Minimized models and grammar-informed initialization for supertagging with highly ambiguous lexicons

Sujith Ravi[1] **Jason Baldridge**[2] **Kevin Knight**[1]

[1]University of Southern California
Information Sciences Institute
Marina del Rey, California 90292
{sravi,knight}@isi.edu

[2]Department of Linguistics
The University of Texas at Austin
Austin, Texas 78712
jbaldrid@mail.utexas.edu

Abstract

We combine two complementary ideas for learning supertaggers from highly ambiguous lexicons: grammar-informed tag transitions and models minimized via integer programming. Each strategy on its own greatly improves performance over basic expectation-maximization training with a bitag Hidden Markov Model, which we show on the CCGbank and CCG-TUT corpora. The strategies provide further error reductions when combined. We describe a new two-stage integer programming strategy that efficiently deals with the high degree of ambiguity on these datasets while obtaining the full effect of model minimization.

1 Introduction

Creating accurate part-of-speech (POS) taggers using a tag dictionary and unlabeled data is an interesting task with practical applications. It has been explored at length in the literature since Merialdo (1994), though the task setting as usually defined in such experiments is somewhat artificial since the tag dictionaries are derived from tagged corpora. Nonetheless, the methods proposed apply to realistic scenarios in which one has an electronic part-of-speech tag dictionary or a hand-crafted grammar with limited coverage.

Most work has focused on POS-tagging for English using the Penn Treebank (Marcus et al., 1993), such as (Banko and Moore, 2004; Goldwater and Griffiths, 2007; Toutanova and Johnson, 2008; Goldberg et al., 2008; Ravi and Knight, 2009). This generally involves working with the standard set of 45 POS-tags employed in the Penn Treebank. The most ambiguous word has 7 different POS tags associated with it. Most methods have employed some variant of Expectation Maximization (EM) to learn parameters for a bigram or trigram Hidden Markov Model (HMM). Ravi and Knight (2009) achieved the best results thus far (92.3% word token accuracy) via a Minimum Description Length approach using an integer program (IP) that finds a minimal bigram grammar that obeys the tag dictionary constraints and covers the observed data.

A more challenging task is learning supertaggers for lexicalized grammar formalisms such as Combinatory Categorial Grammar (CCG) (Steedman, 2000). For example, CCGbank (Hockenmaier and Steedman, 2007) contains 1241 distinct supertags (lexical categories) and the most ambiguous word has 126 supertags. This provides a much more challenging starting point for the semi-supervised methods typically applied to the task. Yet, this is an important task since creating grammars and resources for CCG parsers for new domains and languages is highly labor- and knowledge-intensive. Baldridge (2008) uses grammar-informed initialization for HMM tag transitions based on the universal combinatory rules of the CCG formalism to obtain 56.1% accuracy on ambiguous word tokens, a large improvement over the 33.0% accuracy obtained with uniform initialization for tag transitions.

The strategies employed in Ravi and Knight (2009) and Baldridge (2008) are complementary. The former reduces the model size globally given a data set, while the latter biases bitag transitions toward those which are more likely based on a universal grammar without reference to any data. In this paper, we show how these strategies may be combined straightforwardly to produce improvements on the task of learning supertaggers from lexicons that have not been filtered in any way.[1] We demonstrate their cross-lingual effectiveness on CCGbank (English) and the Italian CCG-TUT

[1]See Banko and Moore (2004) for a description of how many early POS-tagging papers in fact used a number of heuristic cutoffs that greatly simplify the problem.

corpus (Bos et al., 2009). We find a consistent improved performance by using each of the methods compared to basic EM, and further improvements by using them in combination.

Applying the approach of Ravi and Knight (2009) naively to CCG supertagging is intractable due to the high level of ambiguity. We deal with this by defining a new two-stage integer programming formulation that identifies minimal grammars efficiently and effectively.

2 Data

CCGbank. CCGbank was created by semi-automatically converting the Penn Treebank to CCG derivations (Hockenmaier and Steedman, 2007). We use the standard splits of the data used in semi-supervised tagging experiments (e.g. Banko and Moore (2004)): sections 0-18 for training, 19-21 for development, and 22-24 for test.

CCG-TUT. CCG-TUT was created by semi-automatically converting dependencies in the Italian Turin University Treebank to CCG derivations (Bos et al., 2009). It is much smaller than CCGbank, with only 1837 sentences. It is split into three sections: newspaper texts (NPAPER), civil code texts (CIVIL), and European law texts from the JRC-Acquis Multilingual Parallel Corpus (JRC). For test sets, we use the first 400 sentences of NPAPER, the first 400 of CIVIL, and all of JRC. This leaves 409 and 498 sentences from NPAPER and CIVIL, respectively, for training (to acquire a lexicon and run EM). For evaluation, we use two different settings of train/test splits:

TEST 1 Evaluate on the NPAPER section of *test* using a lexicon extracted only from NPAPER section of *train*.

TEST 2 Evaluate on the entire *test* using lexicons extracted from (a) NPAPER + CIVIL, (b) NPAPER, and (c) CIVIL.

Table 1 shows statistics for supertag ambiguity in CCGbank and CCG-TUT. As a comparison, the POS word token ambiguity in CCGbank is 2.2: the corresponding value of 18.71 for supertags is indicative of the (challenging) fact that supertag ambiguity is greatest for the most frequent words.

3 Grammar informed initialization for supertagging

Part-of-speech tags are atomic labels that in and of themselves encode no internal structure. In contrast, supertags are detailed, structured labels; a universal set of grammatical rules defines how categories may combine with one another to project syntactic structure.[2] Because of this, properties of the CCG formalism itself can be used to constrain learning—prior to considering any particular language, grammar or data set. Baldridge (2008) uses this observation to create grammar-informed tag transitions for a bitag HMM supertagger based on two main properties. First, categories differ in their complexity and less complex categories tend to be used more frequently. For example, two categories for *buy* in CCGbank are (S[dcl]\NP)/NP and ((((S[b]\NP)/PP)/PP)/(S[adj]\NP))/NP; the former occurs 33 times, the latter once. Second, categories indicate the form of categories found adjacent to them; for example, the category for sentential complement verbs ((S\NP)/S) expects an NP to its left and an S to its right.

Data	**Distinct**	**Max**	**Type ambig**	**Tok ambig**
CCGbank	1241	126	1.69	18.71
CCG-TUT				
NPAPER+CIVIL	849	64	1.48	11.76
NPAPER	644	48	1.42	12.17
CIVIL	486	39	1.52	11.33

Table 1: Statistics for the *training* data used to extract lexicons for CCGbank and CCG-TUT. **Distinct**: # of distinct lexical categories; **Max**: # of categories for the most ambiguous word; **Type ambig**: per word type category ambiguity; **Tok ambig**: per word token category ambiguity.

Categories combine via rules such as application and composition (see Steedman (2000) for details). Given a lexicon containing the categories for each word, these allow derivations like:

$$\begin{array}{ccccc}
\text{Ed} & \text{might} & \text{see} & \text{a} & \text{cat}\\
\overline{NP} & \overline{(S\backslash NP)/(S\backslash NP)} & \overline{(S\backslash NP)/NP} & \overline{NP/N} & \overline{N}\\
 & \multicolumn{2}{c}{\overline{\hspace{5cm}}{>}\mathbf{B}} & \multicolumn{2}{c}{\overline{\hspace{2cm}}{>}}\\
 & \multicolumn{2}{c}{(S\backslash NP)/NP} & \multicolumn{2}{c}{NP}\\
 & \multicolumn{4}{c}{\overline{\hspace{8cm}}{>}}\\
 & \multicolumn{4}{c}{S\backslash NP}\\
\multicolumn{5}{c}{\overline{\hspace{6cm}}{>}}\\
\multicolumn{5}{c}{S}
\end{array}$$

Other derivations are possible. In fact, every pair of adjacent words above may be combined directly. For example, *see* and *a* may combine through forward composition to produce the category (S\NP)/N, and *Ed*'s category may type-raise to S/(S\NP) and compose with *might*'s category.

Baldridge uses these properties to define tag

[2]Note that supertags can be lexical categories of CCG (Steedman, 2000), elementary trees of Tree-adjoining Grammar (Joshi, 1988), or types in a feature hierarchy as in Head-driven Phrase Structure Grammar (Pollard and Sag, 1994).

transition distributions that have higher likelihood for simpler categories that are able to combine. For example, for the distribution $p(t_i|t_{i-1}{=}NP)$, (S\NP)\NP is more likely than ((S\NP)/(N/N))\NP because both categories may combine with a preceding NP but the former is simpler. In turn, the latter is more likely than NP: it is more complex but can combine with the preceding NP. Finally, NP is more likely than (S/NP)/NP since neither can combine, but NP is simpler.

By starting EM with these tag transition distributions and an unfiltered lexicon (word-to-supertag dictionary), Baldridge obtains a tagging accuracy of 56.1% on ambiguous words—a large improvement over the accuracy of 33.0% obtained by starting with uniform transition distributions. We refer to a model learned from basic EM (uniformly initialized) as EM, and to a model with grammar-informed initialization as EM$_{GI}$.

4 Minimized models for supertagging

The idea of searching for minimized models is related to classic Minimum Description Length (MDL) (Barron et al., 1998), which seeks to select a small model that captures the most regularity in the observed data. This modeling strategy has been shown to produce good results for many natural language tasks (Goldsmith, 2001; Creutz and Lagus, 2002; Ravi and Knight, 2009). For tagging, the idea has been implemented using Bayesian models with priors that indirectly induce sparsity in the learned models (Goldwater and Griffiths, 2007); however, Ravi and Knight (2009) show a better approach is to directly minimize the model using an integer programming (IP) formulation. Here, we build on this idea for supertagging.

There are many challenges involved in using IP minimization for supertagging. The 1241 distinct supertags in the tagset result in 1.5 million tag bigram entries in the model and the dictionary contains almost 3.5 million word/tag pairs that are relevant to the test data. The set of 45 POS tags for the same data yields 2025 tag bigrams and 8910 dictionary entries. We also wish to scale our methods to larger data settings than the 24k word tokens in the test data used in the POS tagging task.

Our objective is to find the smallest supertag grammar (of tag bigram types) that explains the entire text while obeying the lexicon's constraints. However, the original IP method of Ravi and Knight (2009) is intractable for supertagging, so we propose a new two-stage method that scales to the larger tagsets and data involved.

4.1 IP method for supertagging

Our goal for supertagging is to build a minimized model with the following objective:

> $\underline{IP_{original}}$: Find the smallest supertag grammar (i.e., tag bigrams) that can explain the entire text (the test word token sequence).

Using the full grammar and lexicon to perform model minimization results in a very large, difficult to solve integer program involving billions of variables and constraints. This renders the minimization objective $IP_{original}$ intractable. One way of combating this is to use a reduced grammar and lexicon as input to the integer program. We do this without further supervision by using the HMM model trained using basic EM: entries are pruned based on the tag sequence it predicts on the test data. This produces an *observed* grammar of distinct tag bigrams (G_{obs}) and lexicon of observed lexical assignments (L_{obs}). For CCGbank, G_{obs} and L_{obs} have 12,363 and 18,869 entries, respectively—far less than the millions of entries in the full grammar and lexicon.

Even though EM minimizes the model somewhat, many bad entries remain in the grammar. We prune further by supplying G_{obs} and L_{obs} as input (G, L) to the IP-minimization procedure. However, even with the EM-reduced grammar and lexicon, the IP-minimization is still very hard to solve. We thus split it into two stages. The first stage (Minimization 1) finds the smallest grammar $G_{min1} \subset G$ that explains the set of word bigram *types* observed in the data rather than the word sequence itself, and the second (Minimization 2) finds the smallest augmentation of G_{min1} that explains the full word sequence.

Minimization 1 (MIN1). We begin with a simpler minimization problem than the original one ($IP_{original}$), with the following objective:

> $\underline{IP_{\mathrm{min\,1}}}$: Find the smallest set of tag bigrams $G_{min1} \subset G$, such that there is at least one tagging assignment possible for every word bigram type observed in the data.

We formulate this as an integer program, creating binary variables $gvar_i$ for every tag bigram $g_i = t_j t_k$ in G. Binary link variables connect tag bigrams with word bigrams; these are restricted

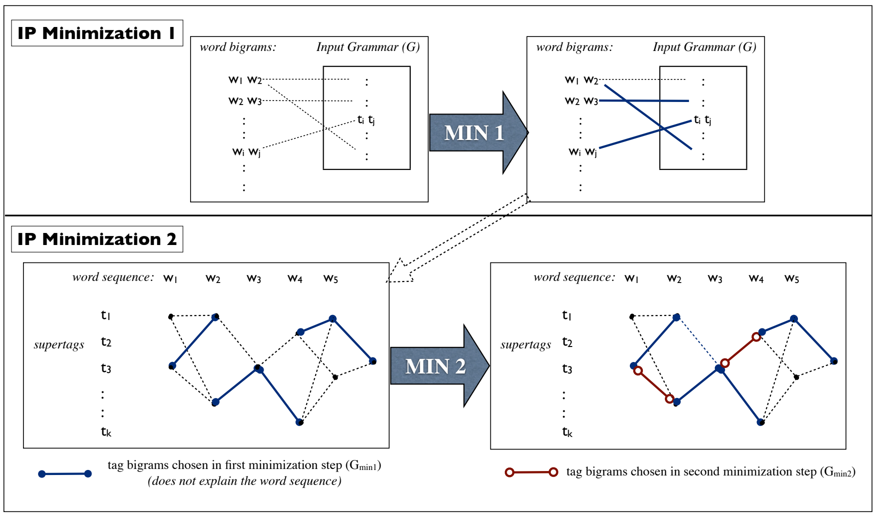
IP Minimization 1
word bigrams:
Input Grammar (G)
w1 w2
w2 w3
ti tj
wi wj
MIN 1
IP Minimization 2
word sequence: w1 w2 w3 w4 w5
supertags
t1
t2
t3
tk
MIN 2
tag bigrams chosen in first minimization step (Gmin1)
(does not explain the word sequence)
tag bigrams chosen in second minimization step (Gmin2)

Figure 1: Two-stage IP method for selecting minimized models for supertagging.

to the set of links that respect the lexicon L provided as input, i.e., there exists a link variable $link_{jklm}$ connecting tag bigram $t_j t_k$ with word bigram $w_l w_m$ only if the word/tag pairs (w_l, t_j) and (w_m, t_k) are present in L. The entire integer programming formulation is shown Figure 2.

The IP solver[3] solves the above integer program and we extract the set of tag bigrams G_{min1} based on the activated grammar variables. For the CCG-bank test data, MIN1 yields 2530 tag bigrams. However, a second stage is needed since there is no guarantee that G_{min1} can explain the test data: it contains tags for all word bigram types, but it cannot necessarily tag the full word sequence. Figure 1 illustrates this. Using only tag bigrams from MIN1 (shown in blue), there is no fully-linked tag path through the network. There are missing links between words w_2 and w_3 and between words w_3 and w_4 in the word sequence. The next stage fills in these missing links.

Minimization 2 (MIN2). This stage uses the original minimization formulation for the supertagging problem $IP_{original}$, again using an integer programming method similar to that proposed by Ravi and Knight (2009). If applied to the observed grammar G_{obs}, the resulting integer program is hard to solve.[4] However, by using the partial solution G_{min1} obtained in MIN1 the IP optimization speeds up considerably. We implement this by fixing the values of all binary grammar variables present in G_{min1} to 1 before optimization. This reduces the search space significantly, and CPLEX finishes in just a few hours. The details of this method are described below.

[3]We use the commercial CPLEX solver.

[4]The solver runs for days without returning a solution.

Minimize: $\sum_{\forall g_i \in G} gvar_i$

Subject to constraints:

1. For every word bigram $w_l w_m$, there exists at least one tagging that respects the lexicon L.

$$\sum_{\forall\ t_j \in L(w_l),\ t_k \in L(w_m)} link_{jklm} \geq 1$$

where $L(w_l)$ and $L(w_m)$ represent the set of tags seen in the lexicon for words w_l and w_m respectively.

2. The link variable assignments are constrained to respect the grammar variables chosen by the integer program.

$$link_{jklm} \leq gvar_i$$

where $gvar_i$ is the binary variable corresponding to tag bigram $t_j t_k$ in the grammar G.

Figure 2: IP formulation for Minimization 1.

We instantiate binary variables $gvar_i$ and $lvar_i$ for every tag bigram (in G) and lexicon entry (in L). We then create a network of possible taggings for the word token sequence $w_1 w_2 .... w_n$ in the corpus and assign a binary variable to each link in the network. We name these variables $link_{cjk}$, where c indicates the column of the link's source in the network, and j and k represent the link's source and destination (i.e., $link_{cjk}$ corresponds to tag bigram $t_j t_k$ in column c). Next, we formulate the integer program given in Figure 3.

Figure 1 illustrates how MIN2 augments the grammar G_{min1} (links shown in blue) with addi-

Minimize: $\sum_{\forall g_i \in G} gvar_i$

Subject to constraints:

1. Chosen link variables form a left-to-right path through the tagging network.

$$\forall_{c=1..n-2} \forall_k \sum_j link_{cjk} = \sum_j link_{(c+1)kj}$$

2. Link variable assignments should respect the chosen grammar variables.

for every link: $link_{cjk} \leq gvar_i$

where $gvar_i$ corresponds to tag bigram $t_j t_k$

3. Link variable assignments should respect the chosen lexicon variables.

for every link: $link_{cjk} \leq lvar_{w_c t_j}$

for every link: $link_{cjk} \leq lvar_{w_{c+1} t_k}$

where w_c is the c^{th} word in the word sequence $w_1 ... w_n$, and $lvar_{w_c t_j}$ is the binary variable corresponding to the word/tag pair w_c/t_j in the lexicon L.

4. The final solution should produce at least one complete tagging path through the network.

$$\sum_{\forall_{j,k}} link_{1jk} \geq 1$$

5. Provide minimized grammar from MIN1as partial solution to the integer program.

$$\forall_{g_i \in G_{min1}} gvar_i = 1$$

Figure 3: IP formulation for Minimization 2.

tional tag bigrams (shown in red) to form a complete tag path through the network. The minimized grammar set in the final solution G_{min2} contains only 2810 entries, significantly fewer than the original grammar G_{obs}'s 12,363 tag bigrams.

We note that the two-stage minimization procedure proposed here is not guaranteed to yield the optimal solution to our original objective $IP_{original}$. On the simpler task of unsupervised POS tagging with a dictionary, we compared our method versus *directly* solving $IP_{original}$ and found that the minimization (in terms of grammar size) achieved by our method is close to the optimal solution for the original objective and yields the same tagging accuracy far more efficiently.

Fitting the minimized model. The IP-minimization procedure gives us a minimal grammar, but does not fit the model to the data. In order to estimate probabilities for the HMM model for supertagging, we use the EM algorithm but with certain restrictions. We build the transition model using only entries from the minimized grammar set G_{min2}, and instantiate an emission model using the word/tag pairs seen in L (provided as input to the minimization procedure). All the parameters in the HMM model are initialized with uniform probabilities, and we run EM for 40 iterations. The trained model is used to find the Viterbi tag sequence for the corpus. We refer to this model (where the EM output (G_{obs}, L_{obs}) was provided to the IP-minimization as initial input) as **EM+IP**.

Bootstrapped minimization. The quality of the observed grammar and lexicon improves considerably at the end of a single EM+IP run. Ravi and Knight (2009) exploited this to iteratively improve their POS tag model: since the first minimization procedure is seeded with a noisy grammar and tag dictionary, iterating the IP procedure with progressively better grammars further improves the model. We do likewise, bootstrapping a new EM+IP run using as input, the observed grammar G_{obs} and lexicon L_{obs} from the last tagging output of the previous iteration. We run this until the chosen grammar set G_{min2} does not change.[5]

4.2 Minimization with grammar-informed initialization

There are two complementary ways to use grammar-informed initialization with the IP-minimization approach: (1) using EM$_{GI}$ output as the starting grammar/lexicon and (2) using the tag transitions directly in the IP objective function. The first takes advantage of the earlier observation that the quality of the grammar and lexicon provided as initial input to the minimization procedure can affect the quality of the final supertagging output. For the second, we modify the objective function used in the two IP-minimization steps to be:

$$\text{Minimize:} \sum_{\forall g_i \in G} w_i \cdot gvar_i \qquad (1)$$

where, G is the set of tag bigrams provided as input to IP, $gvar_i$ is a binary variable in the integer program corresponding to tag bigram $(t_{i-1}, t_i) \in G$, and w_i is negative logarithm of $p_{gii}(t_i|t_{i-1})$ as given by Baldridge (2008).[6] All other parts of

[5]In our experiments, we run three bootstrap iterations.

[6]Other numeric weights associated with the tag bigrams could be considered, such as 0/1 for uncombin-

the integer program including the constraints remain unchanged, and, we acquire a final tagger in the same manner as described in the previous section. In this way, we combine the minimization and GI strategies into a single objective function that finds a minimal grammar set while keeping the more likely tag bigrams in the chosen solution. **EM$_{GI}$+IP$_{GI}$** is used to refer to the method that uses GI information in both ways: EM$_{GI}$ output as the starting grammar/lexicon *and* GI weights in the IP-minimization objective.

5 Experiments

We compare the four strategies described in Sections 3 and 4, summarized below:

EM HMM uniformly initialized, EM training.
EM+IP IP minimization using initial grammar provided by EM.
EM$_{GI}$ HMM with grammar-informed initialization, EM training.
EM$_{GI}$+IP$_{GI}$ IP minimization using initial grammar/lexicon provided by EM$_{GI}$ and additional grammar-informed IP objective.

For EM+IP and EM$_{GI}$+IP$_{GI}$, the minimization and EM training processes are iterated until the resulting grammar and lexicon remain unchanged. Forty EM iterations are used for all cases.

We also include a baseline which randomly chooses a tag from those associated with each word in the lexicon, averaged over three runs.

Accuracy on ambiguous word tokens. We evaluate the performance in terms of tagging accuracy with respect to gold tags for ambiguous words in held-out test sets for English and Italian. We consider results with and without punctuation.[7]

Recall that unlike much previous work, we do not collect the lexicon (tag dictionary) from the test set: this means the model must handle unknown words and the possibility of having missing lexical entries for covering the test set.

Precision and recall of grammar and lexicon. In addition to accuracy, we measure precision and recall for each model on the observed bitag grammar and observed lexicon on the test set. We calculate them as follows, for an observed grammar or lexicon X:

$$Precision = \frac{|\{X\} \cap \{Observed_{gold}\}|}{|\{X\}|}$$

$$Recall = \frac{|\{X\} \cap \{Observed_{gold}\}|}{|\{Observed_{gold}\}|}$$

This provides a measure of model performance on bitag *types* for the grammar and lexical entry *types* for the lexicon, rather than tokens.

able/combinable bigrams.

[7]The reason for this is that the "categories" for punctuation in CCGbank are for the most part not actual categories; for example, the period "." has the categories "." and "S". As such, these supertags are outside of the categorial system: their use in derivations requires phrase structure rules that are not derivable from the CCG combinatory rules.

Model	**ambig**	**ambig -punc**	**all**	**all -punc**
Random	17.9	16.2	27.4	21.9
EM	38.7	35.6	45.6	39.8
EM+IP	52.1	51.0	57.3	53.9
EM$_{GI}$	56.3	59.4	61.0	61.7
EM$_{GI}$+IP$_{GI}$	**59.6**	**62.3**	**63.8**	**64.3**

Table 2: Supertagging accuracy for CCGbank sections 22-24. Accuracies are reported for four settings—(1) *ambiguous* word tokens in the test corpus, (2) ambiguous word tokens, ignoring punctuation, (3) *all* word tokens, and (4) all word tokens except punctuation.

5.1 English CCGbank results

Accuracy on ambiguous tokens. Table 2 gives performance on the CCGbank test sections. All models are well above the random baseline, and both of the strategies individually boost performance over basic EM by a large margin. For the models using GI, accuracy ignoring punctuation is higher than for all almost entirely due to the fact that "." has the supertags "." and S, and the GI gives a preference to S since it can in fact combine with other categories, unlike "."—the effect is that nearly every sentence-final period (˜5.5k tokens) is tagged S rather than ".".

EM$_{GI}$ is more effective than EM+IP; however, it should be kept in mind that IP-minimization is a general technique that can be applied to any sequence prediction task, whereas grammar-informed initialization may be used only with tasks in which the interactions of adjacent labels may be derived from the labels themselves. Interestingly, the gap between the two approaches is greater when punctuation is ignored (51.0 vs. 59.4)—this is unsurprising because, as noted already, punctuation supertags are not actual cate-

	EM	EM+IP	EM$_{GI}$	EM$_{GI}$+IP$_{GI}$
Grammar				
Precision	7.5	32.9	52.6	68.1
Recall	26.9	13.2	34.0	19.8
Lexicon				
Precision	58.4	63.0	78.0	80.6
Recall	50.9	56.0	71.5	67.6

Table 3: Comparison of grammar/lexicon observed in the *model tagging* vs. *gold tagging* in terms of precision and recall measures for supertagging on CCGbank data.

gories, so EM$_{GI}$ is unable to model their distribution. Most importantly, the complementary effects of the two approaches can be seen in the improved results for EM$_{GI}$+IP$_{GI}$, which obtains about 3% better accuracy than EM$_{GI}$.

Accuracy on all tokens. Table 2 also gives performance when taking all tokens into account. The HMM when using full supervision obtains 87.6% accuracy (Baldridge, 2008),[8] so the accuracy of 63.8% achieved by EM$_{GI}$+IP$_{GI}$ nearly halves the gap between the supervised model and the 45.6% obtained by basic EM semi-supervised model.

Effect of GI information in EM and/or IP-minimization stages. We can also consider the effect of GI information in *either* EM training *or* IP-minimization to see whether it can be effectively exploited in both. The latter, EM+IP$_{GI}$, obtains 53.2/51.1 for all/no-punc—a small gain compared to EM+IP's 52.1/51.0. The former, EM$_{GI}$+IP, obtains 58.9/61.6—a much larger gain. Thus, the better starting point provided by EM$_{GI}$ has more impact than the integer program that includes GI in its objective function. However, we note that it should be possible to exploit the GI information more effectively in the integer program than we have here. Also, our best model, EM$_{GI}$+IP$_{GI}$, uses GI information in both stages to obtain our best accuracy of 59.6/62.3.

P/R for grammars and lexicons. We can obtain a more-fine grained understanding of how the models differ by considering the precision and recall values for the grammars and lexicons of the different models, given in Table 3. The basic EM model has very low precision for the grammar, indicating it proposes many unnecessary bitags; it achieves better recall because of the sheer number of bitags it proposes (12,363). EM+IP prunes that set of bitags considerably, leading to better precision at the cost of recall. EM$_{GI}$'s higher recall and precision indicate the tag transition distributions do capture general patterns of linkage between adjacent CCG categories, while EM ensures that the data filters out combinable, but unnecessary, bitags. With EM$_{GI}$+IP$_{GI}$, we again see that IP-minimization prunes even more entries, improving precision at the loss of some recall.

Similar trends are seen for precision and recall on the lexicon. IP-minimization's pruning of inappropriate taggings means more common words are not assigned highly infrequent supertags (boosting precision) while unknown words are generally assigned more sensible supertags (boosting recall). EM$_{GI}$ again focuses taggings on combinable contexts, boosting precision and recall similarly to EM+IP, but in greater measure. EM$_{GI}$+IP$_{GI}$ then prunes some of the spurious entries, boosting precision at some loss of recall.

Tag frequencies predicted on the test set. Table 4 compares gold tags to tags generated by all four methods for the frequent and highly ambiguous words *the* and *in*. Basic EM wanders far away from the gold assignments; it has little guidance in the very large search space available to it. IP-minimization identifies a smaller set of tags that better matches the gold tags; this emerges because other determiners and prepositions evoke similar, but not identical, supertags, and the grammar minimization pushes (but does not force) them to rely on the same supertags wherever possible. However, the proportions are incorrect; for example, the tag assigned most frequently to *in* is ((S\NP)\(S\NP))/NP though (NP\NP)/NP is more frequent in the test set. EM$_{GI}$'s tags correct that balance and find better proportions, but also some less common categories, such as (((N/N)\(N/N))\((N/N)\(N/N)))/N, sneak in because they combine with frequent categories like N/N and N. Bringing the two strategies together with EM$_{GI}$+IP$_{GI}$ filters out the unwanted categories while getting better overall proportions.

5.2 Italian CCG-TUT results

To demonstrate that both methods and their combination are language independent, we apply them to the Italian CCG-TUT corpus. We wanted to evaluate performance out-of-the-box because

[8]A state-of-the-art, fully-supervised maximum entropy tagger (Clark and Curran, 2007) (which also uses part-of-speech labels) obtains 91.4% on the same train/test split.

Lexicon	**Gold**	**EM**	**EM+IP**	**EM$_{GI}$**	**EM$_{GI}$+IP$_{GI}$**
the → *(41 distinct tags in L_{train})*	*(14 tags)*	*(18 tags)*	*(9 tags)*	*(25 tags)*	*(12 tags)*
NP[nb]/N	5742	0	4544	4176	4666
((S\NP)\(S\NP))/N	14	5	642	122	107
(((N/N)\(N/N))\((N/N)\(N/N)))/N	0	0	0	698	0
((S/S)/S[dcl])/(S[adj]\NP)	0	733	0	0	0
PP/N	0	1755	0	3	1
:	:	:	:	:	:
in → *(76 distinct tags in L_{train})*	*(35 tags)*	*(20 tags)*	*(17 tags)*	*(37 tags)*	*(14 tags)*
(NP\NP)/NP	883	0	649	708	904
((S\NP)\(S\NP))/NP	793	0	911	320	424
PP/NP	177	1	33	12	82
((S[adj]\NP)/(S[adj]\NP))/NP	0	215	0	0	0
:	:	:	:	:	:

Table 4: Comparison of tag assignments from the gold tags versus model tags obtained on the test set. The table shows tag assignments (and their counts for each method) for *the* and *in* in the CCGbank test sections. The number of *distinct* tags assigned by each method is given in parentheses. L_{train} is the lexicon obtained from sections 0-18 of CCGbank that is used as the basis for EM training.

Model	**TEST 1**	**TEST 2** *(using lexicon from:)*		
		NPAPER+CIVIL	NPAPER	CIVIL
Random	9.6	9.7	8.4	9.6
EM	26.4	26.8	27.2	29.3
EM+IP	34.8	32.4	34.8	34.6
EM$_{GI}$	43.1	**43.9**	44.0	40.3
EM$_{GI}$+IP$_{GI}$	**45.8**	43.6	**47.5**	**40.9**

Table 5: Comparison of supertagging results for CCG-TUT. Accuracies are for ambiguous word tokens in the test corpus, ignoring punctuation.

	EM	**EM+IP**	**EM$_{GI}$**	**EM$_{GI}$+IP$_{GI}$**
Grammar				
Precision	23.1	26.4	44.9	46.7
Recall	18.4	15.9	24.9	22.7
Lexicon				
Precision	51.2	52.0	54.8	55.1
Recall	43.6	42.8	46.0	44.9

Table 6: Comparison of grammar/lexicon observed in the *model tagging* vs. *gold tagging* in terms of precision and recall measures for supertagging on CCG-TUT.

bootstrapping a supertagger for a new language is one of the main use scenarios we envision: in such a scenario, there is no development data for changing settings and parameters. Thus, we determined a train/test split beforehand and ran the methods exactly as we had for CCGbank.

The results, given in Table 5, demonstrate the same trends as for English: basic EM is far more accurate than random, EM+IP adds another 8-10% absolute accuracy, and EM$_{GI}$ adds an additional 8-10% again. The combination of the methods generally improves over EM$_{GI}$, except when the lexicon is extracted from NPAPER+CIVIL. Table 6 gives precision and recall for the grammars and lexicons for CCG-TUT—the values are lower than for CCGbank (in line with the lower baseline), but exhibit the same trends.

6 Conclusion

We have shown how two complementary strategies—grammar-informed tag transitions and IP-minimization—for learning of supertaggers from highly ambiguous lexicons can be straightforwardly integrated. We verify the benefits of both cross-lingually, on English and Italian data. We also provide a new two-stage integer programming setup that allows model minimization to be tractable for supertagging without sacrificing the quality of the search for minimal bitag grammars.

The experiments in this paper use large lexicons, but the methodology will be particularly useful in the context of bootstrapping from smaller ones. This brings further challenges; in particular, it will be necessary to identify novel entries consisting of seen word and seen category and to predict unseen, but valid, categories which are needed to explain the data. For this, it will be necessary to forgo the assumption that the provided lexicon is always obeyed. The methods we introduce here should help maintain good accuracy while opening up these degrees of freedom. Because the lexicon *is* the grammar in CCG, learning new word-category associations is grammar generalization and is of interest for grammar acquisition.

Finally, such lexicon refinement and generalization is directly relevant for using CCG in syntax-based machine translation models (Hassan et al., 2009). Such models are currently limited to languages for which corpora annotated with CCG derivations are available. Clark and Curran (2006) show that CCG parsers can be learned from sentences labeled with just supertags—without full derivations—with little loss in accuracy. The improvements we show here for learning supertaggers from lexicons without labeled data may be able to help create annotated resources more efficiently, or enable CCG parsers to be learned with less human-coded knowledge.

Acknowledgements

The authors would like to thank Johan Bos, Joey Frazee, Taesun Moon, the members of the UT-NLL reading group, and the anonymous reviewers. Ravi and Knight acknowledge the support of the NSF (grant IIS-0904684) for this work. Baldridge acknowledges the support of a grant from the Morris Memorial Trust Fund of the New York Community Trust.

References

J. Baldridge. 2008. Weakly supervised supertagging with grammar-informed initialization. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 57–64, Manchester, UK, August.

M. Banko and R. C. Moore. 2004. Part of speech tagging in context. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, page 556, Morristown, NJ, USA.

A. R. Barron, J. Rissanen, and B. Yu. 1998. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760.

J. Bos, C. Bosco, and A. Mazzei. 2009. Converting a dependency treebank to a categorial grammar treebank for Italian. In *Proceedings of the Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*, pages 27–38, Milan, Italy.

S. Clark and J. Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 144–151, New York City, USA, June.

S. Clark and J. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).

M. Creutz and K. Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL Workshop on Morphological and Phonological Learning*, pages 21–30, Morristown, NJ, USA.

Y. Goldberg, M. Adler, and M. Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of the ACL*, pages 746–754, Columbus, Ohio, June.

J. Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198.

S. Goldwater and T. L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the ACL*, pages 744–751, Prague, Czech Republic, June.

H. Hassan, K. Sima'an, and A. Way. 2009. A syntactified direct translation model with linear-time decoding. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1182–1191, Singapore, August.

J. Hockenmaier and M. Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

A. Joshi. 1988. Tree Adjoining Grammars. In David Dowty, Lauri Karttunen, and Arnold Zwicky, editors, *Natural Language Parsing*, pages 206–250. Cambridge University Press, Cambridge.

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

B. Merialdo. 1994. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171.

C. Pollard and I. Sag. 1994. *Head Driven Phrase Structure Grammar*. CSLI/Chicago University Press, Chicago.

S. Ravi and K. Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 504–512, Suntec, Singapore, August.

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

Kristina Toutanova and Mark Johnson. 2008. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 1521–1528, Cambridge, MA. MIT Press.