Discriminative Methods for Improving Named Entity Extraction on Speech Data

James Horlock, Simon King

Centre for Speech Technology Research
University of Edinburgh, Edinburgh
J.Horlock@ed.ac.uk, Simon.King@ed.ac.uk

Abstract

In this paper we present a method of discriminatively training language models for spoken language understanding; we show improvements in named entity F-scores on speech data using these improved language models. A comparison between theoretical probabilities associated with manual markup and the actual probabilities of output markup is used to identify probabilities requiring adjustment. We present results which support our hypothesis that improvements in F-scores are possible by using either previously used training data or held out development data to improve discrimination amongst a set of N-gram language models.

1. Introduction

This paper describes a novel approach to training languague models, for named entity extraction from speech, using discriminative techniques. We adopt a Hidden Markov Model (HMM) style approach to address the task of standard named entity (people, places, etc) extraction from a non-standard (speech) source.

Our system is designed for speech data, and takes as input word lattices produced by a HMM recogniser. Converting to accept standard text as input is trivial as any of the multiple isomorphisms from text to lattices will provide suitable input. Speech data is traditionally treated as noisy text in information extraction tasks, ie previous work [1] [2] [3] [4] [5] uses the 1 best recogniser output. By using speech lattices instead of this noisy text, it is possible to obtain higher F-scores in named entity results.

F-scores are the standard measure used to score information extraction systems, they are calculated as the goemetric mean of Recall (the number of correct data over the number of potential correct data) and Precision (the number of correct data over the number of incorrect data).

Having built a system capable of named entity extraction from a combination of language models, we then use discriminative methods to improve the individual language models resulting in improved F-scores.

2. System Overview

The mathematical model we use is very similar to that developed by BBN [1] and Mitre [2], however, our system is designed to work with speech lattices not text. We consider the task of named entity extraction from speech to be that of finding the set of words $W_1^L = (w_1...w_L)$ and its corresponding set of entities $E_1^L = (e_1...e_L)$.

The task is therefore to maximise the joint probability $P(E_1^L, W_1^L)$ which we break down in equation 1.

$$\begin{aligned} &P(E_1^L, W_1^L) \\ &\quad = P(e_1).P(E_2^L, W_1^L|e_1) \\ &\quad = P(e_1).P(w_1|e_1).P(E_2^L, W_2^L|e_1, w_1) \\ &\quad = P(e_1).P(w_1|e_1).P(e_2|e_1, w_1) \\ &\qquad .P(e_2^L, W_2^L|e_1, w_1, e_2) \\ &\quad ... \\ &\quad = \prod_{i=1}^{L} P(e_i|E_1^{i-1}, W_1^{i-1}).P(w_i|E_1^i, W_1^{i-1}) \end{aligned} \tag{1}$$

Making standard trigram assumptions we approximate these probabilities by the products shown in equation 2.

$$\begin{aligned} &P(E_1^L, W_1^L) \\ &\quad \approx \prod_{i=1}^{L} P(e_i|E_{i-3}^{i-1}, W_{i-3}^{i-1}).\prod_{i=1}^{L} P(w_i|E_{i-2}^{i}, W_{i-3}^{i-1}) \end{aligned} \tag{2}$$

The task of named entity extraction from speech is therefore to maximise both products in equation 2. The first product may be regarded as the probability of an entity sequence, whilst the second product is the probability of a word sequence. We therefore consider the finite state automaton shown in figure 1, where the first product from equation 2 is associated with transitions between states in and the second product is associated with state outputs.

The training data is divided into 8 sections corresponding to the 7 named entities and the remaining "not a named entity" data. After applying the Witten Bell [6] training algorithm to produce 8 (one for each entity + 1) language models capable of estimating all the required probabilities, ie both state transition probabilities and the state output probabilities. Thus the respective probabilities from equation 2 are estimated from data. Standard N-gram back-off is used to evaluate unknown probabilities.

Due to the nature of the finite state automaton, figure 1, it is possible to use a Viterbi search to find the values of W_1^L and E_1^L which maximise equation 2, in a very efficient manner. We use Viterbi to solve equation 3 giving us the desired word and entity sequence.

$$\arg\max_{E_1^L, W_1^L} \prod_{i=1}^{L} P(e_i|E_{i-3}^{i-1}, W_{i-3}^{i-1}).P(w_i|E_{i-2}^{i}, W_{i-3}^{i-1}) \tag{3}$$

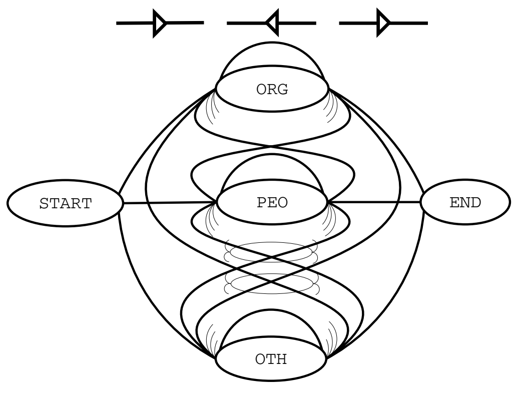

Figure 1: *Simplification of the topology used for the statistical model. Direction of transitions are indicated by arrows above the transitions to avoid confusion.*

3. Data and Baselines

3.1. Data

For this task we have used a subset of the Hub4 task of the DARPA broadcast news workshop data available from the Linguistic Data Consortium [7]. This data contains transcripts of broadcast news, some of which has been marked up with named entities and includes the official development set used in the 1998 evaluations. We also make use of speech lattices corresponding to the official development set. We use the official development set as our final test set and consequently need to split the marked up training material into separate parts for training and development.

3.2. Baselines

By way of baselines we refer to SPRACH-S and SPRACH-R [3], one of which is statistical and the other rule based. These were also tested on this official development set. SPRACH-S scored 80 and 68 on manual transcription and speech data respectively; whereas SPRACH-R scored 69 and 59. All cited results refer to F-score.

3.3. Our Initial System

We also have as our third baseline, the system described in section 2. We have results for our baseline system on training data, our development data, and our test set. These results are 97.77, 83.65 and 74.34 respectively. The substantial difference between the development and test set are as a direct result of speech recognition errors; the development set is a manual transcript, whilst the test set is from speech lattices with Lattice Error Rate (LER) of 5% which has a net effect on Word Error Rate (WER) of approximately 20% for the various transcripts.

4. Discriminative Training of Language Models

As we have already indicated, we use 8 language models to predict probabilities (of words and entities), and take the product of these probabilities to find the best sequence of words and entities.

If we consider $P(w_i = `James'|E^i_{i-2}, W^{i-1}_{i-3})$, and for simplicity we consider only the current entity, for example $P(w_i = `James'|e_i = `PERSON', E^{i-1}_{i-2}, W^{i-1}_{i-3})$ hereafter referred to as $P(James|PERSON)$. Then intuitively we might conclude that $P(James|PERSON) > P(James|LOCATION)$.

There is, however, no way of the LOCATION language model indicating (or even knowing) that the probability of James being a LOCATION is very low, it is assumed that because the word didn't occur in LOCATION training data, and did occur in PERSON data that the comparison of probabilities yields the desired result.

If we assume that neither 'James' nor 'Geneva' appeared in the LOCATION training material; the LOCATION language model will find both James and Geneva to be equally likely. We would prefer $P(James|LOCATION) < P(Geneva|LOCATION)$ since James is likely within the PERSON language model.

We now introduce the concept of discriminative language models and their application to the problem at hand.

Until now we have been using a Viterbi search over named entity sequences matching words; in doing so we have traversed multiple language models.

We suggest that lack of presence in LOCATION training data provides insufficient information to calculate the probability of the word in LOCATION context. Knowledge about the word's presence in other Non-LOCATION training data should be useful in the calculation of the relevant probability. In this instance since $P(James|PERSON)$ is high, the fact that $P(James|LOCATION)$ is not low enough is overcome. We suggest that it is possible to improve results by each language model knowing to discriminate against N-grams that have a high proability under competing language models.

In this section we introduce a method for iteratively making the language models discriminate against non-entity data and then in the next section we provide results on a test corpus having trained our discriminative language models both on the original training data and also on a held out development set.

There is a homomorphism (two-way mapping) between named entity output and the most likely probability path (or equivalently between the named entity output and the path through the model topology figure 1). It is therefore possible to find the sequence of probabilities which correspond to any output sequence of named entity marked up words. By this process it is possible to find not only the probabilities used to generate the output, but also the theoretical probabilities that would have been used to generate the correct markup.

Essentially the process uses two copies of the same data. The one has the markup added automatically as described in section 2, the other has manual markup. By using the homomorphism it is possible to identify which probabilities were used to generate the markup of the first data set, and similarly it is possible to find the theoretical probabilities associated with the manual markup.

A comparison between the used probabilities and the manual "theoretical probabilities" make it possible to find which probabilities, when adjusted, would result in the correct paths being chosen in subsequent runs.

The comparison between these 2 lists of probabilities will result in 3 new lists of probabilities. The first list is the probabilities that occurred in both paths. We assume that these are correct, and consequently ignore this list. The second list is the list of probabilities which occured in the chosen path but did not occur in the manual path. We assume that these probabilities are too high, since they were chosen when they should not have been, and therefore decrease them. The third list is the list of probabilities which occurred in the manual path, but did not

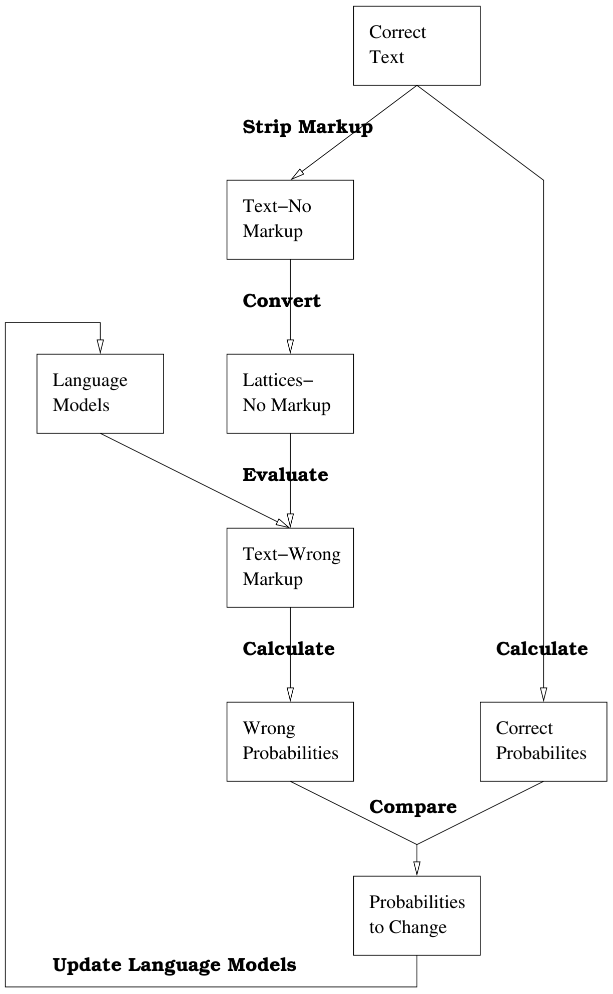

Figure 2: *Schematic diagram of iteration process for updating language models.*

occur in the chosen path. We assume that these probabilities are too low and therefore increase them, since if they had been higher the correct path would have been more likely to have been chosen.

This process is illustrated in the schematic flowchart (figure 2) which shows how to find probabilities which need to be adjusted and adjust them. The flowchart also shows reveals how this process is iterative since the updated language models can be used to re-evaluate the data.

There are a number of issues that arise from adjusting these probabilities: How much should each probability be adjusted? How many times should the same probability be adjusted? What data is suitable for using to adjust the probabilities? We conducted 3 experiments to answer these questions.

5. Experiments

5.1. Experiment 1

The first experiment reused the training data to adjust the probabilities associated with n-grams. We used the above method, illustrated in figure 2, to find a list of probabilities that were too high, ie those which over-emphasised an incorrect path, and a list of probabilities which were to low, ie those which under-emphasised a correct path. The probabilities from each list were adjusted by multiplying by 1.12 or 0.89 (equivalent to ±0.05 in log probabilities) depending on whether it was to be increased or decreased.

In the first experiment the probabilities were adjusted every time a given n-gram occurred in the lists. N-grams which occurred numerous times were adjusted numerous times. The results for this experiment were disappointing with F-measure dropping in all instances including the training data to $\simeq 45$ after a single iteration.

The problem was found to be probabilities that were incorrect were occurring frequently and thus when updated repeatedly the final language models bore little to no resemblance to the original.

Table 1: *F-Measures on training data after successive iterations of discriminative training on the training data.*

Adjustment Factors	Original	First	Second	Third
1.58 0.63	97.77	97.98	98.47	97.72
1.26 0.79	97.77	98.45	98.86	98.79
1.12 0.89	97.77	98.35	98.50	98.86
1.06 0.94	97.77	98.11	98.36	98.51

Table 2: *F-Measures on development set after successive iterations of discriminative training on the training data.*

Adjustment Factors	Original	First	Second	Third
1.58 0.63	83.65	82.40	83.17	82.13
1.26 0.79	83.65	84.30	83.96	83.63
1.12 0.89	83.65	83.74	83.45	83.59
1.06 0.94	83.65	84.35	84.28	84.39

5.2. Experiment 2

The second experiment also used the training data to adjust the probabilities associated with n-gram phrases. A number of different adjustment factors were examined to find optimal values for how much to adjust the incorrect probabilities by. These values corresponded to adjusting the log probabilities by ±0.2, ±0.1, ±0.05 and ±0.025.

Instead of adjusting once for each mistake, a tally of all mistakes were made. If the total number of times that the probability was too high was greater than the total number of times that the probability was too low then the probability was decreased; if the total number of times that the probability was too low was greater that the total number of times that the probability was too high then the probability was increased.

The results after 1, 2 and 3 iterations of the procedure on the different data sets are shown in tables 1, 2 and 3. The results show that there is fluctuation in improvements, but that there is improvement to be gained even by reusing the training data providing the adjustment factor is kept small.

5.3. Experiment 3

The third experiment used the development data to adjust the probabilities associated with n-gram phrases. This time the probabilities were adjusted by either 1.06 or 0.94 (the best result from experiment 2); the same method for calculating the number of times to update was used as in experiment 2. We ran only a single iteration of the process described in figure 2, the results for which are shown in table 4.

Table 3: *F-Measures on test set after successive iterations of discriminative training on the training data.*

Adjustment Factors	Original	First	Second	Third
1.58 0.63	74.34	74.30	74.27	72.26
1.26 0.79	74.34	74.27	74.30	74.02
1.12 0.89	74.34	74.10	74.42	74.32
1.06 0.94	74.34	74.44	74.23	74.41

Table 4: *F-Measures after discrimination on development set.*

	Training	Development	Testing
Before	97.77	83.65	74.34
After	97.76	85.29	74.49

These results show that the decrease in F-measure on the training data, which was to be expected, was negligible. The increase, however, on the development data, which was also to be expected, was significant. The most important improvement to note was the increase in F-measure on the test set. We are therefore able to conclude that this method improves named entity extraction when a development set is used for the discriminative training process.

Although the system was developed for speech lattices, one final evaluation of the system on the manual transcription was run for comparison with the SPRACH system baselines. This was done after using the development files for the discriminative language models. The manual transcription did not contain capitalisation and punctuation that had been added by annotators; it was rather a sequence of correct words. The results for this experiment yielded an F-measure of 84.01% (without discriminative training 83.21).

These results are summarised in 3, alongside previous work baselines.

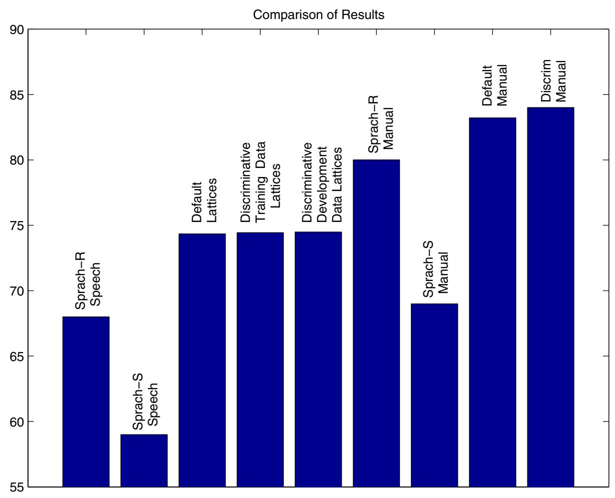

Figure 3: *Comparison of results, showing baselines and new system results on lattices and manual transcriptions.*

6. Conclusions and Future Work

We conclude from this investigation that it is possible to use data to improve trained language models for the purpose of named entity extraction. We have shown that both the original training data and also a held out development set can be used for this purpose.

The fact that the training data can be used to improve results on both a development and a test set shows that the language models had not been trained to a point of over-fitting the data prior to the discriminative training.

It is likely that there is scope for further improvement in these results.

Having shown that a linear relation between frequency of probabilities to be updated and amount to update by (experiment 1) was disadvantageous, whilst always updating probabilities by the same amount was useful (experiment 2). In the future we intend to investigate sigmoidal functions of frequency for finding adjustment factors. We also hope to find ways of updating language model probabilities based on unlabelled data.

7. Acknowledgements

We would like to thank Phil Woodland of Cambridge University Engineering Department for the use of HTK speech lattices.

8. References

[1] D Bikel, S Miller, R Schwartz and R Weischedel *NYMBLE: A High-Performance Learning Name Finder* Proc. Applied Natural Language Processing 1997

[2] D Palmer, J Burger and M Ostendorf *Information Extraction from Broadcast News Speech Data* Proc. Darpa Broadcast News Workshop 1999

[3] S Renals, Y Gotoh, R Gaizauskas and M Stevenson *Baseline IE-NE Experiments Using the Sprach/Lasie System* Proc. Darpa Broadcast News Workshop 1999

[4] D Appelt and D Martin *Named Entity Extraction from Speech: Approach and Results using the TextPro System* Proc. Darpa Broadcast News Workshop 1999

[5] H Kim *Named Entity Recognition form Speech and Its Use in the Generation of Enhanced Speech Recognition Output* PhD Thesis, Cambridge University 2001

[6] T Witten and I Bell *The Zero-Frequency Problem: Estimating the Probabilities of Novel Events in Adaptive Text Compression.* IEEE Transactions on Information Theory 1991

[7] Linguistic Data Consortium 1998 *Catalogue references LDC98E10 & LDC98E11* www.ldc.upenn.edu