IDIAP COMMUNICATION

Language modeling based on neural clustering of words

Vesa Siivola

IDIAP–Com 00-02

April 2000

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 − 27 − 721 77 11
fax +41 − 27 − 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

[a] Dalle Molle Institute for Perceptive Artificial Intelligence PO Box 592 CH-1920 Martigny, Switzerland

Abstract

This document describes a neural method for clustering words and its use in language modeling for speech recognizers. The method is based on clustering the words which appear on similar local context and estimating the parameters needed for language modeling based on these clusters. The language model used is similar to the traditional n-grams.

Contents

Chapter 1

Introduction

The inspiration for this work comes from many sources. The first and oldest is a simple picture, which I saw as a student in my university's neural networks laboratory. In this picture (also found in [Honkela et al., 1995], original idea in [Ritter and Kohonen, 1989]), there were a number of words, and each word seemed to be surrounded by similar words, so that adjectives would be in one corner, verbs in middle and so on. Even though it was apparent, that this was made with a small and simple vocabulary, this aroused my curiosity. How was this done ?

The second source of inspiration was very similar to the first. In data retrieval from a large number of articles, it is very useful to be able to find similar articles. The techniques for doing this are well researched and well known [Salton, 1971].

The third basis is the principle of using "tagged" words in speech recognition. Each word is tagged by its general class (for ex. verb, name, pronoun). The knowledge of this class is used to improve the modeling accuracy of the language model. The tagging is usually done by hand or by complex programs [Gaizauskas et al., 1995].

In this work, these ideas are brought together to form a language model based on word clusters. The idea is similar to the traditional n-grams, but this approach should lend itself better for use with large vocabularies and for language model adaptation.

1.1 Acknowledgments

I would like to thank IDIAP and professor Hervé Bourlard for the opportunity to work on this problem. I thank doctor Mikko Kurimo for helping to formulate these ideas as well as for keeping me from getting lost in the jungle of "not so feasible and a bit too complex" ones. I would also like to thank Guilia Bernardis for the hand tagged material used in early experiment as well as interesting discussions about the subject of this work. I thank Todd Stephenson for helping me to use his code for manipulating sparse matrices in my work and even modifying his code to better fit my purposes. I thank doctor Andrew Morris for proofreading this document.

Chapter 2

Theory

This theory chapter is divided in four sections. In the first, I try to motivate, why it would be useful to cluster words. In the second, the principles of clustering words neurally are explained. In the third, the basics of n-gram language modeling are explained and the fourth one extends these principles to clusters of words.

2.1 Motivation

There are several reasons, why it would be useful to have similar words associated with each other. First of all, we could significantly reduce the number of parameters we need to construct a language model. Instead of having to estimate the transitions from every word in our vocabulary to itself and every other word (up to n^{th} order), we could approximate the transitions between groups, thus greatly reducing the number of parameters. This small set of parameters can be estimated more reliably from smaller training data set. Adding new words would be easier. Just assign a word into its corresponding group and you have already calculated quite reliable estimates for the transition parameters. Adapting the language model to changing subjects of the incoming speech stream should be much easier due to the reduced number of parameters and due the fact, that semantically similar words (monkey, baboon, gorilla) should appear in the same cluster.

2.2 Clustering the Words

How would we assign all of the words known to our speech recognition system to clusters or classes ? What would we do with a word that fits several classes (like play, which is a noun and a verb) ? One way would be to tag the words by hand, but that would be very tedious for bigger vocabularies. There also exists complex, rule-based taggers. The method proposed here is similar to one used in [Ritter and Kohonen, 1989]. It is based on the closest neighbors of a word in text.

First, we assign each word w_i in our vocabulary a vector $\mathbf{v}_i$ of length 1. The correct way to do this is to assign each word a binary vector orthogonal to all other vectors. In this work, random scalar vectors are used to speed up calculations. These random vectors should be independent and mostly orthogonal to each other. Next, we take a big text corpus. For each occurrence w_{ix} of word w_i, we take the word that precedes it w^p_{ix} and add it to the list $W^p_i = \{w^p_{i1}, w^p_{i2}, \ldots, w^p_{im}\}$. We do the same for the word w^f_{ix} that follows the reference word $W^f_i = \{w^f_{i1}, w^f_{i2}, \ldots, w^f_{im}\}$. Then, we take the mean of the corresponding random vectors and concatenate these to form a feature vector $\mathbf{f}_i$, corresponding to word w_i:

$$\mathbf{f}_i = \left[\frac{\sum_{a=0}^{m} \mathbf{v}^p_{ia}}{m}, \quad \frac{\sum_{b=0}^{m} \mathbf{v}^f_{ib}}{m} \right] \tag{2.1}$$

That is, if the number of elements in the random vector was l, the number of elements in the feature vector will be $2l$.

To make the idea clear, we show a simple example where we construct a feature vector $\mathbf{f}_{cat}$ for the word "cat".

A	cat	is	crossing	a	street.	The	cat	has	a	long	tail.
$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\mathbf{v}_5$	$\mathbf{v}_6$	$\mathbf{v}_7$	$\mathbf{v}_8$	$\mathbf{v}_9$	$\mathbf{v}_{10}$	$\mathbf{v}_{11}$	$\mathbf{v}_{12}$

$$\mathbf{f}_{cat} = \left[\frac{\mathbf{v}_1 + \mathbf{v}_7}{2}, \frac{\mathbf{v}_3 + \mathbf{v}_9}{2}\right]$$

Now, let's have a brief thought on what we have accomplished this far: We have mapped each word into a $2l$-dimensional sphere, so that the words occurring in similar contexts are in relatively close proximity. Now, all that is needed to construct the word groups, is to identify clusters of these feature vectors. In this work, the som_pak–package [Kohonen et al., 1996] from Helsinki University of Technology is used to perform the clustering, but any other method for clustering could probably also be used.

What happens to the words with several meanings (as the word "play" mentioned earlier)? These words have more than one context in which they appear, so summing these up should form their own cluster somewhere between the classes they represent. Of course, if the verb meaning of a word is much more frequent than the noun meaning, it is likely that the latter gets ignored. In practice, it seems that there are nice clusters of words with double meanings formed (figure 3.1).

One of the interesting properties of this mapping is that you can automatically assign a group to an out-of-vocabulary word, if you have seen the word occurring in a sentence. Of course, the more sentences where the word occurs, the better the word will match the assigned cluster. It is also possible to add a word to a group by hand, by looking up the appropriate group, if the number of clusters is not prohibitive.

2.3 Language Modeling

2.3.1 Basics

When we "do speech recognition", what are we actually doing ? In strict mathematical terms, we are looking for the most probable word sequence. For humans, this would be affected by the hints we perceive and by the a priori information we have: the sounds we hear, the words we know, our knowledge of the subject of discussion, our knowledge of the speaker, the gestures of the speaker, the lip movements of the speaker, etc. Traditionally, for a speech recognizer, the most probable word sequence is affected by the received acoustic vectors and the a priori information known to recognizer: phoneme models, pronunciation dictionaries and language models. We are thus searching for the most probable word sequence $\hat{W}$, given the acoustic input X and the parameters λ for our a priori models.

$$\hat{W} = \text{argmax}_W P(W|X, \lambda) \tag{2.2}$$

Now, let us split our model parameters λ to parameters of the language model λ_l and other model parameters λ_o. Using Bayes' rule and assuming that the acoustic model is independent of the language model we can modify eq. (2.2) into a more convenient one:

$$P(W|X, \lambda_l, \lambda_o) = \frac{P(W|\lambda_l)\ P(X|W, \lambda_o)}{P(X|\lambda_o)} \tag{2.3}$$

Now, we have separated the computation of the probability $P(W|\lambda_l)$ of word sequence W from other computations. This is the part that we will be concentrating upon, that is, the responsibility of the language model. The probability $P(X|W,\lambda_o)$ of acoustic sequence X, given the word sequence W is governed by pronunciation dictionaries, phone models and on the low level usually by Hidden Markov Models. We can also see that the denominator, the probability of a acoustic sequence $P(X|\lambda_o)$ is the same for all sentences, so we can omit it from maximization (or calculate it only once).

2.3.2 N-grams

As derived in the previous subsection, the mission of our language model is to calculate the probability of word sequence $W = w_1, w_2, w_3, \ldots, w_T$. This calculation can be divided into the calculation of the probability of each word in following manner:

$$\begin{aligned} P(W) &= P(w_1, w_2, w_3, \ldots, w_T) \\ &= P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2, w_1) \cdot \ldots \cdot P(w_T|w_{T-1}, w_{T-2}, w_{T-3}, \ldots, w_1) \end{aligned} \quad (2.4)$$

To estimate all of the probabilities needed in this calculation is of course impossible in practice. From this equation, it is easy to form an approximation using only information about two preceding words. This kind of approximate model is called a trigram model and is widely used in speech recognizers:

$$\begin{aligned} P_{trigram}(W) &= P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2, w_1) \cdot P(w_4|w_3, w_2) \cdot \ldots \cdot P(w_T|w_{T-1}, w_{T-2}) \\ &= P(w_1) \cdot P(w_2|w_1) \cdot \prod_{k=3}^{T} P(w_k|w_{k-1}, w_{k-2}) \end{aligned} \quad (2.5)$$

Of course, the above is easy to extend to any kinds of n-grams:

$$P_{n\_gram}(W) = P(w_1) \cdot P(w_2|w_1) \cdot P(w_3|w_2, w_1) \cdot P(w_{n-1}|w_{n-2}, \ldots, w_1) \cdot \prod_{k=n}^{T} P(w_k|w_{k-1}, \ldots, w_{k-n}) \quad (2.6)$$

The problem with n-gram models when n is big, is that to get reasonable estimates for the probabilities, a huge text corpus is needed. Of course, it also takes a lot of memory to use all of these probabilities. That is why, in practice, we are limited to trigrams.

The n-gram model can be also thought of as a Markov-chain. The first order case, where there would be probabilities for moving from one word to another, would correspond to the bigram case, the second order Markov model corresponding to the trigram case and so on.

2.4 Word Clusters and Language Modeling

How could we use these clusters of words to model a language ? First, to simplify the notation, we define that word w_x, having feature vector $\mathbf{f}_x$, belongs to group G_x^y, which has the cluster center $\mathbf{c}_y$, if the cluster center $\mathbf{c}_y$ is closest to feature $\mathbf{f}_x$ according to the distance metric $d()$:

$$w_x \in G_x^y \mid \mathbf{c}_y = \text{argmin}_{\mathbf{c}_i} d(\mathbf{f}_x, \mathbf{c}_i) \quad (2.7)$$

Here, we keep the notation similar to the trigram model for simplicity. The following equations should be trivial to modify to suit other n-grams. Now, we can approximate (more details in appendix A) the probability of a trigram R from (2.5) by

$$\begin{aligned} P(R) &= P(w_n|w_{n-1}, w_{n-2}) \\ &\approx P(w_n|G_n^{i_n})\ P(G_n^{i_n}|G_{n-1}^{i_{n-1}}, G_{n-2}^{i_{n-2}}) \mid w_n \in G_n^{i_n}, w_{n-1} \in G_{n-1}^{i_{n-1}}, w_{n-2} \in G_{n-2}^{i_{n-2}}) \end{aligned} \quad (2.8)$$

Here, the probability $P(R)$ consists of two parts: The probability of getting from the two previous groups to the current one and the relative probability of a word belonging to a group. This formulation

can be thought of as a second order Hidden Markov Model, where the clusters correspond to states and the recognized word to the emitted symbol.

The probability approximators can be estimated as follows:

$$\hat{p}(w_n|G_n^{i_n}) = \frac{\#w_n}{\#w_x} \mid w_x \in G_n^{i_n} \tag{2.9}$$

$$\hat{p}(G_n^{i_n}|G_{n-1}^{i_{n-1}}, G_{n-2}^{i_{n-2}}) = \frac{\#(w_{n-2}, w_{n-1}, w_n)}{\sum_{x=0}^{X} \#(w_{n-2}, w_{n-1}, w_x)} \mid w_n \in G_n^{i_n}, w_{n-1} \in G_{n-1}^{i_{n-1}}, w_{n-2} \in G_{n-2}^{i_{n-2}} \tag{2.10}$$

where X is the number of the words in the vocabulary.

These approximators have some problems. In spoken language, there are lots of words that often appear together. For example, the sentence "A cup of ..." is much more common than "A cup on ...". If we have grouped all of the prepositions to one group and use only the transition probabilities between groups, this information is completely lost. This is the tradeoff we do when moving from n-grams to cluster models. Equation 2.10 should work also, if the words were randomly assigned to a large number of clusters, but should provide much better approximations if the clusters are somehow sensible. The approximator in equation 2.9 is quite rough. It is possible to find another, smoother approximator for this (see eq. A.7).

Chapter 3

Experiments

3.1 Clustering

The first preliminary experiment was to test how well the clustering of words works. The tricky part here is to define what is a good clustering. The test performed here was very closely dictated by the material available. G. Bernardis has tagged a corpus of address queries in French. A clustering was performed on this material. This clustering was compared to hand tagging. Strictly scientifically taken, this experiment is not completely sensible: Why should the tags chosen by a human be the best possible solution for the kind of language modeling proposed in this work ? Why should we use unsupervised clustering in this kind of limited task ? However, this can be regarded as a proof-of-concept experiment, showing that the clusters are sensible in respect to tags chosen by a human.

The data consists of 4300 free form queries, like the two following:

```
"bonjour" "veuillez" "m'" "indiquer" "le" "numéro" "de" "téléphone" "de" "JORDAN"
"CHRISTIAN" "aa" "ORSONNENS" "s'" "il" "vous" "plaît"

"monsieur" "FLEURY" "YVAN-ALBERT" "au" "chalet" "EDELWEISS" "aa" "CORBEYRIER" "s'"
"il" "vous" "plaît" "madame"
```

Only the words written in capital letters were hand tagged. Words like YVAN-ALBERT were treated as a single word. There was some overlap in hand-tagged classes, that is a word could be both a first name and a street name, for example.

The size of vocabulary was about 5500, of which 4000 were used to construct the map of 21 cluster centers. The length of the random vectors was 170. The feature vector was built using the two preceding and two following words of the reference word, thus the feature vector had the length of 680. Each cluster was marked with a tag corresponding to the most frequent class in its proximity. Then, the other 1500 were used to test this clustering. Each of these words were tagged by the same

Table 3.1: Comparison of hand tagged classes and statistically formed clusters

	# hand tagged	# correctly clustered	% correctly clustered
First name	150	106	71
Family name	621	581	94
Street name	292	189	64
Town name	281	264	94
Name of institution	3	0	0
Out of hand tagged vocabulary	195	16	8

tag as their closest cluster center.

As the data set was quite limited, having very few repetitions of each word, I consider the results presented in table 3.1 to be good. Since there were only very few institution names in the whole corpus, no cluster was formed representing those. The low number of correct out-of-hand-tagged-vocabulary recognitions is also easily explained by the testing method. For example, the word "plaît" would certainly go to its own category, since it is usually at the end of sentence and preceded by words "s' " and "il". Since we chose the words to be used for training in random, each word having equal probability of ending up in the training set regardless of its a frequency of appearance, it is possible that this common word was not in the training set and a cluster for it was not formed.

Last, figure 3.1 shows a clustering using the same data as in the next section, 3.2.1. The 5 most common words for each cluster are printed. Note that not all clusters have 5 words. For example words FEW, LITTLE, MAJOR, SMALL and LARGE have been assigned to the same cluster.

3.2 Perplexity

3.2.1 Data

This experiment uses the same text data as the following speech recognition experiment. The data used for training was taken from Linguistic Data Consortium's TDT-2 English Text Corpus [LDC]. The correct transcriptions from CNN news were used (and not the outputs of a speech recognizer). Text from newswires of New York Times and Associated Press Worldstream Services was also used. All this totaled to more than 50 million words. The test data for perplexity scores was a news transcription from HUB4 evaluation database [HUB-4, 1998]. The test data for the speech recognition test was the audio data corresponding to this transcript. The vocabulary consisted of 20001 words (20000 + out of vocabulary symbol). It should be noted, that a big part of the training data was newswires (written text) as opposed to the test data, which was speech (for speech recognition tests) and speech transcription (for perplexity scores).

3.2.2 Definition of Perplexity

Perplexity can be used to measure how well a language model describes the language. It can be thought as of the average number of choices the language model has when it decides, what is the next word. So for a vocabulary of size N, where each word is equally probable, the perplexity for any text is N. For a language model that describes only one certain text, that is, only one text string is possible, the perplexity is 1. The formal definition is:

$$P_p = P(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}} \tag{3.1}$$

This equation applied to our clustering system (eq. 2.8) yields

$$P_p = \left(P(w_1)P(w_2|w_1) \left[\prod_{k=3}^{N} P(G_k^{i_k}|G_{k-1}^{i_{k-1}}, G_{k-2}^{i_{k-2}}) P(w_k|G_k^{i_k}) \middle| w_k \in G_k^{i_k}, w_{k-1} \in G_{k-1}^{i_{k-1}}, w_{k-2} \in G_{k-2}^{i_{k-2}} \right] \right)^{-\frac{1}{N}} \tag{3.2}$$

3.2.3 Perplexity results

Perplexity scores in relation to the number of clusters are presented in figure 3.2. To put them into perspective, the perplexity using only unigram probabilities and the perplexity for a trigram model is plotted. The perplexity score for trigrams was calculated using backoff to bigrams and unigrams and Good-Turing discounting, as produced by CMU language modeling toolkit [Clarkson and Rosenfeld, 1997]. Pure trigrams were not tested, since this test would not reflect anything, but the data sparsity problem

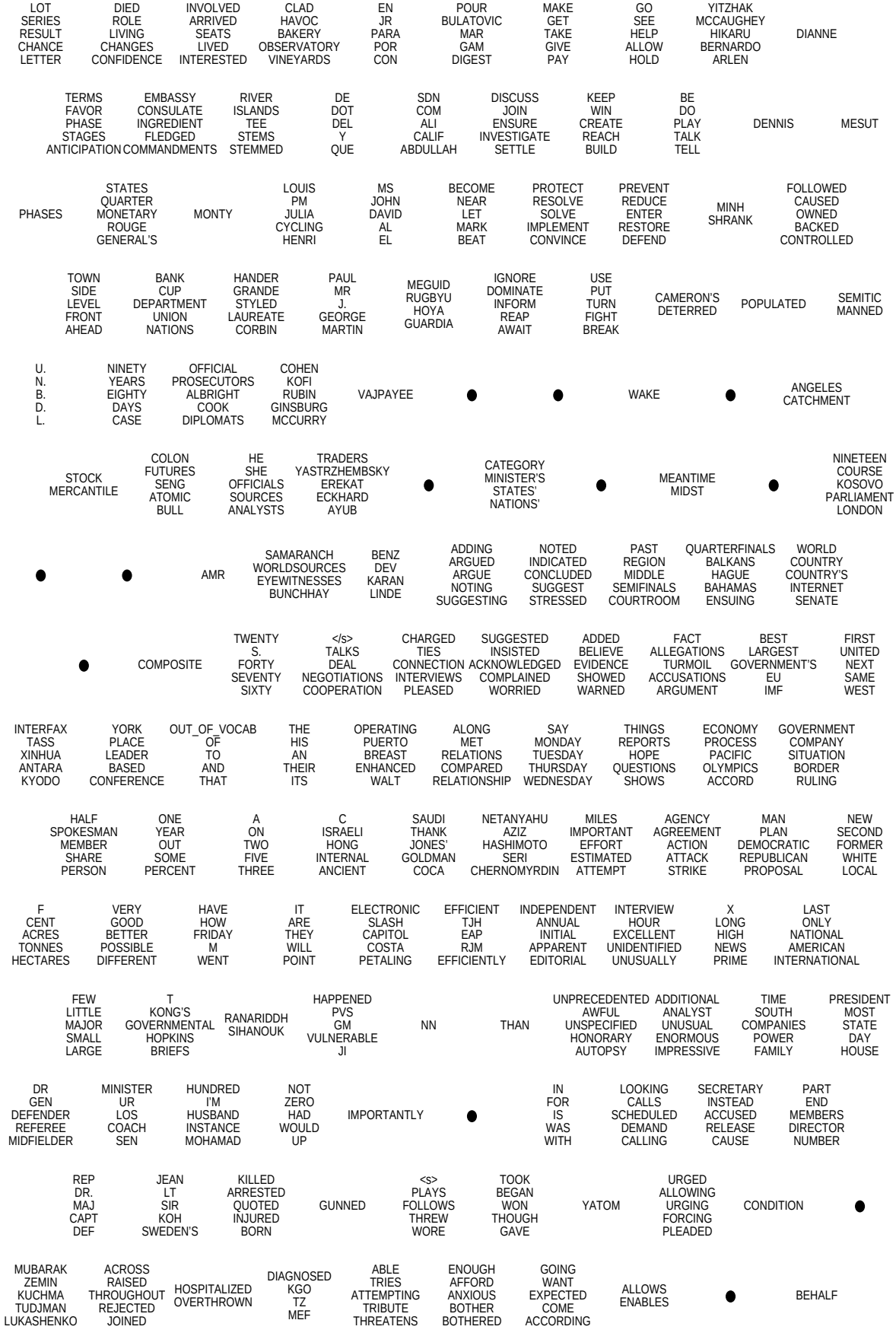

Figure 3.1: *5 most common words in a cluster, 150 cluster centers*

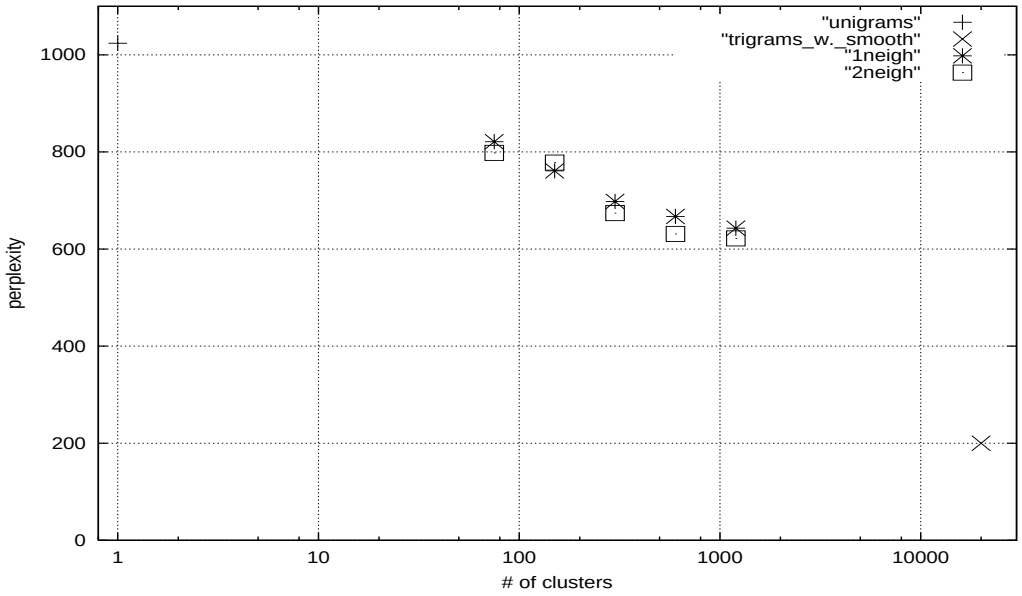

Figure 3.2: *Perplexity and number of clusters (log scale). 1neigh is for clustering using feature vector constructed from closest neighbors of word, 2neigh from two closest ones.*

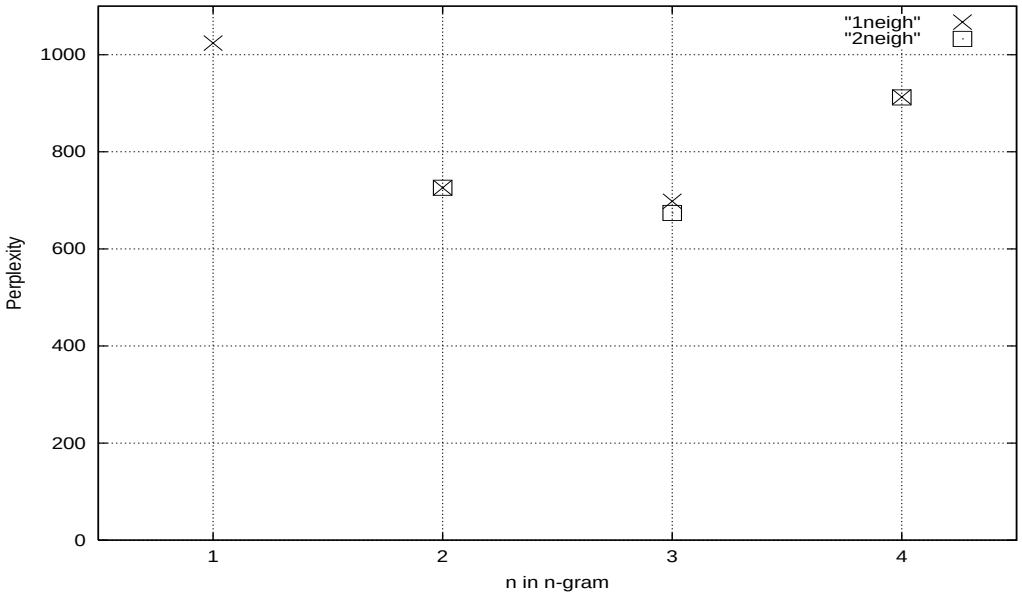

Figure 3.3: *Perplexity and the order of the model, 300 clusters*

for trigrams. No explicit smoothing was used in the clustering approaches. Transitions deemed impossible by the clustering model were given a minimum probability of 10^{-7} in order to produce meaningful scores, but to not add too much extra probability mass to calculations. The results correspond to a 3rd order model (that is, a 2nd order Markov chain).

The tests were conducted using different number of cluster centers. The clustering was done using only 1 adjacent word form either side of the reference word, or with using 2 adjacent words from either side to construct the feature vector (see eq. 2.1).

The effect of the order of the model was also tested. The results are presented in figure 3.3. These tests were all conducted using 300 cluster centers.

3.3 Speech recognition

The language model used in the speech recognition experiments is the same as the one used in perplexity testing, subsection 3.2.1. The Abbot speech recognition system [Robinson et al., 1996] was used. The outputs of the neural network used for acoustic modeling, that is, the phone posteriori probabilities, were used to speed up the testing. The Noway–decoder [Renals and Hochberg, 1995] is a part of Abbot and is responsible for combining the acoustic probabilities, the phoneme probabilities, pronunciation dictionaries and the language model. It was modified to use the clusters described in the theory section 2.4 and arbitrary large n-grams.

The results with respect to different numbers of clusters is presented in figure 3.4. These values are for a 3rd order model (that is, a 2nd order Markov chain). It can be seen that the perplexity scores reported in the previous section can only be considered as approximative and do not translate very well to speech recognition results. Features where the closest neighbors of a word were used gave worse perplexity scores, but better speech recognition results than the features using two closest neighbors. It was also tested how the order of the model affects the results. 300 cluster centers were used for these calculations. The results are given in figure 3.5.

The baseline trigram score (and consequently all other scores) were worse than usually given for this test set. Amongst the possible reasons for this are that during the decoding phase, the search paths were trunctuated quite early to reduce the decoding time and also the language model training data did not probably match the test data in the best possible way.

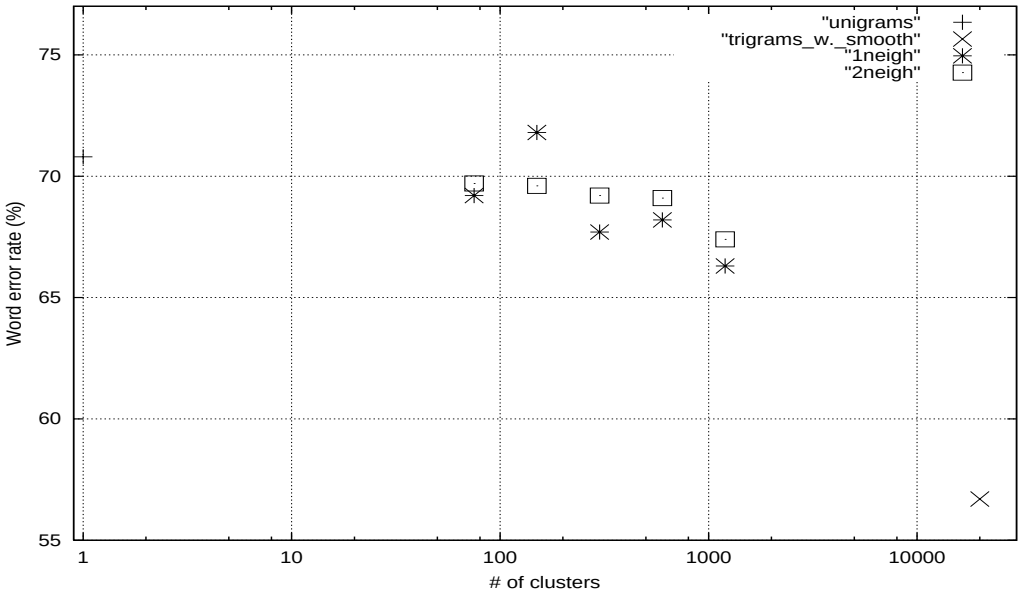

Figure 3.4: *Speech recognition results and the number of clusters (log scale). 1neigh is for clustering using feature vector constructed from closest neighbors of word, 2neigh from two closest ones*

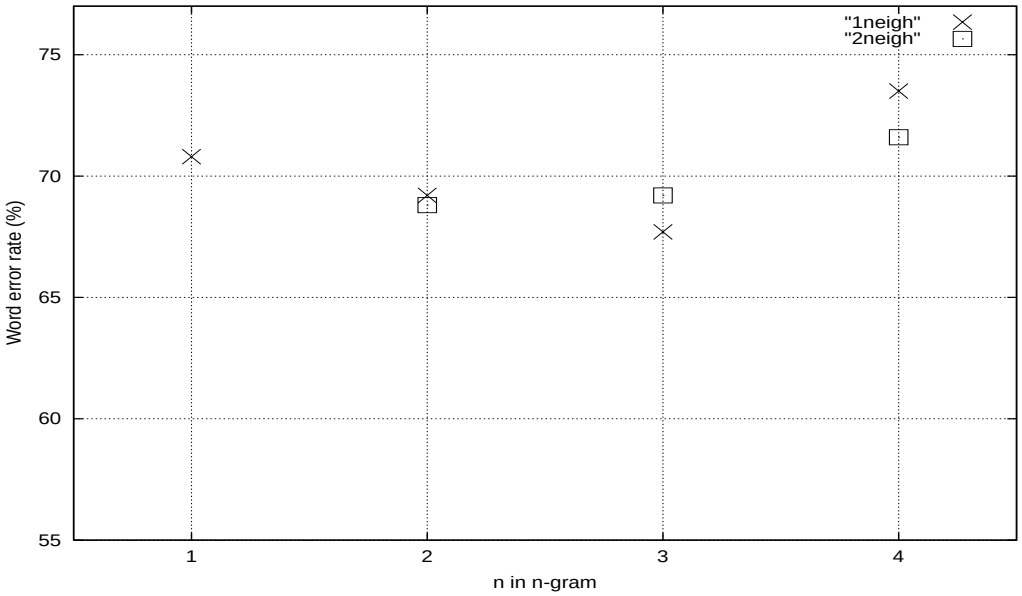

Figure 3.5: *Speech recognition results and the order of the model, 300 clusters.*

Chapter 4

Conclusions

The method for constructing a language model presented in this work seems to be capable of reducing the number of parameters needed at the expense of the recognition results. The approach would probably be most useful when the vocabulary needed is very big and it becomes very hard to collect enough data to form reliable estimates for the transition probabilities.

Based on preliminary tests, the results seem to be rather sensitive to the clustering parameters. It is possible, that fine tuning these parameters or using a different algorithm to perform clustering would improve the results. When there are a lot of clusters, the same data scarcity problem as with trigrams becomes apparent. The same applies for n-grams, where n is bigger than 3. This could be helped with similar smoothing methods that are used with traditional n-grams [Andersen, 1998].

This approach should lend itself to a variety of adaptation methods, but before concentrating on these, the base performance needs to be improved. One possible way to improve the performance would be to pick out the collocations, which appear relatively much more often together than separately and model these collocations as one word [Andersen, 1998].

Bibliography

[Andersen, 1998] Andersen, J. M. (1998). Baseline system for hybrid speech recognition on french (experiments on bref). IDIAP-COM 07, IDIAP.

[Clarkson and Rosenfeld, 1997] Clarkson, P. and Rosenfeld, R. (1997). Statistical language modeling using the CMU–Cambridge toolkit. In *Proceedings ESCA Eurospeech*. See also http://svr-www.eng.cam.ac.uk/~prc14/toolkit.html.

[Gaizauskas et al., 1995] Gaizauskas, R., Wakao, T., Humphreys, K., Cunningham, H., and Wilks, Y. (1995). Description of the LaSIE system as used for MUC-6. In *Proc. Sixth Message Understanding Confrence (MUC-6)*, Maryland.

[Honkela et al., 1995] Honkela, T., Pulkki, V., and Kohonen, T. (1995). Contextual relations of words in Grimm tales analyzed by self-organizing maps. In Fogelman-Soulie, F. and Gallinari, P., editors, *ICANN-95, IEEE International Conference on Artificial Neural Networks*, volume 2, Paris.

[HUB-4, 1998] (1998). 1998 Hub-4 broadcast news technology evaluation project. http://www.itl.nist.gov/iaui/894.01/hub4_98/hub4_98.htm.

[Kohonen et al., 1996] Kohonen, T., Hynninen, J., Kangas, J., and Laaksonen, J. (1996). SOM_PAK: The self–organizing map programming package. Technical Report TKK-F-A31, Helsinki University of Technology, Laboratory of Computer and Information Scence. Code available at http://www.cis.hut.fi/nnrc/som_pak.

[LDC] Linguistic data consortium. http://morph.ldc.upenn.edu/Projects/TDT2/.

[Renals and Hochberg, 1995] Renals, S. and Hochberg, M. (1995). Decoder technology for connectionist large vocabulary speech recognition. Technical report, Dept. of Computer Science, University of Sheffield. Dept. of Computer Science Memo +CS-95-17.

[Ritter and Kohonen, 1989] Ritter, H. and Kohonen, T. (1989). Self–organizing semantic maps. *Biol. Cyb.*, 61(4).

[Robinson et al., 1996] Robinson, T., Hochberg, M., and Renals, S. (1996). *Automatic Speech and Speaker Recognition - Advanced Topics*, chapter 10. Kluwer Academic Publishers. Editors Lee, C.H., Paliwa, K.K. and Soong, F.K.

[Salton, 1971] Salton, G. (1971). *The SMART Retrieval System – Experiments in Automatic Document Processing*. Prentice–Hall.

Appendix A

From n-grams to cluster model

This appendix is provided to give the reader a bit more in–depth review about the approximations leading from n-gram models (eq. 2.6) to clustered models (eq. 2.8).

First, we do some simple manipulations on the trigram probability equation:

$$
\begin{aligned}
& P(w_n|w_{n-1}, w_{n-2}) \\
&= \sum_{i_n=0}^{N} P(w_n, G_n^{i_n}|w_{n-1}, w_{n-2}) \\
&= \sum_{i_n=0}^{N} P(w_n|w_{n-1}, w_{n-2}, G_n^{i_n}) P(G_n^{i_n}|w_{n-1}, w_{n-2}) \\
&= \sum_{i_n=0}^{N} P(w_n|w_{n-1}, w_{n-2}, G_n^{i_n}) \sum_{i_{n-1}=0}^{N} P(G_n^{i_n}, G_{n-1}^{i_{n-1}}|w_{n-1}, w_{n-2}) \\
&= \sum_{i_n=0}^{N} P(w_n|w_{n-1}, w_{n-2}, G_n^{i_n}) \sum_{i_{n-1}=0}^{N} P(G_n^{i_n}|w_{n-1}, w_{n-2}, G_{n-1}^{i_{n-1}}) P(G_{n-1}^{i_{n-1}}|w_{n-1}, w_{n-2}) \\
&= \sum_{i_n=0}^{N} P(w_n|w_{n-1}, w_{n-2}, G_n^{i_n}) \sum_{i_{n-1}=0}^{N} \sum_{i_{n-2}=0}^{N} P(G_n^{i_n}, G_{n-1}^{i_{n-2}}|w_{n-1}, w_{n-2}, G_{n-1}^{i_{n-1}}) P(G_{n-1}^{i_{n-1}}|w_{n-1}, w_{n-2}) \\
&= \sum_{i_n=0}^{N} P(w_n|w_{n-1}, w_{n-2}, G_{n-2}^{i_n}) \sum_{i_{n-1}=0}^{N} \sum_{i_{n-2}=0}^{N} P(G_n^{i_n}|w_{n-1}, w_{n-2}, G_{n-1}^{i_{n-1}}, G_{n-2}^{i_{n-2}}) P(G_{n-1}^{i_{n-1}}|w_{n-1}, w_{n-2}) \\
&\quad \cdot P(G_{n-2}^{i_{n-2}}|w_{n-1}, w_{n-2}, G_{n-1}^{i_{n-1}})
\end{aligned}
\tag{A.1}
$$

where N is the number of clusters. Now we need to start approximating: First, the probability of a cluster G_x only depends on the corresponding word w_x:

$$
\sum_{i_n=0}^{N} P(w_n|w_{n-1}, w_{n-2}, G_n^{i_n}) \sum_{i_{n-1}=0}^{N} \sum_{i_{n-2}=0}^{N} P(G_n^{i_n}|w_{n-1}, w_{n-2}, G_{n-1}^{i_{n-1}}, G_{n-2}^{i_{n-2}}) P(G_{n-1}^{i_{n-1}}|w_{n-1}) P(G_{n-2}^{i_{n-2}}|w_{n-2}) \tag{A.2}
$$

The transition from cluster to another is sufficiently modeled by

$$
\sum_{i_n=0}^{N} P(w_n|w_{n-1}, w_{n-2}, G_n^{i_n}) \sum_{i_{n-1}=0}^{N} \sum_{i_{n-2}=0}^{N} P(G_n^{i_n}|G_{n-1}^{i_{n-1}}, G_{n-2}^{i_{n-2}}) P(G_{n-1}^{i_{n-1}}|w_{n-1}) P(G_{n-2}^{i_{n-2}}|w_{n-2}) \tag{A.3}
$$

The essential information about transitions is included in the term $P(G_n^{i_n}|G_{n-1}^{i_{n-1}}, G_{n-2}^{i_{n-2}})$. This is clearly true, when each word has been assigned to its own cluster and is the more inaccurate the less there

are cluster centers. Thus, we can get rid of trigrams (but we need not to, if we only want to smooth them).

$$\sum_{i_n=0}^{N} P(w_n|G_n^{i_n}) \sum_{i_{n-1}=0}^{N} \sum_{i_{n-2}=0}^{N} P(G_n^{i_n}|G_{n-1}^{i_{n-1}}, G_{n-2}^{i_{n-2}}) P(G_{n-1}^{i_{n-1}}|w_{n-1}) P(G_{n-2}^{i_{n-2}}|w_{n-2}) \tag{A.4}$$

The probabilities $P(G_x^y|w_x)$ can be approximated by

$$P(G_x^y|w_x) = \frac{e^{-d(\mathbf{c}_y, \mathbf{f}_x)}}{\sum_i e^{-d(\mathbf{c}_i, \mathbf{f}_x)}} \tag{A.5}$$

where word w_x is associated with feature vector $\mathbf{f}_x$ and G_x^y with cluster center $\mathbf{c}_y$. The division is made to keep the contribution of all cluster centers summing to one. $d()$ is a distance metric, for example Euclidean.

If we want to further simplify, we can assign each word to only one cluster, that is $P(G_x^y|w_x)$ for cluster y is 1 and 0 for others. We use notation $w_x \in G_x^y$ for assigning the word to closest cluster:

$$w_x \in G_x^y \mid \mathbf{c}_y = \text{argmin}_{\mathbf{c}_i} d(\mathbf{f}_x, \mathbf{c}_i) \tag{A.6}$$

This leads us to

$$\sum_{i_n=0}^{N} P(w_n|G_n^{i_n}) P(G_n^{i_n}|G_{n-1}^{i_{n-1}}, G_{n-2}^{i_{n-2}}) \;\Big|\; w_{n-1} \in G_{n-1}^{i_{n-1}}, w_{n-2} \in G_{n-2}^{i_{n-2}} \tag{A.7}$$

Assigning word w_n to only one cluster:

$$P(w_n|G_n^{i_n}) P(G_n^{i_n}|G_{n-1}^{i_{n-1}}, G_{n-2}^{i_{n-2}}) \;\Big|\; w_n \in G_n^{i_n}, w_{n-1} \in G_{n-1}^{i_{n-1}}, w_{n-2} \in G_{n-2}^{i_{n-2}} \tag{A.8}$$

And thus we have arrived at the equation 2.8.