Acta Universitatis Sapientiae
Electrical and Mechanical Engineering, 1 (2009) 187-197

Text Conditioning and Statistical Language Modeling Aspects for Romanian Language

József DOMOKOS[1,2], Gavril TODEREAN[2]

[1] Department of Electrical Engineering, Faculty of Technical and Human Sciences, Sapientia University, Tîrgu Mureş, Romania,
e-mail: domi@ms.sapientia.ro
[2] Department of Communications, Faculty of Electronics, Telecommunications and Information Technology, Technical University of Cluj-Napoca, Cluj-Napoca, Romania,
e-mail: toderean@pro3soft.ro

Manuscript received March 30, 2009; revised June 30, 2009.

Abstract: In this paper we present a synthesis of the theoretical fundamentals and some practical aspects of statistical (n-gram) language modeling which is a main part of a large vocabulary statistical speech recognition system. There are presented the unigram, bigram and trigram language models as well as the add one, Witten-Bell and Good-Turing estimator based Katz back-off smoothing algorithms. The perplexity measure of a language model used for evaluation is also described.
The practical experiments were made on Romanian Constitution corpus. Text normalization steps before the language model generation are also presented. The results are ARPA-MIT format language models for Romanian language. The models were tested and compared using perplexity measure.
Finally some conclusions are drawn based on the experimental results.

Keywords: Romanian statistical language modeling, natural language processing, text conditioning, ARPA-MIT language model format, n-gram language model, smoothing, perplexity.

1. Introduction

Statistical speech recognition is based on Hidden Markov Models (HMMs). Such a system, depicted in *Fig. 1*, is built using multiple chained HMMs for acoustic modeling and language modeling [1], [2], [3].

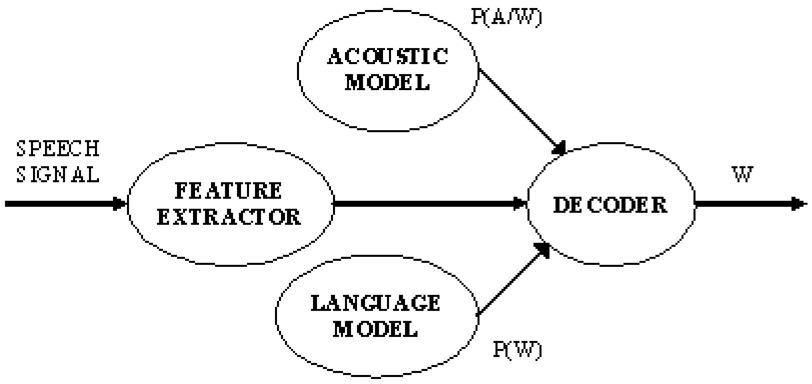

Figure 1: Statistical speech recognition system architecture.

The system presented in *Figure 1*, can be described mathematically as follows [1][4]: we have a set of acoustic vectors $A = \{a_1, a_2, ..., a_n\}$ and we are searching the most probable word sequence: $W^* = \{w_1, w_2, ..., w_m\}$.

$$W^* = \underset{w}{\operatorname{argmax}} \{P(W \mid A)\} \tag{1}$$

Using the Bayes formula, we can transcribe (1) as follows:

$$W^* = \underset{w}{\operatorname{argmax}} \left\{ \frac{P(A \mid W) \cdot P(W)}{P(A)} \right\} \tag{2}$$

We know, that probability of acoustic vector *P(A)* is constant, and we have:

$$W^* = \underset{w}{\operatorname{argmax}} \{P(A \mid W) \cdot P(W)\} \tag{3}$$

In (3) we can distinguish [4], [5], [6], [7]:

- *P(W)* – the language model;
- *P(A|W)* – the acoustic model.

The acoustic modeling part of the speech recognition system can be developed using HMMs, Gaussian Mixture Models (GMMs) or Artificial Neural Networks (ANNs).

The language modeling part of the system can be one of the following [2]:

- statistical language model;
- context-free grammar (CFG);
- Probabilistic context-free grammar (PCFG).

In this paper we want to present the statistical n-gram type language model which is the most powerful and the most widely used one, and we want to create Romanian language models in ARPA-MIT [8] standard format for large vocabulary continuous speech recognition systems.

We have developed so far the feature extractor module and also the acoustic modeling part (using artificial neural networks) of an automatic speech recognition system.

2. Statistical language modeling

The speech can be considered a stochastic process and every linguistic unit (phoneme, syllabus, or word) can be considered a random variable with a random probability distribution. If we are talking at word level, the n-gram language models try to estimate the probability of the next word based on the history (the last *n-1* preceding words)) [1], [4], [6].
The language model try to estimate the probability of word sequence: $w_1^n = (w_1, w_2, \ldots, w_n)$, which is:

$$p(w_1^n) = p(w_1) \cdot p(w_2 \mid w_1) \cdot p(w_3 \mid w_1^2) \ldots \cdot p(w_n \mid w_1^{n-1}) \tag{4a}$$

$$p(w_1^n) = p(w_1) \cdot \prod_{k=1}^{n} p(w_k \mid w_1^{k-1}) \tag{4b}$$

Using Markov assumption, the history can be reduced to the last *n-1* words, and we have:

$$p(w_k \mid w_1^{k-1}) \approx p(w_k \mid w_{k-n+1}^{k-1}) \tag{5}$$

Even (5) is hard to compute for $n > 3$ because we need a large training corpus to properly evaluate the probabilities.

For $n = 1 \ldots 3$ we have:

- Unigram language model ($n=1$);
- Bigram language model ($n=2$);
- Trigram language model ($n=3$).

A. Unigram language model

The unigram language model considers all words independent. This means that no history information is involved.

$$P(w_k \mid w_1^{k-1}) \approx P(w_k) \tag{6}$$

If we use (4), the probability estimation for the unigram model will be:

$$p(w_1^n) = \prod_{k=1}^{n} p(w_k) \tag{7}$$

B. Bigram language model

Bigram language model takes into consideration one word for history.

$$P(w_k \mid w_1^{k-1}) \approx P(w_k \mid w_{k-1}) \tag{8}$$

If we substitute (8) into (4), we have the probability estimation formula for bigram language model:

$$p(w_1^n) = p(w_1) \cdot \prod_{k=2}^{n} p(w_k \mid w_{k-1}) \tag{9}$$

C. Trigram language model

Trigram language model uses a two-word history.

$$P(w_k \mid w_1^{k-1}) \approx P(w_k \mid w_{k-1}, w_{k-2}) = P(w_k \mid w_{k-2}^{k-1}) \tag{10}$$

The probability estimation formula is given by (11).

$$p(w_1^n) = p(w_1) \cdot p(w_2 \mid w_1) \cdot \prod_{k=3}^{n} p(w_k \mid w_{k-1}, w_{k-2}) \tag{11}$$

3. Probability estimation and smoothing

The probabilities for (7), (9), and (11) can be simply calculated using MLE (Maximum Likelihood Expectation) algorithm [2]. Thus we have the following MLE estimators for unigram, bigram and trigram language models:

$$p(w_k) = \frac{n_k}{N}, \tag{12}$$

$$p(w_k \mid w_{k-1}) \approx \frac{Nr.(w_{k-1}, w_k)}{Nr.(w_{k-1})}, \tag{13}$$

$$p(w_k \mid w_{k-1}, w_{k-2}) \approx \frac{Nr.(w_{k-2}, w_{k-1}, w_k)}{Nr.(w_{k-2}, w_{k-1})}, \tag{14}$$

Where:

n_k – is the number of occurrences of word w_k;

N – is the total number of words in training corpus;

Nr. (...) – is the number of occurrences of a specific word sequence.

These probabilities calculated using MLE algorithm do not provide useful results. In order to be able to use the probabilities in language modeling experiments, they must be smoothed [4], [6], [7], [9].

Smoothing means that a probability mass is retained from high probabilities to be reallocated to zero or small probability values. There are a lot of useful smoothing techniques [4], [9], [10]:

- Add one or Laplace smoothing;
- Good-Turing estimator;

- Back-off or Katz smoothing;
- Kneser - Ney smoothing;
- Jelinek - Mercer smoothing or interpolation.

For practical experiments we used Good - Turing estimator and back-off smoothing. We had also implemented the add-one and Witten-Bell smoothing algorithms in Microsoft Visual Studio environment using C++ programming language.

A. Good – Turing estimator

The Good-Turing estimator comes from biology where it was used for species estimation. The general form of the estimator is [4]:

$$P(X) = \frac{r^*}{N}, \quad \text{where}, r^* = (r+1) \cdot \frac{E(N_{r+1})}{E(N_r)} \tag{15}$$

In (15) we have the following notations:

r is the number of occurrences of word X;

N_r is the number of words which occurs exactly r times in the training corpus;

N is the total number of words from the training corpus;

E is an estimation function for N_r;

r^* is the adjusted number of occurrences;

The total value of probability calculated using Good-Turing estimator is always smaller than *1*. The remaining probability mass is reallocated to the unseen words from the vocabulary. The simplest way to choose the estimation function E [5] is presented in (16).

$$\frac{E(n+1)}{E(n)} = \frac{n}{n+1} \cdot \left(1 - \frac{E(1)}{N}\right) \tag{16}$$

B. Katz back – off smoothing

Back-off smoothing was firstly introduced by Katz. He showed that MLE estimation of probabilities is good enough if the number of occurrences of a word is bigger than a threshold value $K = 6$ [4].

All the probabilities for n-gram word sequences which have an occurrence number between *0* and K will be smoothed using Good-Turing estimator to save probability mass for unseen word sequences. If a word sequence has zero occurrences we try to estimate its probability using the inferior *(n-1)*-gram model. If the occurrence is still zero for this inferior model we continue to back-off to a lower model. Finally if we reach the unigram model, we have the relative frequency of a word bigger than zero.

For a trigram back-off model we have the fallowing relations:

$$\hat{p}(w_3 \mid w_1, w_2) = \begin{cases} f(w_3 \mid w_1, w_2), & \text{if nr. } (w_1, w_2, w_3) \geq K \\ \alpha \cdot \mathrm{Q_T}(w_3 \mid w_1, w_2), & \text{if } 0 < \text{nr.}(w_1, w_2, w_3) < K \\ \beta \cdot \hat{p}(w_3 \mid w_2), & \text{else} \end{cases} \tag{17}$$

$$\hat{p}(w_3 \mid w_2) = \begin{cases} f(w_3 \mid w_2), & \text{if nr. } (w_2, w_3) \geq L \\ \alpha \cdot \mathrm{Q_T}(w_3 \mid w_2), & \text{if } 0 < \text{nr.}(w_2, w_3) < L \\ \beta \cdot f(w_3), & \text{else} \end{cases} \tag{18}$$

4. Language model evaluation

Language model evaluation can be done in different ways, for instance using [4][6]:

- random sentence generation;
- words reordering in sentences;
- perplexity;
- integration in an existing speech recognition system.

In our experiments we used perplexity to measure the quality of language models. Perplexity is the most used measurement for language model evaluation.

Perplexity can be defined using entropy from information theory. For a random variable $X=\{x_1, x_2, \ldots, x_N\}$, the entropy can be defined:

$$H(X) = -\sum_{x \in X} p(x) \cdot \log_2 p(x) \tag{19}$$

Instead of entropy, we use the entropy rate calculated as follows:

$$\frac{1}{N} H(w_1^n) = -\frac{1}{N} \sum_{w \in V} p(w_1^n) \cdot \log_2 p(w_1^n) \tag{20}$$

For a real language we should consider infinitely long word sequences:

$$\frac{1}{N} H(w_1^n) = \lim_{x \to \infty} -\frac{1}{N} \sum_{w \in V} p(w_1^n) \cdot \log_2 p(w_1^n) \tag{21}$$

Using the Shannon – McMillan - Breiman theorem, if the language is stationary and ergodic (which is true for the natural languages), the above formula can be simplified:

$$H(L) = \lim_{x \to \infty} \left(-\frac{1}{N} \log_2 p(w_1^n) \right) \tag{22}$$

Finally we use a large training corpus to estimate probabilities p* and we have the *logprob* value instead of the entropy rate:

$$LP = -\frac{1}{N}\log_2 p^*(w_1^n) \tag{23}$$

Perplexity is defined as:

$$PP = 2^{LP} \tag{24}$$

5. Text conditioning

Collecting sufficient language model training data for good speech recognition performance in a new domain is often difficult. There are some text corpora for Romanian, which can be used for language modeling, but they are not normalized. This chapter presents the text normalization steps which are used to make these data more suitable for language model training.

Text is unlike speech in a variety of ways. For example, a written text may also include numbers, abbreviations, acronyms, punctuation, and other "non-standard words" (NSWs) which are not written in their spoken form. In order to effectively use this text for language modeling, these items must be converted to their spoken forms. This process has been referred to as text conditioning or normalization and is often used in text-to-speech systems.

State of the art language modeling tools like HLMTools [8], SRI LM [11] or CMU LM [12] do not provide professional text conditioning tools. A set of text conditioning tools are available from the Linguistic Data Consortium (LDC). A more systematic approach to the NSW normalization problem is referred to here as the NSW tools [13], [14]. These tools perform text normalization using a set of ad-hoc rules, converting numerals to words and expanding abbreviations listed in a table. They also use models trained on data from several categories. The NSW tools perform well in a variety of domains, unlike the LDC tools which were developed for business news.

Text normalization for Romanian is a hard process because of the diacritic characters as well.

Our system performs the following basic conditionings:

- it segments the text into sentences on the basis of punctuation, marking with <s> and </s> tags the beginning and the end of the sentences and puts only one sentence per line;
- eliminates most punctuation symbols;
- converts numbers into words;
- converts the whole text to uppercase;
- deletes all empty lines;
- eliminates redundant white spaces;
- replaces diacritic characters;

6. Experimental results

The built language models are based on Romanian Constitution text corpus. This little corpus contains 9936 words including both the train part and the test part (a total number of words). We count n-grams up to *n = 4,* however the corpus size does not allow us to compute valuable trigram and four-gram probabilities as you can see from perplexity results.

In *Table 1* the four-grams with the greatest frequency of appearance can be seen.

Table 2 presents the most probable 15 words from Romanian Constitution corpus.

Table 1. Most frequent four-grams in Romanian Constitution corpus

Word 1	Word 2	Word 3	Word 4	Number of appearance
</S>	<S>	DREPTUL	LA	27
DE	LEGE	</S>	<S>	16
SE	STABILESC	PRIN	LEGE	12
</S>	<S>	DREPTUL	DE	11
PRIN	LEGE	ORGANICA	</S>	10
LEGE	ORGANICA	</S>	<S>	10
</S>	<S>	CAMERA	DEPUTATILOR	9
CONDITIILE	LEGII	</S>	<S>	8
IN	CONDITIILE	LEGII	</S>	8
CAMERA	DEPUTATILOR	SI	SENATUL	8

The total number of distinct words in corpus is 1928, grouped in 718 sentences. 963 of them had more than one appearance.

We generated a dictionary from the most probable 963 words from the corpus (in fact these words appear more than once in the training corpus), and then we mapped all the other words into an unknown word class. We than generated the unigram, bigram, and trigram language models with Katz cut-off based on the corpus. The built language models were stored in ARPA MIT standard format.

Language model evaluation was made using perplexity measure for the three models. The perplexity results of the created models using the 963 - word dictionary, are presented in *Table 3*.

We made a second experiment with a smaller dictionary, containing only the words with appearance greater than 2 (626 words). The perplexity results of our second experiment using the dictionary with 626 words are synthesized in *Table 4*.

Table 2. The most probable 15 words from Romanian Constitution

Word	Appearance	Word	Appearance	Word	Appearance
</S>	718	A	195	AL	71
<S>	718	LA	151	DREPTUL	70
DE	470	SE	128	PRIN	69
SI	405	SAU	116	PENTRU	68
IN	287	ESTE	82	CU	66

Table 1. Perplexity results for Romanian Constitution corpus using a 963-word dictionary.

Model	Perplexity
Unigram	559.74
Bigram	397.37
Trigram	419.52

Table 2. Perplexity results for Romanian Constitution corpus using a 626-word dictionary.

Model	Perplexity
Unigram	509.64
Bigram	332.24
Trigram	419.23

7. Conclusions and future works

We can see from *Table 2* that the most frequent words in Romanian Constitution corpus are the prepositions: *“de”, “şi”,” în”, “a”, “la”, “se”, “sau”, “al”, “prin”*, *“cu”* etc. The verb *“este”(to be)* has surprisingly high frequency of appearance, and because of the text corpus domain we also have the noun *“dreptul”(law)* in the first 15 words in order of frequency of appearance.

The sentence start tags and sentence end tags are also present in the four-grams and in the most frequent word list because of the short sentences of the corpus.

The best results are achieved using bigram model. The trigram model cannot improve the results because there is insufficient data for training the model.

We can see from the results that if the number of words increases, the perplexity of the model increases too, and the model has weaker quality. The models built by eliminating the words with occurrences smaller than a threshold are simpler and perform better. This threshold can be experimentally settled. This technique is called in the literature n-gram pruning [1], [2].

The words with single occurrence do not improve the model quality, they will raise perplexity and they should be eliminated from the vocabulary. We have drawn the same conclusion in our previous work [1] based on the Susanne Corpus.

In conclusion the used n-gram model dimension should be chosen considering the amount of training data available. This little corpus is good enough for preliminary testing of a text conditioning and language modeling tool and we can train well enough just unigram and bigram language models.

As future work, we would like to improve the text conditioning tool with diacritic restoration feature, and try to generate language models based upon a much bigger training corpus of Romanian journal articles and to implement the state of the art Kneser - Ney smoothing algorithm [4],[8],[10],[11].

In order to compare our language modeling tools with others we will try to use open source language modeling toolkits (e.g., CMU-LM [9], SRILM [7]) on the same corpora.

We also try to collect a larger Romanian text corpus based on WEB resources.

Acknowledgements

This research work is supported by the Institute for Research Programs of the Sapientia University.

References

[1] Jelinek, F., "Statistical Methods for speech recognition", *The MIT Press*, 2001.

[2] Becchetti, C., Ricotti, L. P., "Speech recognition. Theory and C++ implementations", *John Wiley & sons*, 1999.

[3] Domokos, J., Toderean, G., Buza, O., "Statistical Language modeling on Susanne corpus", *IEEE International Conference COMMUNICATIONS 2008, Proceedings*, pp. 69-72, 2008.

[4] Jurafsky, D., Martin, J. H., "Speech and language processing. An introduction to Natural language Processing", *Computational Linguistics and Speech Recognition, Prentice Hall*, 2000.

[5] Huang, X., Acero, A., Hon, H., "Spoken Language Processing. A Guide to Theory, Algorithm & System Development", *Prentice Hall*, 2001.

[6] Manning, C., Heinrich, S., "Foundations of statistical language processing", *The MIT Press*, 1999.

[7] Rosenfeld, R., "Two decades of statistical language modeling: where do we go from here?"**,** *Proceedings of the IEEE*, Vol. 88, pp. 1270 – 1278, 2000.

[8] Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P., "The HTK Book"*, Cambridge University Engineering Department*, 2005.

[9] Chen, S., Goodman J., "An Empirical Study of Smoothing Techniques for Language Modeling", *Harvard Computer Science Technical report TR-10-98*, 1998.

[10] Goodman, J., Gao, J., "Language model size reduction by pruning and clustering", ICSLP-2000, *International Conference on Spoken Language Processing*, Beijing, 2000.

[11] Stolcke, A., "SRILM - An Extensible Language Modeling Toolkit", *Proceedings of International Conference on Spoken Language Processing*, Denver, Colorado, 2002.

[12] Clarkson, P. R., Rosenfeld, R., "Statistical Language Modeling Using the CMU-Cambridge Toolkit", *Proceedings ESCA Eurospeech*, 1997.

[13] Paul, D. B., Baker, J. M., **"**The design for the wall street journal-based CSR corpus"**,** *Workshop on Speech and Natural Language, Proceedings*, pp. 357 – 362, 1992.

[14] Schwarm, S., Ostendorf, M., **"**Text normalization with varied data sources for conversational speech language modeling"**,** *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '02*, Vol. 1, pp. 789 – 792, 2002.