SuperSense Tagging with a Maximum Entropy Classifier and Dynamic Programming

Giuseppe Attardi, Luca Baronti, Stefano Dei Rossi, Maria Simi

Università di Pisa, Dipartimento di Informatica, Largo B. Pontecorvo 3,
56127 Pisa, Italy
{attardi, barontil, deirossi, simi}@di.unipi.it

Abstract. Our participation to the SuperSense tagging task relies on a flexible and customizable tagging tool, developed as part of the Tanl suite, for use in various tagging tasks, including PoS tagging and Named Entity tagging. The tagger is based on a Maximum Entropy classifier and uses dynamic programming to select accurate sequences of tags. It extracts three different kinds of features: attributes features, related to the position of the attributes surrounding the current token; local features that are morphological features extracted from the analysis of the current word and the context in which it appears; global features that are properties holding at the document level. Features were explicitly customized for the SuperSense task.

Keywords: SuperSense Tagging, Word Net, Maximum Entropy, dynamic programming

1 Description of the System

SuperSense tagging (SST) consists in annotating nouns, verbs, adjectives and adverbs in a text, within a general semantic taxonomy defined by the WordNet lexicographer classes (called SuperSenses) [1].

SST can be regarded as a special case of chunking, hence we implemented a SuperSense tagger by extending and customizing a generic chunker, which we developed as part of Tanl pipeline [2] and which is based on the work of Chieu & Ng [3]. This generic chunker was also used for implementing the Tanl NER, that achieves state of the art accuracy on the CoNLL 2003 benchmarks for English.

In [4] we reported preliminary results in SST (F1 score of 79.1), which represented a significant improvement on state-of-the art performance for Italian in this task. The task was simpler, with respect to Evalita 2011, in several respects; in particular, the annotation of complex noun phrases such as "*Presidente della Repubblica*" was not contemplated.

The tagger uses a Maximum Entropy classifier for learning how to chunk texts and dynamic programming in order to select sequences of tags with the highest probability. The tagger design is flexible and allows choosing which features are

relevant for a specific tagging task, and from which tokens or tokens attributes they should be extracted.

Maximum Entropy is effective for chunking, since it does not assume independence of features. It is also a more efficient technique than SVM and, complemented with dynamic programming, can achieve similar levels of accuracy.

1.1 Maximum Entropy and Dynamic Programming

The Maximum Entropy framework estimates probabilities based on the principle of making as few assumptions as possible, other than the constraints imposed. Such constraints are derived from training data, expressing some relationship between features and outcome. The probability distribution that satisfies the above property is the one with the highest entropy, it is unique, and agrees with the maximum-likelihood distribution. The distribution has the following exponential form [5]:

$$p(o \mid h) = \frac{1}{Z(h)} \prod_{j=1}^{k} \alpha_j^{f_j(h,o)},$$

where *o* refers to the outcome, *h* is the history or context, and *Z(h)* is a normalization function. The features used in the Maximum Entropy framework are binary. An example of a feature function is:

$$f_j(h,o) = \begin{cases} 1 & \text{if } o = B - noun.location, \text{FORM} = Washington \\ 0 & \text{otherwise} \end{cases}$$

The parameters α_j are estimated by a procedure called Generalized Interactive Scaling (GIS) [6]. This is an iterative procedure that improves the estimation of parameters at each iteration.

Since the Maximum Entropy classifier assigns tags to each token independently, it may produce inadmissible sequences of tags. Hence a dynamic programming technique is applied to select correct sequences. A probability is assigned to a sequence of tags t_1, t_2,…, t_n for sentence *s*, based on the probability of the transition between two consecutive tags $P(t_{i+1} \mid t_i)$, and the probability of a tag $P(t_i \mid s)$, obtained from the probability distribution computed by Maximum Entropy:

$$P(t_1, t_2, ..., t_n) = \prod_{i=1}^{n} P(t_i \mid s) P(t_i \mid t_{i-1})$$

In principle the algorithm should compute the sequence with maximum probability. We use instead a dynamic programming solution which operates on a window of size $w = 5$, long enough for most SuperSenses. For each position *n*, we compute the best probability $PB(t_n)$ considering the n-grams of length $k < w$ preceding t_n:

$$PB(t_n) = \max_k PB(t_{n-k-1}) \ldots PB(t_{n-1})$$

A baseline is computed, assuming that the *k*-gram is made all of ‘O’ (outside) tags:

$$PB_O(t_n) = \max_k PB(t_{n-k-1}) \, P(t_{n-k} = \text{O}) \ldots P(t_{n-1} = \text{O})$$

Similarly for each class *C* we compute:

$$PB_C(t_n) = \max_k PB(t_{n-k-1})\ P(t_{n-k} = \mathrm{C})\ ...\ P(t_{n-1} = \mathrm{C})$$

and finally:

$$PB(t_n) = \max(PB_O(t_n), \max_C PB_C(t_n)$$

1.2 Features Specification

The modular architecture of the chunker offers the possibility to specify the features to extract using a textual configuration file. In particular three different kind of features can be specified:

- **attributes features:** represent certain attributes (e.g.: PoS, Lemma, NE) of surrounding tokens, expressed by the relative positions w.r.t. to the current token; for example POSTAG -1 0 means: use as context features for the current token the PoS of the previous token and of the current token, in position 0;
- **local features:** other binary morphological features extracted from the analysis of the current word and the context in which it appears; for example "*previous word is capitalized*";
- **global features:** properties holding at the document level. For instance, if a word in a document was previously annotated with a certain tag, then it is likely that other occurrences of the same word should be tagged similarly. Global features represent these properties. They are particularly useful in cases where the word context is ambiguous but the word appeared previously in a simpler context.

1.3 Dataset and Testing Phase

The training set was in a tab-separated columns format with one token per line and four columns corresponding to FORM, LEMMA, PoS and SuperSense in the IOB2 notation.

Before the beginning of the tests, we prepared the dataset for a proper validation process. The sentences available in the training set were shuffled and divided into three separate sets:

- A training set (about the 70% of the corpus) used to train the models;
- A validation set (about 20% of the corpus) used to choose the best model;
- A test set (about 10% of the corpus) used to evaluate the performance.

To compute the baseline result we used a basic configuration with no attributes features and with the following standard set of local features:

- *Features of Current Word*: first word of sentence and capitalized; first word of sentence and not capitalized; two parts joined by a hyphen.
- *Features from Surrounding Words*: both previous, current and following words are capitalized; both current and following words are capitalized; both current and previous words are capitalized; word is in a sequence within quotes.

With 100 iterations of the Maximum Entropy algorithm we obtained an F-score of 71.07 on the validation set.

The testing process consisted in a process of feature selection involving the creation of many configuration files with different combination of features. In particular about 300 positional permutations of the attribute features were tested along with the variation of other parameters like the number of iterations, the *cutoff feature* (an option that prevents the tagger to learn from features that appear a number of times below a given threshold), and *refine feature* (an option to split the IOB tags into a more refined set).

The performance of each system was computed testing the model on the validation set and comparing the accuracy with that of the other systems. Then the same configuration file was used to train a new model on a dataset resulting from the merging of the training set and the validation set, and the performance was tested on the test set. This validation process was done to make sure that the performance does not degrade on new and unknown data because of overfitting on the validation set.

The best run on the validation set obtained a F-score of 80.01, about 10 points higher than the baseline.

1.4 Submission

For the final submission we have chosen the four runs with the best and most balanced performance on the validation and test set. We decided to participate only to the closed task because we didn't obtain any performance improvement from the use of external dictionaries and gazetteers, such as ItalWordNet (IWN) [7].

In the following sections we will describe the features used to create the four runs, called **run [1-4]**.

Attributes Features. The table below shows the positional parameters of the attributes features used for the four runs.

Table 1. Attributes features for the four runs

	Run 1-2	**Run 3-4**
FORM	0	0
POSTAG	-2 0 1 2	0 1
CPOSTAG	-1 0	-1 0
LEMMA	-1 0	0

For example LEMMA -1 0 tells the tagger to use as features the LEMMA of the previous (-1) and of the current (0) token. The CPOSTAG is the coarse-grained POS tag that corresponds to the first letter of the POSTAG.

Local Features. The standard set of local features described above for the baseline was used for all the runs. An additional set of local feature was used for run 3 and run

4 with the aim to improve the performance of the tagger on the classes of SuperSenses with low F-score. Such classes are: verb.emotion, verb.possession, verb.contact and verb.creation. A list of the most common non-ambiguous verbs in those classes was obtained from the training set and they were added as local features for the current LEMMA. The list of verbs is the following:

- *verb.emotion*: sperare, interessare, preoccupare, piacere, mancare, temere, amare;
- *verb.possession*: vendere, perdere, offrire, pagare, ricevere, raccogliere;
- *verb.contact*: porre, mettere, cercare, colpire, portare, cercare, toccare;
- *verb.creation*: realizzare, creare, produrre, aprire, compiere.

Global Features. The *refine* option which performed well for tasks with a lower number of classes like Named Entity Recognition, proved to be less relevant for SST where the number of classes and the level of ambiguity is already high, so we didn't use it for the runs. Also changing the threshold value of the *cutoff* option to values > 1 showed no improvements on the performance of the system, so we left it to 0.

Different numbers of training iterations were used for the four runs, in particular:

- *run 1*: 100;
- *run 2*: 150;
- *run 3*: 200;
- *run 4*: 500.

2 Results

Table 2. UniPI systems results on the closed subtask

	Accuracy	Precision	Recall	FB1
UniPI - run 3	**88.50%**	**76.82%**	**79.76%**	**78.27**
UniPI - run 2	88.34%	76.69%	79.38%	78.01
UniPI - run 1	88.30%	76.64%	79.33%	77.96
UniPI - run 4	88.27%	76.48%	79.29%	77.86

Run 3 was the best performing system for the Evalita 2011 SST closed task.

3 Discussion

Analysing the data of all the experiments performed while tuning the system, we observed that our Maximum Entropy tagger achieves the best F1 results with a reduced number of iterations, i.e. between 100 and 200 iterations (**Fig. 1**). This is really important information for future tunings of the tagger: to be able to fix one important parameter decreases the number of experiments to be performed, and has also a positive effect on the execution time for training the system.

It is worth nothing that, consistently with this evaluation, the best results on the test set were obtained with run 3, 2 and 1 with 100, 150 and 200 iterations, while run 4, with 500 iterations, obtained the worst score.

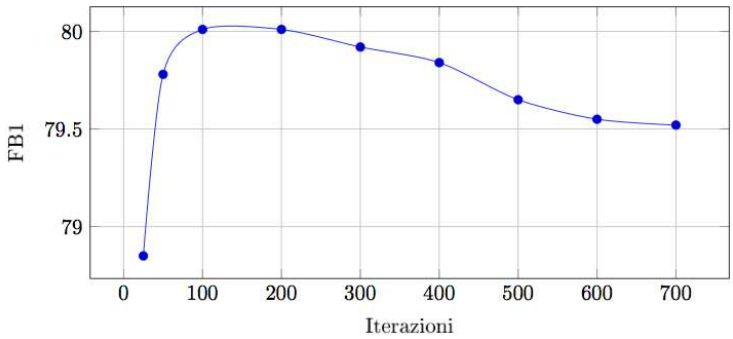

Fig. 1. Maximum Entropy performance vs. number of iterations measured on 300 runs performed during the feature selection phase.

References

1. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
2. Attardi, G., Dei Rossi, S., Simi, M.: The Tanl Pipeline. In: Proceedings of Workshop on Web Services and Processing Pipelines in HLT, co-located LREC 2010, Malta (2010)
3. Chieu, H.L., Ng, H.T.: Named Entity Recognition with a Maximum Entropy Approach. In: Proceedings of CoNLL-2003, pp. 160-163. Edmonton, Canada (2003)
4. Attardi, G., Dei Rossi, S., Di Pietro, G., Lenci, A., Montemagni, S., Simi, M.: A Resource and Tool for SuperSense Tagging of Italian Texts. In: Proceedings of 7th Language Resources and Evaluation Conference (LREC 2010), pp. 17--23, Malta (2010)
5. Della Pietra, S., Della Pietra, V., Lafferty, J.: Inducing Features of Random Fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(4), pp. 380–393 (1997)
6. Darroch, J. N., Ratcliff, D.: Generalized Iterative Scaling for Log-Linear Models. Annals of Mathematical Statistics, 43(5), pp. 1470--1480 (1972)
7. Roventini, A., Alonge, A., Calzolari, N., Magnini, B., Bertagna, F.: ItalWordNet: a Large Semantic Database for Italian. In: Proceedings LREC 2000, Athens (2000)