On an Empirical Study of Smoothing Techniques for a Tiny Language Model

Freha Mezzoudj
University of Science and Technology-Mohamed Boudiaf (USTO-MB), Oran, Algeria
Hassiba Benbouali University of Chlef, Algeria,
fmezzoudj@gmail.com

Mourad Loukam
Hassiba Benbouali University of Chlef, Algeria
Natural Language Processing Team, LMA Laboratory
mourad.loukam@univ-chlef.dz

Abdelkader Benyettou
University of Science and Technology-Mohamed Boudiaf (USTO-MB), Oran, Algeria
SIMPA Laboratory
a_benyettou@yahoo.fr

ABSTRACT

The language models (LM) are an important module in many areas of natural language processing, in particular speech recognition and machine translation. In this experimental work, we present the most popular smoothing methods and their effects on statistical language modelling. We compare the behavior of twelve smoothing algorithms that have been developed in speech and natural language processing fields, using a small but novel text corpus of French radio show transcription to construct and improve tiny language models. The perplexity (average word branching factor), which measures the performance of our LM, ranked from 195.9 to 165.4. The best result is obtained by Modified Kneser-Ney algorithm, with the interpolation version. The details of the experimentation are given. We consider the obtained results good and in agreement with the literature.

General Terms

Algorithms, Measurement, Performance, Experimentation, Languages

Keywords

Language model, n-gram, perplexity, Additive smoothing, Absolute discounting, Witten-bell smoothing, Kneser-Ney smoothing, interpolation, backoff

1. INTRODUCTION

Actually, the language models (LM) constitute one of the key components in several applications that produce natural language text, such as large vocabulary speech recognition [13],[18], spoken language understanding, entity disambiguation [4], statistical machine translation [20],[12], information retrieval [23],[7], language text identification [1], handwriting recognition [9] and so on [17].

IPAC 15, November 23-25, 2015, Batna, Algeria

http://dx.doi.org/10.1145/2816839.2816878 ...$15.00.

The LM is a statistical approach used for assigning probabilities to sequences of words that form valid sentences in the language. One solution is to make the assumption based on a Markovian principal, that the probability of a word in a sentence is conditioned on only the previous $n-1$ words, namely an n-gram language model [16]. It is good to develop these LM from very large text corpus [19] but some problems appear : first, the training needs expensive computational resources and secondly, in any particular application the amount of specific domain data is frequently limited. Therefore, several approaches were proposed to limit the storage size of large LMs [11].

The goal of automatic speech recognition system (ASR) is to convert a human speech signal into a text form [15] so the machine tries to match sounds with word sequences. The LM provides context to distinguish between words and sentences that sound are similar but with different means. These ambiguities are easier to resolve when evidence from the LM is incorporated with the pronunciation model and the acoustic model.

Also, in statistical machine translation (SMT), we are faced with a sequence of words e in the source language and we are looking for its best translation f into the target language. Although we use an LM to evaluate the probability of the produced sequences of words in order to choose the best translation of a source words with a good order, given the context.

Data sparsity is a major problem in building MLs: most possible word sequences or sentences will not be observed in training (so with zero probability). However they will never be considered in transcription or translation. The LMs are often smoothed to avoid these situations. Assigning all strings a nonzero probability helps preventing errors in speech recognition, machines translation, etc. Many techniques were proposed for smoothing n-gram models in order to better estimate probabilities, where there are insufficient data to estimate probability accurately [3].

Given the fact that an important relation linked respectively the speech recognition, processing language and machine learning, the goal of this paper is to summarise the smoothing algorithms for modelling language and discusses their implementation and performance. We compare the behavior of the most popular smoothing algorithms for training a LM dedicated at later stage for ASR using a transcription text corpus of a French radiobroadcast.

This paper is organised as follows: In section 2, we in-

troduce the language modelling approach. In section 3, we describe briefly some of the smoothing methods we evaluated. The major experiments and results are presented and discussed in section 4. Section 5 presents conclusion and outline directions for future work.

2. LANGUAGE MODEL

The statistical language modelling is a technique of natural language processing (NLP). We distinguish two great families of language models; models based on grammar and probabilistic models. The modeling with grammar generally generates a correct/incorrect type of answer without accuracy. However the probabilistic models furnish probabilistic quantified answers.

This last technique is widely used in NLP research. It attempts to measure how likely any given piece of text is probable, in any given language, by building a probabilistic model of the considered language and assigning a probability value to the text using this model.

The LMs are usually estimated from collections of text in the chosen language. The LM tries to capture the structure of a language through word frequencies. It describes the probability of a valid sentence using the sequence of its words. For a sentence s composed of words $w_1...w_l$, we can express its probability p(s) as:

$$\begin{aligned} p(s) &= p(w_1)p(w_2|w_1)p(w_3|w_1w_2)..p(w_i|w_1..w_{l-1}) \\ &= p(w_1)\prod_{i=1}^{l} p(w_i|h_i) \end{aligned}$$

where h_i is the history of the word w_i.

2.1 N-gram model language

The most popular form of probabilistic LM is the n-gram, which is notable for its simplicity, computational efficiency and surprising power [6]. In n-gram models, we make the approximation that the probability of a word depends only on the $n-1$ immediately preceding word, giving us:

$$p(s) \approx p(w_1)\prod_{i=n}^{l} p(w_i|w_{i-n+1}^{i-1}) \quad (1)$$

To estimate $p(w_i|w_{i-n+1}^{i-1})$, we take the number of times the n-gram $_{i-n+1}^{i-1}$ occurs in some text and normalize:

$$p(s) = \frac{c(w_{i-n+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i-1})} \quad (2)$$

where w_i^j denotes the words $w_i...w_j$ and $c(w_i^j)$ is the n-gram w_i^j occurrences in the training set.

2.2 Perplexity

A measure of the language model complexity is the mathematical quantity known as language perplexity, which is the geometric meaning of the word branching factor (or the average number of words that follow any given word of the language). It can be derivate from the probability that the model assigns to the sentence s probabilities $p(s) = p(w_i|w_{i-n+1}^{i-1})$, using the equation 1. Then for a text T composed of n sentences, the probability of all sentences is given by:

$$p(T) = \prod_{i=1}^{l_T} p(t_i) \quad (3)$$

We can derive a compression algorithm that encodes the text T using $(-log_b p(T))$, where $b = 2$ is the base with respect to which the information is measured (e.g., bits). The perplexity can be defined as:

$$PP(T) = 2^{-\frac{1}{W_T} log_2 p(T)} \quad (4)$$

where W_T is the length of the text T measured in words. For example, for 5000 words Wall Street Journal Task, the language perplexity (using a bigram language model) is 130 [16]. On comparing perplexity of two LMs, the lesser is for a better model.

3. SMOOTHING METHODS

Typically, however, the n-gram model probabilities are calculated from the frequency counts using the equation 2, the ML performance can decrease when confronted with any n-gram model that has not been explicitly seen before. Instead, some form of smoothing is necessary for more sophisticated models.

The term *smoothing* refers to the adjustment of the maximum likelihood estimator of word probabilities in order to improve the accuracy of the language model as a whole. Also, this term is used to describe the techniques which tend to make distributions more uniform, by adjusting low probabilities such as zero probabilities upward, and high probabilities downward [3].

3.1 Interpolation or backoff

Several smoothing methods have been proposed in recent years such as: Additive smoothing (1948), Good-Turing Estimate (1953), Jelinkel-Mercer smoothing (1980, 1992), Katz smoothing (1987), Witten-bell smoothing (1990, 1991), Absolute discount (1990, 1991), Ristad's naturel discount (1995), Kneser-Ney smoothing (1995, 1997), Modified Kneser-Ney smoothing [3].

In general, these methods can be classified into two categories. The first category, algorithms with backoff distribution: the idea is if an n-gram has a non-zero count then we use its distribution. Otherwise, we back-off to the lower-order distribution (generally, we use trigram otherwise we can use bigram or unigram). Katz smoothing is the canonical example of backoff smoothing. The second category is algorithms based on interpolation, where unigram, bigram and trigram are mixed. Several smoothing algorithms, such as Jelinek-Mercer smoothing, are expressed as the linear interpolation of higher-order and lower-order n-gram models. Some of these algorithms are found in the two version implementations [3],[2],[21].

3.2 Some smoothing methods

The *Additive smoothing* is one of the simplest methods. To avoid zero probabilities, we pretend that each n-gram occurs δ times more than it actually does, where $0 < \delta \leq 1$, and instead to use the equation 2, we will use :

$$p_{add}(w_i|w_{i-n+1}^{i-1}) = \frac{\delta + c(w_{i-n+1}^{i})}{\delta|V| + \sum_{w_i} c(w_{i-n+1}^{i})} \quad (5)$$

where V is the vocabulary, or set of all words considered.

However the *Ney's Absolute discount backoff* model uses discount as the constant of subtracts, but with standard interpolated version, the n-gram probability will be interpolated with lower-order estimates. The result of the interpolation is encoded as a standard backoff model and can be evaluated as such: The interpolation happens at estimation time. That is, instead of the equation 2 we have:

$$p_{abs}(w_i|w_{i-n+1}^{i-1}) = \frac{max\{c(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})}) + (1 - \lambda_{w_{i-n+1}^{i-1}})p_{abs}(w_i|w_{i-n+2}^{i-1}) \quad (6)$$

where D is estimated by: $D = \frac{n_1}{n_1+2n_2}$, and n_1, n_2 are the total number of n-grams with exactly one and two counts, respectively, in the training data.

The *modified Kneser-Ney smoothing* which is an extension of absolute discounting, used three different parameters, D_1, D_2, D_{3+}, that are applied to n-grams with one, two and three or more counts, respectively. The equation probability will be:

$$p_{kn}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i}))}{\sum_{w_i} c(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1})p_{kn}(w_i|w_{i-n+2}^{i-1}) \quad (7)$$

where

$$D(c) = \begin{cases} 0 & c = 0 \\ D_1 & c = 1 \\ D_2 & c = 2 \\ D_{3+} & c \geqslant 3 \end{cases}$$

and $\gamma(w_{i-n+1}^{i-1})$ is defined by a combination of D_1, D_2 and D_{3+}.

A good review of others methods with details can be found in [3],[2],[23].

4. EXPERIMENTS

For our experiments, we take a data from ETAPE corpus [8] we split it manually in three subsets for the train, held-out and test tasks. A pretreatment process was applied on all data:

1. eliminating empty line from the whole text,
2. tokenizing the text so that each sentence is on a single line (with <s> and </s> symbols,
3. converting uppercase characters at the beginning of the line to lower case,
4. eliminating all non-word, non-sentence and punctuation items.

With this tokenization, the size of our data sets in terms of sentences and words are shown in table 1. The word counts include added end-of-sentence tokens. All our LMs are built with the SRILM toolkit [21]. This package is developed, maintained and distributed under an open source community license by SRI International's Speech Technology and Research Laboratory in California. This toolkit is not available in pre-compile form and must be compiled and installed manually [14]. Most LM research and development at SRI and in different universities in world is based on SRILM.

Table 1: Corpus size statistics

Corpus	Sentence count	Word count
Etape-train	18 083	285 735
Etape-held-out	1 004	12 790
Etape-test	1 004	18 427

In order to achieve a good comparison, we have focused on finding the best n for the n-gram according to our context. Multiple experimentations are done with n-grams of various order, different n for $\{1, 2, 3, 4, 5\}$. Using the held-out data, we show how these factors affect the relative performance of smoothing technique. We determined that trigram will be used. The results of this first step are shown in the figure 1 and the table 2. where the ppl1 is the average perplexity

Table 2: Perplexity for different values of n (for n-gram)

n (n-gram)	ppl	ppl1
1	493.5	910.6
2	189.8	318.7
3	**174.9**	244.5
4	174.9	244.7
5	175.1	244.9

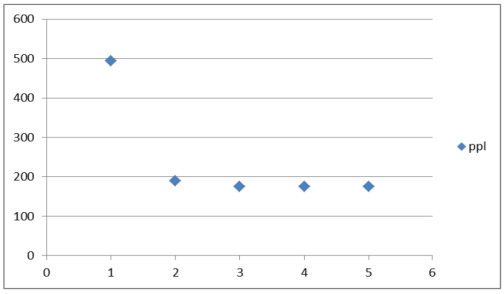

Figure 1: Choice of n-gram value.

per word excluding the beginning <s> and ending </s> tokens.

Secondly, we calculate trigram count on train data. We use this information to generate LM. In each time we change the smoothing method. For some methods, such as Additive smoothing and Absolute discount (with its two versions: backoff and interpolation) that are related to some parameters, we do cross-validation using held-out data in order to perform the results (see the figure 2 and the table 3).

For *Additive smoothing*, Lidstone and Jeffreys [3] advocate taking $\delta = 1$ but with our tests we found that $\delta = 0.003$ gives better results.

The *Ney's Absolute discount backoff* model uses discount as the constant of subtracts, the value of the discount parameter λ affects the performance of the LM, for $\lambda = 0.71$ the performance of LM is better. Some results of cross-validation are shown in the table 4 and the figure 3.

Similar optimisation is done with the second version of *Absolute discount*, a good value for the discount parameter $\lambda = 0.82$ gives perplexity of 181.5 for LM. Some results can be shown in the table 5 and the figure 4.

Table 3: Parameter optimization for Additive smoothing on held-out data

δ	ppl	ppl1
1	1 085.0	2 163.9
0.001	212.2	360.9
0.003	**198.9**	335.9
0.005	200.4	338.2
0.007	200.3	345.5
0.009	208.9	345.2
0.01	235.8	404.4
0.02	211.4	358.7
0.1	376.7	676.1

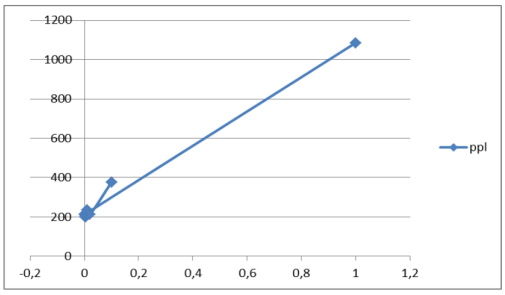

Figure 2: Optimization behavior of parameter Additive smoothing (Table 3).

Table 4: Parameter optimisation for Absolute discount algorithm (with interpolation) on held-out data

λ	ppl	ppl1
1	195.4	328.9
0.9	185.8	311.2
0.8	197.2	299.2
0.71	**177.9**	296.8
0.72	177.9	296.9
0.75	178.1	297.3
0.7	177.9	296.8
0.6	179.5	299.6
0.5	183.4	306.9

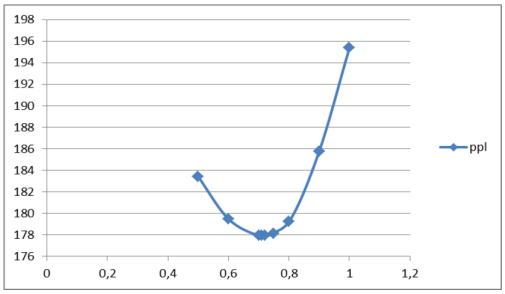

Figure 3: Optimization behavior of parameter Absolute discount (Table 4).

Finally, we generate trigram models with optimised parameter values, and with several smoothing methods. The standard method generated in SRILM; if no discounting method is specified, Good-Turing combined by Kartz method is used with optimised parameters for 3-grams ($gt1-min = 1, gt1-max = 7; gt2-min = 1, gt2-max = 7, gt3-min = 2, gt3-max = 7$) which corresponds to the first line in the table 6. If we specify a Good-Turing with other parameters the LM performance will decrease, which corresponds to the second line of the table 6.

We also found that some methods such as *additive smoothing* or *Ristad's naturel discount* have very poor performance, they respectively give perplexities of 195.9 and 183.5 . The most important result shown in the table 6, is that the *Modified Kneser-Ney* using *interpolation* achieves better result, which grants with [3],[22],[10] and [5].

Table 5: Parameter optimisation for Absolute discount algorithm (with interpolation) on held-out data

λ	ppl	ppl1
0.1	276.8	482.5
0.2	234.8	402.8
0.3	214.1	363.8
0.4	201.3	340.0
0.5	192.8	324.1
0.6	186.9	313.3
0.7	183.2	306.5
0.8	181.5	303.5
0.82	**181.5**	303.4
0.9	182.8	305.9

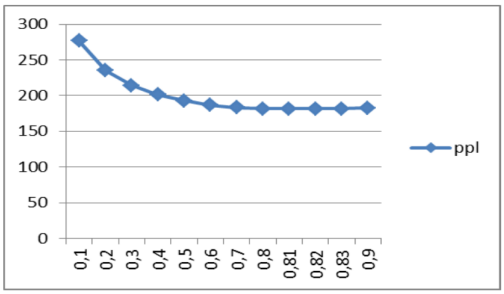

Figure 4: Parameter optimization behavior of interpolated Absolute discount (table 5).

Table 6: Perplexity of different smoothing algorithms on test data

Method	Paramter	ppl	ppl1
Standard (Good-Turing)		174.9	244.6
Good-turing optimized		177.2	248.0
Add-Smooth	$\delta = 0.003$	195.9	275.9
Absolute-discount-backoff	$\lambda = 0.71$	175.9	246.2
Absolute-discount-Interpolation	$\lambda = 0.82$	179.9	252.0
Witten-bell-backoff		178.1	249.3
Witten-bell-interpolation		179.9	252.2
Ristad's-naturel-discount.		183.5	257.4
Original K-Ney-backoff		169.7	236.8
Original K-Ney-interpolation		167.5	233.6
Modified K-Ney-backoff		173.5	242.4
Modified K-Ney-interpolation		**165.4**	230.5

5. CONCLUSIONS

Smoothing is a fundamental technique for statistical modelling. It can be used in a lot of important applications. The performance of smoothing methods differs and it is not easy to precise the best one when experimentation conditions change. However, in terms of normalized performance, interpolated models are significantly superior to backoff models. From our experiments, we can also see that optimising some free parameters on hold out data can really improve the performance of an algorithm.

In this work, we have measured the performance of algorithms through the perplexity to check the generalisation ability of our LM. For future work, we will use LM; with the appropriate smoothing algorithm, as a module for specific application ; speech recognition, and performed experiments measured by the word error rate (WER).

6. ACKNOWLEDGMENTS

We would like to thank Denis Jouvet and David Langlois from Loria Inria, Nancy for their appreciated help. Thanks to Stolcke Andreas for SRILM toolkit and for his availability.

7. REFERENCES

[1] R. D. Brown. Finding and identifying text in 900+ languages. *Digital Investigation*, 9:S34–S43, 2012.

[2] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318. Association for Computational Linguistics, 1996.

[3] S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393, 1999.

[4] C. Dalvi, Bhavana aand Xiong and J. Callan. A language modeling approach to entity recognition and disambiguation for search queries. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 45–54. ACM, 2014.

[5] J. Dumoulin. Smoothing of ngram language models of human chats. In *Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS), 2012 Joint 6th International Conference on*, pages 1–4. IEEE, 2012.

[6] J. T. Goodman. A bit of progress in language modeling. *Computer Speech & Language*, 15(4):403–434, 2001.

[7] P. Goyal, L. Behera, and T. M. McGinnity. A novel neighborhood based document smoothing model for information retrieval. *Information retrieval*, 16(3):391–425, 2013.

[8] G. Gravier and G. Adda. Evaluations en traitement automatique de la parole (etape). *Evaluation Plan, Etape*, 2011.

[9] M. Hamdani, P. Doetsch, M. Kozielski, A. E.-D. Mousa, and H. Ney. The rwth large vocabulary arabic handwriting recognition system. In *Document Analysis Systems (DAS), 2014 11th IAPR International Workshop on*, pages 111–115. IEEE, 2014.

[10] A. Hasan, S. Islam, and M. Rahman. A comparative study of witten bell and kneser-ney smoothing methods for statistical machine translation. *Journal of Information Technology*, 1:1–6, 2012.

[11] K. Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197. Association for Computational Linguistics, 2011.

[12] P. Koehn and B. Haddow. Towards effective use of training data in statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 317–321. Association for Computational Linguistics, 2012.

[13] L. Lamel, J.-L. Gauvain, V. B. Le, I. Oparin, and S. Meng. Improved models for mandarin speech-to-text transcription. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 4660–4663. IEEE, 2011.

[14] N. Madnani. Querying and serving n-gram language models with python. *The Python Papers*, 4(2):2009, 2009.

[15] L. R. Rabiner. Speech recognition in machines. *The MIT Encyclopedia of the Cognitive Sciences, R. Wilson and F. K. Keil,*, 1999.

[16] L. R. Rabiner and B. Juang. Statistical methods for the recognition and understanding of speech. *Encyclopedia of language and linguistics*, 2004.

[17] R. Rosenfield. Two decades of statistical language modeling: Where do we go from here? 2000.

[18] A. Rousseau, P. Deléglise, and Y. Estève. Enhancing the ted-lium corpus with selected data for language modeling and more ted talks. In *Proc. of LREC*, pages 3935–3939, 2014.

[19] H. Schwenk, A. Rousseau, and M. Attik. Large, pruned or continuous space language models on a gpu for statistical machine translation. In *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics, 2012.

[20] R. Sennrich. Perplexity minimization for translation model domain adaptation in statistical machine translation. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 539–549. Association for Computational Linguistics, 2012.

[21] A. Stolcke et al. Srilm-an extensible language modeling toolkit. In *INTERSPEECH*, 2002.

[22] M. Sundermeyer, R. Schlüter, and H. Ney. On the estimation of discount parameters for language model smoothing. In *INTERSPEECH*, pages 1433–1436, 2011.

[23] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.