Fast and Robust Part-of-Speech Tagging Using Dynamic Model Selection

Jinho D. Choi
Department of Computer Science
University of Colorado Boulder
choijd@colorado.edu

Martha Palmer
Department of Linguistics
University of Colorado Boulder
mpalmer@colorado.edu

Abstract

This paper presents a novel way of improving POS tagging on heterogeneous data. First, two separate models are trained (generalized and domain-specific) from the same data set by controlling lexical items with different document frequencies. During decoding, one of the models is selected dynamically given the cosine similarity between each sentence and the training data. This dynamic model selection approach, coupled with a one-pass, left-to-right POS tagging algorithm, is evaluated on corpora from seven different genres. Even with this simple tagging algorithm, our system shows comparable results against other state-of-the-art systems, and gives higher accuracies when evaluated on a mixture of the data. Furthermore, our system is able to tag about 32K tokens per second. We believe that this model selection approach can be applied to more sophisticated tagging algorithms and improve their robustness even further.

1 Introduction

When it comes to POS tagging, two things must be checked. First, a POS tagger needs to be tested for its robustness in handling heterogeneous data.[1] Statistical POS taggers perform very well when their training and testing data are from the same source, achieving over 97% tagging accuracy (Toutanova et al., 2003; Giménez and Màrquez, 2004; Shen et al., 2007). However, the performance degrades increasingly as the discrepancy between the training and testing data gets larger. Thus, to ensure robustness, a tagger needs to be evaluated on several different kinds of data. Second, a POS tagger should be tested for its speed. POS tagging is often performed as a pre-processing step to other tasks (e.g., parsing, chunking) and it should not be a bottleneck for those tasks. Moreover, recent NLP tasks deal with very large-scale data where tagging speed is critical.

To improve robustness, we first train two separate models; one is optimized for a general domain and the other is optimized for a domain specific to the training data. During decoding, we dynamically select one of the models by measuring similarities between input sentences and the training data. Our hypothesis is that the domain-specific and generalized models perform better for sentences similar and not similar to the training data, respectively. In this paper, we describe how to build both models using the same training data and select an appropriate model given input sentences during decoding. Each model uses a one-pass, left-to-right POS tagging algorithm. Even with the simple tagging algorithm, our system gives results that are comparable to two other state-of-the-art systems when coupled with this dynamic model selection approach. Furthermore, our system shows noticeably faster tagging speed compared to the other two systems.

For our experiments, we use corpora from seven different genres (Weischedel et al., 2011; Nielsen et al., 2010). This allows us to check the performance of each system on different kinds of data when run individually or selectively. To the best of our knowledge, this is the first time that a POS tagger has been evaluated on such a wide variety of data in English.

[1] We use the term "heterogeneous data" as a mixture of data collected from several different sources.

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 363–367,
Jeju, Republic of Korea, 8-14 July 2012.

2 Approach

2.1 Training generalized and domain-specific models using document frequency

Consider training data as a collection of documents where each document contains sentences focusing on a similar topic. For instance, in the Wall Street Journal corpus, a document can be an individual file or all files within each section.[2] To build a generalized model, lexical features (e.g., n-gram word-forms) that are too specific to individual documents should be avoided so that a classifier can place more weight on features common to all documents.

To filter out these document-specific features, a threshold is set for the document frequency of each lowercase simplified word-form (LSW) in the training data. A simplified word-form (SW) is derived by applying the following regular expressions sequentially to the original word-form, w. 'replaceAll' is a function that replaces all matches of the regular expression in w (the 1st parameter) with the specific string (the 2nd parameter). In a simplified word, all numerical expressions are replaced with 0.

1. w.replaceAll(`\d%`, 0) (e.g., 1% → 0)
2. w.replaceAll(`\$\d`, 0) (e.g., $1 → 0)
3. w.replaceAll(`^\.\d`, 0) (e.g., .1 → 0)
4. w.replaceAll(`\d(,|:|-|\/|\.)\d`, 0) (e.g., 1,2|1:2|1-2|1/2|1.2 → 0)
5. w.replaceAll(`\d+`, 0) (e.g., 1234 → 0)

A LSW is a decapitalized SW. Given a set of LSW's whose document frequencies are greater than a certain threshold, a model is trained by using only lexical features associated with these LSW's. For a generalized model, we use a threshold of 2, meaning that only lexical features whose LSW's occur in at least 3 documents of the training data are used. For a domain-specific model, we use a threshold of 1.

The generalized and domain-specific models are trained separately; their learning parameters are optimized by running n-fold cross-validation where n is the total number of documents in the training data and grid search on Liblinear parameters c and B (see Section 2.4 for more details about the parameters).

[2]For our experiments, we treat each section of the Wall Street Journal as one document.

2.2 Dynamic model selection during decoding

Once both generalized and domain-specific models are trained, alternative approaches can be adapted for decoding. One is to run both models and merge their outputs. This approach can produce output that is potentially more accurate than output from either model, but takes longer to decode because the merging cannot be processed until both models are finished. Instead, we take an alternative approach, that is to select one of the models dynamically given the input sentence. If the model selection is done efficiently, this approach runs as fast as running just one model, yet can give more robust performance.

The premise of this dynamic model selection is that the domain-specific model performs better for input sentences similar to its training space, whereas the generalized model performs better for ones that are dissimilar. To measure similarity, a set of SW's, say T, used for training the domain-specific model is collected. During decoding, a set of SW's in each sentence, say S, is collected. If the cosine similarity between T and S is greater than a certain threshold, the domain-specific model is selected for decoding; otherwise, the generalized model is selected.

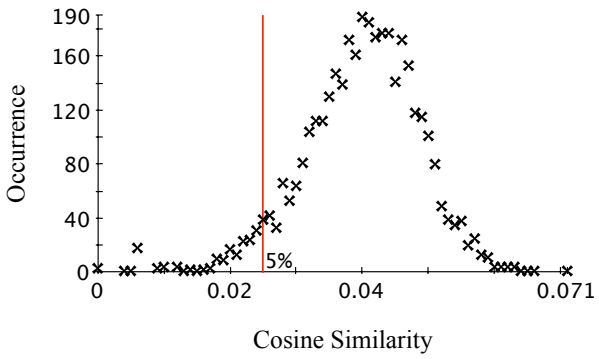

Figure 1: Cosine similarity distribution: the y-axis shows the number of occurrences for each cosine similarity during cross-validation.

The threshold is derived automatically by running cross-validation; for each fold, both models are run simultaneously and cosine similarities of sentences on which the domain-specific model performs better are extracted. Figure 1 shows the distribution of cosine similarities extracted during our cross-validation. Given the cosine similarity distribution, the similarity at the first 5% area (in this case, 0.025) is taken as the threshold.

2.3 Tagging algorithm and features

Each model uses a one-pass, left-to-right POS tagging algorithm. The motivation is to analyze how dynamic model selection works with a simple algorithm first and then apply it to more sophisticated ones later (e.g., bidirectional tagging algorithm).

Our feature set (Table 1) is inspired by Giménez and Màrquez (2004) although ambiguity classes are derived selectively for our case. Given a word-form, we count how often each POS tag is used with the form and keep only ones above a certain threshold. For both generalized and domain-specific models, a threshold of 0.7 is used, which keeps only POS tags used with their forms over 70% of the time. From our experiments, we find this to be more useful than expanding ambiguity classes with lower thresholds.

Lexical	$f_{i\pm\{0,1,2,3\}}$, $(m_{i-2,i-1})$, $(m_{i-1,i})$, $(m_{i-1,i+1})$, $(m_{i,i+1})$, $(m_{i+1,i+2})$, $(m_{i-2,i-1,i})$, $(m_{i-1,i,i+1})$, $(m_{i,i+1,i+2})$, $(m_{i-2,i-1,i+1})$, $(m_{i-1,i+1,i+2})$
POS	$p_{i-\{3,2,1\}}$, $a_{i+\{0,1,2,3\}}$, $(p_{i-2,i-1})$, $(a_{i+1,i+2})$, (p_{i-1}, a_{i+1}), (p_{i-2}, p_{i-1}, a_i), $(p_{i-2}, p_{i-1}, a_{i+1})$, (p_{i-1}, a_i, a_{i+1}), $(p_{i-1}, a_{i+1}, a_{i+2})$
Affix	$c_{:1}$, $c_{:2}$, $c_{:3}$, $c_{n:}$, $c_{n-1:}$, $c_{n-2:}$, $c_{n-3:}$
Binary	initial uppercase, all uppercase/lowercase, contains 1/2+ capital(s) not at the beginning, contains a (period/number/hyphen)

Table 1: Feature templates. i: the index of the current word, f: SW, m: LSW, p: POS, a: ambiguity class, c_*: character sequence in w_i (e.g., $c_{:2}$: the 1st and 2nd characters of w_i, $c_{n-1:}$: the n-1'th and n'th characters of w_i). See Giménez and Màrquez (2004) for more details.

2.4 Machine learning

Liblinear L2-regularization, L1-loss support vector classification is used for our experiments (Hsieh et al., 2008). From several rounds of cross-validation, learning parameters of ($c = 0.2$, $e = 0.1$, $B = 0.4$) and ($c = 0.1$, $e = 0.1$, $B = 0.9$) are found for the generalized and domain-specific models, respectively (c: cost, e: termination criterion, B: bias).

3 Related work

Toutanova et al. (2003) introduced a POS tagging algorithm using bidirectional dependency networks, and showed the best contemporary results. Giménez and Màrquez (2004) used one-pass, left-to-right and right-to-left combined tagging algorithm and achieved near state-of-the-art results. Shen et al. (2007) presented a tagging approach using guided learning for bidirectional sequence classification and showed current state-of-the-art results.[3]

Our individual models (generalized and domain-specific) are similar to Giménez and Màrquez (2004) in that we use a subset of their features and take one-pass, left-to-right tagging approach, which is a simpler version of theirs. However, we use Liblinear for learning, which trains much faster than their classifier, Support Vector Machines.

4 Experiments

4.1 Corpora

For training, sections 2-21 of the Wall Street Journal (WSJ) from OntoNotes v4.0 (Weischedel et al., 2011) are used. The entire training data consists of 30,060 sentences with 731,677 tokens. For evaluation, corpora from seven different genres are used: the MSNBC broadcasting conversation (BC), the CNN broadcasting news (BN), the Sinorama news magazine (MZ), the WSJ newswire (NW), and the GALE web-text (WB), all from OntoNotes v4.0. Additionally, the Mipacq clinical notes (CN) and the Medpedia articles (MD) are used for evaluation of medical domains (Nielsen et al., 2010). Table 2 shows distributions of these evaluation sets.

4.2 Accuracy comparisons

Our models are compared with two other state-of-the-art systems, the Stanford tagger (Toutanova et al., 2003) and the SVMTool (Giménez and Màrquez, 2004). Both systems are trained with the same training data and use configurations optimized for their best reported results. Tables 3 and 4 show tagging accuracies of all tokens and unknown tokens, respectively. Our individual models (Models D and G) give comparable results to the other systems. Model G performs better than Model D for BC, CN, and MD, which are very different from the WSJ. This implies that the generalized model shows its strength in tagging data that differs from the training data. The dynamic model selection approach (Model S) shows the most robust results across genres, although Models D and G still can perform

[3]Some semi-supervised and domain-adaptation approaches using external data had shown better performance (Daume III, 2007; Spoustová et al., 2009; Søgaard, 2011).

	BC	BN	CN	MD	MZ	NW	WB	Total
Source	MSNBC	CNN	Mipacq	Medpedia	Sinorama	WSJ	ENG	-
Sentences	2,076	1,969	3,170	1,850	1,409	1,640	1,738	13,852
All tokens	31,704	31,328	35,721	34,022	32,120	39,590	34,707	239,192
Unknown tokens	3,077	1,284	6,077	4,755	2,663	983	2,609	21,448

Table 2: Distributions of evaluation sets. The Total column indicates a mixture of data from all genres.

	BC	BN	CN	MD	MZ	NW	WB	Total
Model D	91.81	95.27	87.36	90.74	93.91	97.45	93.93	92.97
Model G	92.65	94.82	88.24	91.46	93.24	97.11	93.51	93.05
Model S	**92.26**	95.13	88.18	**91.34**	**93.88**	**97.46**	93.90	**93.21**
G over D	50.63	36.67	68.80	40.22	21.43	9.51	36.02	41.74
Stanford	87.71	**95.50**	**88.49**	90.86	92.80	97.42	**94.01**	92.50
SVMTool	87.82	95.13	87.86	90.54	92.94	97.31	93.99	92.32

Table 3: Tagging accuracies of all tokens (in %). Models D and G indicate domain-specific and generalized models, respectively and Model S indicates the dynamic model selection approach. "G over D" shows how often Model G is selected over Model D using the dynamic selection (in %).

	BC	BN	CN	MD	MZ	NW	WB	Total
Model S	**60.97**	77.73	68.69	**67.30**	**75.97**	**88.40**	76.27	**70.54**
Stanford	19.24	**87.31**	**71.20**	64.82	66.28	**88.40**	**78.15**	64.32
SVMTool	19.08	78.35	66.51	62.94	65.23	86.88	76.47	47.65

Table 4: Tagging accuracies of unknown tokens (in %).

better for individual genres (except for NW, where Model S performs better than any other model).

For both all and unknown token experiments, Model S performs better than the other systems when evaluated on a mixture of the data (the Total column). The differences are statistically significant for both experiments (McNemar's test, $p < .0001$). The Stanford tagger gives significantly better results for unknown tokens in BN; we suspect that this is where their bidirectional tagging algorithm has an advantage over our simple left-to-right algorithm.

4.3 Speed comparisons

Tagging speeds are measured by running each system on the mixture of all data. Our system and the Stanford system are both written in Java; the Stanford tagger provides APIs that allow us to make fair comparisons between the two systems. The SVMTool is written in Perl, so there is a systematic difference between the SVMTool and our system.

Table 5 shows speed comparisons between these systems. All experiments are evaluated on an Intel Xeon 2.57GHz machine. Our system tags about 32K tokens per second (0.03 milliseconds per token), which includes run-time for both POS tagging and model selection.

	Stanford	SVMTool	Model S
tokens / sec.	421	1,163	31,914

Table 5: Tagging speeds.

5 Conclusion

We present a dynamic model selection approach that improves the robustness of POS tagging on heterogeneous data. We believe that this approach can be applied to more sophisticated algorithms and improve their robustness even further. Our system also shows noticeably faster tagging speed against two other state-of-the-art systems. For future work, we will experiment with more diverse training and testing data and also more sophisticated algorithms.

Acknowledgments

This work was supported by the SHARP program funded by ONC: 90TR0002/01. The content is solely the responsibility of the authors and does not necessarily represent the official views of the ONC.

References

Hal Daume III. 2007. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL'07, pages 256–263.

Jesús Giménez and Lluís Màrquez. 2004. SVMTool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, LREC'04.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. 2008. A Dual Coordinate Descent Method for Large-scale Linear SVM. In *Proceedings of the 25th international conference on Machine learning*, ICML'08, pages 408–415.

Rodney D. Nielsen, James Masanz, Philip Ogren, Wayne Ward, James H. Martin, Guergana Savova, and Martha Palmer. 2010. An architecture for complex clinical question answering. In *Proceedings of the 1st ACM International Health Informatics Symposium*, IHI'10, pages 395–399.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided Learning for Bidirectional Sequence Classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, ACL'07, pages 760–767.

Anders Søgaard. 2011. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ACL'11, pages 48–52.

Drahomíra "johanka" Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. 2009. Semi-supervised Training for the Averaged Perceptron POS Tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL'09, pages 763–771.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, NAACL'03, pages 173–180.

Ralph Weischedel, Eduard Hovy, Martha Palmer, Mitch Marcus, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue. 2011. OntoNotes: A Large Training Corpus for Enhanced Processing. In Joseph Olive, Caitlin Christianson, and John McCary, editors, *Handbook of Natural Language Processing and Machine Translation*. Springer.