Simple semi-supervised training of part-of-speech taggers

Anders Søgaard
Center for Language Technology
University of Copenhagen
soegaard@hum.ku.dk

Abstract

Most attempts to train part-of-speech taggers on a mixture of labeled and unlabeled data have failed. In this work stacked learning is used to reduce tagging to a classification task. This simplifies semi-supervised training considerably. Our prefered semi-supervised method combines tri-training (Li and Zhou, 2005) and disagreement-based co-training. On the Wall Street Journal, we obtain an error reduction of 4.2% with SVMTool (Gimenez and Marquez, 2004).

1 Introduction

Semi-supervised part-of-speech (POS) tagging is relatively rare, and the main reason seems to be that results have mostly been negative. Merialdo (1994), in a now famous negative result, attempted to improve HMM POS tagging by expectation maximization with unlabeled data. Clark et al. (2003) reported positive results with little labeled training data but negative results when the amount of labeled training data increased; the same seems to be the case in Wang et al. (2007) who use co-training of two diverse POS taggers. Huang et al. (2009) present positive results for self-training a simple bigram POS tagger, but results are considerably below state-of-the-art.

Recently researchers have explored alternative methods. Suzuki and Isozaki (2008) introduce a semi-supervised extension of conditional random fields that combines supervised and unsupervised probability models by so-called MDF parameter estimation, which reduces error on Wall Street Journal (WSJ) standard splits by about 7% relative to their supervised baseline. Spoustova et al. (2009) use a new pool of unlabeled data tagged by an ensemble of state-of-the-art taggers in every training step of an averaged perceptron POS tagger with 4–5% error reduction. Finally, Søgaard (2009) stacks a POS tagger on an unsupervised clustering algorithm trained on large amounts of unlabeled data with mixed results.

This work combines a new semi-supervised learning method to POS tagging, namely tri-training (Li and Zhou, 2005), with stacking on unsupervised clustering. It is shown that this method can be used to improve a state-of-the-art POS tagger, SVMTool (Gimenez and Marquez, 2004). Finally, we introduce a variant of tri-training called tri-training with disagreement, which seems to perform equally well, but which imports much less unlabeled data and is therefore more efficient.

2 Tagging as classification

This section describes our dataset and our input tagger. We also describe how stacking is used to reduce POS tagging to a classification task. Finally, we introduce the supervised learning algorithms used in our experiments.

2.1 Data

We use the POS-tagged WSJ from the Penn Treebank Release 3 (Marcus et al., 1993) with the standard split: Sect. 0–18 is used for training, Sect. 19–21 for development, and Sect. 22–24 for testing. Since we need to train our classifiers on material distinct from the training material for our input POS tagger, we save Sect. 19 for training our classifiers. Finally, we use the (untagged) Brown corpus as our unlabeled data. The number of tokens we use for training, developing and testing the classifiers, and the amount of unlabeled data available to it, are thus:

	tokens
train	44,472
development	103,686
test	129,281
unlabeled	1,170,811

Proceedings of the ACL 2010 Conference Short Papers, pages 205–208,
Uppsala, Sweden, 11-16 July 2010.

The amount of unlabeled data available to our classifiers is thus a bit more than 25 times the amount of labeled data.

2.2 Input tagger

In our experiments we use SVMTool (Gimenez and Marquez, 2004) with model type 4 run incrementally in both directions. SVMTool has an accuracy of 97.15% on WSJ Sect. 22-24 with this parameter setting. Gimenez and Marquez (2004) report that SVMTool has an accuracy of 97.16% with an optimized parameter setting.

2.3 Classifier input

The way classifiers are constructed in our experiments is very simple. We train SVMTool and an unsupervised tagger, Unsupos (Biemann, 2006), on our training sections and apply them to the development, test and unlabeled sections. The results are combined in tables that will be the input of our classifiers. Here is an excerpt:[1]

Gold standard	SVMTool	Unsupos
DT	DT	17
NNP	NNP	27
NNP	NNS	17*
NNP	NNP	17
VBD	VBD	26

Each row represents a word and lists the gold standard POS tag, the predicted POS tag and the word cluster selected by Unsupos. For example, the first word is labeled 'DT', which SVMTool correctly predicts, and it belongs to cluster 17 of about 500 word clusters. The first column is blank in the table for the unlabeled section.

Generally, the idea is that a classifier will learn to trust SVMTool in some cases, but that it may also learn that if SVMTool predicts a certain tag for some word cluster the correct label is another tag. This way of combining taggers into a single end classifier can be seen as a form of stacking (Wolpert, 1992). It has the advantage that it reduces POS tagging to a classification task. This may simplify semi-supervised learning considerably.

2.4 Learning algorithms

We assume some knowledge of supervised learning algorithms. Most of our experiments are implementations of wrapper methods that call off-the-shelf implementations of supervised learning algorithms. Specifically we have experimented with support vector machines (SVMs), decision trees, bagging and random forests. Tri-training, explained below, is a semi-supervised learning method which requires large amounts of data. Consequently, we only used very fast learning algorithms in the context of tri-training. On the development section, decisions trees performed better than bagging and random forests. The decision tree algorithm is the C4.5 algorithm first introduced in Quinlan (1993). We used SVMs with polynomial kernels of degree 2 to provide a stronger stacking-only baseline.

[1] The numbers provided by Unsupos refer to clusters; "*" marks out-of-vocabulary words.

3 Tri-training

This section first presents the tri-training algorithm originally proposed by Li and Zhou (2005) and then considers a novel variant: tri-training with disagreement.

Let L denote the labeled data and U the unlabeled data. Assume that three classifiers c_1, c_2, c_3 (same learning algorithm) have been trained on three bootstrap samples of L. In tri-training, an unlabeled datapoint in U is now labeled for a classifier, say c_1, if the other two classifiers agree on its label, i.e. c_2 and c_3. Two classifiers inform the third. If the two classifiers agree on a labeling, there is a good chance that they are right. The algorithm stops when the classifiers no longer change. The three classifiers are combined by majority voting. Li and Zhou (2005) show that under certain conditions the increase in classification noise rate is compensated by the amount of newly labeled data points.

The most important condition is that the three classifiers are diverse. If the three classifiers are identical, tri-training degenerates to self-training. Diversity is obtained in Li and Zhou (2005) by training classifiers on bootstrap samples. In their experiments, they consider classifiers based on the C4.5 algorithm, BP neural networks and naive Bayes classifiers. The algorithm is sketched in a simplified form in Figure 1; see Li and Zhou (2005) for all the details.

Tri-training has to the best of our knowledge not been applied to POS tagging before, but it has been applied to other NLP classification tasks, incl. Chinese chunking (Chen et al., 2006) and question classification (Nguyen et al., 2008).

1: **for** $i \in \{1..3\}$ **do**
2: $S_i \leftarrow bootstrap\_sample(L)$
3: $c_i \leftarrow train\_classifier(S_i)$
4: **end for**
5: **repeat**
6: **for** $i \in \{1..3\}$ **do**
7: **for** $x \in U$ **do**
8: $L_i \leftarrow \emptyset$
9: **if** $c_j(x) = c_k(x)(j, k \neq i)$ **then**
10: $L_i \leftarrow L_i \cup \{(x, c_j(x)\}$
11: **end if**
12: **end for**
13: $c_i \leftarrow train\_classifier(L \cup L_i)$
14: **end for**
15: **until** none of c_i changes
16: apply majority vote over c_i

Figure 1: Tri-training (Li and Zhou, 2005).

3.1 Tri-training with disagreement

We introduce a possible improvement of the tri-training algorithm: If we change lines 9–10 in the algorithm in Figure 1 with the lines:

if $c_j(x) = c_k(x) \neq c_i(x)(j, k \neq i)$ **then**
$L_i \leftarrow L_i \cup \{(x, c_j(x)\}$
end if

two classifiers, say c_1 and c_2, only label a datapoint for the third classifier, c_3, if c_1 and c_2 agree on its label, but c_3 *disagrees*. The intuition is that we only want to strengthen a classifier in its weak points, and we want to avoid skewing our labeled data by easy data points. Finally, since tri-training with disagreement imports less unlabeled data, it is much more efficient than tri-training. No one has to the best of our knowledge applied tri-training with disagreement to real-life classification tasks before.

4 Results

Our results are presented in Figure 2. The stacking result was obtained by training a SVM on top of the predictions of SVMTool and the word clusters of Unsupos. SVMs performed better than decision trees, bagging and random forests on our development section, but improvements on test data were modest. Tri-training refers to the original algorithm sketched in Figure 1 with C4.5 as learning algorithm. Since tri-training degenerates to self-training if the three classifiers are trained on the same sample, we used our implementation of tri-training to obtain self-training results and validated our results by a simpler implementation. We varied poolsize to optimize self-training. Finally, we list results for a technique called co-forests (Li and Zhou, 2007), which is a recent alternative to tri-training presented by the same authors, and for tri-training with disagreement (tri-disagr). The p-values are computed using 10,000 stratified shuffles.

Tri-training and tri-training with disagreement gave the best results. Note that since tri-training leads to much better results than stacking alone, it is unlabeled data that gives us most of the improvement, not the stacking itself. The difference between tri-training and self-training is near-significant (p <0.0150). It seems that tri-training with disagreement is a competitive technique in terms of accuracy. The main advantage of tri-training with disagreement compared to ordinary tri-training, however, is that it is very efficient. This is reflected by the average number of tokens in L_i over the three learners in the worst round of learning:

	av. tokens in L_i
tri-training	1,170,811
tri-disagr	173

Note also that self-training gave very good results. Self-training was, again, much slower than tri-training with disagreement since we had to train on a large pool of unlabeled data (but only once). Of course this is not a standard self-training set-up, but self-training informed by unsupervised word clusters.

4.1 Follow-up experiments

SVMTool is one of the most accurate POS taggers available. This means that the predictions that are added to the labeled data are of very high quality. To test if our semi-supervised learning methods were sensitive to the quality of the input taggers we repeated the self-training and tri-training experiments with a less competitive POS tagger, namely the maximum entropy-based POS tagger first described in (Ratnaparkhi, 1998) that comes with the maximum entropy library in (Zhang, 2004). Results are presented as the second line in Figure 2. Note that error reduction is much lower in this case.

	BL	stacking	tri-tr.	self-tr.	co-forests	tri-disagr	error red.	p-value
SVMTool	97.15%	97.19%	**97.27%**	97.26%	97.13%	**97.27%**	4.21%	<0.0001
MaxEnt	96.31%	-	**96.36%**	**96.36%**	96.28%	**96.36%**	1.36%	<0.0001

Figure 2: Results on Wall Street Journal Sect. 22-24 with different semi-supervised methods.

5 Conclusion

This paper first shows how stacking can be used to reduce POS tagging to a classification task. This reduction seems to enable robust semi-supervised learning. The technique was used to improve the accuracy of a state-of-the-art POS tagger, namely SVMTool. Four semi-supervised learning methods were tested, incl. self-training, tri-training, co-forests and tri-training with disagreement. All methods increased the accuracy of SVMTool significantly. Error reduction on Wall Street Journal Sect. 22-24 was 4.2%, which is comparable to related work in the literature, e.g. Suzuki and Isozaki (2008) (7%) and Spoustova et al. (2009) (4–5%).

References

Chris Biemann. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *COLING-ACL Student Session*, Sydney, Australia.

Wenliang Chen, Yujie Zhang, and Hitoshi Isahara. 2006. Chinese chunking with tri-training learning. In *Computer processing of oriental languages*, pages 466–473. Springer, Berlin, Germany.

Stephen Clark, James Curran, and Mike Osborne. 2003. Bootstrapping POS taggers using unlabeled data. In *CONLL*, Edmonton, Canada.

Jesus Gimenez and Lluis Marquez. 2004. SVMTool: a general POS tagger generator based on support vector machines. In *LREC*, Lisbon, Portugal.

Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. 2009. Improving a simple bigram HMM part-of-speech tagger by latent annotation and self-training. In *NAACL-HLT*, Boulder, CO.

Ming Li and Zhi-Hua Zhou. 2005. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541.

Ming Li and Zhi-Hua Zhou. 2007. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE Transactions on Systems, Man and Cybernetics*, 37(6):1088–1098.

Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. 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.

Tri Nguyen, Le Nguyen, and Akira Shimazu. 2008. Using semi-supervised learning for question classification. *Journal of Natural Language Processing*, 15:3–21.

Ross Quinlan. 1993. *Programs for machine learning*. Morgan Kaufmann.

Adwait Ratnaparkhi. 1998. *Maximum entropy models for natural language ambiguity resolution*. Ph.D. thesis, University of Pennsylvania.

Anders Søgaard. 2009. Ensemble-based POS tagging of italian. In *IAAI-EVALITA*, Reggio Emilia, Italy.

Drahomira Spoustova, Jan Hajic, Jan Raab, and Miroslav Spousta. 2009. Semi-supervised training for the averaged perceptron POS tagger. In *EACL*, Athens, Greece.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*, pages 665–673, Columbus, Ohio.

Wen Wang, Zhongqiang Huang, and Mary Harper. 2007. Semi-supervised learning for part-of-speech tagging of Mandarin transcribed speech. In *ICASSP*, Hawaii.

David Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.

Le Zhang. 2004. Maximum entropy modeling toolkit for Python and C++. University of Edinburgh.