An empirical study of statistical language models: n-gram language models vs. neural network language models

Freha Mezzoudj*

Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf,
BP 1505 El Mnaouer, 31000, Oran, Algeria
Email: freha.mezzoudj@univ-usto.dz
and
Université Hassiba Benbouali Chlef,
Ouled Fares, 02000, Chlef, Algeria
*Corresponding author

Abdelkader Benyettou

Signal Image et Parole (SIMPA) Laboratory,
Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf,
BP 1505 El Mnaouer, 31000,
Oran, Algeria
Email: abdelkader.benyettou@univ-usto.dz

Abstract: Statistical language models are an important module in many areas of successful applications such as speech recognition and machine translation. And n-gram models are basically the state-of-the-art. However, due to sparsity of data, the modelled language cannot be completely represented in the n-gram language model. In fact, if new words appear in the recognition or translation steps, we need to provide a smoothing method to distribute the model probabilities over the unknown values. Recently, neural networks were used to model language based on the idea of projecting words onto a continuous space and performing the probability estimation in this space. In this experimental work, we compare the behaviour of the most popular smoothing methods with statistical n-gram language models and neural network language models in different situations and with different parameters. The language models are trained on two corpora of French and English texts. Good empirical results are obtained by the recurrent neural network language models.

Keywords: language models; n-grams; Kneser-Ney smoothing; modified Kneser-Ney smoothing; Good-Turing smoothing; interpolation; back-off; feed-forward neural networks; continuous space language models; CSLM; recurrent neural networks; RNN; speech recognition; machine translation.

Reference to this paper should be made as follows: Mezzoudj, F. and Benyettou, A. (2018) 'An empirical study of statistical language models: n-gram language models vs. neural network language models', *Int. J. Innovative Computing and Applications*, Vol. 9, No. 4, pp.189–202.

Biographical notes: Freha Mezzoudj received her Magister's degree in Computer Science at the University of Science and Technology of Oran Mohamed Boudiaf, USTO-MB, Algeria in 2011. She is a PhD student at the same university. She is an Assistant Professor since 2012 at the Department of Computer Science, Hassiba Benbouali University of Chlef, Algeria. Her research interests include issues related to artificial intelligence, automatic speech recognition, statistical language modelling, deep learning, formal modelling and bioinformatics. She has published research papers at many national and international conference proceedings.

Abdelkader Benyettou is a Professor at the Department of Computer Science, University of Science and Technology of Oran Mohammed Boudiaf, USTO-MB, Algeria. His research interests are related to pattern recognition, artificial intelligence, automatic speech recognition, neural networks, hidden Markov models, software engineering, computing in mathematics and so on. He is also a Director of Signal Image PArole (SIMPA) Laboratory at the USTO-MB, Algeria. He has published research papers at national and international journals, conference proceedings as well as chapters of books.

Copyright © 2018 Inderscience Enterprises Ltd.

This paper is a revised and expanded version of a paper entitled 'On an empirical study of smoothing techniques for a tiny language model' presented at Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication, Batna, Algeria, 23–25 November 2015.

1 Introduction

Statistical language models are widely used in NLP systems like automatic speech recognition (ASR), statistical machine translation (SMT), spoken language understanding, speech synthesis, document classification and many others ([Rosenfeld, 2000).

Language model adopt a statistical approach for assigning probabilities to sequences of words that form valid sentences in a language. An assumption based on a Markovian principle is made, the probability of a word in a sentence is conditioned on only by the previous $n - 1$ words (Rabiner and Juang, 2005).

We distinguish three great families of language models: models based on grammar, statistical n-grams models and neural network models. Models based on grammar generally generate a correct/incorrect type of answer without accuracy. However, the statistical or probabilistic models based on n-grams furnish probabilistically quantified answers. The third kind of language modelling is based on neural networks.

In ASR, a language model is the core component that incorporates syntactic and semantic constraints of a given natural language. The goal of an ASR system is to convert a human speech signal into a text form with the machine trying to match sounds with word sequences. Indeed, the language model provides a context to distinguish between words and sentences which sounds are similar and convey with different meanings. These ambiguities are easier to resolve when evidence from the language model is incorporated with the pronunciation model and the acoustic model.

In 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.

The language model is used to evaluate the probability of the produced sequences of words in order to choose the best translation of source words with a good order in a given context.

In the state-of-the-art of modelling language, n-gram models are considered as the conventional language models due to their simplicity and good performance (Goodman, 2001). However, the problem with n-gram language models is that many word sequences do not appear in the training corpus even if it is large. These words counts and probabilities are equal to zero, so they will never be considered in transcription or translation. An n-gram model lacks an explicit estimation for the probability of these unseen n-grams. One solution is given with smoothing techniques which refer to the adjustment of the word probabilities and tend to make distributions more uniform by adjusting low probabilities upward, and high probabilities downward (Chen and Goodman, 1999).

Classic n-gram language models cope with rare or unseen sequences using smoothing methods in order to prevent errors in language modelling, speech recognition or machine translation, etc. Many techniques were proposed for this task (Chen and Goodman, 1999). However, the neural network language models (NNLM) which are based on learning the features representation (Bengio et al., 2003; Schwenk, 2007; Mikolov et al., 2010) have no notion of discrete counts. They embed words in a continuous space in which probability inference is performed using the hidden layer neurons.

Given the fact that an important relation links the language processing and machine learning, the goal of this paper is to summarise and compare the behaviour and the performance of the most popular smoothing algorithms for n-gram models and the most used NNLM.

In order to evaluate the performance of language models, we can combine them with acoustic models (or translation models) and embed them in a recognition system and measure how much the application improves. Language models can also be evaluated efficiently without access to a speech recogniser or a translator, specially when language models are trained on in-domain data (Chen et al., 1998). In this paper, we focus on modelling language, so we compare between the different language models which is an easy way to do and computentially less expensive.

The language models presented in this paper were trained on two English and French recent data relating to real applications with different sizes:

1 text transcriptions of French radiobroadcast used on ETAPE evaluation campaign (Gravier et al., 2012)

2 on an English text used during the NAACL 2012 workshop (Callison-Burch et al., 2012) on SMT.

This paper is organised as follows: In Section 2, we introduce the n-gram language modelling approach and we describe some of the most popular smoothing n-gram methods. In Section 3, we explain the architecture of the feed-forward and the recurrent neural network (RNN) for language modelling. In Section 4, we introduce some comparative related works. The major experiments and results are presented and discussed in Section 5. Section 6 concludes the paper and provides some future research directions.

2 N-gram language models

Natural languages are known to have a layered structure, a hidden and deeper structure that represents the meaning and core semantic relations within a sentence, and a surface form found in normal written texts or a spoken language, as formulated in linguistic theories (Chomsky, 2014) such as generative grammar of Chomsky.

The task of statistical language modelling is to create statistical models that are able to capture the regularities of a natural language and measure how likely any given piece of text is probable in any given language. So, the language model tries to capture the structure of a language through word frequencies. It assigns a probability of a valid sentence using the sequence of its words. The probabilities are estimated using maximum likelihood estimation (MLE) from the training corpus and assigned to different words.

The most popular form of a statistical or probabilistic language model is the n-gram model (Jelinek et al., 1991) which is notable for its simplicity, computational efficiency and surprising power (Goodman, 2001).

For a sentence s composed of words $w_1 \ldots w_l$, we can express its probability $p(s)$ by following equations:

$$\begin{aligned} p(s) &= p(w_1)\,p(w_2 \mid w_1)\,p(w_3 \mid w_1 w_2)..p(w_l \mid w_1..w_{l-1}) \\ &= p(w_1)\prod_{i=2}^{l} p(wi \mid h_i) \end{aligned}$$

where h_i is the history of the word w_i.

The intuition of the n-gram model is that instead of computing the probability of a word given its entire history, we can approximate the history by just the last few words in a training text corpus. So, we consider that the probability of a word depends only on the $n - 1$ immediately preceding words, giving by equation (1):

$$p(s) \approx p(w_1)\prod_{i=n}^{l} p\left(w_i \mid w_{i-n+1}^{i-1}\right) \tag{1}$$

where w_i^j denotes the words $w_i \ldots w_j$.

To estimate $p(w_i \mid w_{i-n+1}^{i-1})$ using MLE, we time the n-gram w_{i-n+1}^{i-1} occurrences in a text and normalise them as in equation (2):

$$p\left(w_{i-n+1}^{i-1}\right) = \frac{ct\left(w_{i-n+1}^{i}\right)}{\sum_{w_i} ct\left(w_{i-n+1}^{i-1}\right)} \tag{2}$$

where $ct(w_i^j)$ is the count of n-gram w_i^j in the training corpus.

We introduce some useful concepts for the rest of the paper such as perplexity, vocabulary and so on.

2.1 Perplexity

In order to evaluate the performance of a language model, we embed it in an application and measure how much the application improves. Thus, for speech recognition, we can compare the performance of many language models by running the speech recogniser many times with each language model, and seeing which one gives the most accurate transcription. This kind of evaluation can be expensive.

Instead, we can use a metric to quickly evaluate potential improvements in a language model independently of any application using a test set. This measure of language models complexity is the mathematical quantity known as *perplexity*, which is the geometric average of the probabilities of the words that follow any given word in the language (Jelinek, 1997). It can be derivated from the probability that the model assigns to the sentence s using equation (1). Then for a test text T composed of n sentences, the probability of all sentences is given by equation (3):

$$p(T) = \prod_{i=1}^{W_T} p(w_i) \tag{3}$$

We can derive a compression algorithm that encodes the text T using $(-\log_b p(T))$; called the cross-entropy, where $b = 2$ is the base with respect to which the information is measured (e.g., bits). The perplexity can be defined by equation (4):

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

where W_T is the length of the text T measured in words.

For example, for 5,000 words *Wall Street Journal Task*, the language model gives the perplexity of 130, it means that there are 130 possible choices on average for every word prediction position. Therefore, when comparing two language models, the one which gives the lower perplexity is regarded as the appropriate model.

2.2 Vocabulary

There are many machine learning algorithms to automatically induce the vocabulary from a corpus of a text. As an example, we can select as the vocabulary the N most frequent words. The number of free parameters of the language model increases rapidly as vocabulary size increases. In general, for large vocabulary systems a vocabulary usually does not entirely cover the language vocabulary. Thus the language model vocabulary should be opened, i.e., it should account the potential appearance of unknown words – UNK (or out-of-vocabulary words – OOV) in a sentence and should correctly estimate their probabilities.

Traditionally the open vocabulary problem in language model training is solved by replacing all unknown words by the special token UNK. Then the model is trained in a conventional manner as if UNK were a normal word. The use of a closed vocabulary language model which has no provisions for unknown words is not appreciable.

2.3 Smoothing methods

Typically, the n-gram model probabilities are calculated from the frequency counts using equation (2). The language model performance can decrease when confronted with any n-gram model that has not been explicitly seen before. For example, having seen the sentence 'The cat is walking in the bedroom' in the training corpus, does not help us to generalise the sentence 'A cat is walking in the garden' in the step test if the word 'garden' does not appear in the training corpus. Instead, we need to estimate the probability of events unseen in the training corpus. If an n-gram does not appear in the training corpus then its probability is equal to zero. But no word sequence should have zero probability, this cannot be right. Furthermore, MLE produces poor estimates when the counts are non-zero but still small. The solution is given with smoothing techniques.

The 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. It tends to make distributions more uniform by adjusting low probabilities such as zero probabilities upward, and high probabilities downward (Chen and Goodman, 1999).

Smoothing techniques are a very important issue in language modelling. They contribute to diminish the effect of data sparsity problem and they significantly improve the modelling and recognition performance. Several smoothing methods have been proposed such as: additive smoothing, Good-Turing estimation (Good, 1953), Jelinek-Mercer smoothing (Jelinek, 1980), Katz smoothing (Katz, 1987), Witten-Bell smoothing (Witten and Bell, 1991), absolute discount (Ney and Essen, 1991), Kneser-Ney smoothing (Kneser and Ney, 1995), modified Kneser-Ney smoothing (Chen and Goodman, 1999), stupid back-off (Brants et al., 2007), etc.

In general, these methods can be classified into two categories. The first category contains algorithms with back-off distribution, they are based on the idea that: if an n-gram has a non-zero count then we use its distribution, otherwise, we must back-off to the lower-order distribution in order to avoid the zero probability. Generally, we use trigrams if not we can use bigrams or unigram. The Katz smoothing is the canonical example of back-off smoothing. The second category is containing the algorithms based on interpolation of higher-order and lower-order n-grams. Generally, we mix unigram, bigrams and trigrams counts for probability estimation. Jelinek-Mercer smoothing is a canonical example of the interpolation smoothing.

We discuss here the two most important smoothing methods: Kneser-Ney smoothing and the modified Kneser-Ney smoothing.

2.3.1 Kneser-Ney smoothing

The original or unmodified Kneser-Ney smoothing (Kneser and Ney, 1995) is an extension of the Ney's absolute discounting where the lower-order distribution combines with a higher-order distribution. However, instead of using equation (2), we will use equation (5):

$$p_{uMKN}\left(w_i \mid w_{i-n+1}^{i-1}\right) = \begin{cases} \dfrac{\max\left(ct\left(w_{i-n+1}^{i}\right) - D, 0\right.}{\sum_{w_i} ct\left(w_{i-n+1}^{i}\right)} & ct\left(w_{i-n+1}^{i}\right) > 0 \\ \gamma\left(w_{i-n+1}^{i-1}\right) p_{uMKN}\left(w_i \mid w_{i-n+2}^{i-1}\right) & ct\left(w_{i-n+1}^{i}\right) = 0 \end{cases} \quad (5)$$

where $\gamma(w_i \mid w_{i-n+1}^{i-1})$ is chosen to make the distribution sum to 1. An interpolated version exists too, using equation (6):

$$p_{uMKN}\left(w_i \mid w_{i-n+1}^{i-1}\right) = \frac{\max\left\{ct\left(w_{i-n+1}^{i}\right) - D, 0\right\}}{\sum_{w_I} ct\left(w_{i-n+1}^{i}\right)} + \gamma\left(w_{i-n+1}^{i-1}\right) p_{uMKN}\left(w_i \mid w_{i-n+2}^{i-1}\right) \quad (6)$$

2.3.2 Modified Kneser-Ney smoothing

The modified Kneser-Ney smoothing (Chen and Goodman, 1999) uses 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 probability is estimated by equation (7):

$$p_{MKN}\left(w_i \mid w_{i-n+1}^{i-1}\right) = \frac{ct\left(w_{i-n+1}^{i}\right) - D\left(ct\left(w_{i-n+1}^{i}\right)\right)}{\sum_{w_i} ct\left(w_{i-n+1}^{i}\right)} + \gamma\left(w_{i-n+1}^{i-1}\right) p_{MKN}\left(w_i \mid w_{i-n+2}^{i-1}\right) \quad (7)$$

where:

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

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

3 Neural network language models

The simple back-off and/or interpolated n-gram models are still considered to be the state-of-the-art, but an alternative approach based on the use of neural networks was recently introduced. This approach also called deep learning has emerged as a new area of machine learning research. The deep learning is based on the use of:

1 models with multiple layers or stages of nonlinear information processing (usually neural networks)

2 methods for supervised or unsupervised learning of feature representation at successive layers.

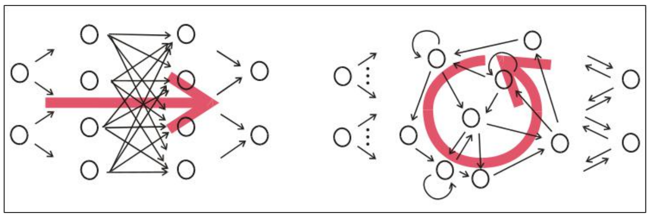
Neural networks which are a basic machine learning technique can be seen as nonlinear projections of the input features. They are considered as universal approximators to any arbitrary continuous function (Hornik et al., 1989). They are capable of processing and learning from complex input data and solving different kinds of complicated tasks.

In smoothed classic n-gram language models, the words probabilities are estimated with MLE using the words historic with equation (2) or its variants: equations (5), (6) and equation (7), etc. However, NNLM deal with the problem of data sparsity by learning representations for words in a continuous space and the probabilities are inferred with hidden layer (s). This idea allows us to obtain the new word probability regardless of whether the n-grams were seen in training or not. Thus n-gram distributions are expressed as a smooth function of the word representation and can take into account underlying similarities between words (Bengio et al., 2003).

For example assume, we observed the word 'dog' many times during training, but we observed the word 'cat' only a few times, or not at all. If each of the words is associated with its own dimension, occurrences of the word 'dog' will not tell us anything about the occurrences of the word 'cat'. However, in the continuous representation, the learned vector for the word 'dog' may be similar to the learned vector from the word 'cat'. This situation allows the model to share the statistical strength between the two events. Also, having seen the sentence 'A dog was running in a yard' in the training corpus, it should help us to generalise the sentence 'The cat is walking in the room' almost similarly. We can explain this situation by the fact that the words: 'dog' and 'cat' (resp. 'the' and 'a', 'room' and 'yard', 'is' and 'was', etc.) have similar semantic and grammatical roles, so they are automatically clustered together in the continuous space of their representation using the neural network.

Many variants of the NNLM were proposed during the last years based on the feed-forward neural network (Bengio et al., 2003; Schwenk, 2007), the recurrent architecture (Mikolov et al., 2010; 2011b) (see Figure 1), etc.

Figure 1 Typical structure of a feed-forward network (left) and a recurrent network (right) (see online version for colours)



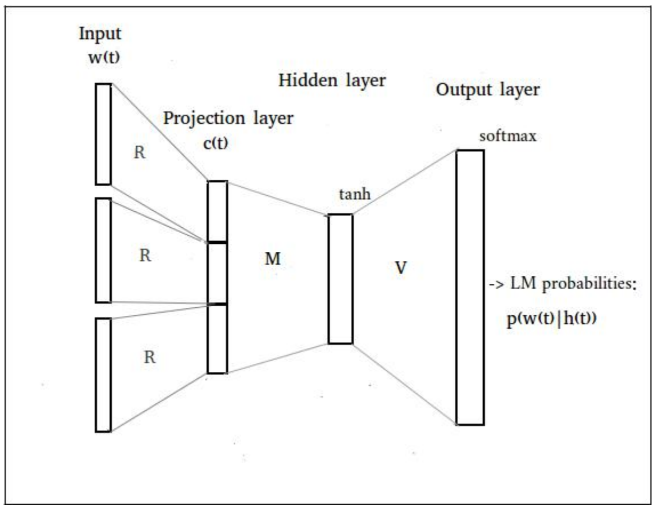
Source: Jaeger (2002)

Feed-forward networks (FNN) include networks with fully connected layers, such as the multi-layer perceptron. In FNN, activation is piped through the network from input units to output units from left to right. However, RNN are designed to model sequences. It has (at least one) cyclic path of synaptic connections.

3.1 Feed-forward neural network LM

The standard feed-forward NNLM was proposed by Bengio et al. (2001, 2003). This approach was used and called continuous space language model (CSLM) in Schwenk (2007, 2013). The basic idea is to project the word indices onto a continuous space and to use a probability estimator operating on this space. A neural network is used to simultaneously learn the projection of the words onto the continuous space and to estimate the n-gram probabilities. This is still an n-gram approach, but the language model probabilities are interpolated for any possible context of length $n - 1$ instead of backing-off to shorter contexts. In general, the network has three layers as it is illustrated in Figure 2.

Figure 2 Language model-based feed-forward neural network (NNLM or CSLM) used by Bengio et al. (2001) and Schwenk (2007)



First , the input of the NNLM is formed by using a fixed length history of $n - 1$ words, where each of its previous words are encoded using 1-of-N coding vector and N is the size of the vocabulary. Thus, every word from the vocabulary is associated with a vector with length N, where only one value corresponding to the index of a given word in the vocabulary is one and all other values are 0. Then, these context vectors are concatenated to create the input of the hidden layer with nonlinear activation function. This representation of words is projected linearly to a lower dimensional space, using a shared matrix R, called a projection matrix (in a projection layer). After this layer, a hidden layer with nonlinear activation function is used. On the output layer, each neuron corresponds to the probability of a word calculated by the softmax activation function. The size of the latter layer is therefore equal to the size of the vocabulary.

Formally, we define the CSLM using the matrix-vector multiplication of its layers. The inputs of the neural network are the indices of the $n - 1$ previous words in the vocabulary called context (or history) $h_i = w_{i-n+1} \ldots w_{i-2} w_{i-1}$, represented by 1-of-$N$ coding vector.

The output of this layer is a vector c of $(n - 1)R$ real numbers obtained by concatenating the representations of the context word. The projection matrix R is shared among words at different positions in the history vector and it is automatically learned.

The hidden layer introduces a nonlinear activation function (f is usually hyperbolic tangent or a logistic:

$$d = f(Mc + b) \tag{8}$$

where c is the input vector, M is weight matrix between the input and the hidden layers and b is bias vector for the hidden layer. The vector d is a more abstract representation of the context than c.

The output layer consists of N nodes; each node is associated with one word in the vocabulary. Its activation values are given by equation (9):

$$o = Vd + k \tag{9}$$

where V is weight matrix between the hidden layer and the output layer, k is a bias vector for the output layer.

The softmax activation function is used to ensure that the outputs form a valid probability distribution. The outputs are the posterior probabilities of all words w_i of the vocabulary given the history vectors and normalised by softmax as in equation (10):

$$p_i = \frac{\exp(o_i)}{\sum_{i=1}^{N} \exp(o_i)} \tag{10}$$

Usually, the training is performed with the standard back-propagation algorithm (Bottou, 2012).

The model parameters are optimised on the development (or validation) corpus minimising the following error function given by equation (11):

$$E = Y \log(p_i) + \beta(M^2 + V^2) \tag{11}$$

where Y denotes the desired output probability. The first part of equation (11) is the cross-entropy between the output and the target probability distributions, and the second part is a regularisation term that aims to prevent the neural network from over-fitting the training data (weight decay). The regularisation parameter β has to be determined experimentally.

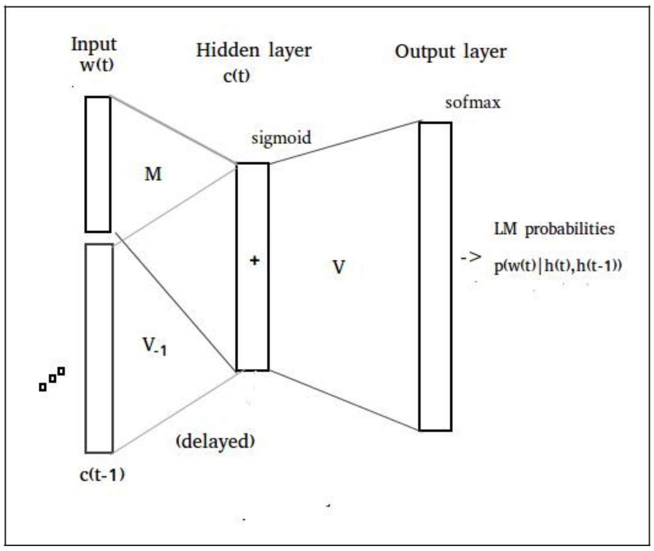
A common way to improve the back-propagation algorithm is to estimate the error and the gradients based on a sample of m examples (such as m is about 10–20 training examples). Many propagations occur before updating the weights, accumulating errors over the samples within a batch. This gives rise to the mini-batch stochastic gradient descent algorithm (Ruder, 2016).

The problem of training with large vocabularies in NNLMs (or CSLM) has received much attention. One strategy is to restrict the vocabulary of the NNLM to a shortlist and reverting to a traditional n-gram language model for other words (Schwenk, 2004). Other strategies consist of limiting the number of training examples using resampling (Schwenk and Gauvain, 2005) or selecting a subset of the training data (Schwenk et al., 2012).

3.2 RNN LM

The RNN is approximately the same as the feed-forward neural network but additional weights from the hidden layer in the previous time step are added. The hidden neurons of the RNN receive input values from both the input neurons and the hidden neurons. This is in contrast to feed-forward neural networks that receive only input values from the input neurons, as it is illustrated in Figure 3.

Figure 3 RNN language model used by Mikolov et al. (2010)



In theory, RNN's hidden layer can learn to represent unlimited memory. The context length is extended to indefinite size by using a recurrent version of neural networks which can handle arbitrary context lengths. The word representation considered by a recurrent neural network language model (RNNLM) is proposed by Mikolov et al. (2010, 2013).

The RNNLM architecture consists of an input layer, a hidden layer with recurrent connections, plus the output layer. The input vector $c(t)$ represents input word at time t encoded using 1-of-N coding vector and the output layer $o(t)$ produces a probability distribution over words.

The hidden layer thus gives a representation of the context history that iteratively accumulates an unbounded number of previous words representations using the recursive equation (12):

$$d(t) = f\left(Mc(t) + V_{-1}c(t-1)\right) \tag{12}$$

where M and V_{-1} are weight matrices between the input layer and the hidden layer.

The output layer consists of N nodes, each node is associated with one word in the vocabulary. Its activation values are given by equation (13):

$$o = Vd(t) \tag{13}$$

The softmax activation function is used to calculate the posterior probabilities of all words w_i of the vocabulary given the full history, using equation (14):

$$p_i = \exp(o_i) \Big/ \sum_{i=1}^{N} \exp(o_i) \tag{14}$$

The biases are not used in RNN, as no significant improvement of performance was observed (Mikolov et al., 2010). RNNs are typically trained using back-propagation through time (BPTT) (Werbos, 1990) in which the network is unrolled through time, and the error signal is back-propagated through multiple time steps.

4 Related works

Previous works have compared or/and improved smoothing techniques under various conditions. A good analysis is done by Chen and Goodman (1996, 1999). These works are considered as a key reference in this field. An extensive empirical comparison of smoothing techniques is performed and measured through cross-entropy. Different factors such as training set size, vocabulary file, etc., relative to the performance of these methods are tested. At last, they confirmed that for these smoothed models, the interpolated versions generally have better normalised cross-entropies while the back-off versions have more ideal expected-to-actual count ratios.

Goodman (2001) gives a proof helping to justify Kneser-Ney smoothing and describes implementation tricks and details for handling large data sizes, optimising parameters smoothing, etc. For n-gram language models, he examines up to 20-grams, but shows that even for the largest models, performance has plateaued by 5 to 7 grams.

In a practical application a falsely recognised numeral can change important content information inside the sentence more than other types of errors. In Donaj and Kacic (2014) a processed corpora of inflective language such as Slovenian, Czech, Russian or Polish languages was used to build n-gram language models smoothed using GT and Katz back-off methods for numerals. The authors proposed to sort numerals into classes. These class-based language models modelled only the classes themselves and not the words. Results of experiments showed that significant improvements were obtained on numeral-rich domains.

In Sundermeyer et al. (2011) an estimation of the discount parameters for language model smoothing is proposed. In the widely-used Kneser-Ney family of smoothing algorithms, the discount parameters can be computed directly using approximation formulas minimising the log-likelihood of the training data. The authors investigated the optimisation of discount parameters for large amounts of training data. Experiments on large English and French corpora show that increasing the number of parameters and optimising them on validation data can improve the perplexity.

In Hasan et al. (2012) analyse the performance of two different smoothing techniques for language modelling in the scope of machine translation: Witten-Bell smoothing and Kneser-Ney smoothing. The first evaluation on a parallel corpus shows that Kneser-Ney discounting method performs better than Witten-Bell method for low value of the order (n). With the increment of order, all the methods show quite similar performance. The 6-gram language models with Kneser-Ney and Witten-Bell discounting methods gave the best score.

In Dumoulin (2012) the study shows that the performance of smoothing algorithms may depend on the application of the language model. For example, unigram models with interpolation smoothing may perform better with information retrieval applications while trigram models with back-off smoothing might perform better for speech recognition. It examines the relative performance of some selected smoothing methods with bigram language models created using chat data.

In Huang et al. (2013) the smoothing methods, GT and advanced Good-Turing (AGT) are evaluated on Chinese training corpora (Giga word). The ten language models created by different size of corpus are evaluated sequentially. The experiments supported that the GT is always superior to that of AGT for all ten training models. The model with middle size of corpus of 180 M Chinese words always achieve the best performance of language model. Using cut-off value, all n-grams seen lesser then, this predefined threshold are discarded. Also, it is obvious that the smallest cut-off parameter $k \in \{1, 2, \ldots, 11\}$ gave the lowest perplexity for all language models.

The standard n-gram smoothing techniques for language modelling are still good in several aspects. Besides their good performance in terms of word error rate, they are essential for efficient decoding, and due to fast training times, they can benefit from arbitrarily large amounts of training data. In addition, any modelling approach can usually be improved by interpolation with a smoothed n-gram language model (Mikolov et al., 2010).

Advances in modelling language introduce exponential models (Chen, 2009) and neural network models (Bengio et al., 2003; Mnih and Hinton, 2007; Schwenk, 2007). We note that some studies (Chelba et al., 2013) have shown that NNLM perform better than the n-gram language models. However, training these language models is very costly for large vocabularies.

In Mikolov et al. (2010, 2011b, 2011a) an extensive experiment was performed. RNNLM were compared to several n-gram models and to more advanced models, namely the maximum entropy model, the random clustering language model, the random forest language model, the structured language model, etc. The authors observed that the basic RNNLM gave good results.

In Jozefowicz et al. (2016) the language modelling experiments were performed on the 1B Word Benchmark dataset (Chelba et al., 2013). The authors noticed that RNNLMs can be trained on large amounts of data, and outperform competing models including carefully tuned n-grams. Thus, a large language model based on long short-term memory (LSTM) (Sak et al., 2014) which is a specific RNN architecture that was designed to model temporal sequences more accurately than conventional RNNs, performs much better than n-gram models.

In Ziang (2017) noted that data noising is an effective technique for regularising neural network models used in application domains such as vision and acoustic modelling for speech. This technique based on noising primitives has not been developed for language modelling. In this paper, the authors incorporate well-understood generative assumptions that are known to be helpful in the domain. By examining the expected pseudo-counts from applying the noising schemes, they draw a connection between input noising in RNNLM and smoothing in n-gram models. They demonstrate performance gains when applying the proposed schemes to language modelling and machine translation.

In this present work, we compare the most important smoothing algorithms for n-gram language models: GT method, original Kneser-Ney smoothing and modified Kneser-Ney smoothing considering or not explicit vocabulary file and/or out-of-vocabulary option. Feed-forward language models based on the continuous space (CSLM) and RNNLM are also investigated and tested. The language models are trained on two corpora of French and English texts.

5 Experiments and evaluation

5.1 Used corpora

The language models presented in this paper were trained on two English and French recent data relating to real applications with different sizes:

1 text transcriptions of French radiobroadcast used on ETAPE evaluation campaign (Gravier et al., 2012)

2 on an English text used during the NAACL 2012 workshop (Callison-Burch et al., 2012) on SMT.

This difference between the two corpora, will allow us to evaluate and compare the performance of language models on different points of view.

5.1.1 ETAPE data

We took the ETAPE training set, which contains about three hundred thousand words and was used in Mezzoudj et al. (2015b). We recall that, ETAPE (Gravier and Adda, 2011) is a project targeting the organisation of evaluation campaigns in the field of automatic speech processing for the French language. It is partially funded by the French National Research Agency (ANR) and the *Association Francophone de la Communication Parlée* (AFCP). So we used training data shared during this evaluation campaign.

First, a pre-treatment process was applied on all data:

- eliminating empty line from the whole text
- tokenising the text so that each sentence is on a single line
- converting uppercase characters at the beginning of the line to lower case
- eliminating all non-word, non-sentence and punctuation items.

In language modelling, using only on the test set introduces a bias that makes the probabilities all look too high and causes huge inaccuracies in perplexity. So we use a particular unseen test set so often that we implicitly tune to its characteristics called the development set (also called validation set). How do we divide our development test data into training, development, and test sets?

After pre-treatment process of ETAPE data, we split it manually in three subsets. The first 18K sentences which correspond to 90% of data are used for training, the next 5% of data for development and the final 5% of data for test, as recommended in Haton et al. (2006). The size of the corpus in terms of sentences and words is shown in Table 1.

Table 1 ETAPE corpora statistics

Corpus	*Sentence count*	*Word count*
Etape-train.	18,083	249,569
Etape-dev.	1,004	10,782
Etape-test	1,004	16,419

Since our training data is very close in topic and style to the test data, the results should give the real performance of the models even though the size of the training data is small. The model parameters are optimised on the development corpus.

5.1.2 NAACL data

We used NAACL data package which was shared during the NAACL'2012 workshop on SMT (Callison-Burch et al., 2012). We used nc7 and eparl7 data to train language models, newstest2010 data for development and newstest2011 data for test, as divided and recommended during this workshop. These texts were tokenised with the standard Moses tools (Koehn et al., 2007). Its size in terms of sentences and words are shown in Table 2.

Table 2 NAACL corpora statistics

Corpus	*Sentence count*	*Word count*
nc7-train.	212,517	5,085,447
eparl7-train.	2,218,201	59,940,634
newstest2010-dev.	2,489	61,924
newstest2011-test	3,003	74,833

For language modelling on NAACL corpus, we note that we use two training sets for the creation of the language model (see Table 2). The usefulness of using two (or more) training sets is that besides the training data corresponding to the domain of interest (called in-domain data), the training of a good language model can take benefit of data coming from different sources or domains (called non-domain data).

The conventional approach is a three-step process. In the first, we train a separate language model on each individual training corpus: a small amount of in-domain data (in this case, the nc7 corpus) and large amount of non-domain data (such as eparl7). In the second step, the weights of the linear interpolation (LI) of the individual models are estimated so as to maximise the likelihood of the interpolated language model development data via expectation-maximisation (EM) algorithm. Finally, the individual models are interpolated according to the optimal weight values (μ_i) using the following equation:

$$p^{LI}(w \mid h) = \sum_i \mu_i p_i(w \mid h)$$

where *i* is the index of the individual language models and $p^{LI}(w|h)$ is the interpolated language model. The interpolation weight μ_i of the individual language models, satisfying $\sum_i \mu_i = 1$, is typically tuned to optimise the development data perplexity (Jelinek, 1980). This technique was explored thoroughly in Mezzoudj et al. (2015a).

It is important to differentiate between interpolations of the different levels of n-grams which allow the smoothing of the language model itself (Chen and Goodman, 1996) and between interpolating several language models (Jelinek, 1980) to get one.

All the experiments were done on the available machine, Intel Core i5-2430M CPU 2.40GHz 4 (5,9 Go of memory and 62,5 Go of hard disk). The results are evaluated using perplexity, a standard intrinsic evaluation metric for language models.

5.2 *Performance of n-gram language models*

5.2.1 *ETAPE data*

In this paper, all the n-gram language models were built with the SRILM toolkit (Stolcke et al., 2002, 2011) a commonly used, off-the-shelf toolkit for building and applying statistical language models. This package is developed, maintained and distributed under an open source community license by International's Speech Technology and Research Laboratory (SRI) of California.

Before, training any n-gram language model, we note that the most important factors that influence the quality of the resulting n-gram model are the choice of the order *n* and of the smoothing technique. Concerning the order *n*, the longest the context on which we train the model is, the most coherent the sentences are. In the unigram sentences, there is no coherent relation between words. The bigram sentences have some local word-to-word coherence. The 3-gram and 4-gram sentences are better for modelling language and cheaper in calculation than 5, 6 or 7-gram. So, we report results while using 4-gram with the three popular smoothing methods: GT smoothing, Kneser-Ney smoothing (with its two versions) and modified Kneser-Ney smoothing (also with its two versions).

Two vocabulary files were generated independently and explicitly by a unigram model as proposed in the tutorial of CSLM toolkit (Schwenk, 2013) from the available data, the ETAPE data and the NAACL data respectively. For the ETAPE corpus, we generated a vocabulary file of 15,662 words using the training (Etape-train.), development (Etape-dev.) and test data (Etape-test). Naturally, the size of the vocabulary grows with the corpus size that it is extracted from. For the NAACL corpus, we generated a vocabulary file of 166,952 words, using the available training (nc7-train and eparl7-train), development (newtest2010-dev) and test data (newstest2011-test).

The development (dev) and test (test) perplexities of language models trained on ETAPE and NAACL data using the two versions of the modified Kneser-Ney smoothing method are given in Tables 3 and 4.

In these experiments, for accurate comparison, we considered or not explicit vocabulary file (Vocab.) and/or out-of-vocabulary (OOV) options. The crossed-out terms show the non considered options during training.

In Table 4, the language models interpolated linearly using the two language models trained on the two training sets (nc7 and eparl7) of NAACL data are noted *all*.

Table 3 The use (or not) of vocabulary and/or OOV for language modelling on ETAPE data

Method	*Interpo. MKN*		*Back-off MKN*	
	pp-dev	*pp-test*	*pp-dev*	*pp-test*
OOV, Vocab.	242.5	244.6	221.2	*221.3*
OOV, ~~Vocab.~~	241.7	244.3	159.6	*157.6*
~~OOV~~, Vocab.	242.2	*164.7*	221.0	184.6
~~OOV, Vocab~~.	163.2	*161.3*	175.7	173.8

Note: The crossed-out terms show the non-considered options during the training.

Habitually in literature, we notice that back-off methods for smoothing language models do not do well but the models smoothed using the interpolation of n-grams with higher-order and the lower-order distribution are good (Chen and Goodman, 1999). However, the best results are achieved using the back-off modified Kneser-Ney algorithm (back-off MKN) using the (*-kndiscount*) command with several experiments performed on the two corpora, as reported in Tables 3 and 4.

In SRILM Toolkit, the use of a closed vocabulary mode is set by the command line option (*-unk*). When an open-vocabulary language model is used, the appearance of unknown word in a recognition hypothesis is treated by the n-grams containing the UNK token. This is where the trouble lies. The direct use of probabilities of n-grams ending in UNK is not quite correct. The problem is that the probability of such n-grams is actually the sum of the probabilities for all potential n-grams ending in unknown words (with the same context), including those absent in training corpus.

Table 4 The use (or not) of vocabulary (Voc.) and/or OOV for language modelling on NAACL data with two versions of modified Kneser-Ney back-off (noted Back-off MKN) and interpolated (noted Interpo. MKN)

Method	*Train*	*Interpo. MKN*			*Back-off MKN*		
		pp-dev	*pp-test*	*weight*	*pp-dev*	*pp-test*	*weight*
OOV, Vocab.	nc7	477.7	501.5	0.51	478.1	496.6	0.45
	eparl7	451.4	510.9	0.49	410.4	455.4	0.55
	All	343.8	*376.4*	1	338.2	*365.8*	1
OOV, ~~Vocab.~~	nc7	307.3	314.8	0.52	310.2	314.9	0.49
	eparl7	326.8	370.4	0.48	308.3	341.8	0.51
	All	261.4	286.7	1	264.0	*285.7*	1
~~OOV~~, Vocab.	nc7	477.7	391.6	0.49	478.1	399.4	0.44
	eparl7	451.4	385.2	0.51	410.4	367.2	0.56
	All	344.3	*288.3*	1	338.1	293.7	1
~~OOV, Vocab~~.	nc7	315.5	325.3	0,47	327.1	334.2	0.44
	eparl7	319.9	365.1	0.53	311.6	346.5	0.56
	All	260.7	*287.3*	1	286.2	*291.0*	1

Note: The crossed-out terms show the non-considered options during the training.

For training the 4-gram language models, we note that we considered the vocabulary file which corresponds to the command (*-vocab*) and we used (*-unk*) switch for SRILM tools as the data contain UNK tokens for unknown words.

When we used the vocabulary and the OOV in the training, the back-off unmodified Kneser-Ney algorithm (back-off uMKN) gave better results than the interpolated unmodified Kneser-Ney algorithm (interpo. uMKN), which corresponds to the first line in Tables 3 and 4.

In Mezzoudj et al. (2015b) noticed that the smoothing methods with interpolation work well. But for more precision, this situation is true if we do not use an explicit vocabulary file and (/or) do not consider the OOV option using SRILM toolkit, as we see at last line in Tables 3 and 4. However, considering these two parameters OOV and vocabulary are essential for correct language model training, so for all future experiments conducted, these parameters should be used.

If any discounting method is chosen in SRILM toolkit, the standard method used to train the language model is GT method (Good, 1953) combined with Katz method (Katz, 1987).

Table 5 LM perplexity with different smoothing algorithms on development (dev.) and test ETAPE data

LM	*pp-dev*	*pp-test*
Good-Turing LM	380.3	387.7
Back-off uMKNLM	219.5	*220.1*
Interpo. uMKNLM	255.6	258.4
Back-off MKNLM	221.2	*221.3*
Interpo. MKNLM	242.5	244.6

We noted that, a simple way of reducing the language model size is to apply pruning technique using a discount-cut-off, so all n-grams seen lesser then a predefined threshold are discarded. The default GT discount is at most seven cut-offs (Stolcke et al., 2002). In our case, we do not specify any smoothing method and any cut-off parameters, so we use the default GT. The perplexity of the standard 4-gram GT LM is 387.7 (see the first line of Table 5).

We also generate four 4-gram models with two versions back-off and interpolated of unmodified Kneser-Ney methods (back-off uMKNLM, interpo. uMKNLM) and of modified Kneser-Ney methods (back-off MKNLM, interpo. MKNLM). The most important result presented in Table 5 is that the language models trained with back-off Kneser-Ney versions achieve better results with close perplexities of 220.1 and 221.3 vs. the other language model versions.

5.2.2 NAACL data

We conducted similar experiments with NAACL data. The 4-gram language models were obtained by interpolating the two language models trained on the in-domain (nc7-train) and the non-domain data (eparl7-train) respectively. We used a smoothing method with the adequate vocabulary and the OOV option each time. The perplexities of the final interpolated language model (all) are given in Table 6.

The result achieved by the interpolated modified Kneser-Ney method (interpo. MKN) was of 376.4 besides the perplexity achieved by the back-off modified Kneser-Ney method (back-off MKN) of 365.8. Also, the modified back-off Kneser-Ney version achieved the best result with perplexity of 365.8 vs. the perplexity (of 376.4) obtained by the interpolated modified Kneser-Ney algorithm.

In general, the back-off smoothing versions work well when we include vocabulary file explicitly in the training of language models which may be related to the implementations proposed in the SRILM toolkit (Stolcke et al., 2002). So limiting the vocabulary using (*-ngram-*

count) command without (*-vocab*) can directly trouble the count-of-count frequencies, since we are effectively mapping low-frequency tokens to the OOV. Presumably, this situation will affect the interpolated Kneser-Ney in a bad way. We observe that the 4-gram model with back-off modified Kneser-Ney (back-off MKNLM) smoothing on NAACL data is performing the best among n-gram models with a perplexity of 365.8 (see Table 6), and we used it further as a baseline.

Table 6 LM perplexity with different smoothing algorithms on development (dev.) and test NAACL data

LM	*pp-dev.*	*pp-test*
Good-turing LM	487.4	533.6
Back-off uMKN LM	338.2	367.3
Interpo. uMKN LM	356.4	392.3
Back-off MKN LM	338.2	*365.8*
Interpo. MKN LM	343.8	376.4

5.3 Performance of NNLM

5.3.1 ETAPE data

All the feed-forward language models were built with the open-source CSLM toolkit (Schwenk, 2013). This toolkit, programmed using C++, is modular and relies on highly optimised mathematics libraries for the computational intensive parts such as the Basic Linear Algebra Subprograms (BLAS), including support for GPU cards.

The CSLM implemented is a multi-layer neural network: the input layer (first layer) projects all words in the context h_i onto the projection layer (second layer), the hidden layer (third layer) and the output layer (fourth layer) achieves the nonlinear probability estimation.

The CSLM is able to calculate the probabilities of all words in the vocabulary of the corpus. However, due to too high computational complexity, CSLM is mainly used to calculate the probabilities of a subset of the whole vocabulary. This subset is called a short-list, it consists of the most frequent words in the vocabulary. The CSLM redistributes the probability mass of other words using the standard n-gram language model.

Also, we used the proposed strategy of restricting the vocabulary of the CSLM to a shortlist and reverting to a traditional n-gram language model for other words (Schwenk, 2004). So the CSLMs were trained on the same training data sentences using CSLM toolkit based on a 4-gram language model.

The choice of the neural network hyper-parameters has to be done manually: type of activation function, choice of architecture (how many hidden layers, their size), number of training epochs, which features are presented at the input layer, etc.

We used three neural network architectures: a simple CSLM (with one hidden layer), a large CSLM (large number of neurons per layer) and a deep CSLM (with three hidden layers). This flexibility is a good property. In many cases, these changes led to improvements in accuracy compared to basic models (with simple architecture).

Following the existing setup proposed in the CSLM tutorial, we used values close to the number of neurons by a layer and shortlist length. However, we chose the other parameters of the neural network in relation with the size of the vocabulary and the textual data such as the input features and their size. For the nonlinear activation function, we used hyperbolic tangent (htan).

We trained the simple feed-forward neural network CSLM for 10 epochs. We consider the input layer with the same dimension as vocabulary size which is 15,662, the projection layer has a dimension of 256 for each word followed by one 768 × 192-dimensional tanh hidden layer. The softmax output layer has a dimension of 1,024 (which is the size of the used shortlist).

The large feed-forward neural network CSLM is trained for 10 epochs. The projection layer has a dimension of 256, followed by one 768 × 192-dimensional tanh hidden layer and a softmax output layer of dimension 8,192.

The deep feed-forward neural networks CSLMs were trained for 10 epochs. The projection layer has a dimension of 256, followed by three (768 × 512), (512 × 256) and (256 × 192)-dimensional tanh hidden layers and a softmax output layer of dimension 8,192.

We wanted to use deeper (more than three layers) and larger architectures with more data but unfortunately we are discomforted and limited by the material resources.

The RNNLM were built with the open-source toolkit of Mikolov et al. (2011c). It is an open source and freely available toolkit for training statistical language models based on RNN and hash-based maximum entropy models. The toolkit includes techniques for reducing computational complexity (classes in the output layer and direct connections between input and output layer). In this toolkit, a common compromise choice is to use mini-batches (Ruder, 2016) with a batch of small size and with stochastically selected samples in order to improve the back-propagation algorithm performance.

According to Mikolov et al. (2011c) the optimal size of the hidden layer depends on the size of the training data. For less than 1M words, 50–200 neurons is usually enough, for 1M–10M words use 200–300 (using larger hidden layer usually does not degrade the performance but makes the training progress slowly). For our RNNLM, we created language models with 50 neurons in the hidden layer. We used 50 classes to speed up the training progress. We used mini-batches of size ten and unroll for four steps when performing BPTT algorithm (Werbos, 1990).

The timings and perplexities of language models trained on ETAPE data are presented in Table 7. The training of the CSLM took respectively 30 min, 75 min and 49 min for simple, large and deep neural networks architectures. For the RNNLM trained on the same data, the experiment took 101 min for the stages of training and test. We see that the n-gram language model smoothed with the modified Kneser-Ney (MKN) is considerably faster than NNLM (CSLM and RNNLM) but the RNNLM gives the best performance. Unfortunately and not as expected, the CSLM performance for the three architectures simple, large and deep (resp. with perplexities 295.6, 455.3 and 272.7) were not better than that of the n-gram language models. Knowing that the feed-forward neural networks are intensive training data, these results may be due to the amount of ETAPE data (about 250 K) and may not be sufficient for good CSLM training. However, the RNNLM had a good perplexity of 153.6 than all other language models. The experiments prove that RNNLMs can be competitive with CSLM and n-gram back-off language models that are trained on same data.

Table 7 LM perplexity on dev. and test data (ETAPE)

LM	*ppl dev*	*ppl test*	*Time[min]*
Back-off MKN LM	221.2	*221.3*	15
Simple CSLM	278.2	295.6	30
Large CSLM	432.6	455.3	75
Deep CSLM	253.3	272.7	49
RNLM	159.4	*153.6*	101

5.3.2 NAACL data

Using the two training subsets of NAACL (in-domain and non-domain data), we trained two RNNLMs and we interpolated them linearly using LI (Jelinek, 1980).

We used three neural network architectures: a simple CSLM (with one hidden layer), a large CSLM (large number of neurons per layer) and a deep CSLM (with three hidden layers). We trained the simple feed-forward neural network CSLM for 10 epochs. We considered the input layer with the same dimension as NAACL vocabulary size which is 166,952, the projection layer has a dimension of 256 for each word followed by one 768 × 192-dimensional tanh hidden layer. The softmax output layer has a dimension of 1,024 (which is the size of the used shortlist).

For RNNLM, we created language models with 100 neurons in the hidden layer. We used 50 classes to speed up the training progress. We used mini-batches of size ten and unroll for four steps when performing BPTT algorithm.

Table 8 NNLM perplexity on dev. and test data (NAACL)

LM	*pp- dev*	*pp- test*	*Time[h]*
Back-off MKN LM	338.2	365.8	1
Simple CSLM	327.9	356.4	49
Large CSLM	331.8	359.5	90
Deep CSLM	334.4	362.5	72
RNNLM	248.85	*264.9*	133

The timings and perplexities of language models trained on NAACL data are presented in Table 8. We see that the training of the 4-gram MKN language model is considerably very faster (with 1 h) than NNLM (CSLM and RNNLM). The training of the CSLM took 49 h, 90 h and 72 for simple, large and deep neural networks architectures respectively. The shortest experiment took almost two days (49 h) for the CSLM training with the simple neural network architecture. The perplexity of the simple CSLM of 356.4 is the best result compared to large, deep feed-forward neural networks and 4-gram language model.

The CSLMs with its three architectures: simple, large and deep trained on NAACL corpus perform better than n-gram language model (with perplexity of 365.8).

For the RNNLM trained on NAACL data, the experiment took almost five days and a half (133 h). The perplexity of RNNLM is 264.9 which is the best one for NAACL data (see Table 8). Unfortunately, the RNN suffers from drawbacks that limit their practical usefulness and makes it very slow.

6 Conclusions

Statistical language modelling is one of the most important NLP tasks. Language models are core of speech recognition, machine translation and many other applications. In this paper, we investigated language modelling under average conditions in terms of material strength and data quantity. The goal is to analyse the behaviour of different language models based on n-grams using the Kneser-Ney smoothing family methods and neural networks: the feed-forward neural networks with CSLM and RNNLM on two corpora from two different languages (French and English) and with different sizes. One with more than two hundred thousand words (ETAPE corpus), and a larger one with above five million words (NAACL corpus).

For n-gram language models, the probability of a words sequence is approximated by the product of conditional probabilities of $n - 1$ previous words. Unfortunately, the n-gram language model is defined in a discrete binary space of word indices with no way to learn similarities between words. An unseen n-gram is generalised by smoothing techniques using back-off or interpolation of the lower order n-gram models. According to the literature, the interpolated smoothing methods for n-gram language models are better than the back-off one.

The experiments on the two corpora ETAPE and NACCL have shown that the standard n-gram models still remain notable for their simplicity and computational efficiency, the training experiments lasted few minutes to an hour and gave medium-performance results. Considering the vocabulary file and the OOV option, the n-gram language models smoothed using the back-off modified Kneser-Ney (back-off MKN) methods gave better results than the interpolated versions.

NNLM try to address the sparsity data problems by learning continuous-valued word representations. For the CSLM shortlist, an unseen word during training is generally based on distance and similarity between the words in the continuous space. If the unseen word in some sequences is close enough to some seen words during training, a similar probability will be assigned to it. For the other words out of the shortlist, their history is represented by context of $n - 1$

words using the probabilities estimated by an n-gram language model. According to the results, the CSLM can be useful for language modelling but like n-gram language models, the CSLM can have a finite and discrete context. Based on feed-forward neural network, the CSLM requires a large amount of data for a good training. The results obtained with the CSLM trained on NAACL data are more interesting than those obtained with the ETAPE data vs. the results obtained using n-gram language models.

With the RNNLM, the history of the words is learned by the hidden layer neurons using recurrent connections. RNNLM manages to exploit the available data at the maximum thanks to the use of the recurrence which enhance the word quality representation with a long context. Experimental results using RNNLM have shown that they can reduce perplexities compared with both n-gram language models and feed-forward CSLMs. However, neural language models are generally notoriously slow in training compared to the standard n-gram language models.

We could not see the influence of the difference of the French and English languages on training language models. In contrast, the quantity of data affects the performance of language models but if the amount of learning data is large, the CSLM and the RNNLMs become slower.

In a parallel work, we are doing an extended comparative study with all smoothing methods of the n-gram language models. In future work, we are going to compare advanced language models trained on larger amounts of Arabic data.

References

Bengio, Y., Ducharme, R., Vincent, P. and Janvin, C. (2001) ‘A neural probabilistic language model’, in *NIPS*, IEEE, pp.933–938.

Bengio, Y., Ducharme, R., Vincent, P. and Janvin, C. (2003) ‘A neural probabilistic language model’, *The Journal of Machine Learning Research*, February, Vol. 3, pp.1137–1155.

Bottou, L. (2012) ‘Stochastic gradient descent tricks’, in *Neural Networks: Tricks of the Trade*, pp.421–436, Springer, Berlin, Heidelberg.

Brants, T., Popat, A.C., Xu, P., Och, F.J. and Dean, J. (2007) ‘Large language models in machine translation’, in *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Citeseer.

Callison-Burch, C. et al. (2012) ‘Findings of the 2012 workshop on statistical machine translation’, in *Proceedings of the Seventh Workshop on Statistical Machine Translation*, June, Montréal, Canada, pp.10–51.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P. and Robinson, T. (2013) *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling*, arXiv preprint arXiv:1312.3005.

Chen, S.F. (2009) ‘Shrinking exponential language models’, in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, pp.468–476.

Chen, S.F. and Goodman, J. (1996) ‘An empirical study of smoothing techniques for language modeling’, in *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, pp.310–318.

Chen, S.F. and Goodman, J. (1999) ‘An empirical study of smoothing techniques for language modeling’, *Computer Speech & Language*, Vol. 13, No. 4, pp.359–393.

Chen, S.F., Beeferman, D. and Rosenfeld, R. (1998) ‘Evaluation metrics for language models’, *Proc. DARPA Broadcast. News Transcription and Understanding Workshop*, pp.275–280.

Chomsky, N. (2014) *Aspects of the Theory of Syntax*, Vol. 11, MIT Press, Cambridge.

Donaj, G. and Kacic, Z. (2014) ‘Manual sorting of numerals in an inffective language for language modelling’, *International Journal of Speech Technology*, Vol. 17, No. 3, pp.281–289.

Dumoulin, J. (2012) ‘Smoothing of n-gram language models of human chats’, in *2012 Joint 6th International Conference on Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS)*, IEEE, pp.1–4.

Good, I.J. (1953) ‘The population frequencies of species and the estimation of population parameters’, *Biometrika*, Vol. 40, Nos. 3/4, pp.237–264.

Goodman, J.T. (2001) ‘A bit of progress in language modeling’, *Computer Speech & Language*, Vol. 15, No. 4, pp.403–434.

Gravier, G. and Adda, G. (2011) ‘Evaluations en traitement automatique de la parole’, *Evaluation Plan*, Etape.

Gravier, G., Adda, G., Paulson, N., Carrée, M., Giraudel, A. and Galibert, O. (2012) ‘The ETAPE corpus for the evaluation of speech-based TV content processing in the French language’, in *LREC-Eighth International Conference on Language Resources and Evaluation*.

Hasan, A., Islam, S. and Rahman, M. (2012) ‘A comparative study of Witten Bell and Kneser-Ney smoothing methods for statistical machine translation’, *Journal of Information Technology*, June, Vol. 1, pp.1–6.

Haton, J-P., Cerisara, C., Fohr, D., Laprie, Y. and Smaali, K. (2006) *Reconnaissance automatique de la parole: Du Signal à son Interprétation*, Dunod.

Hornik, K., Stinchcombe, M. and White, H. (1989) ‘Multilayer feed-forward networks are universal approximators’, *Neural Networks*, Vol. 2, No. 5, pp.359–366.

Huang, F-L., Yu, M-S. and Hwang, C-Y. et al. (2013) ‘An empirical study of Good-Turing smoothing for language models on different size corpora of Chinese’, *Journal of Computer and Communications*, Vol. 1, No. 05, p.14.

Jaeger, H. (2002) ‘Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach’, *GMD-Forschungszentrum Informationstechnik*.

Jelinek, F. (1980) ‘Interpolated estimation of Markov source parameters from sparse data’, *Pattern Recognition in Practice*, The Netherlands Amsterdam,, North- Holland, May, pp.381–397.

Jelinek, F. (1997) *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, MA.

Jelinek, F., Mercer, R.L. and Roukos, S. (1991) ‘Principles of lexical language modeling for speech recognition.’, in Furui, S. and Sondhi, M.M. (Eds.): *Advances in Speech Signal Processing*, pp.651–699, Marcel Decker, New York.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N. and Wu, Y. (2016) *Exploring the Limits of Language Modeling*, arXiv preprint arXiv:1602.02410.

Katz, S. (1987) 'Estimation of probabilities from sparse data for the language model component of a speech recognizer', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 35, No. 3, pp.400–401.

Kneser, R. and Ney, H. (1995) 'Improved backing-off for m-gram language modeling', in *1995 International Conference on Acoustics, Speech, and Signal Processing, 1995, ICASSP-95*, IEEE, Vol. 1, pp.181–184.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C. and Zens, R. et al. (2007) 'Moses: open source toolkit for statistical machine translation', in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, Association for Computational Linguistics, pp.177–180.

Mezzoudj, F., Langlois, D., Jouvet, D. and Benyettou, A. (2015a) 'Textual data selection for language modelling in the scope of automatic speech recognition', in *International Conference on Natural Language and Speech Processing*, Algeria.

Mezzoudj, F., Loukam, M. and Benyettou, A. (2015b) 'On an empirical study of smoothing techniques for a tiny language model', in *Proceedings of IPAC 15*, ACM, 23–25 November, Batna, Algeria, pp.67–80.

Mikolov, T., Karafiat, M., Burget, L., Cernocky, J. and Khudanpur, S. (2010) 'Recurrent neural network based language model', in *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, 26–30 September, Makuhari, Chiba, Japan, pp.1045–1048.

Mikolov, T., Deoras, A., Kombrink, S., Burget, L. and Cernocky, J. (2011a) 'Empirical evaluation and combination of advanced language modeling techniques', in *INTERSPEECH*, No. 1, pp.605–608.

Mikolov, T., Kombrink, S., Burget, L., Cernocky, J.H. and Khudanpur, S. (2011b) 'Extensions of recurrent neural network language model', in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp.5528–5531.

Mikolov, T., Kombrink, S., Deoras, A., Burget, L. and Cernocky, J. (2011c) 'RNNLM-recurrent neural network language modeling toolkit', in *Proc. of the 2011 ASRU Workshop*, pp.196–201.

Mikolov, T., Yih, W-T. and Zweig, G. (2013) 'Linguistic regularities in continuous space word representations', in *HLT-NAACL*, Vol. 13, pp.746–751.

Mnih, A. and Hinton, G. (2007) 'Three new graphical models for statistical language modelling', in *Proceedings of the 24th International Conference on Machine Learning*, ACM, pp.641–648.

Ney, H. and Essen, U. (1991) 'On smoothing techniques for bigram-based natural language modelling', in *1991 International Conference on Acoustics, Speech, and Signal Processing, 1991, ICASSP-91*, IEEE, pp.825–828.

Rabiner, L.R. and Juang, B. (2005) 'Statistical methods for the recognition and understanding of speech', in *Encyclopedia of Language and Linguistics*, 2nd ed.

Robert, C.P., Chopin, N. and Rousseau, J. (2009) 'Harold Jeffreys's theory of probability revisited', *Statistical Science*, Vol. 24, No. 2, pp.141–172.

Rosenfeld, R. (2000) 'Two decades of statistical language modeling: where do we go from here?', *Proceedings of the IEEE*, Vol. 88, No. 8, pp.1270–1278.

Ruder, S. (2016) *An Overview of Gradient Descent Optimization Algorithms*, arXiv preprint arXiv:1609.04747.

Sak, H., Senior, A. and Beaufays, F. (2014) *Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition*, arXiv preprint arXiv:1402.1128.

Schwenk, H. (2004) 'Efficient training of large neural networks for language modeling', in *2004 IEEE International Joint Conference on Neural Networks, 2004, Proceedings*, IEEE, Vol. 4, pp.3059–3064.

Schwenk, H. (2007) 'Continuous space language models', *Computer Speech & Language*, Vol. 21, No. 3, pp.492–518.

Schwenk, H. (2013) 'CSLM: a modular open-source continuous space language modeling toolkit', in *INTERSPEECH*.

Schwenk, H. and Gauvain, J-L. (2005) 'Training neural network language models on very large corpora', in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, pp.201–208.

Schwenk, H., Rousseau, A. and Attik, M. (2012) '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*, Association for Computational Linguistics, pp.11–19.

Stolcke, A. et al. (2002) 'Srilm-an extensible language modeling toolkit', in *INTERSPEECH*.

Stolcke, A., Zheng, J., Wang, W. and Abrash, V. (2011) 'SRILM at sixteen: update and outlook', in *Proceedings of IEEE Automatic Speech Recognition and Understanding Workshop*, p.5.

Sundermeyer, M., Schlüter, R. and Ney, H. (2011) 'On the estimation of discount parameters for language model smoothing', in *INTERSPEECH*, pp.1433–1436.

Werbos, P.J. (1990) 'Backpropagation through time: what it does and how to do it', in *Proceedings of the IEEE*, Vol. 78, No. 10, pp.1550–1560.

Witten, I.H. and Bell, T.C. (1991) 'The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression', *IEEE Transactions on Information Theory*, Vol. 37, No. 4, pp.1085–1094.

Ziang, X. et al. (2017) *Data Noising as Smoothing in Neural Network Language Models*, arXiv preprint arXiv:1703.02573.