Automatic correction of part-of-speech corpora

Uwe Reichel Lia Saki Bucar Shigemori

Institute of Phonetics and Speech Processing (IPS), Munich, Germany
{reichelu|lia}@phonetik.uni-muenchen.de

Abstract

In this study a simple method for automatic correction of part-of-speech corpora is presented, which works as follows: Initially two or more already available part-of-speech taggers are applied on the data. Then a sample of differing outputs is taken to train a classifier to predict for each difference which of the taggers (if any) delivered the correct output. As classifiers we employed instance-based learning, a C4.5 decision tree and a Bayesian classifier. Their performances ranged from 59.1 % to 67.3 %. Training on the automatically corrected data finally lead to significant improvements in tagger performance.

1 Introduction

Part-of-speech (POS) taggers can be divided in rule-based and data-driven systems. Rule-based approaches like ENGTWOL [1] operate on a) dictionaries containing word forms, their POS tags, and syntactic information, and b) context sensitive rules to derive the correct tags during application. The most common data-driven approaches are Markov taggers [2], Maximum Entropy based taggers [3], and Transformation-based taggers [4].

Compared to rule-based taggers, data-driven taggers have the well-known advantages of fast development, noise robustness, and language independence. However, a huge amount of training material is needed for their design to face two major problems:

- context-dependent POS assignment
- tagging of words unseen in the training data (out-of-vocabulary OOV cases)

Increasing the amount of training data leads to more reliable counts to disambiguate context-dependent wordform-POS relations and might reduce the number of OOV cases in application. The latter issue however needs more elaboration within the context of the *large number of rare events* distribution

(see e.g. [5], [6]), which is outside the scope of this study. However, some observations about type coverage are presented in section 3.

Since manually tagging a training corpus of reasonable size (comprising several 100 K or more tokens) is very time expensive, an automatisation of this work would be appealing.

2 Data and labelling

The data used in this study comprises parts of the *European Corpus Initiative Multilingual Corpus 1* containing German newspaper texts (about 380 K word tokens), which were pre-tagged by two already established taggers for German: the TnT tagger [7] (TnT) and the Tree tagger [8] (STT) using the Stuttgart-Tübingen tag set. Both taggers face the OOV problem by further examining string suffixes of word forms which, in German, provide information about the word class. STT combines the Markov framework with a decision tree classifier.

For a sample of 20% of the total data, two experts labelled all cases of differing POS outputs marking whether STT, TnT or none of them yielded the correct output. Note that this procedure is based on the simplifying assumption that all instances of tagger agreements are tagged correctly.

3 Corpus size dependence of a tagger's performance

3.1 The Tagger

The data-driven tagger used in this study (SVT) is a generalisation of a basic Markov tagger, and was presented in greater detail in [9]. Basically, the aim to estimate the most probable tag sequence $\hat{T}$ given the word sequence W can be formulated as follows:

$$\hat{T} \quad = \quad \arg\max_{T} \Big[P(T|W) \Big].$$

To estimate $P(T|W)$, Bayes formula is applied, the constant denominator is omitted, and two simplifying assumption are made to get reliable counts for probability estimations:

- The probability of word w_i depends only on its tag t_i
- The probability of tag t_i depends only on a limited tag history *t-history*$_i$

This leads to the following formula:

$$\hat{T} \quad = \quad \arg\max_{t_1 \ldots t_n} \Big[\prod_{i=1}^{n} P(t_i|\text{t-history}_i) P(w_i|t_i) \Big].$$

For tagger SVT $P(t_i|\text{t-history}_i)$ is replaced by a linearly interpolated trigram model and w_i is replaced by a list of word representations leading to a reformulation of $P(w_i|t_i)$. Our model is thus given by

$$\hat{T} \quad = \quad \arg\max_{t_1 \ldots t_n} \Big[\prod_{i=1}^{n} \frac{1}{P(t_i)} \sum_j u_j P(\text{t}_i|\text{t-history}_{ij}) \sum_k v_k P(\text{t}_i|\text{w-representation}_{ik}) \Big].$$

The utilised word representations consist of string suffixes of different lengths determined by successor variety peaks (cf. [10]).

3.2 Corpus size effects on tagging performance

To examine the influence of the amount of training data on tagger performance, the STT-tagged training corpus token size was increased in 10 K steps ranging from 10 K to 370 K. For each size the SVT tagger was tested on a held out constant test set after the training.

As can be seen in Figure 1 the performance curve (left-hand side) follows the type coverage curve giving the percentage of types in the test data already seen in training (right-hand side) quite closely. The steep increase of both functions for 370 K tokens can be attributed to the approaching passages of training and test part within the text corpus resulting in a larger proportion of shared vocabulary and text style features.

It can be concluded, that for the given amount of training material our tagger's performance still depends heavily on the choice of training and test sets and therefore does not yet show an asymptotical behaviour. This finding clearly indicates the need for more data.

4 Efficient data acquisition

Manual POS tagging is a time consuming task, and the same holds for automatic tagging followed by manual correction. Hence an automatisation of tagging as well as of correction would be highly desirable. Our approach to these automatisations is described in the subsequent sections.

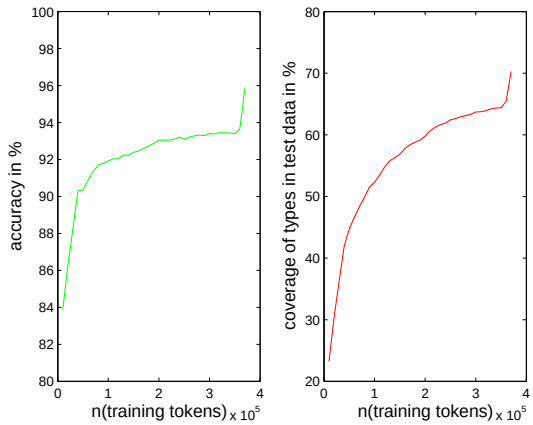

Figure 1: **Left:** Performance as a function of training data size. **Right:** Type coverage in the test data as a function of training data size.

4.1 Basic idea

The development phase of our corpus creation model consists of three steps:

1. An initial POS corpus is generated by applying two or more already available POS taggers on raw text data.

2. For a sample of differing tagging outputs the correct alternatives are marked manually.

3. A classifier C is trained on this labelled sample in order to predict the correct tagging alternative.

The manual effort required by step 2 is very low compared to manual POS corpus generation or correction since it is restricted to differing outputs only. Note that instances of tagger agreements are assumed to be correct. This issue is discussed later on.

The creation of new POS data in the application phase simply consists of the following steps:

1. Raw text is tagged by the same taggers used in the development phase.

2. For each tagging disagreement the trained classifier C chooses the correct tag alternative (if any).

5 Identification of the correct tagging alternative

Three supervised machine learning methods were trained on the manually judged output differences of the taggers STT and TnT: an instance-based approach, a C4.5 decision tree and a Bayesian Classifier.

Features and targets The feature vectors of size 8 consist of the tag history of length 2 of both taggers (4 features) as well as the observation which of the taggers (if any) made a mistake within this history (2 features) and the current output of the taggers (2 features).

The classification targets $c \in C$ are "STT-output is correct", "TnT-output is correct", and "none of the outputs is correct".

Instance-based learning For training an instance-based classifier IB the feature vectors were binarised. We used k-nearest neighbour classification with k set to 15. In application, the classification function $\hat{f}$ for feature vector x_q given the k nearest training vectors $\{x_1, \ldots, x_k\}$ is shown in the following equation.

$$\hat{f}(x_q) \quad = \quad \arg\max_{c \in C} \sum_{i=1}^{k} w_i \delta(c, f(x_i)).$$

$\delta(c, f(x_i))$ is the Kronecker function being 1 if class c equals the class $f(x_i)$ of training vector x_i, and 0 elsewise. w_i weights the contribution of each neighbour x_i decreasing quadratically with respect to its distance d to x_q:

$$w_i \quad = \quad \frac{1}{d(x_q, x_i)^2}.$$

For d the Jaccard distance between binary vectors was chosen by measuring the proportion of differing non-zero bits.

C4.5 decision tree In a C4.5 decision tree DT [11], each feature vector is represented as a path from the root to the leaf connected to the vector's class. Each non-terminal node is associated with an attribute according to which the set is further divided. During the recursive branching of the tree in the training procedure the choice of the attribute A most suited to split the set of objects is guided by a measure closely related to information gain which gives the mean reduction of the amount of bits needed to encode the object class given that

the value of A is known. The branching stops if all objects belong to the same class or cannot be further distinguished by their features. Subsequent recursive pruning to avoid over-adaption is guided by the comparison of the expected error rate of a subtree and the error rate expected after its condension to a leaf.

Bayesian Classifier A Bayesian classifier BC returns the class $\hat{f}(x_q)$ with the highest estimated probability given the feature vector x_q consisting of the features $x_{q1}, \ldots, x_{qn}$ which can be reformulated by Bayes formula, omission of the constant denominator, and the simplifying assumption of feature independence as shown below:

$$\begin{aligned} \hat{f}(x_q) &= \arg\max_{c \in C} \left[P(c|x_q) \right] \\ \hat{f}(x_q) &= \arg\max_{c \in C} \left[P(c) \cdot \prod_{i=1}^{n} P(x_{qi}|c) \right] \end{aligned}$$

6 Results

In this study we have evaluated

- the capability of our models to predict the correct tag alternative, and
- the performance differences of the SVT tagger trained on the pre-tagged and the automatically corrected data.

6.1 Correct tag prediction

The classifiers' performances were compared by ten-fold cross-validation. A baseline model BL was added, which simply adopted the output of that tagger, which performed better for the training data.

The left part of Figure 2 shows the mean performances of the classifiers in predicting the correct tag alternative. The performances range from 49.3 % (baseline model) to 67.3 % (C4.5 decision tree). The baseline model is significantly outperformed by all other models (Wilcoxon test of paired samples, $p = 0.002$), and the decision tree significantly outperforms all other models. No significant performance difference was found for the Bayes classifier and the K-Nearest-Neighbours classifier.

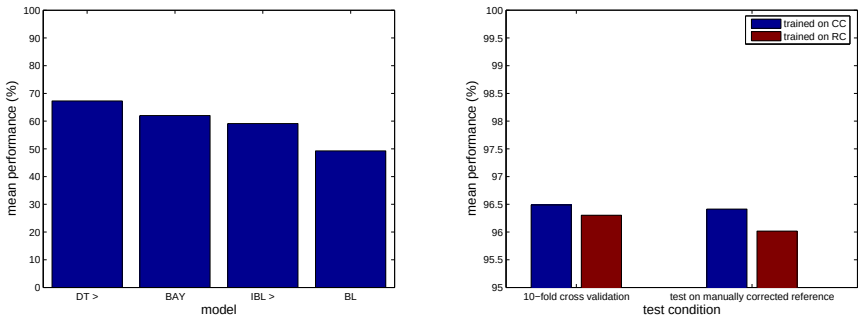

Figure 2: **Left:** Mean performances of the models in identifying the correct tagger output. DT: C4.5 decision tree, BAY: Bayes classifier, IBL: K-Nearest-Neighbours classifier, BL: baseline classifier. $>$: "significantly better than" (Wilcoxon test of paired samples, $p = 0.002$). **Right:** Tagging Performances based on the corrected training corpus CC and on the uncorrected reference RC.

6.2 Effects of automatic corpus correction

For examination whether or not the automatic correction of the corpus increases the performance of our SVT tagger the reference corpus (RC) was given by the STT tagger output, and the corrected corpus (CC) by applying the C4.5 classifier on the tagging differences of STT and TnT. For "Both tags wrong"-classifications the STT output was chosen by default.

Improvements of tagger performance. The SVT tagger was tested on ten subsets from that part of the corpus which was manually labelled to develop the tag-correction classifiers. SVT performed significantly better after being trained on CC (excluding the test corpus) than after training on RC (Wilcoxon test of paired samples, p = 0.004). The mean performances amounted 96.4% and 96.0% respectively. These results can be found in the right part of Figure 2.

Data consistency vs. correctness. RC is less correct than CC, but at the same time it is expected to be more consistent than CC, since it is based on a single tagger, while CC is based on two. To test whether consistency or correctness of the training data leads to a better tagging performance, we carried out two ten-fold cross validations for the SVT tagger trained and tested on CC and on RC. It turned out, that again SVT performed slightly but nevertheless significantly better on CC (mean: 96.5%) than on RC (mean: 96.3%; Wilcoxon test of paired

samples, p = 0.002). This result shows the slight superiority of correctness to consistency.

7 Discussion

Simplifying assumption We have not examined yet, whether or not the simplifying assumption stating that both taggers deliver the correct output whenever they agree, is too strong. In any case the assumption could be weakened by using more taggers for data labelling and/or by using taggers which considerably differ in their mode of operation.

Effects of automatic corpus correction The results show that although the performances in predicting the correct tag alternative are still quite poor (less than 70 %), the accuracy of a tagger trained on the automatically corrected training data increases significantly. This promising finding makes further work on improving tag alternative prediction worthwhile.

Acquiring new POS training material As could be seen in section 3, POS tagging requires large training corpora. Once given a corpus correction system like the one presented in this study, POS training material simply can be augmented by double pre-tagging additional text and identifying the correct alternative. Sentences containing “Both tags wrong”-decisions could be dropped or corrected manually.

Boosting The method described here could be transformed into a kind of very simple boosting algorithm combining the two weaker classifiers STT and TnT to form a stronger one just by replacing the classes introduced in section 5 with the correct POS tags. Further studies are needed to test the quality of such an approach.

Further Applications The data correction procedure proposed in this study is principally not bound to a specific domain as POS tagging but could be used whenever at least two labelling systems are already available and their errors can be identified. One alternative domain would be correcting a training corpus for grapheme-to-phoneme conversion.

References

[1] A. Voutilainen, "A syntax-based part of speech analyser," in *Proc. of the Seventh Conference of the European Chapter of the Association for Computation al Linguistics*. Dublin: Association for Computational Linguistics, 1995, pp. 157–164.

[2] F. Jelinek, "Markov source modeling of text generation," in *The Impact of Processing Techniques on Communications*, ser. NATO ASI series, J. Skwirzynski, Ed. Dordrecht: M. Nijhoff, 1985, vol. E91.

[3] A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Proc. Conference on Empirical Methods in Natural Language Processing*, 1996, pp. 133–142.

[4] E. Brill, "Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging," *Computational Linguistics*, vol. 21, no. 4, pp. 543–566, 1995.

[5] H. Baayen, *Word frequency distributions*. Dordrecht: Kluwer, 2000.

[6] R. Sproat, Ed., *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*. Dordrecht: Kluwer, 1998.

[7] T. Brants, "Tnt – a statistical part-of-speech tagger," in *Proc. ANLP-2000*, Seattle, WA, 2000.

[8] H. Schmid, "Improvements in Part-of-Speech Tagging with an Application to German," in *EACL, SIGDAT*, Dublin, 1995.

[9] U. Reichel, "Improving data driven part-of-speech tagging by morphologic knowledge induction," in *Proc. Advances in Speech Technology AST*, Maribor, 2005.

[10] M. Nascimento and A. da Cunha, "An experiment stemming non-traditional text," in *SPIRE'98 Proceedings*, Santa Cruz de La Sierra, Bolivia, 1998.

[11] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo: Morgan Kaufmann, 1993.