Experiments in tagger combination: arbitrating, guessing, correcting, suggesting

Giuseppe Attardi[1], Antonio Fuschetto[1], Francesco Tamberi[1], Maria Simi[1], and Eva Maria Vecchi[2]

[1] Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, I-56127 Pisa, Italy
{attardi,simi}@di.unipi.it
[2] Istituto di Linguistica Computazionale, CNR, Via Giuseppe Moruzzi 1, I-56124 Pisa, Italy

Abstract. The paper reports experiments with several strategies of tagger combination, using two well known taggers, TreeTagger and Hunpos. The most successful experiment,which achieved the best score in the Evalita 2009 Open Task, is a hybrid solution combining an easiest first iterative strategy with hand-written arbitration rules and a quality lexicon. The system used in the Closed Task uses a similar combination of two variants of Hunpos, together with correction/guessing rules.

Keywords: NLP, PoS Tagging, Evaluation.

1 Introduction

In the SemaWiki project pipeline [1] we have been using two well-known taggers: TreeTagger [2] and Hunpos [3]. In most cases of agreement between the taggers, the tagging was correct; in cases of disagreement the mistakes appeared to be of a very different nature. Hence we explored means to combine the results of the two taggers in order to improve their accuracy. In our experiments we tried several combination strategies:

1. an easiest-first iterative strategy;
2. arbitration rules, for deciding which of the predictions to trust;
3. correction and guessing rules for classification of new words;
4. suggestions: having one tagger to provide suggestions to the other.

2 The Taggers

TreeTagger [2] is a statistical part-of-speech tagger, which can be trained to new languages supplying to it a lexicon and a tagged training corpus.

Probabilistic models based on *n*-grams usually estimate the probability of a tagged sequence of words with first or second order Markov models. The TreeTagger method differs from the classical methods in the way it estimates the transition probabilities,

i.e. the probability of a tag given the previous ones. Instead of using maximum likelihood estimation, in order to address the problem of sparse data, zero frequencies and ungrammaticalities, TreeTagger uses a binary decision tree.

We reworked some parts of the implementation of TreeTagger to improve its performance, by using memory mapping for model data, adding UTF-8 support and tuning the decision tree analysis. This version is available as part of the Tanl toolkit [1].

Hunpos [3] is an open source reimplementation of TnT [4].

3 Open task

A large Italian lexicon of 1,267,677 forms, developed as part of the SemaWiki project, was used as an external resource, both with TreeTagger and Hunpos.

The full-form lexicon is generated from a base lexicon of 65,500 lemmas, initially inspired by [5], and updated along several years and cross-checked with other online dictionaries ([6], [7]). The lexicon was extended to provide information on transitive verbs, on superlatives and on diminutives and aligned to the conventions of the Tanl POS specifications [8]. The generation of the full-form lexicon was done with a script derived from a set of inflection rules supplied by Achim Stein.

Our baseline for this task is the accuracy of Hunpos (96.27%), which is slightly better than that of TreeTagger (95.32%). The efficiency of Hunpos is impressive: ~0.03min for training, ~0.07min for tagging the Test Set.

3.1 Arbitrating and guessing

For the first two runs of the open task the two taggers were used as black boxes and an elaborate arbitration strategy was used to resolve disagreements between them, taking for granted the output in case of agreement. Statistics computed on the development corpus indicate an agreement between the two taggers of 95.26% with an accuracy of 93.80%. A perfect arbitration strategy in case of disagreement could at best achieve a 4.74% increase in performance, which corresponds to an overall accuracy of 98.54%.

The arbitration strategy is a hybrid solution which uses a set of arbitration rules, accounting for most common mistakes, followed by a selection between the two outputs based on a statistical estimate of their plausibility.

Arbitration rules are used for deciding, in case of disagreement, which tagger to trust in a specific context. Many rules were suggested by looking at the outputs of the two taggers but only a few proved effective. Only six rules were selected with a series of experiments. In the following we indicate with tt_i and th_i the tags predicted for token w_i by TreeTagger and Hunpos respectively.

1. *if* $tt_i \in$ {'Vm2sc', 'Vcp1s'} *and* th_i = 'SP' *then* th_i
2. *if* th_i = 'SP' *and* beginning of sentence (*bos*) *then* tt_i
3. *if* th_i = 'CS' *then* th_i
4. *if* tt_i = 'B' *then* tt_i

5. *if* tt_i begins with 'S' and w_i is a new word, present in the lexicon *then* tt_i
6. *if* both tt_i and th_i begin with 'S' and w_i is a new word not in the lexicon
   then classify_unknown(w_i, th_i, bos)

The last rule is not a pure arbitration rule, but rather a *guessing* rule, since it uses a function (*classify_unknown*) which injects some heuristics for guessing the correct category of unknown names, i.e. not found in the training and absent from the lexicon.

The statistical combination method uses a simple score, computed on the training set, as a measure of plausibility of the tags proposed by the taggers. In particular we estimate the probability of a pos tag (without morphology) given the previous and next tag and compute a plausibility score as the average of the two probabilities:

$$S(t_i) = (P(t_i \mid t_{i-1}) + P(t_i \mid t_{i+1}))/2 \quad (1)$$

$P(t_i \mid t_k)$ is estimated on the training corpus by $F(t_i; t_k) / F(t_i)$, i.e. by the frequency of the bigram $(t_i; t_k)$ relative to the frequency of tag t_i. In case one of the surrounding tags is missing, the conditional probability of the tag given the missing neighbor is set to 0.

Following an idea presented in [9], an iterative *easiest-first* strategy is responsible for *arbitrating* among the two taggers' outputs, in case the rules in the previous phase fail to make a decision. The iterative strategy works as follows:

1. Start with tags assigned in the previous phase; initialize a *threshold*;
2. *For each* token w_i, with predicted tags tt_i and th_i:
   If either w_{i-1} or w_{i+1} have an assigned tag,
   return tt_i when $S(tt_i) > S(th_i)$ *and* $S(tt_i) > threshold$;
   return th_i when $S(th_i) > S(tt_i)$ *and* $S(th_i) > threshold$;
3. Decrease *threshold*; go back to 2 until all words are tagged.

In the first and second run we used the arbitration strategy described above, using TreeTagger and a *right-to-left* version of Hunpos, which performed slightly better than *left-to-right* Hunpos on the development set (96.45%). With this strategy we obtained a significant improvement in accuracy on the development set, i.e. 97.23%.

3.2 Suggesting and arbitrating

For the last two runs we explored a technique that we had successfully applied to parsing, i.e. exploiting hints from one parser in a second parsing step [10]. In the case of POS tagging, we exploited the fact that TreeTagger accepts in the input a list of possible tags for each token. Hence we modified Hunpos to produce the log likelihood score for each of its predictions.

The outputs from a base run of Hunpos and of TreeTagger on the test file are analyzed: where the taggers disagree but the likelihood score from Hunpos is higher than a threshold, its prediction is added as hint in the test file. The threshold is different depending on the type of tag, according to an overall accuracy for the parser that was estimated from the development set. The test file, augmented with suggestions from Hunpos, is passed to a second tagging stage by TreeTagger, producing the final output. In principle the process could be iterated, by using the

output of the taggers again, but these would require modifying Hunpos in order to accept hints as well.

We also tested a naïve combination, based on the overall accuracy of the taggers on each POS tag, which achieved an accuracy of 96.31% on the development test set, compared to a 96.68% of the fancier combination.

Our final run was produced by applying the arbitration strategy described earlier to the output of the previous tagger combination and the output of Hunpos, obtaining an improvement of 0.31%.

3.3 Results

We submitted four runs for the Open Task, summarized in Table 1 in terms of the tagging accuracy on all words (TA) and unknown words (UWTA).

The difference between SemaWiki 1 and 2 is the corpus used for training the taggers and for computing the scores: in the first run we used only the training data provided to the participants; in the second run, we used also the data provided for development. The difference in performance gives us a measure of the contribution of using the development corpus for domain adaptation, from newspaper style to Wikipedia style of texts.

The best run for the open task was the second one using the hybrid arbitration strategy with training performed on the joined train and development corpora. The PoS accuracy of 96.75% is also the best result for the Evalita 2009 PoS tagging open task. Nevertheless this result is lower than expected, considering that on the development corpus we had achieved 97.23%. This drop might be due to overfitting by the selected rules. Moreover, Hunpos *left-to-right* performs better (96.54%) than Hunpos *right-to-left* (96.26%) on the Test Set, so also the choice of direction for Hunpos was an overfit.

The accuracy measured discarding the morphological features, i.e. considering just the fine-grained POS tags, shows a consistent improvement of about 0.28% in all runs, meaning that morphology accounts for only a small percentage of errors.

Table 1. Open task results

Run	POS TA	CPOS TA	POS UWTA	CPOS UWTA
SemaWiki 2	**96.75%**	**97.03%**	**94.62%**	**95.30%**
SemaWiki 1	96.44%	96.73%	94.27%	95.07%
SemaWiki 4	96.38%	96.67%	93.13%	93.81%
SemaWiki 3	96.14%	96.42%	92.55%	93.24%
Evalita best	**96.75%**	**97.03%**	**94.62%**	**95.30%**

The accuracy on the unknown words is also the best result for the Evalita 2009 PoS tagging open task, since it closely follows the accuracy computed on all words with a 2–3% drop. The tagger is also relatively efficient (2.05min for obtaining the models, 55sec for tagging the test set).

4 Closed task

For the closed task, using TreeTagger was out of question since it performs quite badly without a lexicon. Hunpos performance instead, with 95.22% accuracy on the development set, is quite remarkable also without lexicon. A *right-to-left* version of Hunpos achieves 95.14% accuracy, but the disagreement between the *left-to-right* and the *right-to-left* version is insignificant (on the order of 0.02%).

The two taggers being so close in accuracy, an arbitration strategy, like the one used for the closed task, would not be very promising.

4.1 Correcting and guessing

For this task we tried a different approach by considering *likelihood* scores assigned by the two taggers to their own predictions; more precisely for each tag the taggers output a *log-likelihood* measure, i.e. the logarithm of the probability of the predicted tag. We used *log-likelihood* to devise a set of initial tags to rely on.

The strategy goes through two stages:

1. If the *log-likelihood* score of one of the taggers is above a given *threshold* (we used –3 in the runs) than *return* the tag *else* leave the tag unassigned for the second stage.
2. An *easiest-first* iterative strategy, similar to the one used for the open task, is used to fill the holes.

Stage 1 is used as a basis for the iterative strategy, since, differently from the open task, which uses taggers agreement and basic *arbitration* rules before the iterative strategy, taggers' agreement is too high in this case and does not leave much room for improvement. The iterative strategy of stage 2 is still based on plausibility scores estimated on the train corpus, in the same way we have discussed for the open task.

Besides this, before assigning a tag, we use *correction rules*, in both the first and second stage. In particular in the first stage we use only two rules for correcting tokens improperly classified as proper names, a weakness of Hunpos; in the second stage, in addition to those, we use three more correction/guessing rules for dealing with unknown nouns, adjectives, verbs. The guessing rules also embed a specific strategy for guessing the morphology of unknown words: the morphology of word forms is derived once for all from a large corpus of non-annotated text by taking into account common determiners and adjectives appearing before the form.

As before, the difference between the first and second run in the final submission is the corpus used for training the taggers and for computing the scores: the first run uses only the training data; the second run, both the training and development data.

4.2 Results

Our best result, obtained with the second run, is 95.73%. The accuracy of the best scoring system of the Evalita 2009 PoS tagging closed task is 96.34%, a difference of 0.61%. The loss in performance due to the morphology is 0.79%, higher than in the closed task, as one would expect since morphology, and in particular the *n* feature (for

'underspecified') of nouns and adjectives, is hard to derive from the local context. Our strategy for guessing the morphology was not effective enough.

Table 2 summarizes the results in tagging accuracy in the Closed Task.

Table 2. Closed task results

Team	POS TA	CPOS TA	POS UWTA	CPOS UWTA
SemaWiki 2	95.73%	96.52%	90.15%	93.47%
SemaWiki 1	95.24%	96.00%	87.40%	90.72%
Evalita best	**96.34%**	**96.91%**	**91.41%**	**93.81%**

5 Conclusions

The taggers we developed for Evalita 2009 are accurate and efficient. However the quite elaborate arbitration strategy of our best run in the Open Task achieves only a 0.21% improvement over the accuracy of Hunpos *left-to-right* on the test set (96.54%). This is definitely not enough to justify the considerable decrease in efficiency in tagging, due to the use of two taggers and the combination strategy. The good accuracy of our system is mainly due to the quality of the lexicon, which accounts for a +1.05% increase in accuracy when used with Hunpos alone.

Acknowledgments. We are grateful to all the SemaWiki team for producing quality resources for this task and in particular to Simonetta Montemagni for supervising the linguistic soundness and providing advice on critical cases. The SemaWiki project was partially funded by the Fondazione Cassa di Risparmio di Pisa.

References

1. Attardi, G., et al.: Tanl (Text Analytics and Natural Language Processing): Analisi di Testi per il Semantic Web e il Question Answering, http://medialab.di.unipi.it/wiki/SemaWiki
2. Schmid, H.: Probabilistic Part-of-Speech Tagging Using Decision Trees. In: Proceedings of the International Conference on New Methods in Language Processing, pp. 44--49 (1994)
3. Halácsy, P., Kornai, A., Oravecz, C.: HunPos – an open source trigram tagger. In: Proceedings of the Demo and Poster Sessions of the 45th Annual Meeting of the ACL, pp. 209--212 (2007)
4. Brants, T.: TnT–A Statistical Part-of-Speech Tagger. In: Proceedings of ANLP-NAACL Conf. (2000)
5. Zingarelli: Il nuovo Zingarelli minore. Zanichelli (2008)
6. Gabrielli: Il Grande Italiano, http://dizionari.repubblica.it/
7. De Mauro, T.: Il Dizionario della lingua italiana, http://www.demauroparavia.it/
8. Tanl POS Tagset. http://medialab.di.unipi.it/wiki/Tanl_POS_Tagset (2007)
9. Tsuruoka, Y., Tsujii, J.: Bidirectional Inference with the Easiest-First Strategy for Tagging Sequence Data. In: Proceedings of HLT-EMNLP, pp. 467--474 (2005)
10. Attardi, G., Dell'Orletta, F.: Reverse Revision and Linear Tree Combination for Dependency Parsing. In: Proceedings of NAACL HLT (2009)