*. Oxford University Press.

Q. Zhao and M. Marcus. 2009. A simple unsupervised learner for pos disambiguation rules given only a minimal lexicon. In *EMNLP*.