Learning a Part-of-Speech Tagger from Two Hours of Annotation

Dan Garrette
Department of Computer Science
The University of Texas at Austin
dhg@cs.utexas.edu

Jason Baldridge
Department of Linguistics
The University of Texas at Austin
jbaldrid@utexas.edu

Abstract

Most work on weakly-supervised learning for part-of-speech taggers has been based on unrealistic assumptions about the amount and quality of training data. For this paper, we attempt to create true low-resource scenarios by allowing a linguist just two hours to annotate data and evaluating on the languages Kinyarwanda and Malagasy. Given these severely limited amounts of either type supervision (tag dictionaries) or token supervision (labeled sentences), we are able to dramatically improve the learning of a hidden Markov model through our method of automatically generalizing the annotations, reducing noise, and inducing word-tag frequency information.

1 Introduction

The high performance achieved by part-of-speech (POS) taggers trained on plentiful amounts of labeled word tokens is a success story of computational linguistics (Manning, 2011). However, research on learning taggers using type supervision (e.g. tag dictionaries or morphological transducers) has had a more checkered history. The setting is a seductive one: by labeling the possible parts-of-speech for high frequency words, one might learn accurate taggers by incorporating the type information as constraints to a semi-supervised generative learning model like a hidden Markov model (HMM). Early work showed much promise for this strategy (Kupiec, 1992; Merialdo, 1994), but successive efforts in recent years have continued to peel away and address layers of unrealistic assumptions about the size, coverage, and quality of the tag dictionaries that had been used (Toutanova and Johnson, 2008; Ravi and Knight, 2009; Hasan and Ng, 2009; Garrette and Baldridge, 2012). This paper attempts to strip away further layers so we can build better intuitions about the effectiveness of type-supervised and token-supervised strategies in a realistic setting of POS-tagging for low-resource languages.

In most previous work, tag dictionaries are extracted from a corpus of annotated tokens. To explore the type-supervised scenario, these have been used as a proxy for dictionaries produced by linguists. However, this overstates their effectiveness. Researchers have often manually *pruned* tag dictionaries by removing low-frequency word/tag pairs; this violates the assumption that frequency information is not available. Others have also created tag dictionaries by extracting every word/tag pair in a large, labeled corpus, *including the test data*—even though actual applications would never have such complete lexical knowledge. Dictionaries extracted from corpora are also biased towards including only the most likely tag for each word type, resulting in a cleaner dictionary than one would find in real scenario. Finally, tag dictionaries extracted from annotated tokens benefit from the annotation process of labeling *and* review *and* refinement over an extended collaboration period. Such high quality annotations are simply not available for most low-resource languages.

This paper describes an approach to learning a POS-tagger that can be applied in a truly low-resource scenario. Specifically, we discuss techniques that allow us to learn a tagger given only

Proceedings of NAACL-HLT 2013, pages 138–147,
Atlanta, Georgia, 9–14 June 2013.

the amount of labeled data that a human annotator could provide in *two hours*. Here, we evaluate on the languages Malagasy and Kinyarwanda, as well as English as a control language. Furthermore, we are interested in whether type-supervision or token-supervision is more effective, given the strict time constraint; accordingly, we had annotators produce both a tag dictionary and a set of labeled sentences.

The data produced under our conditions differs in several ways from the labeled data used in previous work. Most obviously, there is less of it. Instead of using hundreds of thousands of labeled tokens to construct a tag dictionary (and hundreds of thousands more as unlabeled (raw) data for training), we only use the 1k-2k labeled tokens or types provided by our annotators within the timeframe. Our training data is also much noisier than the data from a typical corpus: the annotations were produced by a single non-native-speaker working alone for two hours. Therefore, dealing with the size and quality of training data were core challenges to our task.

To learn a POS-tagger from so little labeled data, we developed an approach that starts by generalizing the initial annotations to the entire raw corpus. Our approach uses label propagation (LP) (Talukdar and Crammer, 2009) to infer tag distributions on unlabeled tokens. We then apply a novel *weighted* variant of the model minimization procedure originally developed by Ravi and Knight (2009) to estimate sequence and word-tag frequency information from an unlabeled corpus by approximating the minimal set of tag bigrams needed to explain the data. This combination of techniques turns a tiny, unweighted, initial tag dictionary into a weighted tag dictionary that covers the entire corpus's vocabulary. This weighted information limits the potential damage of tag dictionary noise and bootstraps frequency information to approximate a good starting point for the learning of an HMM using expectation-maximization (EM), and far outperforms just using EM on the raw annotations themselves.

2 Data

Our experiments use Kinyarwanda (KIN), Malagasy (MLG), and English (ENG). KIN is a Niger-Congo language spoken in Rwanda. MLG is an Austronesian language spoken in Madagascar. Both KIN and MLG are low-resource and KIN is morphologically-rich. For each language, the word tokens are divided into four sets: training data to be labeled by annotators, raw training data, development data, and test data. For consistency, we use 100k raw tokens for each language.

Data sources For ENG, we used the Penn Treebank (PTB) (Marcus et al., 1993). Sections 00-04 were used as raw data, 05-14 as a dev set, and 15-24 (473K tokens) as a test set. The PTB uses 45 distinct POS tags. The KIN texts are transcripts of testimonies by survivors of the Rwandan genocide provided by the Kigali Genocide Memorial Center. The MLG texts are articles from the websites[1] *Lakroa* and *La Gazette* and Malagasy Global Voices,[2] a citizen journalism site.[3] Texts in both KIN and MLG were tokenized and labeled with POS tags by two linguistics graduate students, each of which was studying one of the languages. The KIN and MLG data have 14 and 24 distinct POS tags, respectively, and were developed by the annotators.

Time-bounded annotation One of our main goals is to evaluate POS-tagging for low-resource languages in experiments that correspond better to a real-world scenario than previous work. As such, we collected two forms of annotation, each constrained by a two-hour time limit. The annotations were done by the same linguists who had annotated the KIN and MLG data mentioned above. Our experiments are thus relevant to the reasonable context in which one has access to a linguist who is familiar with the target language and a given set of POS tags.

The first annotation task was to directly produce a dictionary of words to their possible POS tags—i.e., collecting an actual tag dictionary of the form that is typically *simulated* in POS-tagging experiments. For each language, we compiled a list of word types, ordered starting with most frequent, and presented it to the annotator with a list of admissible POS tags. The annotator had two hours to specify POS tags for as many words as possible. The word types and frequencies used for this task were taken from the raw training data and did not include the test sets. This

[1] `www.lakroa.mg` and `www.lagazette-dgi.com`

[2] `mg.globalvoicesonline.org/`

[3] The public-domain data is available at `github.com/dhgarrette/low-resource-pos-tagging-2013`

data is used for what will call *type-supervised* training. The second task was annotating full sentences with POS tags, again for two hours. We refer to this as *token-supervised* training.

Having both sets of annotations allows us to investigate the relative value of each with respect to training taggers. Token-supervision provides valuable frequency and tag context information, but type-supervision produces larger dictionaries. This can be seen in Table 1, where the dictionary size column in the table gives the number of unique word/tag pairs derived from the data.

We also wanted to directly compare the two annotators to see how the differences in their relative annotation speeds and quality would affect the overall ability to learn an accurate tagger. We thus had them complete the same two tasks for English. As can be seen in Table 1, there are clear differences between the two annotators. Most notably, annotator B was faster at annotating full sentences while annotator A was faster at annotating word types.

	sent.	tok.	dict.
KIN human sentences A	90	1537	750
KIN human TD A			1798
MLG human sentences B	92	1805	666
MLG human TD B			1067
ENG human sentences A	86	1897	903
ENG human TD A			1644
ENG human sentences B	107	2650	959
ENG human TD B			1090

Table 1: Statistics for Kinyarwanda, Malagasy, and English data annotated by annotators A and B.

3 Approach

Our approach to learning POS-taggers is based on Garrette and Baldridge (2012), which properly separated test data from learning data, unlike much previous work. The input to our system is a raw corpus and either a human-generated tag dictionary or human-tagged sentences. The majority of the system is the same for both kinds of labeled training data, but the following description will point out differences. The system has four main parts, in order:

1. Tag dictionary expansion
2. Weighted model minimization
3. Expectation maximization (EM) HMM training
4. MaxEnt Markov Model (MEMM) training

3.1 Tag dictionary expansion

In a low-resource setting, most word types will not be found in the initial tag dictionary. EM-HMM training uses the tag dictionary to limit ambiguity, so a sparse tag dictionary is problematic because it does not sufficiently confine the parameter space.[4] Small dictionaries also interact poorly with the model minimization of Ravi et al. (2010): if there are too many unknown words, and every tag must be considered for them, then the minimal model will simply be the one that assumes that they all have the same tag.

For these reasons, we automatically *expand* an initial small dictionary into one that has coverage for most of the vocabulary. We use label propagation (LP)—specifically, the Modified Adsorption (MAD) algorithm (Talukdar and Crammer, 2009)[5]—which is a graph-based technique for spreading labels between related items. Our graphs connect token nodes to each other via feature nodes and are seeded with POS-tag labels from the human-annotated data.

Defining the LP graph Our LP graph has several types of nodes, as shown in Figure 1. The graph contains a TOKEN node for each token of the labeled corpus (when available) and raw corpus. Each word type has one TYPE node that is connected to its TOKEN nodes. Both kinds of nodes are connected with *feature* nodes. The PREVWORD_x and NEXTWORD_x nodes represent the features of a token being preceded by or followed by word type x in the corpus. These *bigram* features capture extremely simple syntactic information. To capture shallow morphological relatedness, we use *prefix* and *suffix* nodes that connect word types that share prefix or suffix character sequences up to length 5. For each node-feature pair, the connecting edge is weighted as $1/N$ where N is the number of nodes connected to the particular feature.

[4] This is of course not the case for purely unsupervised taggers, though we also note that it is not at all clear they are actually learning taggers for part-of-speech.

[5] The open-source MAD implementation is provided through Junto: `github.com/parthatalukdar/junto`

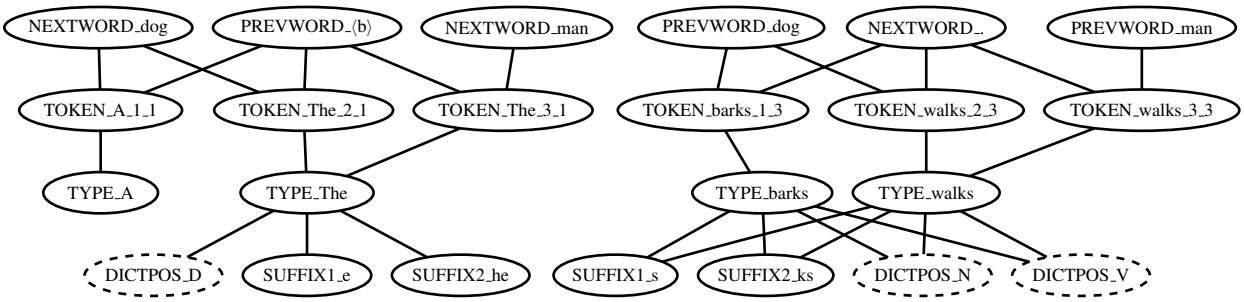

Figure 1: Subsets of the LP graph showing regions of connected nodes. Graph represents the sentences "A dog barks .", "The dog walks .", and "The man walks ."

We also explored the effectiveness of using an external dictionary in the graph since this is one of the few available sources of information for many low-resource languages. Though a standard dictionary probably will not use the same POS tag set that we are targeting, it nevertheless provides information about the relatedness of various word types. Thus, we use nodes DICTPOS_p that indicate that a particular word type is listed as having POS p in the dictionary. Crucially, *these tags bear no particular connection to the tags we are predicting*: we still target the tags defined by the linguist who annotated the types or tokens used, which may be more or less granular than those provided in the dictionary. As external dictionaries, we use English Wiktionary (614k entries), `malagasyworld.org` (78k entries), and `kinyarwanda.net` (3.7k entries).[6]

Seeding the graph is straightforward. With token-supervision, labels for tokens are injected into the corresponding TOKEN nodes with a weight of 1.0. In the type-supervised case, any TYPE node that appears in the tag dictionary is injected with a uniform distribution over the tags in its tag dictionary entry.

Toutanova and Johnson (2008) (also, Ravi and Knight (2009)) use a simple method for predicting possible tags for unknown words: a set of 100 most common suffixes are extracted and then models of *P(tag|suffix)* are built and applied to unknown words. However, these models suffer with an extremely small set of labeled data. Our method uses character affix feature nodes along with *sequence* feature nodes in the LP graph to get distributions over unknown words. Our technique thus subsumes theirs as it can infer tag dictionary entries for words whose suffixes do not show up in the labeled data (or with enough frequency to be reliable predictors).

Extracting a result from LP LP assigns a label distribution to every node. Importantly, each individual TOKEN gets its own distribution instead of sharing an aggregation over the entire word type. From this graph, we extract a new version of the raw corpus that contains tags for each token. This provides the input for model minimization.

We seek a small set of likely tags for each token, but LP gives each token a distribution over the entire set of tags. Most of the tags are simply noise, some of which we remove by normalizing the weights and excluding tags with probability less than 0.1. After applying this cutoff, the weights of the remaining tags are re-normalized. We stress that this tag dictionary cutoff is not like those used in past research, which were done with respect to frequencies obtained from labeled tokens: we use either no word-tag frequency information (type-supervision) or very small amounts of word-tag frequency information indirectly through LP (token-supervision).[7]

Some tokens might not have any associated tag labels after LP. This occurs when there is no path from a TOKEN node to any seeded nodes or when all tags for the TOKEN node have weights less than the threshold. Since we require a distribution for every token, we use a default distribution for such cases. Specifically, we use the unsupervised emission probability initialization of Garrette and Baldridge (2012), which captures both the estimated *frequency* of a tag and its *openness* using only a

[6] Wiktionary (`wiktionary.org`) has only 3,365 entries for Malagasy and 9 for Kinyarwanda.

[7] See Banko and Moore (2004) for further discussion of these issues.

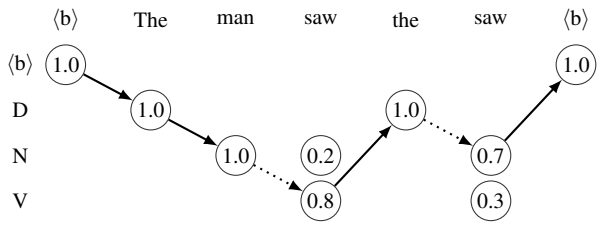

Figure 2: Weighted, greedy model minimization graph showing a potential state between the stages of the tag bigram choosing algorithm. Solid edges: selected bigrams. Dotted edges: holes in the path.

small tag dictionary and unlabeled text.

Finally, we ensure that tokens of words in the original tag dictionary are only assigned tags from its entry. With this filter, LP of course does not add new tags to known words (without it, we found performance drops). If the intersection of the small tag dictionary entry and the token's resulting distribution from LP (after thresholding) is empty, we fall back to the filtered and renormalized *default* distribution for that token's type.

The result of this process is a sequence of (initially raw) tokens, each associated with a distribution over a subset of tags. From this we can extract an *expanded* tag dictionary for use in subsequent stages that, crucially, provides tag information for words not covered by the human-supplied tag dictionary. This expansion is simple: an unknown word type's set of tags is the union of all tags assigned to its tokens. Additionally, we add the full entries of word types given in the original tag dictionary.

3.2 Weighted model minimization

EM-HMM training depends crucially on having a clean tag dictionary and a good starting point for the emission distributions. Given only raw text and a tag dictionary, these distributions are difficult to estimate, especially in the presence of a very sparse or noisy tag dictionary. Ravi and Knight (2009) use model minimization to remove tag dictionary noise and induce tag frequency information from raw text. Their method works by finding a minimal set of tag bigrams needed to explain a raw corpus.

Model minimization is a natural fit for our system since we start with little or no frequency information and automatic dictionary expansion introduces noise. We extend the greedy model minimization procedure of Ravi et al. (2010), and its enhancements by Garrette and Baldridge (2012), to develop a novel *weighted* minimization procedure that uses the tag weights from LP to find a minimal model that is biased toward keeping tag bigrams that have consistently high weights across the entire corpus. The new weighted minimization procedure fits well in our pipeline by allowing us to carry the tag distributions forward from LP instead of simply throwing that information away and using a traditional tag dictionary.

In brief, the procedure works by creating a graph such that each possible tag of each raw-corpus token is a vertex (see Figure 2). Any edge that would connect two tags of adjacent tokens is a potential tag bigram choice. The algorithm first selects tag bigrams until every token is covered by at least one bigram, then selects tag bigrams that fill gaps between existing edges until there is a complete bigram path for every sentence in the raw corpus.[8]

Ravi et al. (2010) select tag bigrams that cover the most new words (stage 1) or fill the most holes in the tag paths (stage 2). Garrette and Baldridge (2012) introduced the tie-breaking criterion that bigram choices should seek to introduce the smallest number of new *word/tag pairs* possible into the paths. Our criteria adds to this by using the tag weights on each token: a tag bigram b is chosen by summing up the node weights of any not-yet covered words touched by the tag bigram b, dividing this sum by one plus the number of *new* word/tag pairs that would be added by b, and choosing the b that maximizes this value.[9]

Summing node weights captures the intuition of Ravi et al. (2010) that good bigrams are those which have high coverage of new words: each newly covered node contributes additional (partial) counts. However, by using the weights instead of full counts, we also account for the confidence assigned by LP. Dividing by the number of new word/tag pairs added focuses on bigrams that reuse existing tags for words

[8] Ravi et al. (2010) include a third phase of iterative model fitting; however, we found this stage to be not only expensive, but also unhelpful because it frequently yields negative results.

[9] In the case of token-supervision, we pre-select all tag bigrams appearing in the labeled corpus since these are assumed to be known high-quality tag bigrams and word/tag pairs.

and thereby limits the addition of new tags for each word type.

At the start of model minimization, there are no selected tag bigrams, and thus no valid path through any sentence in the corpus. As bigrams are selected, we can begin to cover subsequences and eventually full sentences. There may be multiple valid taggings for a sentence, so after each new bigram is selected, we run the Viterbi algorithm over the raw corpus using the set of selected tag bigrams as a hard constraint on the allowable transitions. This efficiently identifies the highest-weight path through each sentence, if one exists. If such a path is found, we remove the sentence from the corpus and store the tags from the Viterbi tagging. The algorithm terminates when a path is found for every raw corpus sentence. The result of weighted model minimization is this set of tag paths. Since each path represents a valid tagging of the sentence, we use this output as a noisily labeled corpus for initializing EM in stage three.

3.3 Tagger training

Stage one provides an expansion of the initial labeled data and stage two turns that into a corpus of noisily labeled sentences. Stage three uses the EM algorithm initialized by the noisy labeling and constrained by the expanded tag dictionary to produce an HMM.[10] The initial distributions are smoothed with one-count smoothing (Chen and Goodman, 1996). If human-tagged sentences are available as training data, then we use their counts to supplement the noisy labeled text for initialization and we add their counts into every iteration's result.

The HMM produced by stage three is not used directly for tagging since it will contain zero-probabilities for test-corpus words that were unseen during training. Instead, we use it to provide a Viterbi labeling of the raw corpus, following the "auto-supervision" step of Garrette and Baldridge (2012). This material is then concatenated with the token-supervised corpus (when available), and used to train a Maximum Entropy Markov Model tagger.[11] The MEMM exploits subword features and generally produces 1-2% better results than an HMM trained on the same material.

4 Experiments[12]

Experimental results are shown in Table 2. Each experiment starts with an initial data set provided by annotator A or B. Experiment (1) simply uses EM with the initial small tag dictionary to learn a tagger from the raw corpus. (2) uses LP to infer an expanded tag dictionary and tag distributions over raw corpus tokens, but then takes the highest-weighted tag from each token for use as noisily-labeled training data to initialize EM. (3) performs greedy model-minimization on the LP output to derive that noisily-labeled corpus. Finally, (4) is the same as (3), but additionally uses external dictionary nodes in the LP graph. In the case of token-supervision, we also include (0), in which we simply used the tagged sentences as supervised data for an HMM without EM (followed by MEMM training).

The results show that performance improves with our LP and minimization techniques compared to basic EM-HMM training. LP gives large across-the-board improvements over EM training with only the original tag dictionary (compare columns 1 & 2). Weighted model minimization further improves results for type-supervision settings, but not for token supervision (compare 2 & 3).

Using an external dictionary in the LP graph has little effect for KIN, probably due to the available dictionary's very small size. However, MLG with its larger dictionary obtains an improvement in both scenarios. Results on ENG are mixed; this may be because the PTB tagset has 45 tags (far more than the dictionary) so the external dictionary nodes in the LP graph may consequently serve to collapse distinctions (e.g. singular and plural) in the larger set.

Our results show differences between token- and type-supervised annotations. Tag dictionary expansion is helpful no matter what the annotations look like: in both cases, the initial dictionary is too small for effective EM learning, so expansion is necessary. However, model minimization only benefits the type-supervised scenarios, leaving token-supervised performance unchanged. This suggests

[10] An added benefit of this strategy is that the EM algorithm with the expanded dictionary runs much more quickly than without it since it does not have to consider every possible tag for unknown words, averaging 20x faster on PTB experiments.

[11] We use OpenNLP: `opennlp.apache.org`.

[12] Our code is available at `github.com/dhgarrette/low-resource-pos-tagging-2013`

Human Annotations	0. No EM			1. EM only			2. With LP			3. LP+min			4. LP(ed)+min		
Initial data	T	K	U	T	K	U	T	K	U	T	K	U	T	K	U
KIN tokens A	**72**	90	58	55	82	32	71	86	58	71	86	58	71	86	58
KIN types A				63	77	32	78	83	69	**79**	83	70	**79**	83	70
MLG tokens B	74	89	49	68	87	39	74	89	49	74	89	49	**76**	90	53
MLG types B				71	87	46	72	81	57	74	86	56	**76**	86	60
ENG tokens A	63	83	38	62	83	37	**72**	85	55	**72**	85	55	**72**	85	56
ENG types A				66	76	37	75	81	56	**76**	83	56	74	81	55
ENG tokens B	70	87	44	70	87	43	**78**	90	60	**78**	90	60	**78**	89	61
ENG types B				69	83	38	75	82	61	**78**	85	61	**78**	86	61

Table 2: Experimental results. Three languages are shown: Kinyarwanda (KIN), Malagasy (MLG), and English (ENG). The letters A and B refer to the annotator. LP(ed) refers to label propagation including nodes from an external dictionary. Each result given as percentages for Total (T), Known (K), and Unknown (U).

that minimization is working as intended: it induces frequency information when none is provided. With token-supervision, the annotator provides some information about which tag transitions are best and which emissions are most likely. This is missing with type-supervision, so model minimization is needed to bootstrap word/tag frequency guesses.

This leads to perhaps our most interesting result: in a time-critical annotation scenario, it seems better to collect a simple tag dictionary than tagged sentences. While the tagged sentences certainly contain useful information regarding tag frequencies, our techniques can learn this missing information automatically. Thus, having wider coverage of word type information, and having that information be focused on the most frequent words, is more important. This can be seen as a validation of the last two decades of work on (simulated) type-supervision learning for POS-tagging—with the caveat that the additional effort we do is needed to realize the benefit.

Our experiments also allow us to compare how the data from different annotators affects the quality of taggers learned. Looking at the direct comparison on English data, annotator B was able to tag more sentences than A, but A produced more tag dictionary entries in the type-supervision scenario. However, it appears, based on the EM-only training, that the annotations provided by B were of higher quality and produced more accurate taggers in both scenarios. Regardless, our full training procedure is able to substantially improve results in all scenarios.

Table 3 gives the recall and precision of the tag dictionaries for MLG for settings 1, 2 and 3. The initial, human-provided tag dictionary unsurprisingly has the highest precision and lowest recall. LP expands that data to greatly improve recall with a large drop in precision. Minimization culls many entries and improves precision with a small relative loss in recall. Of course, this is only a rough indicator of the quality of the tag dictionaries since the word/tag pairs of the test set only partially overlap with the raw training data and annotations.

Tag Dictionary Source	R	P
(1) human-annotated TD	18.42	29.33
(2) LP output	35.55	2.62
(3) model min output	30.49	4.63

Table 3: Recall (R) and precision (P) for tag dictionaries versus the test data in a "MLG types B" run.

Because gold-standard annotations are available for the English sentences, we also ran *oracle* experiments using labels from the PTB corpus (essentially, the kind of data used in previous work). We selected the same amount of labeled tokens or word/tag pairs as were obtained by the annotators. We found similar patterns of improved performance by using LP expansion and model minimization, and all accuracies are improved compared to their human-annotator equivalents (about 2-6%). Overall accuracy for both type and token supervision comes to 78-80%.

#Errors	11k	6k	5k	4k	3k
Gold	TO	NNP	NN	JJ	NNP
Model	IN	NN	JJ	NN	JJ

Table 4: Top errors from an "ENG types B" run.

Error Analysis One potential source of errors comes directly from the annotators themselves. Though our approach is designed to be robust to annotation errors, it cannot correct all mistakes. For example, for the "ENG types B" experiment, the annotator listed IN (preposition) as the only tag for word type "to". However, the test set only ever assigns tag TO for this type. This single error accounts for a 2.3% loss in overall tagging accuracy (Table 4).

In many situations, however, we are able to automatically remove improbable tag dictionary entries, as shown in Table 5. Consider the word type "for". The annotator has listed RP (particle) as a potential tag, but only five out of 4k tokens have this tag. With RP included, EM becomes confused and labels a majority of the tokens as RP when nearly all should be labeled IN. We are able to eliminate RP as a possibility, giving excellent overall accuracy for the type. Likewise for the comma type, the annotator has incorrectly given ":" as a valid tag, and LP, which uses the tag dictionary, pushes this label to many tokens with high confidence. However, minimization is able to correct the problem.

Finally, the word type "opposition" provides an example of the expected behavior for unknown words. The type is not in the tag dictionary, so EM assumes all tags are valid and uses many labels. LP expands the starting dictionary to cover the type, limiting it to only two tags. Minimization then determines that NN is the best tag for each token.

for	*IN	*RP	JJ	NN	CD
(1) EM	1,221	2764		9	5
(2) LP	4,003				
(3) min	4,004		1		
gold	3,999	5			
, (comma)	*,	*:	JJS	PTD	VBP
(1) EM	24,708		4	3	3
(2) LP	15,505	9226			1
(3) min	24,730				
gold	24,732				
opposition	NN	JJ	DT	NNS	VBP
(1) EM	24	4	1	4	4
(2) LP	41	4			
(3) min	45				
gold	45				

Table 5: Tag assignments in different scenarios. A star indicates an entry in the human-provided TD.

5 Related work

Goldberg et al. (2008) trained a tagger for Hebrew using a manually-created lexicon which was not derived from an annotated corpus. However, their lexicon was constructed by trained lexicographers over a long period of time and achieves very high coverage of the language with very good quality. In contrast, our annotated data was created by untrained linguistics students working alone for just two hours.

Cucerzan and Yarowsky (2002) learn a POS-tagger from existing linguistic resources, namely a dictionary and a reference grammar, but these resources are not available, much less digitized, for most under-studied languages.

Subramanya et al. (2010) apply LP to the problem of tagging for domain adaptation. They construct an LP graph that connects tokens in low- and high-resource domains, and propagate labels from high to low. This approach addresses the problem of learning appropriate tags for unknown words within a language, but it requires that the language have at least one high-resource domain as a source of high quality information. For low-resource languages that have no significant annotated resources available in *any* domain, this technique cannot be applied.

Das and Petrov (2011) and Täckström et al. (2013) learn taggers for languages in which there are no POS-annotated resources, but for which parallel texts are available between that language and a high-resource language. They project tag information from the high-resource language to the lower-resource language via alignments in the parallel text. However, large parallel corpora are not available for most low-resource languages. These are also expensive resources to create and would take considerably more effort to produce than the monolingual resources that our annotators were able to generate

in a two-hour timeframe. Of course, if they are available, such parallel text links could be incorporated into our approach.

Furthermore, their approaches require the use of a universal tag set shared between both languages. As such, their approach is only able to induce POS tags for the low-resource language if the same tag set is used to tag the high-resource language. Our approach does not rely on any such universal tag set; we learn whichever tags the human annotator chooses to use when they provide their annotations. In fact, in our experiments we learn much more detailed tag sets than the fairly coarse universal tag set used by Das and Petrov (2011) or Täckström et al. (2013): we learn a tagger for the full Penn Treebank tag set of 45 tags versus the 12 tags in the universal set.

Ding (2011) constructed an LP graph for learning POS tags on Chinese text by propagating labels from an initial tag dictionary to a larger set of data. This LP graph contained Wiktionary word/POS relationships as features as well as Chinese-English word alignment information and used it to directly estimate emission probabilities to initialize an EM training of an HMM.

Li et al. (2012) train an HMM using EM and an initial tag dictionary derived from Wiktionary. Like Das and Petrov (2011), they use a universal POS tag set, so Wiktionary can be directly applied as a wide-coverage tag dictionary in their case. Additionally, they evaluate their approach on languages for which Wiktionary has high coverage—which would certainly not get far with Kinyarwanda (9 entries). Our approach does not rely on a high-coverage tag dictionary nor is it restricted to use with a small tag set.

6 Conclusions and future work

With just two hours of annotation, we obtain 71-78% accuracy for POS-tagging across three languages using both type and token supervision. Without tag dictionary expansion and model minimization, performance is much worse, from 63-74%. We dramatically improve performance on unknown words: the range of 37-58% improves to 53-70%.

We also have a provisional answer to whether annotation should be on types or tokens: use type-supervision if you also expand and minimize. These methods can identify missing word/tag entries and estimate frequency information, and they produce as good or better results compared to starting with token supervision. The case of Kinyarwanda was most dramatic: 71% accuracy for token-supervision compared to 79% for type-supervision. Studies using more annotators and across more languages would be necessary to make any stronger claim about the relative efficacy of the two strategies.

Acknowledgements

We thank Kyle Jerro, Vijay John, Katrin Erk, Yoav Goldberg, Ray Mooney, Slav Petrov, Oscar Täckström, and the reviewers for their assistance and feedback. This work was supported by the U.S. Department of Defense through the U.S. Army Research Office (grant number W911NF-10-1-0533) and via a National Defense Science and Engineering Graduate Fellowship for the first author. Experiments were run on the UTCS Mastodon Cluster, provided by NSF grant EIA-0303609.

References

Michele Banko and Robert C. Moore. 2004. Part-of-speech tagging in context. In *Proceedings of COLING*, Geneva, Switzerland.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of ACL*, Santa Cruz, California, USA.

Silviu Cucerzan and David Yarowsky. 2002. Bootstrapping a multilingual part-of-speech tagger in one person-day. In *Proceedings of CoNLL*, Taipei, Taiwan.

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL-HLT*, Portland, Oregon, USA.

Weiwei Ding. 2011. Weakly supervised part-of-speech tagging for Chinese using label propagation. Master's thesis, University of Texas at Austin.

Dan Garrette and Jason Baldridge. 2012. Type-supervised hidden Markov models for part-of-speech tagging with incomplete tag dictionaries. In *Proceedings of EMNLP*, Jeju, Korea.

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

Kazi Saidul Hasan and Vincent Ng. 2009. Weakly supervised part-of-speech tagging for morphologically-rich,

resource-scarce languages. In *Proceedings of EACL*, Athens, Greece.

Julian Kupiec. 1992. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language*, 6(3).

Shen Li, João Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of EMNLP*, Jeju Island, Korea.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Proceedings of CICLing*.

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

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

Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proceedings of ACL-AFNLP*.

Sujith Ravi, Ashish Vaswani, Kevin Knight, and David Chiang. 2010. Fast, greedy model minimization for unsupervised tagging. In *Proceedings of COLING*.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings EMNLP*, Cambridge, MA.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. In *Transactions of the ACL*. Association for Computational Linguistics.

Partha Pratim Talukdar and Koby Crammer. 2009. New regularized algorithms for transductive learning. In *Proceedings of ECML-PKDD*, Bled, Slovenia.

Kristina Toutanova and Mark Johnson. 2008. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of NIPS*.