Parallel Algorithms for Unsupervised Tagging

Sujith Ravi
Google
Mountain View, CA 94043
sravi@google.com

Sergei Vassilivitskii
Google
Mountain View, CA 94043
sergeiv@google.com

Vibhor Rastogi[*]
Twitter
San Francisco, CA
vibhor.rastogi@gmail.com

Abstract

We propose a new method for unsupervised tagging that finds minimal models which are then further improved by Expectation Maximization training. In contrast to previous approaches that rely on manually specified and multi-step heuristics for model minimization, our approach is a simple greedy approximation algorithm DMLC (DISTRIBUTED-MINIMUM-LABEL-COVER) that solves this objective in a single step.

We extend the method and show how to efficiently parallelize the algorithm on modern parallel computing platforms while preserving approximation guarantees. The new method easily scales to large data and grammar sizes, overcoming the memory bottleneck in previous approaches. We demonstrate the power of the new algorithm by evaluating on various sequence labeling tasks: Part-of-Speech tagging for multiple languages (including low-resource languages), with complete and incomplete dictionaries, and supertagging, a complex sequence labeling task, where the grammar size alone can grow to millions of entries. Our results show that for all of these settings, our method achieves state-of-the-art scalable performance that yields high quality tagging outputs.

1 Introduction

Supervised sequence labeling with large labeled training datasets is considered a solved problem. For instance, state of the art systems obtain tagging accuracies over 97% for part-of-speech (POS) tagging on the English Penn Treebank. However, learning accurate taggers without labeled data remains a challenge. The accuracies quickly drop when faced with data from a different domain, language, or when there is very little labeled information available for training (Banko and Moore, 2004).

Recently, there has been an increasing amount of research tackling this problem using unsupervised methods. A popular approach is to learn from POS-tag dictionaries (Merialdo, 1994), where we are given a raw word sequence and a dictionary of legal tags for each word type. Learning from POS-tag dictionaries is still challenging. Complete word-tag dictionaries may not always be available for use and in every setting. When they are available, the dictionaries are often noisy, resulting in high tagging ambiguity. Furthermore, when applying taggers in new domains or different datasets, we may encounter new words that are missing from the dictionary. There have been some efforts to learn POS taggers from incomplete dictionaries by extending the dictionary to include these words using some heuristics (Toutanova and Johnson, 2008) or using other methods such as type-supervision (Garrette and Baldridge, 2012).

In this work, we tackle the problem of unsupervised sequence labeling using tag dictionaries. The first reported work on this problem was on POS tagging from Merialdo (1994). The approach involved training a standard Hidden Markov Model (HMM) using the Expectation Maximization (EM) algorithm (Dempster et al., 1977), though EM does not

[*] The research described herein was conducted while the author was working at Google.

Transactions of the Association for Computational Linguistics, 2 (2014) 105–118. Action Editor: Sharon Goldwater.
Submitted 11/2013; Revised 2/2014; Published 4/2014.

perform well on this task (Johnson, 2007). More recent methods have yielded better performance than EM (see (Ravi and Knight, 2009) for an overview).

One interesting line of research introduced by Ravi and Knight (2009) explores the idea of performing model minimization followed by EM training to learn taggers. Their idea is closely related to the classic Minimum Description Length principle for model selection (Barron et al., 1998). They (1) formulate an objective function to find the smallest model that explains the text (model minimization step), and then, (2) fit the minimized model to the data (EM step). For POS tagging, this method (Ravi and Knight, 2009) yields the best performance to date; 91.6% tagging accuracy on a standard test dataset from the English Penn Treebank. The original work from (Ravi and Knight, 2009) uses an integer linear programming (ILP) formulation to find minimal models, an approach which does not scale to large datasets. Ravi et al. (2010b) introduced a two-step greedy approximation to the original objective function (called the MIN-GREEDY algorithm) that runs much faster while maintaining the high tagging performance. Garrette and Baldridge (2012) showed how to use several heuristics to further improve this algorithm (for instance, better choice of tag bigrams when breaking ties) and stack other techniques on top, such as careful initialization of HMM emission models which results in further performance gains. Their method also works under incomplete dictionary scenarios and can be applied to certain low-resource scenarios (Garrette and Baldridge, 2013) by combining model minimization with supervised training.

In this work, we propose a new scalable algorithm for performing model minimization for this task. By making an assumption on the structure of the solution, we prove that a variant of the greedy set cover algorithm always finds an approximately optimal label set. This is in contrast to previous methods that employ heuristic approaches with no guarantee on the quality of the solution. In addition, we do not have to rely on ad hoc tie-breaking procedures or careful initializations for unknown words. Finally, not only is the proposed method approximately optimal, it is also easy to distribute, allowing it to easily scale to very large datasets. We show empirically that our method, combined with an EM training step outperforms existing state of the art systems.

1.1 Our Contributions

- We present a new method, DISTRIBUTED MINIMUM LABEL COVER, DMLC, for model minimization that uses a fast, greedy algorithm with formal approximation guarantees to the quality of the solution.

- We show how to efficiently parallelize the algorithm while preserving approximation guarantees. In contrast, existing minimization approaches cannot match the new distributed algorithm when scaling from thousands to millions or even billions of tokens.

- We show that our method easily scales to both large data and grammar sizes, and does not require the corpus or label set to fit into memory. This allows us to tackle complex tagging tasks, where the tagset consists of several thousand labels, which results in more than one million entries in the grammar.

- We demonstrate the power of the new method by evaluating under several different scenarios—POS tagging for multiple languages (including low-resource languages), with complete and incomplete dictionaries, as well as a complex sequence labeling task of supertagging. Our results show that for all these settings, our method achieves state-of-the-art performance yielding high quality taggings.

2 Related Work

Recently, there has been an increasing amount of research tackling this problem from multiple directions. Some efforts have focused on inducing POS tag clusters without any tags (Christodoulopoulos et al., 2010; Reichart et al., 2010; Moon et al., 2010), but evaluating such systems proves difficult since it is not straightforward to map the cluster labels onto gold standard tags. A more popular approach is to learn from POS-tag dictionaries (Merialdo, 1994; Ravi and Knight, 2009), incomplete dictionaries (Hasan and Ng, 2009; Garrette and Baldridge, 2012) and human-constructed dictionaries (Goldberg et al., 2008).

Another direction that has been explored in the past includes bootstrapping taggers for a new language based on information acquired from other languages (Das and Petrov, 2011) or limited annotation resources (Garrette and Baldridge, 2013). Additional work focused on building supervised taggers for noisy domains such as Twitter (Gimpel et al., 2011). While most of the relevant work in this area centers on POS tagging, there has been some work done for building taggers for more complex sequence labeling tasks such as supertagging (Ravi et al., 2010a).

Other related work include alternative methods for learning sparse models via priors in Bayesian inference (Goldwater and Griffiths, 2007) and posterior regularization (Ganchev et al., 2010). But these methods only encourage sparsity and do not explicitly seek to minimize the model size, which is the objective function used in this work. Moreover, taggers learned using model minimization have been shown to produce state-of-the-art results for the problems discussed here.

3 Model

Following Ravi and Knight (2009), we formulate the problem as that of label selection on the sentence graph. Formally, we are given a set of sequences, $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ where each S_i is a sequence of words, $S_i = w_{i1}, w_{i2}, \ldots, w_{i,|S_i|}$. With each word w_{ij} we associate a set of possible tags T_{ij}. We will denote by m the total number of (possibly duplicate) words (tokens) in the corpus.

Additionally, we define two special words w_0 and w_∞ with special tags $start$ and end, and consider the modified sequences $S'_i = w_0, S_i, w_\infty$. To simplify notation, we will refer to $w_\infty = w_{|S_i|+1}$. The sequence label problem asks us to select a valid tag $t_{ij} \in T_{ij}$ for each word w_{ij} in the input to minimize a specific objective function.

We will refer to a tag pair $(t_{i,j-1}, t_{ij})$ as a *label*. Our aim is to minimize the number of distinct labels used to cover the full input. Formally, given a sequence S'_i and a tag t_{ij} for each word w_{ij} in S'_i, let the induced set of labels for sequence S'_i be

$$L_i = \bigcup_{j=1}^{|S'_i|} \{(t_{i,j-1}, t_{ij})\}.$$

The total number of distinct labels used over all sequences is then

$$\phi = \left| \cup_i L_i \right| = \left| \bigcup_i \bigcup_{j=1}^{|S_i|+1} \{(t_{i,j-1}, t_{ij})\} \right|.$$

Note that the order of the tokens in the label makes a difference as {(NN, VP)} and {(VP, NN)} are two distinct labels.

Now we can define the problem formally, following (Ravi and Knight, 2009).

Problem 1 (Minimum Label Cover)**.** *Given a set $\mathcal{S}$ of sequences of words, where each word w_{ij} has a set of valid tags T_{ij}, the problem is to find a valid tag assignment $t_{ij} \in T_{ij}$ for each word that minimizes the number of distinct labels or tag pairs over all sequences, $\phi = \left| \bigcup_i \bigcup_{j=1}^{|S_i|+1} \{(t_{i,j-1}, t_{ij})\} \right|$.*

The problem is closely related to the classical Set Cover problem and is also NP-complete. To reduce Set Cover to the label selection problem, map each element i of the Set Cover instance to a single word sentence $S_i = w_{i1}$, and let the valid tags T_{i1} contain the names of the sets that contain element i. Consider a solution to the label selection problem; every sentence S_i is covered by two labels (w_0, k_i) and (k_i, w_∞), for some $k_i \in T_{i1}$, which corresponds to an element i being covered by set k_i in the Set Cover instance. Thus any valid solution to the label selection problem leads to a feasible solution to the Set Cover problem ($\{k_1, k_2, \ldots\}$) of exactly half the size.

Finally, we will use $\{\{\ldots\}\}$ notation to denote a multiset of elements, i.e. a set where an element may appear multiple times.

4 Algorithm

In this Section, we describe the DISTRIBUTED-MINIMUM-LABEL-COVER, DMLC, algorithm for approximately solving the minimum label cover problem. We describe the algorithm in a centralized setting, and defer the distributed implementation to Section 5. Before describing the algorithm, we briefly explain the relationship of the minimum label cover problem to set cover.

4.1 Modification of Set Cover

As we pointed out earlier, the minimum label cover problem is at least as hard as the Set Cover prob-

```
Input: A set of sequences $\mathcal{S}$ with each words $w_{ij}$ having possible tags $T_{ij}$.
Output: A tag assignment $t_{ij} \in T_{ij}$ for each word $w_{ij}$ approximately minimizing labels.
Let $\mathcal{M}$ be the multi set of all possible labels generated by choosing each possible tag $t \in T_{ij}$.
    $\mathcal{M} = \bigcup_{i} \left( \bigcup_{j=1}^{|S_i|+1} \bigcup_{\substack{t' \in T_{i,j-1} \\ t \in T_{ij}}} \{\{(t',t)\}\} \right)$   (1)
Let $\mathcal{L} = \emptyset$ be the set of selected labels.
repeat
    Select the most frequent label not yet selected: $(t',t) = \arg\max_{(s',s)\notin\mathcal{L}} |\mathcal{M} \cap (s',s)|$.
    For each bigram $(w_{i,j-1}, w_{ij})$ where $t' \in T_{i,j-1}$ and $t \in T_{ij}$ tentatively assign $t'$ to $w_{i,j-1}$ and $t$ to $w_{ij}$. Add $(t',t)$ to $\mathcal{L}$.
    If a word gets two assignments, select one at random with equal probability.
    If a bigram $(w_{i,j-1}, w_{ij})$ is consistent with assignments in $(t',t)$, fix the tentative assignments, and set $T_{i,j-1} = \{t'\}$ and $T_{ij} = \{t\}$. Recompute $\mathcal{M}$, the multiset of possible labels, with the updated $T_{i,j-1}$ and $T_{ij}$.
until there are no unassigned words
```

Algorithm 1: MLC Algorithm

```
Input: A set of sequences $\mathcal{S}$ with each words $w_{ij}$ having possible tags $T_{ij}$.
Output: A tag assignment $t_{ij} \in T_{ij}$ for each word $w_{ij}$ approximately minimizing labels.
(Graph Creation) Initialize each vertex $v_{ij}$ with the set of possible tags $T_{ij}$ and its neighbors $v_{i,j+1}$ and $v_{i,j-1}$.
repeat
    (Message Passing) Each vertex $v_{ij}$ sends its possibly tags $T_{ij}$ to its forward neighbor $v_{ij+1}$.
    (Counter Update) Each vertex receives the the tags $T_{i,j-1}$ and adds all possible labels $\{(s,s')|s \in T_{i,j-1}, s' \in T_{ij}\}$ to a global counter $(M)$.
    (MaxLabel Selection) Each vertex queries the global counter $\mathcal{M}$ to find the maximum label $(t,t')$.
    (Tentative Assignment) Each vertex $v_{ij}$ selects a tag tentatively as follows: If one of the tags $t,t'$ is in the feasible set $T_{ij}$, it tentatively selects the tag.
    (Random Assignment) If both are feasible it selects one at random. The vertex communicates its assignment to its neighbors.
    (Confirmed Assignment) Each vertex receives the tentative assignment from its neighbors. If together with its neighbors it can match the selected label, the assignment is finalized. If the assigned tag is $T$, then the vertex $v_{ij}$ sets the valid tag set $T_{ij}$ to $\{t\}$.
until no unassigned vertices exist.
```

Algorithm 2: DMLC Implementation

lem. An additional challenge comes from the fact that labels are tags for a pair of words, and hence are related. For example, if we label a word pair $(w_{i,j-1}, w_{ij})$ as (NN, VP), then the label for the next word pair $(w_{ij}, w_{i,j+1})$ has to be of the form (VP, *), i.e., it has to start with VP.

Previous work (Ravi et al., 2010a; Ravi et al., 2010b) recognized this challenge and employed two phase heuristic approaches. Eschewing heuristics, we will show that with one natural assumption, even with this extra set of constraints, the standard greedy algorithm for this problem results in a solution with a provable approximation ratio of $O(\log m)$. In practice, however, the algorithm performs far better than the worst case ratio, and similar to the work of (Gomes et al., 2006), we find that the greedy approach selects a cover approximately 11% worse than the optimum solution.

4.2 MLC Algorithm

We present in Algorithm 1 our MINIMUM LABEL COVER algorithm to approximately solve the minimum label cover problem. The algorithm is simple, efficient, and easy to distribute.

The algorithm chooses labels one at a time, selecting a label that covers as many words as possible in every iteration. For this, it generates and maintains a multi-set of all possible labels $\mathcal{M}$ (Step 3). The multi-set contains an occurrence of each valid label, for example, if $w_{i,j-1}$ has two possible valid tags NN and VP, and w_{ij} has one possible valid tag VP, then $\mathcal{M}$ will contain two labels, namely (NN, VP) and (VP, VP). Since M is a multi-set it will contain duplicates, e.g. the label (NN, VP) will appear for each adjacent pair of words that have NN and VP as valid tags, respectively.

In each iteration, the algorithm picks a label with the most number of occurrences in $\mathcal{M}$ and adds it to the set of chosen labels (Step 6). Intuitively, this is a greedy step to select a label that covers the most number of word pairs.

Once the algorithm picks a label (t', t), it tries to assign as many words to tags t or t' as possible (Step 7). A word can be assigned t' if t' is a valid tag for it, and t a valid tag for the next word in sequence. Similarly, a word can be assigned t, if t is a valid tag for it, and t' a valid tag for the previous word. Some words can get both assignments, in which case we choose one tentatively at random (Step 8). If a word's tentative random tag, say t, is consistent with the choices of its adjacent words (say t' from the previous word), then the tentative choice is fixed as a permanent one. Whenever a tag is selected, the set of valid tags T_{ij} for the word is reduced to a singleton $\{t\}$. Once the set of valid tags T_{ij} changes, the multi-set $\mathcal{M}$ of all possible labels also changes, as seen from Eq 1. The multi-set is then recomputed (Step 9) and the iterations repeated until all of words have been tagged.

We can show that under a natural assumption this simple algorithm is approximately optimal.

Assumption 1 (c-feasibility)**.** *Let $c \geq 1$ be any number, and k be the size of the optimal solution to the original problem. In each iteration, the* MLC *algorithm fixes the tags for some words. We say that the algorithm is c-feasible, if after each iteration there exists some solution to the remaining problem, consistent with the chosen tags, with size at most ck .*

The assumption encodes the fact that a single bad greedy choice is not going to destroy the overall structure of the solution, and a nearly optimal solution remains. We note that this assumption of c-feasibility is not only sufficient, as we will formally show, but is also necessary. Indeed, without any assumptions, once the algorithm fixes the tag for some words, an optimal label may no longer be consistent with the chosen tags, and it is not hard to find contrived examples where the size of the optimal solution doubles after each iteration of MLC.

Since the underlying problem is NP-complete, it is computationally hard to give direct evidence verifying the assumption on natural language inputs. However, on small examples we are able to show that the greedy algorithm is within a small constant factor of the optimum, specifically it is within 11% of the optimum model size for the POS tagging problem using the standard 24k dataset (Ravi and Knight, 2009). Combined with the fact that the final method outperforms state of the art approaches, this leads us to conclude that the structural assumption is well justified.

Lemma 1. *Under the assumption of c-feasibility, the* MLC *algorithm achieves a $O(c \log m)$ approximation to the minimum label cover problem, where $m = \sum_i |S_i|$ is the total number of tokens.*

Proof. To prove the Lemma we will define an objective function $\bar{\phi}$, counting the number of unlabeled word pairs, as a function of possible labels, and show that $\bar{\phi}$ decreases by a factor of $(1 - O(1/ck))$ at every iteration.

To define $\bar{\phi}$, we first define ϕ, the number of labeled word pairs. Consider a particular set of labels, $\mathcal{L} = \{L_1, L_2, \ldots, L_k\}$ where each label is a pair (t_i, t_j). Call $\{t_{ij}\}$ a valid assignment of tokens if for each w_{ij}, we have $t_{ij} \in T_{ij}$. Then the score of $\mathcal{L}$ under an assignment t, which we denote by ϕ_t, is the number of bigram labels that appear in $\mathcal{L}$. Formally, $\phi_t(\mathcal{L}) = | \cup_{i,j} \{\{(t_{i,j-1}, t_{ij}) \cap \mathcal{L}\}\}|$. Finally, we define $\phi(\mathcal{L})$ to be the best such assignment, $\phi(\mathcal{L}) = \max_t \phi_t(\mathcal{L})$, and $\bar{\phi}(\mathcal{L}) = m - \phi(\mathcal{L})$ the number of uncovered labels.

Consider the label selected by the algorithm in every step. By the c-feasibility assumption, there exists some solution having ck labels. Thus, some label from that solution covers at least a $1/ck$ fraction of the remaining words. The selected label (t, t') maximizes the intersection with the remaining feasible labels. The conflict resolution step ensures that in expectation the realized benefit is at least a half of the maximum, thereby reducing $\bar{\phi}$ by at least a

$(1 - 1/2ck)$ fraction. Therefore, after $O(kc \log m)$ operations all of the labels are covered. □

4.3 Fitting the Model Using EM

Once the greedy algorithm terminates and returns a minimized grammar of tag bigrams, we follow the approach of Ravi and Knight (2009) and fit the minimized model to the data using the alternating EM strategy.

In this step, we run an alternating optimization procedure iteratively in phases. In each phase, we initialize (and prune away) parameters within the two HMM components (transition or emission model) using the output from the previous phase. We initialize this procedure by restricting the transition parameters to only those tag bigrams selected in the model minimization step. We train in conjunction with the original emission model using EM algorithm which prunes away some of the emission parameters. In the next phase, we alternate the initialization by choosing the pruned emission model along with the original transition model (with full set of tag bigrams) and retrain using EM. The alternating EM iterations are terminated when the change in the size of the observed grammar (i.e., the number of unique bigrams in the tagging output) is $\leq 5\%$.[1] We refer to our entire approach using greedy minimization followed by EM training as DMLC + EM.

5 Distributed Implementation

The DMLC algorithm is directly suited towards parallelization across many machines. We turn to Pregel (Malewicz et al., 2010), and its open source version Giraph (Apa, 2013). In these systems the computation proceeds in rounds. In every round, every machine does some local processing and then sends arbitrary messages to other machines. Semantically, we think of the communication graph as fixed, and in each round each vertex performs some local computation and then sends messages to its neighbors. This mode of parallel programming directs the programmers to "Think like a vertex."

The specific systems like Pregel and Giraph build infrastructure that ensures that the overall system is fault tolerant, efficient, and fast. In addition, they provide implementation of commonly used distributed data structures, such as, for example global counters. The programmer's job is simply to specify the code that each vertex will run at every round.

We implemented the DMLC algorithm in Pregel. The implementation is straightforward and given in Algorithm 2. The multi-set $\mathcal{M}$ of Algorithm 1 is represented as a global counter in Algorithm 2. The message passing (Step 3) and counter update (Step 4) steps update this global counter and hence perform the role of Step 3 of Algorithm 1. Step 5 selects the label with largest count, which is equivalent to the greedy label picking step 6 of Algorithm 1. Finally steps 6, 7, and 8 update the tag assignment of each vertex performing the roles of steps 7, 8, and 9, respectively, of Algorithm 1.

5.1 Speeding up the Algorithm

The implementation described above directly copies the sequential algorithm. Here we describe additional steps we took to further improve the parallel running times.

Singleton Sets: As the parallel algorithm proceeds, the set of feasible sets associated with a node slowly decreases. At some point there is only one tag that a node can take on, however this tag is rare, and so it takes a while for it to be selected using the greedy strategy. Nevertheless, if a node and one of its neighbors have only a single tag left, then it is safe to assign the unique label [2].

Modifying the Graph: As is often the case, the bottleneck in parallel computations is the communication. To reduce the amount of communication we reduce the graph on the fly, removing nodes and edges once they no longer play a role in the computation. This simple modification decreases the communication time in later rounds as the total size of the problem shrinks.

6 Experiments and Results

In this Section, we describe the experimental setup for various tasks, settings and compare empirical performance of our method against several existing

[1] For more details on the alternating EM strategy and how initialization with minimized models improve EM performance in alternating iterations, refer to (Ravi and Knight, 2009).

[2] We must judiciously initialize the global counter to take care of this assignment, but this is easily accomplished.

baselines. The performance results for all systems (on all tasks) are measured in terms of tagging accuracy, i.e. % of tokens from the test corpus that were labeled correctly by the system.

6.1 Part-of-Speech Tagging Task

6.1.1 Tagging Using a Complete Dictionary

Data: We use a standard test set (consisting of 24,115 word tokens from the Penn Treebank) for the POS tagging task. The tagset consists of 45 distinct tag labels and the dictionary contains 57,388 word/tag pairs derived from the entire Penn Treebank. Per-token ambiguity for the test data is about 1.5 tags/token. In addition to the standard 24k dataset, we also train and test on larger data sets—973k tokens from the Penn Treebank, 3M tokens from PTB+Europarl (Koehn, 2005) data.

Methods: We evaluate and compare performance for POS tagging using four different methods that employ the model minimization idea combined with EM training:

- EM: Training a bigram HMM model using EM algorithm (Merialdo, 1994).
- ILP + EM: Minimizing grammar size using integer linear programming, followed by EM training (Ravi and Knight, 2009).
- MIN-GREEDY + EM: Minimizing grammar size using the two-step greedy method (Ravi et al., 2010b).
- DMLC + EM: This work.

Results: Table 1 shows the results for POS tagging on English Penn Treebank data. On the smaller test datasets, all of the model minimization strategies (methods 2, 3, 4) tend to perform equally well, yielding state-of-the-art results and large improvement over standard EM. When training (and testing) on larger corpora sizes, DMLC yields the best reported performance on this task to date. A major advantage of the new method is that it can easily scale to large corpora sizes and the distributed nature of the algorithm still permits fast, efficient optimization of the global objective function. So, unlike the earlier methods (such as MIN-GREEDY) it is fast enough to run on several millions of tokens to yield additional performance gains (shown in last column).

Speedups: We also observe a significant speedup when using the parallelized version of the DMLC algorithm. Performing model minimization on the 24k tokens dataset takes 55 seconds on a single machine, whereas parallelization permits model minimization to be feasible even on large datasets. Fig 1 shows the running time for DMLC when run on a cluster of 100 machines. We vary the input data size from 1M word tokens to about 8M word tokens, while holding the resources constant. Both the algorithm and its distributed implementation in DMLC are linear time operations as evident by the plot. In fact, for comparison, we also plot a straight line passing through the first two runtimes. The straight line essentially plots runtimes corresponding to a linear speedup. DMLC clearly achieves better runtimes showing even better than linear speedup. The reason for this is that distributed version has a constant overhead for initialization, independent of the data size. While the running time for rest of the implementation is linear in data size. Thus, as the data size becomes larger, the constant overhead becomes less significant, and the distributed implementation appears to complete slightly faster as data size increases.

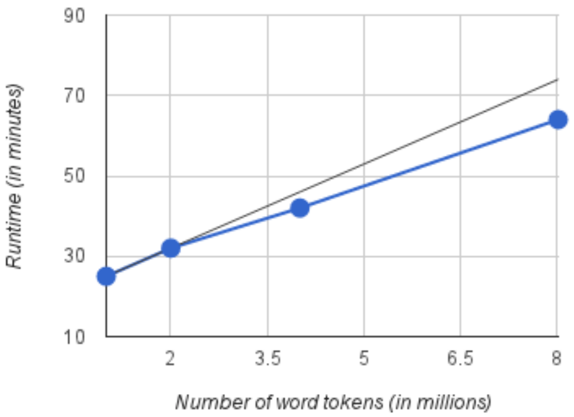

Figure 1: Runtime vs. data size (measured in # of word tokens) on 100 machines. For comparison, we also plot a straight line passing through the first two runtimes. The straight line essentially plots runtimes corresponding to a linear speedup. DMLC clearly achieves better runtimes showing a better than linear speedup.

6.1.2 Tagging Using Incomplete Dictionaries

We also evaluate our approach for POS tagging under other resource-constrained scenarios. Obtain-

Method	Tagging accuracy (%)		
	te=24k	*te*=973k	
	tr=24k	*tr*=973k	*tr*=3.7M
1. EM	81.7	82.3	
2. ILP + EM (Ravi and Knight, 2009)	91.6	–	
3. MIN-GREEDY + EM (Ravi et al., 2010b)	91.6	87.1	
4. DMLC + EM (this work)	91.4	87.5	87.8

Table 1: Results for unsupervised part-of-speech tagging on English Penn Treebank dataset. Tagging accuracies for different methods are shown on multiple datasets. *te* shows the size (number of tokens) in the test data, *tr* represents the size of the raw text used to perform model minimization.

ing a complete dictionary is often difficult, especially for new domains. To verify the utility of our method when the input dictionary is incomplete, we evaluate against standard datasets used in previous work (Garrette and Baldridge, 2012) and compare against the previous best reported performance for the same task. In all the experiments (described here and in subsequent sections), we use the following terminology—*raw* data refers to unlabeled text used by different methods (for model minimization or other unsupervised training procedures such as EM), *dictionary* consists of word/tag entries that are legal, and *test* refers to data over which tagging evaluation is performed.

English Data: For English POS tagging with incomplete dictionary, we evaluate on the Penn Treebank (Marcus et al., 1993) data. Following (Garrette and Baldridge, 2012), we extracted a word-tag dictionary from sections 00-15 (751,059 tokens) consisting of 39,087 word types, 45,331 word/tag entries, a per-type ambiguity of 1.16 yielding a per-token ambiguity of 2.21 on the raw corpus (treating unknown words as having all 45 possible tags). As in their setup, we then use the first 47,996 tokens of section 16 as raw data and perform final evaluation on the sections 22-24. We use the raw corpus along with the unlabeled test data to perform model minimization and EM training. Unknown words are allowed to have all possible tags in both these procedures.

Italian Data: The minimization strategy presented here is a general-purpose method that does not require any specific tuning and works for other languages as well. To demonstrate this, we also perform evaluation on a different language (Italian) using the TUT corpus (Bosco et al., 2000). Following (Garrette and Baldridge, 2012), we use the same data splits as their setting. We take the first half of each of the five sections to build the word-tag dictionary, the next quarter as raw data and the last quarter as test data. The dictionary was constructed from 41,000 tokens comprised of 7,814 word types, 8,370 word/tag pairs, per-type ambiguity of 1.07 and a per-token ambiguity of 1.41 on the raw data. The raw data consisted of 18,574 tokens and the test contained 18,763 tokens. We use the unlabeled corpus from the raw and test data to perform model minimization followed by unsupervised EM training.

Other Languages: In order to test the effectiveness of our method in other non-English settings, we also report the performance of our method on several other Indo-European languages using treebank data from CoNLL-X and CoNLL-2007 shared tasks on dependency parsing (Buchholz and Marsi, 2006; Nivre et al., 2007). The corpus statistics for the five languages (Danish, Greek, Italian, Portuguese and Spanish) are listed below. For each language, we construct a dictionary from the raw training data. The unlabeled corpus from the raw training and test data is used to perform model minimization followed by unsupervised EM training. As before, unknown words are allowed to have all possible tags. We report the final tagging performance on the test data and compare it to baseline EM.

Garrette and Baldridge (2012) treat unknown words (words that appear in the raw text but are missing from the dictionary) in a special manner and use several heuristics to perform better initialization for such words (for example, the probability that an unknown word is associated with a particular tag is conditioned on the openness of the tag). They also use an auto-supervision technique to smooth counts learnt from EM onto new words encountered during testing. In contrast, we do not apply any such technique for unknown words and allow them to be mapped uniformly to all possible tags in the dictionary. For this particular set of experiments, the only difference from the Garrette and Baldridge (2012) setup is that we include unlabeled text from the test data (but without any dictionary tag labels or special heuristics) to our existing word tokens from raw text for performing model minimization. This is a standard practice used in unsupervised training scenarios (for example, Bayesian inference methods) and in general for scalable techniques where the goal is to perform inference on the same data for which one wishes to produce some structured prediction.

Language	**Train**	**Dict**	**Test**
	(tokens)	(entries)	(tokens)
DANISH	94386	18797	5852
GREEK	65419	12894	4804
ITALIAN	71199	14934	5096
PORTUGUESE	206678	30053	5867
SPANISH	89334	17176	5694

Results: Table 2 (column 2) compares previously reported results against our approach for English. We observe that our method obtains a huge improvement over standard EM and gets comparable results to the previous best reported scores for the same task from (Garrette and Baldridge, 2012). It is encouraging to note that the new system achieves this performance without using any of the carefully-chosen heuristics employed by the previous method. However, we do note that some of these techniques can be easily combined with our method to produce further improvements.

Table 2 (column 3) also shows results on Italian POS tagging. We observe that our method achieves significant improvements in tagging accuracy over all the baseline systems including the previous best system (+2.9%). This demonstrates that the method generalizes well to other languages and produces consistent tagging improvements over existing methods for the same task.

Results for POS tagging on CoNLL data in five different languages are displayed in Figure 2. Note that the proportion of raw data in test versus train (from the standard CoNLL shared tasks) is much smaller compared to the earlier experimental settings. In general, we observe that adding more raw data for EM training improves the tagging quality (same trend observed earlier in Table 1: column 2 versus column 3). Despite this, DMLC + EM still achieves significant improvements over the baseline EM system on multiple languages (as shown in Figure 2). An additional advantage of the new method is that it can easily scale to larger corpora and it produces a much more compact grammar that can be efficiently incorporated for EM training.

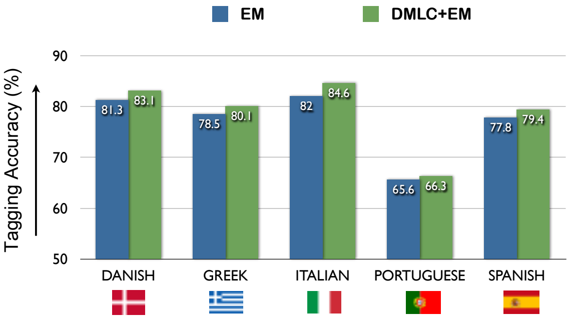

Figure 2: Part-of-Speech tagging accuracy for different languages on CoNLL data using incomplete dictionaries.

6.1.3 Tagging for Low-Resource Languages

Learning part-of-speech taggers for severely low-resource languages (e.g., Malagasy) is very challenging. In addition to scarce (token-supervised) labeled resources, the tag dictionaries available for training taggers are tiny compared to other languages such as English. Garrette and Baldridge (2013) combine various supervised and semi-supervised learning algorithms into a common POS tagger training pipeline to address some of these challenges. They also report tagging accuracy improvements on low-resource languages when using the combined system over any single algorithm. Their system has four main parts, in order: (1) Tag dictionary expansion using label propagation algorithm, (2) Weighted model minimization, (3) Expectation maximization (EM) training of HMMs using auto-supervision, (4) MaxEnt Markov Model (MEMM) training. The entire procedure results in a trained tagger model that can then be applied to tag any raw data.[3] Step 2 in this procedure involves

[3] For more details, refer (Garrette and Baldridge, 2013).

Method	**Tagging accuracy (%)**	
	English (PTB 00-15)	Italian (TUT)
1. Random	63.53	62.81
2. EM	69.20	60.70
3. Type-supervision + HMM initialization (Garrette and Baldridge, 2012)	88.52	72.86
4. DMLC + EM (this work)	88.11	75.79

Table 2: Part-of-Speech tagging accuracy using PTB sections 00-15 and TUT to build the tag dictionary. For comparison, we also include the results for the previously reported state-of-the-art system (method 3) for the same task.

Method	**Tagging accuracy (%)**		
	Total	Known	Unknown
Low-resource tagging using (Garrette and Baldridge, 2013)	80.7 (70.2)	87.6 (90.3)	66.1 (45.1)
Low-resource tagging using DMLC + EM (this work)	81.1 (70.8)	87.9 (90.3)	66.7 (46.5)

Table 3: Part-of-Speech tagging accuracy for a low-resource language (Malagasy) on *All/Known/Unknown* tokens in the test data. Tagging performance is shown for multiple experiments using different (incomplete) dictionary sizes: (a) *small*, (b) *tiny* (shown in parentheses). The new method (row 2) significantly outperforms the existing method with $p < 0.01$ for *small* dictionary and $p < 0.05$ for *tiny* dictionary.

a weighted version of model minimization which uses the multi-step greedy approach from Ravi et al. (2010b) enhanced with additional heuristics that uses tag weights learnt via label propagation (in Step 1) within the minimization process.

We replace the model minimization procedure in their Step 2 with our method (DMLC + EM) and directly compare this new system with their approach in terms of tagging accuracy. Note for all other steps in the pipeline we follow the same procedure (and run the same code) as Garrette and Baldridge (2013), including the same smoothing procedure for EM initialization in Step 3.

Data: We use the exact same setup as Garrette and Baldridge (2013) and run experiments on Malagasy, an Austronesian language spoken in Madagascar. We use the publicly available data[4]: 100k raw tokens for training, a word-tag dictionary acquired with 4 hours of human annotation effort (used for type-supervision), and a held-out test dataset (5341 tokens). We provide the unlabeled corpus from the raw training data along with the word-tag dictionary as input to model minimization and evaluate on the test corpus. We run multiple experiments for different (incomplete) dictionary scenarios: (a) *small* = 2773 word/tag pairs, (b) *tiny* = 329 word/tag pairs.

Results: Table 3 shows results on Malagasy data comparing a system that employs (unweighted) DMLC against the existing state-of-the-art system that incorporates a multi-step weighted model minimization combined with additional heuristics. We observe that switching to the new model minimization procedure alone yields significant improvement in tagging accuracy under both dictionary scenarios. It is encouraging that a better minimization procedure also leads to higher tagging quality on the unknown word tokens (column 4 in the table), even when the input dictionary is tiny.

6.2 Supertagging

Compared to POS tagging, 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.

As described earlier, our approach scales easily to large datasets as well as label sizes. To evaluate it on the supertagging task, we use the same dataset from (Ravi et al., 2010a) and compare against their baseline method that uses an modified (two-step) version

[4] github.com/ dhgarrette/low-resource-pos-tagging-2013

Method	Supertagging accuracy (%)	
	Ambiguous	Total
1. EM	38.7	45.6
2. ILP* + EM (Ravi et al., 2010a)	52.1	57.3
3. DMLC + EM (this work)	55.9	59.3

Table 4: Results for unsupervised supertagging with a dictionary. Here, we report the total accuracy as well as accuracy on just the ambiguous tokens (i.e., tokens which have more than one tagging possibility). *The baseline method 2 requires several pre-processing steps in order to run feasibly for this task (described in Section 6.2). In contrast, the new approach (DMLC) runs fast and also permits efficient parallelization.

of the ILP formulation for model minimization.

Data: We use the CCGbank data for this experiment. This data 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 (Banko and Moore, 2004)—sections 0-18 for training (i.e., to construct the word-tag dictionary), and sections 22-24 for test.

Results: Table 4 compares the results for two baseline systems—standard EM (method 1), and a previously reported system using model minimization (method 2) for the same task. We observe that DMLC produces better taggings than either of these and yields significant improvement in accuracy (+2% overall, +3.8% on ambiguous tokens).

Note that it is not feasible to run the ILP-based baseline (method 2 in the table) directly since it is very slow in practice, so Ravi et al. (2010a) use a set of pre-processing steps to prune the original grammar size (unique tag pairs) from >1M to several thousand entries followed by a modified two-step ILP minimization strategy. This is required to permit their model minimization step to be run in a feasible manner. On the other hand, the new approach DMLC (method 3) scales better even when the data/label sizes are large, hence it can be run with the full data using the original model minimization formulation (rather than a two-step heuristic).

Ravi et al. (2010a) also report further improvements using an alternative approach involving an ILP-based weighted minimization procedure. In Section 7 we briefly discuss how the DMLC method can be extended to this setting and combined with other similar methods.

7 Discussion and Conclusion

We present a fast, efficient model minimization algorithm for unsupervised tagging that improves upon previous two-step heuristics. We show that under a fairly natural assumption of c-feasibility the solution obtained by our minimization algorithm is $O(c \log m)$-approximate to the optimal. Although in the case of two-step heuristics, the first step guarantees an $O(\log m)$-approximation, the second step, which is required to get a consistent solution, can introduce many additional labels resulting in a solution arbitrarily away from the optimal. Our one step approach ensures consistency at each step of the algorithm, while the c-feasibility assumption means that the solution does not diverge too much from the optimal in each iteration.

In addition to proving approximation guarantees for the new algorithm, we show that it is parallelizable, allowing us to easily scale to larger datasets than previously explored. Our results show that the algorithm achieves state-of-the-art performance, outperforming existing methods on several different tasks (both POS tagging and supertagging) and works well even with incomplete dictionaries and extremely low-resource languages like Malagasy.

For future work, it would be interesting to apply a weighted version of the DMLC algorithm where labels (i.e., tag pairs) can have different weight distributions instead of uniform weights. Our algorithm can be extended to allow an input weight distribution to be specified for minimization. In order to initialize the weights we could use existing strategies such as grammar-informed initialization (Ravi et al., 2010a) or output distributions learnt via other methods such as label propagation (Garrette and Baldridge, 2013).

References

2013. Apache giraph. `http://giraph.apache.org/`.

Michele Banko and Robert C. Moore. 2004. Part-of-speech tagging in context. In *Proceedings of COLING*, pages 556–561.

Andrew R Barron, Jorma Rissanen, and Bin Yu. 1998. The Minimum Description Length Principle in Coding and Modeling. *IEEE Transactions of Information Theory*, 44(6):2743–2760.

Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Leonardo Lesmo. 2000. Building a Treebank for Italian: a data-driven annotation schema. In *Proceedings of the Second International Conference on Language Resources and Evaluation LREC-2000*, pages 99–105.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of CoNLL*, pages 149–164.

Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. 2010. Two decades of unsupervised POS induction: How far have we come? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 575–584.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 600–609.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Dan Garrette and Jason Baldridge. 2012. Type-supervised Hidden Markov Models for part-of-speech tagging with incomplete tag dictionaries. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 821–831.

Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 138–147.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, pages 42–47.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2008. EM can find pretty good HMM POS-taggers (when given a good start). In *Proceedings of ACL*, pages 746–754.

Sharon Goldwater and Thomas L. Griffiths. 2007. A fully Bayesian approach to unsupervised part-of-speech tagging. In *ACL*.

Fernando C. Gomes, Cludio N. Meneses, Panos M. Pardalos, and Gerardo Valdisio R. Viana. 2006. Experimental analysis of approximation algorithms for the vertex cover and set covering problems.

Kazi Saidul Hasan and Vincent Ng. 2009. Weakly supervised part-of-speech tagging for morphologically-rich, resource-scarce languages. In *Proceedings of the 12th Conference on the European Chapter of the Association for Computational Linguistics*, pages 363–371.

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

Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Machine Translation Summit X*, pages 79–86.

Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. 2010. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

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

Taesun Moon, Katrin Erk, and Jason Baldridge. 2010. Crouching Dirichlet, Hidden Markov Model: Unsupervised POS tagging with context local tag generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 196–206.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, pages 915–932.

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conferenceof the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, pages 504–512.

Sujith Ravi, Jason Baldridge, and Kevin Knight. 2010a. Minimized models and grammar-informed initialization for supertagging with highly ambiguous lexicons. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 495–503.

Sujith Ravi, Ashish Vaswani, Kevin Knight, and David Chiang. 2010b. Fast, greedy model minimization for unsupervised tagging. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 940–948.

Roi Reichart, Raanan Fattal, and Ari Rappoport. 2010. Improved unsupervised POS induction using intrinsic clustering quality and a Zipfian constraint. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 57–66.

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

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