Statistical Learning Course - Final Project

Prof. Mário Figueiredo

Instituto Superior Técnico

Language Models

August, 2016

David Fernandes Semedo
Faculdade de Ciências e Tecnologia - Universidade Nova de Lisboa
df.semedo@campus.fct.unl.pt

Contents

Chapter 1

Motivation and Scope

The task of computationally understanding language is crucial for several natural language processing (NLP) applications. Traditional approaches consisted of using formal language theory tools, like Finite Automata, to recognise or generate sentences. However, this approach fails to generalise to different (in terms of structure and/or vocabulary) language sequences.

Language modelling methods attempt to capture natural language regularities (e.g. structure) from text corpora, speech audio signals, among others, by estimating the distribution of different natural language phenomena, which in turn allows for the development of complex and effective (NLP) technologies. In short, a Language Model (LM) models the probability of linguistic units at different granularity's such as words, sentences or documents. For many NLP applications like text summarisation, information retrieval (IR), sentence generation, document classification, machine translation, speech recognition, among others, LMs play a central role [29, 37, 49].

Given the infinite set of possible sentences and the fact that natural vocabularies are large, estimating the mentioned distributions requires the estimation of a large number of parameters, making the task of learning a LM not trivial and requiring large amounts of training data. To deal with this complexity and to reduce the computational requirements, some LMs are developed around certain assumptions. However, an implication of these assumptions is that models lose expressiveness and may fail to capture certain language regularities. Consequently, the complexity of the methods used to learn LMs is closely related with the assumptions that they are based on, i.e., less assumptions imply higher complexity. Notwithstanding, even under these assumptions, remarkable results have been achieved using LMs, across a wide range of tasks.

Effective methods that do not make any simplifying assumption, allowing them to capture any kind of linguistic regularity from a theoretical point-of-view, while keeping the computational requirements contained, have been developed [41]. Despite of this potential unlimited expressiveness, for now, these models do not outperform models developed under certain assumptions for some specific tasks. This observation is worth further analysis and comparison between different types of methods to assess the reasons of each method effectiveness for some tasks.

Apart from traditional language modelling, lately several methods extending LMs and incorporating additional elements (e.g. images features) have been

proposed, aiding capturing the semantics of data [23, 27, 38]. The idea is to obtain a LM that not only captures the linguistic regularities but is also capable of correlating/aligning additional elements with those regularities. A good example of such scenario is image captioning, in which the estimated distribution, models not only linguistic units but also images, allowing one to generate text semantically related with that image.

Being able to include such additional elements could be an approach to address the observation made by Rosenfeld [59] in which, from Shannon-style experiments[1] whose objective is to elicit human knowledge of language, it was observed that humans outperform easily language models. The hypothesis is that humans do so by reasoning not only at a linguistic but also at a common sense and domain level. By developing models that are able to correlate/align text with additional elements from data, which provide by themselves a different *perspective* of the data/information, can in principle leverage the task of learning the semantics of the data and in turn, implicitly incorporate knowledge of the domain into those models. This last line of work is closely tied to the task of representing different modalities (e.g. text and image) on the same geometric space (multimodal or cross-modal space) [57]. Being able to abstract from each modality representation and understand information involves interpreting (syntactically, visually, and semantically) each representation, which is something easy for humans but not for machines. However, clever techniques able to learn these abstractions have been developed and integrated into LMs.

In this work, methods for learning LMs will be analysed and discussed, covering from traditional models (e.g. Markov models) to neural language models. State-of-the-art methods for multimodal LM learning will be surveyed. Special emphasis will be given to the statistical properties of the different models, in order not only understand the methods properties and assumptions.

A final application which consists in generating descriptions (sentences) for content, like images, based on different types of LMs, while capturing its semantics, will be assumed through all the discussions.

The remainder of this work is organised as follows. In chapter 2 we start by defining a LM and the notation used throughout this work. The n-gram model will be studied in detail followed by an analysis of metrics for LM quality evaluation. Chapter 3 presents and discusses in depth a traditional LM approach, the *Query Likelihood*, and smoothing techniques. Neural language models, including multimodal LMs, will be covered and surveyed in chapter 4. To end this work, conclusions and possible future steps will be presented in

[1] The experiment consists of asking a human to predict the next element of text.

chapter 5.

Chapter 2

Statistical Language Modelling

2.1 Definitions

To ease the mathematical specification and analysis of the models and methods that will be covered in this work, a theoretical framework specifying a formulation for some objects common to all the methods, is presented in this section.

Let $\mathcal{V}$ be a finite set containing all the words of the language, i.e., the *vocabulary*. Then, the set of all possible sequences of words is the infinite set $\mathcal{V}^*$, where $\mathcal{V}^* = \{\epsilon\} \cup \mathcal{V}^1 \cup \mathcal{V}^2 \dots$ (Kleene closure). Naturally, a *sentence* is a sequence of words $(w_1, \dots, w_i, \dots, w_N)$, for some $N \in \mathbb{N}$, such that each $w_i \in \mathcal{V}$.

Definition 1 (Language Model). *Given a vocabulary $\mathcal{V}$, a language model is defined by $\mathcal{V}$ and by a probability distribution $P(W_1 = w_1, \dots, W_N = w_N)$, where each W_i is a random variable taking the value w_i, such that:*

(i) *For any sequence $(w_1, \dots, w_n) \in \mathcal{V}^*$, $P(W_1 = w_1, \dots, W_n = w_N) \geq 0$*

(ii) $\sum_{(w_1,\dots,w_N)\in\mathcal{V}^*} P(W_1 = w_1, \dots, W_N = w_N) = 1$

The problem of learning a language model corresponds to the problem of estimating $P(W_1 = w_1, \dots, W_N = w_N)$. It is noteworthy to point that this assumes that sentences from the corpus are independent.

It is assumed that training data used to estimate the distribution $P(W_1 = w_1, \dots, W_N = w_N)$, consists of a *corpus* $D = \{s_1, \dots, s_j, \dots, s_{N_c}\}$ with a total of N_c sentences, where each s_j is a sentence, such that $D \subseteq \mathcal{V}^*$. Each sentence s_j is a sequence of words $s_j = (w_1^j, \dots, w_k^j, \dots, w_{N_s}^j)$ where each $w_k^j \in \mathcal{V}$ and N_s denotes the total number of words in s_j. The set of unique words in D is defined as $W^D = \{w_i : w_i \in s_j, \forall\ s_j \in D\}$[1] and the cardinality $|W^D|$ of the set W^D denotes the total number of unique words.

This definition of a LM follows a probabilistic *generative* approach. In other words, the joint probability of words in a sentence is modelled. Therefore, this approach provides a model of how sentences are actually generated. On the other hand, *discriminative* approaches learn boundaries between different classes

[1] A set by definition only contains distinct objects.

(e.g. positive of negative sentences), i.e., they learn how to separate data by some discriminative criteria. In principle, it follows that generative models can learn the underlying structure of data (for the language modelling task, learn the linguistic regularities) provided that the model is correct for the data at hand. However, usually generative models make certain assumptions, enforcing a given structure for data, that may not be reflected in the true structure of data.

In section 3.3 we extend the discussion between the two approaches highlighting the main advantages and disadvantages of both.

2.2 *N*-gram models

As stated in the previous section, the task of learning a LM is equivalent to the task of estimating the probability distribution $P(W_1 = w_1, \ldots, W_N = w_N)$. A straightforward method would be to count the number of times the sequence $(w_1, \ldots, w_N)$ appears and normalise by the total number of sentences. The problem with this approach is that if one wants a LM capable of generalising to a certain degree, this model will not work, since non occurring sentences will be assigned a probability of 0.

An alternative approach is to decompose the target probability distribution at the word-level. Given a sentence of size N, this can be achieved using the chain rule as follows:

$$P(W_1 = w_1, \ldots, W_N = w_N) = \prod_{i=1}^{N} P(W_i = w_i | W_1 = w_1, \ldots, W_{i-1} = w_{i-1}) \tag{2.1}$$

where the target probability distribution is now defined by a product of word probabilities given the precedent words, of each word in the sentence. This definition does not make any kind of independence assumptions, therefore, it does not lose expressiveness. However, from a computational perspective, it involves estimating a huge amount of parameters, making it intractable.

Nevertheless, the expression of equation 2.1 can be approximated. One way to do so is to use *Markov chains*. Intuitively, this is equivalent to bounding the number of precedent words from which each word w_i is conditioned on to a fixed value m. A *Markov chain* of order m is defined as follows.

Definition 2 (Markov Chain of order m). *Given a set of possible states $S = \{w_1, \ldots, w_N\}$, a Markov Chain of order m is defined by sequence of random variables $W_1, W_2, \ldots$ such that the probability of transition from a state $i-1$ to a state i depends only on the previous m states, i.e., depends on the states $[i-m, i-1]$.*

Then, the expression of equation 2.1 can be approximated by a Markov chain of order m as follows:

$$P(W_1 = w_1, \ldots, W_N = w_N) \approx \prod_{i=1}^{N} P(W_i = w_i | W_{i-m} = w_{i-m}, \ldots, W_{i-1} = w_{i-1}) \tag{2.2}$$

Clearly, the assumption over this approximation has a significant impact on the model since in general it is wrong. However, it significantly decreases the number of parameters to estimate. Furthermore, for situations in which the training data is limited (in size and diversity) it allows for better generalisation.

2.2.1 Variable-Length Sentence Generation

Until now it was assumed that the length of the sequences N was fixed. A common approach allowing models based on equation 2.2 to generate sentences of variable-length, is to set the last word $w_N = STOP$, for each sentence, where $STOP$ is a special symbol such that $STOP \notin \mathcal{V}$.

With the addition of the $STOP$ symbol it is possible to generate sentences of variable-length by generating the word i by sampling iteratively from the distribution:

$$P(W_i = w_i | W_{i-m} = w_{i-m}, \ldots, W_{i-1} = w_{i-1}) \tag{2.3}$$

where a m-order Markov Chain is used, until the $STOP$ symbol is generated.

2.2.2 Unigrams

One of the most simple models for approximating equation 2.1 is the *unigram* or *1-gram* model, also known as *bag-of-words*, which assumes that all words are independent and does not model any kind of sequence. The probability distribution of a sentence under this model is defined as follows:

$$P(W_1 = w_1, \ldots, W_N = w_N) \approx \prod_{i=1}^{N} P(W_i = w_i) \tag{2.4}$$

Despite its simplicity and its strong underlying assumption these models are extensively used in several NLP tasks, namely in IR. This comes from the fact that unigrams are not only very computationally efficient but are also often enough to capture the topic of a given document [37]. Furthermore, as it will be detailed in section 3.1, in IR usually LMs are estimated for single documents, therefore drastically reducing the amount of training data available, what may not allow the learning of more complex LMs. Yet, for tasks in which it is important to capture/model structure of the language like speech correction, machine translation, and others, these models are not suited.

In unigram models the order of the words from a sentence is completely irrelevant. For a given corpus, this model can be materialised by a *multinomial distribution* over words. Formally, given a corpus D and the collection W of words in sentences $s_j \in D$, the multinomial distribution is given as follows:

$$\begin{aligned} P(D) = P(W) &= \frac{x^D!}{\prod_{t=1}^{|\mathcal{V}|} x_t^D!} \prod_{i=1}^{|\mathcal{V}|} P_W(w_i)^{x_i^D} \\ &\propto \prod_{i=1}^{|\mathcal{V}|} P_W(w_i)^{x_i^D} \end{aligned} \tag{2.5}$$

where each x_i^D is the frequency (or count) of word $w_i \in \mathcal{V}^2$ in D, $x_D = |W| = \sum_{i=1}^{\mathcal{V}} x_i^D$ is the total number of words in D and $P_W(w_i)$ is a function that assigns a probability to the word w_i. The multinomial coefficient is constant for a particular unigram model (bag-of-words) so it can be dropped while evaluating different models for $P_W(w_i)$. In fact, without the multinomial coefficient, the expression

on the second step of equation 2.5 is the same as the one from equation 2.4, i.e., the unigram distribution expression given that $W = (w_1, \ldots, w_i, \ldots, w_{|W|})$.

2.2.3 *N*-gram Maximum Likelihood Estimates

Given an n-gram model, a first approach to estimate the parameters of the distribution specified by each model is to find the *Maximum Likelihood Estimates* (MLE) of those parameters.

A multinomial distribution over the n-grams, that models the probability of a given count of each n-gram in a corpus D with N n-grams, is assumed. Based on this, let $c(w_1, \ldots, w_i, \ldots, w_n)$ be the number of times that the n-gram $(w_1, \ldots, w_i, \ldots, w_n)$ appears in D. Furthermore, for a n-gram model we have a Markov chain of order n, which means that the probability distribution of a sentence is decomposed by a product of probability expressions of the form $P(W_i = w_i | W_{i-n} = w_{i-n}, \ldots, W_{i-1} = w_{i-1})$. From this, we have that the parameters vector $\boldsymbol{p}$ is defined as:

$$\begin{aligned} \boldsymbol{p} = (&P(W_1 = w_1 | W_{1-n} = w_{1-n}, \ldots, W_0 = w_0), \ \ldots, \\ &P(W_k = w_k | W_{k-n} = w_{k-n}, \ldots, W_{k-1} = w_{k-1}), \ \ldots, \\ &P(W_N = w_N | W_{N-n} = w_{N-n}, \ldots, W_{N-1} = w_{N-1})) \end{aligned} \tag{2.6}$$

i.e., a parameter for the probability of each k n-gram. For simplicity and readability purposes we simply denote the parameters vector as $\boldsymbol{p} = (p_1, \ldots, p_k, \ldots, p_N)$. Additionally, we define $x_k = c(w_1, \ldots, w_i, \ldots, w_n)$, with $x_k \geq 0$, to be the number of occurrences of the n-gram $(w_1, \ldots, w_i, \ldots, w_n)$, and $x = \sum_{k=1}^{N} x_k$ (the total number of n-gram occurrences). The multinomial probability distribution function is then defined as follows:

$$f(x_1, \ldots, x_k, \ldots, x_N; p_1, \ldots, p_k, \ldots, p_N) = \frac{x!}{\prod_{k=1}^{N} x_k!} \prod_{k=1}^{N} p_k^{x_k} \tag{2.7}$$

The rationale is to find the parameters $\boldsymbol{p}$ that best describe the data. This is equivalent to find the parameters that maximise the likelihood $\ell(\boldsymbol{p})$. First, since the function $log(x)$ is monotone increasing in the interval $]0, +\infty[$, then maximising $\ell(\boldsymbol{p})$ is equivalent to maximising $log\ \ell(\boldsymbol{p})$:

$$log\ \ell(\boldsymbol{p}) = f(x_1, \ldots, x_k, \ldots, x_N; \boldsymbol{p}) \tag{2.8}$$

Plugging the expression from equation 2.7 into equation 2.8 and simplifying

[2]The order is irrelevant. Combinations of a given collection of words will have the same probability.

the terms:

$$\begin{aligned} log\ \ell(\boldsymbol{p}) &= log\left(\frac{x!}{\prod_{k=1}^{N} x_k!}\prod_{k=1}^{N} p_k^{x_k}\right) \\ &= log\ x! - log\prod_{k=1}^{N} x_k! + log\prod_{k=1}^{N} p_k^{x_k} \\ &= log\ x! - \sum_{k=1}^{N} log\ x_k! + \sum_{k=1}^{N} x_k log\ p_k \\ &= K + \sum_{k=1}^{N} x_k log\ p_k \end{aligned} \tag{2.9}$$

where K is a constant (independent of parameters $\boldsymbol{p}$) that can be ignored in the maximisation. In order to find the MLE of the parameters $\boldsymbol{p}$, the likelihood function must be maximised taking into account the constraint $\sum_{k=1}^{N} p_k = 1$. This constrained optimisation problem can be solved by using Lagrange Multipliers. A Lagrange Multiplier for constraint $\sum_{k=1}^{N} p_k = 1$ is introduced in the expression of equation 2.9 and the expression to be maximised, the Lagrangian, becomes:

$$L(\boldsymbol{p}, \lambda) = \sum_{k=1}^{N} x_k log\ p_k + \lambda(1 - \sum_{k=1}^{N} p_k) \tag{2.10}$$

From Lagrange multipliers theory, the solution to the problem of maximising $L(\boldsymbol{p}, \lambda)$ must satisfy $\frac{\partial L(\boldsymbol{p},\lambda)}{\partial p_k} = 0$, for each $p_k \in \boldsymbol{p}$. Computing the partial derivative in order to p_k:

$$\frac{\partial L(\boldsymbol{p}, \lambda)}{\partial p_k} = \frac{x_k}{p_k} - \lambda \tag{2.11}$$

Setting the expression of equation 2.11 to zero one obtains:

$$\frac{\partial L(\boldsymbol{p}, \lambda)}{\partial p_k} = 0 \Leftrightarrow \frac{x_k}{p_k} - \lambda = 0 \Leftrightarrow p_k = \frac{x_k}{\lambda} \tag{2.12}$$

Given that $x = \sum_{k=1}^{N} x_k$ and $\sum_{k=1}^{N} p_k = 1$, the value of λ can be achieved by summing through all the p_k's and x_k's, in the left and right side, respectively, of the equation 2.12 as follows:

$$\sum_{k=1}^{N} p_k = \frac{\sum_{k=1}^{N} x_k}{\lambda} \Leftrightarrow 1 = \frac{x}{\lambda} \Leftrightarrow \lambda = x \tag{2.13}$$

Then, the parameters vector estimate $\hat{\boldsymbol{p}}$ is:

$$\hat{\boldsymbol{p}} = \left(\frac{p_1}{x}, \ldots, \frac{p_k}{x}, \ldots, \frac{p_N}{x}\right) \tag{2.14}$$

Therefore, the MLE estimate for each n-gram probability are the relative counts of that n-gram:

$$P(W_i = w_i | W_{i-n} = w_{i-n}, \ldots, W_{i-1} = w_{i-1}) = \frac{c(w_{i-n}, \ldots, w_{i-1}, w_i)}{\sum_{k=1}^{N} c(w_{k-n}, \ldots, w_{k-1}, w_k)} \tag{2.15}$$

Despite its simplicity, the derived MLE has several problems in practice. It is very likely that a given n-gram does not occur in D. Consequently, the numerator of equation 2.15 will be 0, just as $P(W_i = w_i|W_{i-n} = w_{i-n}, \ldots, W_{i-1} = w_{i-1})$. This is a serious issue due to the fact that since the joint probability of a sentence is defined as a product of these probabilities, if one is 0 all of the others will be cancelled out. It may also happen that for any word $w_k \in \mathcal{V}$, the value of $c(w_{i-n}, \ldots, w_{i-1}, w_k)$, i.e. the count of any *(n-1)*-gram $(w_{i-n}, \ldots, w_{i-1})$, is 0. In this situation, the n-gram probability is not even defined.

An approach to solve these problems is to use *smoothing* techniques which will be detailed in section 3.2.

2.3 Evaluating Models Quality

In order to be able to compare different methods for learning and modelling a LM one needs to have a metric (or several) to evaluate each model quality.

2.3.1 Perplexity

Among proposed metrics for this purpose, a common one is *perplexity* [11], which is based on the *cross-entropy* between two probability distributions. The intuition behind perplexity is that a LM can be evaluated by how well the model predicts sentences (joint probability of the words from equation 2.1) generated using the same true distribution as the ones in the corpus D. Crucially, those sentences must be "held-out", i.e. they must not be part of the training data used to learn the model (known as test data).

Entropy measures the degree of uncertainty of a given probability distribution p [12], and is defined as follows:

$$H(p) = -\sum_{w} p(w) log_b\ p(w) \tag{2.16}$$

When the base of the logarithm b is 2, the units of the entropy are regarded as bits. This comes from the fact that entropy is related to how many bits are necessary to encode messages that must travel across a communication channel, from a given source. From now on, it will be considered that $b = 2$.

In the context of LM, and given that the objective is to learn a probability distribution $P(s_j)$ assigning a probability to each sentence $s_j = (w_1^j, \ldots, w_N^j) \in D$, the entropy of $P(s_j)$, also known as *entropy rate*, can be computed by:

$$\begin{aligned}&\frac{1}{N} H(P(s_j)) = \\ &-\frac{1}{N} \sum_{s_j : s_j \in D, |s_j| = N} P(W_1 = w_1^j, \ldots, W_N = w_N^j) log_2\ P(W_1 = w_1^j, \ldots, W_N = w_N^j)\end{aligned} \tag{2.17}$$

By computing the rate, this expression already addresses the issue caused by longer sentences being less likely, otherwise the entropy of $P(s_j)$ would be dependent of the sentences length N.

Ideally we would compute the *true entropy* H_{true} of the language that the LM is trying to model, based on the true probability distribution $P_{true}(s_j)$ and

compare it with the LM learnt. To achieve this, the entropy expression 2.17 must be evaluated over an infinite sentence (infinite number of words):

$$\begin{aligned} H_{true} = \lim_{N\to\infty} -\frac{1}{N} H(P_{true}(s_j)) = \\ \lim_{N\to\infty} -\frac{1}{N} \sum_{s_j : s_j \in D, |s_j|=N} P_{true}(s_j) log_2\ P_{true}(s_j) \end{aligned} \tag{2.18}$$

It happens that this summation is over all possible sentences. By assuming that the source is ergodic [12], the summation can be discarded and the expression simplifies to:

$$H_{true} = \lim_{N\to\infty} -\frac{1}{N} log_2\ P_{true}(s_j) \tag{2.19}$$

where as on equation 2.18, $|s_j| = N$. Based on the ergodicity assumption, given a large enough N, the previous equation can be approximated with:

$$H_{true} \approx -\frac{1}{N} log_2\ P_{true}(s_j) \tag{2.20}$$

Given the true probability distribution $P_{true}(s_j)$ and the LM estimated distribution $P(s_j)$ we are interested in comparing both. To this end, one can use the *cross-entropy* $H(P_{true}(s_j), P(s_j))$, which is defined as:

$$H(P_{true}(s_j), P(s_j)) = \sum_{s_j \in D} P_{true}(s_j) log_2\ P(s_j) \tag{2.21}$$

It is worth noting that cross-entropy is not a symmetric function[3]. Since $P_{true}(s_j)$ is not known we can only use $P(s_j)$. By replacing $P_{true}(s_j)$ on equation 2.19 by $P(s_j)$ we have:

$$\begin{aligned} H_{true} &= \lim_{N\to\infty} -\frac{1}{N} log_2\ P(s_j) \\ &= H(P_{true}(s_j), P(s_j)) \end{aligned} \tag{2.22}$$

This corresponds to the expression of the cross-entropy $H(P_{true}(s_j), P(s_j))$. Now, the approximation performed on equation 2.20 can be applied for further simplification:

$$H(P_{true}(s_j), P(s_j)) = -\frac{1}{N} log_2\ P(s_j) \tag{2.23}$$

Finally, the perplexity metric is defined as:

$$PP = 2^{H(P_{true}(s_j), P(s_j)))} \tag{2.24}$$

where $H(P_{true}(s_j), P(s_j))$ is defined by equation 2.22. The lower the perplexity the better the model. Perplexity is lower bounded by $H(P_{true})$. Therefore, a good LM has a cross-entropy close to $H(P_{true})$. Intuitively, the LM should assign higher probabilities to the most likely sentences, and not waste probability mass on less likely (or even impossible) ones.

[3]This arises from the fact that cross-entropy expression is based on the Kullback–Leibler divergence, which is also not symmetric by itself.

2.3.2 Task-specific metrics

Despite of its wide adoption, for some tasks the perplexity metric does not allow one to know how good the model is for a specific task. Therefore, an alternative approach consists of using metrics specific for a given task. For instance, in the task of speech recognition, which consists of predicting the most likely sequence of words for a given acoustic signal, the *word error rate* (WER), which measures how much does the predicted words differ from the actual sentence, can be used to evaluate the LM. This is based on counting the number of substitutions, deletions and insertions needed to achieve the actual sentence from the predicted one, and then normalising. The same applies for the task of machine translation, and so on.

2.3.3 Sentence Generation metrics

For sentence generation several alternative metrics have been proposed and adopted. The basic objective of these metrics is to provide an automated evaluation metric correlated with human's evaluations of quality.

A common metric is the Bilingual Evaluation Understudy (BLEU) [52]. This metric was proposed to evaluate the quality of machine translated sentences between languages. The BLEU score is obtained by evaluating the precision of overlapping n-grams between the produced sentence and a reference sentence, consequently, it requires good reference sentences. Namely, usually the *precisions* $precision_i$ for $i \in [1, \ldots, n]$, corresponding to the precision regarding i-grams, are computed. Then, the final score is obtained by the expression:

$$\text{BLUE-n} = BP \exp\left(\sum_{i=1}^{n} \lambda_i log\ precision_i\right) \tag{2.25}$$

where each λ_i are weights that can be parametrised and BP is a brevity penalty defined as follows:

$$BP = min\left(1, \frac{produced\_sentence\_length}{reference\_sentence\_length}\right) \tag{2.26}$$

The objective of the brevity penalty BP is to penalise short sentences/translations. The weights λ_i are often set to 1, which simplifies equation 2.25 to:

$$\text{BLUE-n} = BP \prod_{i=1}^{n} precision_i \tag{2.27}$$

The BLEU score is often computed over the entire test set, instead of on an isolated sentence. Sentence generation is not deterministic in the sense that different sentences may have the same meaning, and therefore be correct. To account for variability in sentence production, the BLEU metric is extended using *Multiple Reference Translations* (or multiple reference sentences, in the context of the sentence generation task). These approach introduces two modifications:

1. n-grams $precision_i$ is computed such that matches in any of the references are allowed;
2. For BP, the closest reference (the one that matches the most in terms of precision) length is used.

Despite its wide adoption, some issues have been reported. Concretely, it has been shown by Callison-Burch et al. [8] that BLEU does not always correlate with human evaluations as normally expected, and that it allows a considerable amount of variation in the produced sentences which by itself does not reflect genuine improvements in produced sentences quality. The authors also observed that some sentences with low BLEU scores had better quality, based on human evaluations. In [63] the authors also observed low correlation between the BLEU metric and human evaluations, for the task of sentence generation for describing images.

To determine if the complete meaning is captured by a generated sentence, the most relevant metric is the *recall*. However, the BLEU metric is strongly focused on precision and not on recall. Thus, the METEOR metric [31], which places a stronger emphasis on recall and addresses other aspects, was proposed. This metric is based on the harmonic mean of 1-gram precision and recall, with recall having a higher weight. METEOR works by computing a set of mappings between 1-grams (*alignment*) for a pair of sentences, and chooses the one (to compute the precision/recall) that has less intersections given the order of the words.

To avoid marking as a mistake words that give a sentence the same meaning (e.g. variations of a noun versus an adjective), words are stemmed (i.e. reduced to its stem). Additionally, synonyms are also taken into account. This last feature implementation uses the WordNet [45] ontology, a popular ontology of English words, to find semantically closed words, given a reference 1-gram.

In the BLEU metric short sentences are penalised, in the METEOR metric the amount of not adjacent mappings between the two sentences is penalised. In practice, this last penalisation promotes sentences that have matching n-grams, for high values of n. The METEOR metric offers higher correlation with human evaluations than the BLEU metric but computing the METEOR scores is more computationally expensive.

Several other common metrics have been proposed. The *Recall Oriented Understudy of Gisting Evaluation* [65] (Rouge) like METEOR focus on evaluating the n-grams recall and is very popular for summarisation evaluation. For the specific task of generating sentences (descriptions) for images, the *Consensus-based Image Description Evaluation* [62] (CIDEr) was proposed. The problem of evaluating image descriptions lies in the inherent subjectivity of visual interpretations. To address this issue, this metric captures consensus between a set of human evaluations, i.e., it measures the similarity of a candidate sentence to a

majority of how most people described a given image.

Chapter 3

Traditional Approaches and Techniques

Having introduced statistical language modelling in the previous chapter, common aspects regarding LMs will now be discussed. The general LM approach for information retrieval will be presented. Despite its specificity towards IR, it is an interesting approach that can be generalised under similar assumptions. Then traditional methods and techniques, crucial to develop effective LMs, will be presented and discussed.

3.1 Query Likelihood

The LM approach to IR is referred as *query likelihood* [55]. Given a set of documents $\mathcal{D}$ and a user query q, this consists of, estimating a LM_d for each document d, in fact d is the corpus D, and then by ranking each document by the likelihood of q sentence according to LM_d, i.e., by the distribution $P(d|q)$. As in the previous section, $N = |\mathcal{V}|$. The distribution $P(d|q)$ can be expanded using Bayes rule as follows:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \tag{3.1}$$

where due to the fact that $P(q)$ is the same for all documents $d \in \mathcal{D}$, it is ignored. Additionally, the prior $P(d)$ is also usually ignored by assuming a uniform distribution across all documents, although it can be used to include additional features in the model (e.g. document length), relevant for IR. Thus, the LM approach for IR reduces to the problem of estimating $P(q|d)$. Concretely, under the LM approach, we are interested in estimating the distribution $P(q|\text{LM}_d)$, which can be interpreted as the probability of q being generated from the language model LM_d.

Initially a multiple Bernoulli model over $P(q|\text{LM}_d)$ was considered [67]. However, the Bernoulli distribution does not model directly the counts of terms. Later, a model with a multinomial distribution over $P(q|\text{LM}_d)$ was proposed [19] yielding better results, which naturally incorporates the counts. This last approach consists of a *1*-gram language model, as discussed in section 2.2.2.

Then, the probability distribution $P(q|\mathrm{LM}_d)$ is defined as follows:

$$P(q|\mathrm{LM}_d) = \frac{x!}{\prod_{k=1}^{N} x_k!} \prod_{k=1}^{N} P(w_k|\mathrm{LM}_d)^{x_k} \tag{3.2}$$

where x_k is defined as in section 2.2.3, it denotes the frequency of word $w_k \in \mathcal{V}$ in the document d, and x denotes the sum of the occurrences of all the words.

The distribution of equation 3.2 is computed for all the documents $d \in \mathcal{D}$, yielding the likelihood of q given each LM_d, which by itself consists of the document's ranking. Since the q is the same for all the documents, the multinomial coefficient can be ignored.

Finally, for estimating $P(w_k|\mathrm{LM}_d)$ the maximum likelihood estimator derived in section 2.15 can be used. As already mentioned, this estimator has problems due to the fact that some *1*-grams may not appear in a given document. In the next section we discuss techniques based on *smoothing* that not only mitigate this problem but also introduce additional features relevant to IR.

3.2 Smoothing

Smoothing consists of adjusting the maximum likelihood estimators of a language model to produce more accurate probabilities [9, 68]. This is important to prevent estimated distributions of a LM from assigning zero probability to n-grams. Moreover, smoothing methods attempt to improve model's effectiveness by incorporating non-linguistical features. For situations in which the amount of text is limited, smoothing plays a crucial role in improving the LMs estimations.

Several smoothing techniques have been proposed. Chen and Goodman [9] surveyed common smoothing techniques, and performed an extensive overview of these techniques, describing and presenting each technique characteristics. The performance of each technique is also assessed. Additionally, a detailed study of smoothing techniques applied to IR was performed by Zhai and Lafferty [68]. In this work we describe briefly the most common methods and highlight the best performing ones. A special emphasis will be given to *Dirichlet* smoothing, due to its Bayesian approach.

When presenting the different smoothing techniques, we will take into consideration the maximum likelihood estimator for a n-gram model from equation 2.15. For reading purposes, we define $P_i^n = P(W_i = w_i|W_{i-n} = w_{i-n}, \ldots, W_{i-1} = w_{i-1})$, where n is the order of the n-gram.

3.2.1 Additive Smoothing

Additive smoothing [9] is a simple method which consists of simulating that each n-gram occurred δ times more than it actually does:

$$P_i^n = \frac{\delta + x_i}{\delta|\mathcal{V}| + \sum_{k=1}^{N} x_k} \tag{3.3}$$

where usually $0 < \delta \leq 1$. The special case of $\delta = 1$ is referred as Laplacian smoothing. Despite its simplicity, this method performance is very poor [15].

3.2.2 Jelinek-Mercer Smoothing

The *Jelinek-Mercer* smoothing [22] (JMS) is based on the interpolation of different LMs. Concretely, a general expression for interpolation-based smoothing is defined as:

$$P_i^n = \lambda P_a^n + (1 - \lambda) P_b^m \tag{3.4}$$

where P_a^n is a LM for n-grams and P_b^m is also a LM, but for m-grams, i.e., m can be greater or smaller than n. Moreover, P_b^m can also be a n-gram LM but learnt from a different corpus (e.g. the entire collection). A parameter λ, where $0 \leq \lambda \leq 1$, is used to control the influence of each LM.

The idea of combining higher order with lower order n-grams comes from the fact that in a limited data situation, in which data may not be sufficient to estimate a higher order model, the lower order model can provide valuable information. In [7] an approach based on this last idea is presented by defining recursively the interpolation expression as follows:

$$P_{interp_i}(n) = \lambda_n P_a^n + (1 - \lambda_n) P_{interp_i}(n - 1) \tag{3.5}$$

where P_a^n is instantiated as the n-gram MLE expression. The recursion can be stopped by defining $P_{interp_i}(1) = P_a^1$ or by defining the smoothed 0-gram model as the uniform distribution over the words in the vocabulary, i.e., $P_a^0 = \frac{1}{|\mathcal{V}|}$.

In order to learn the value of the λ parameter one should use different data from the one used to estimate the MLE. One approach consists of reserving a fragment of the data for each learning task (*Held-out* interpolation) or by rotating different fragments of the data for each learning task and taking the average of the results (*Deleted Interpolation*) [22].

Given the amount of data available, it may not be enough to train all the parameters involved (each recursive call of equation 3.5 has one parameter). Jelinek and Mercer [22] proposed splitting the several λ parameters in a given number of partitions or *buckets* such that all the parameters in the same bucket have the same value. The authors suggest to place together parameters which one believes that should have similar values. Several strategies for distributing the parameters have been proposed, which are based on the counts [9].

3.2.3 Good Turing Estimate and Katz Smoothing

The *Good Turing Estimate* [16] (GTE) states that for a given n-gram occurring r times, it should be treated as if it occurred r^* where

$$r^* = (r + 1)\frac{n_{r+1}}{n_r} \tag{3.6}$$

where n_r denotes the number of n-grams occurring exactly r times. In order to obtain a probability for each n-gram, the equation 3.6 is normalised by $N = \sum_{r=0}^{\infty} n_r r^*$. Thus, for a n-gram w_i with $x_i = r$:

$$P_i^n = \frac{r^*}{N} \tag{3.7}$$

Namely, N is equal to the original number of counts in the distribution. The complete derivation of equation 3.7 is presented in [9].

The idea of GTE is to reallocate probability mass from n-grams that occur $r+1$ times to n-grams that occur r times. It is worth noting that GTE cannot be used for cases in which $n_r = 0$, otherwise the expression from equation 3.6 is undefined. Therefore, a smoothing method is required to smooth cases in which $n_r = 0$. Notwithstanding, GTE is used in other smoothing techniques, namely in *Katz Smoothing*.

Katz Smoothing [24] (KS) is a smoothing technique based on GTE, that extends it to accommodate the combination of higher-order with lower-order models. It is characteristed as a *back-off* method since it fallsback to lower-order models to capture reliable information.

Given a n-gram w_k^n with count $r = x_k^n$ where n is the n-gram order, KS defines a correct count $c_{katz}(w_k^n)$ as:

$$c_{katz}(w_k^n) = \begin{cases} d_r r & , \ r > 0 \\ \alpha(w_k^{n-1}) P_{katz}(w_k^{n-1}) & , \ r = 0 \end{cases} \tag{3.8}$$

where d_r is a *discount ratio* applied to all n-grams with nonzero count r. The probability distribution $P_{katz}(w_k^n)$ is defined by normalising the $c_{katz}(w_k^n)$ expression:

$$P_k^n = P_{katz}(w_k^n) = \frac{c_{katz}(w_k^n)}{\sum_{w_j^n} c_{katz}(w_j^n)} \tag{3.9}$$

For w_k^1 (1-grams) the distribution $P_{katz}(w_k^1)$ is the MLE for the 1-gram model, ending the recursion. Thus, the definition of KS is very similar to JMS.

The value of $\alpha(w_k^{n-1})$ is defined such that the total number of counts in the P_{katz} distribution is unchanged, more formally, $\sum_{w_k^n} c_{katz}(w_k^n) = \sum_{w_k^n} x_k^n$, and it is defined as follows:

$$\alpha(w_k^{n-1}) = \frac{1 - \sum_{w_k^n : x_k^n > 0} P_{katz}(w_k^n)}{\sum_{w_k^n : x_k^n = 0} P_{katz}(w_k^{n-1})} \tag{3.10}$$

The value of d_r is derived from the GTE and is defined as:

$$d_r = \begin{cases} 1 & , \ r > m \\ \frac{\frac{r^*}{r} - \frac{(m+1) n_{m+1}}{n_{r-1}}}{1 - \frac{(k+1) n_{m+1}}{n_{r-1}}} & , \ r \leq m \end{cases} \tag{3.11}$$

where in the original work $m = 5$. Namely, large counts are assumed to be reliable thus, they are not discounted. The idea is that each d_r is chosen in such a way that it is proportional to the discount given by the GTE ($1 - \frac{r^*}{r}$). The full derivation is presented in [9].

3.2.4 Absolute discounting

The *Absolute discounting* smoothing method [50] (ADS), like the JMS, is also an interpolation-based method which combines higher with lower order models. Though, instead of multiplying the higher order MLE by a parameter λ, a fixed discount δ, with $0 \leq \delta \leq 1$, is subtracted:

$$P_i^n = P_{abs_i}(n) = \frac{max\{x_i^n - \delta, 0\}}{\sum_{w_k^n} x_k^n} + (1 - \lambda_n) P_{abs_i}(n-1) \tag{3.12}$$

This last expression is normalised to yield a probability distribution by setting:

$$(1 - \lambda_n) = \frac{\delta}{\sum_{w_k^n} x_k^n} N_{1+}(w_{k-1}^{n-1}\bullet) \tag{3.13}$$

where $N_{1+}(w_{k-1}^{n-1}\bullet) = |\{w_k : x_k^n > 0\}|$, and $\bullet$ evokes a free variable w_k to be iterated. Thus, $N_{1+}(w_{k-1}^{n-1}\bullet)$ denotes the number of unique words that follow the context w_{k-1}^{n-1}.

As in JMS, according to Ney et al. [50], the parameter δ should be set using the *deleted estimation* technique. Moreover, the authors propose the estimate:

$$\delta = \frac{n_1}{n_1 + 2n_2} \tag{3.14}$$

where n_r is defined as in section 3.2.3.

3.2.5 Dirichlet Smoothing

The *Dirichlet Smoothing* method [36, 49] (DS) is based on a Bayesian framework, in which a prior distribution (Dirichlet) is selected and used to smooth a given distribution (e.g. MLE). Based on the intuition that longer corpus allow one to estimate a LM more accurately, the DS method makes the smoothing process dependent on the corpus size.

The DS method consists of a LM (a multinomial distribution) for which the *conjugate prior*[1] for Bayesian analysis is the Dirichlet distribution [49], with $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_i, \ldots, \alpha_N)$ being a parameter vector. Thus, this results in the following expression:

$$P_i^n = \frac{x_i^n + \alpha_i}{\sum_{k=1}^N x_k^n + \alpha_k} \tag{3.15}$$

where $P(w_i^n|\mathcal{D})$ is a LM computed on a larger collection $\mathcal{D}$ (e.g. a set of documents).

This result will now be derived. The notation used for the derivation will be the same as in section 2.2.3. We start by assuming that the prior distribution of the multinomial parameters $P(\boldsymbol{p})$ is Dirichlet.

Then, for a corpus $\mathcal{D}$ with we have that:

$$P(\mathcal{D}|\boldsymbol{p}) = \prod_{i=1}^{N} p_i^{x_i} \tag{3.16}$$

In fact, $P(\mathcal{D})$ is the likelihood of the multinomial model, ignoring the multinomial coefficient. Given that $\sum_{k=1}^N p_i = 1$ and $0 \leq p_i \leq 1$, the Dirichlet distribution has the following probability density function:

$$Dir(\boldsymbol{p}|\boldsymbol{\alpha}) = \frac{1}{B(\alpha)} \prod_{i=1}^{N} p_i^{\alpha_i - 1} \tag{3.17}$$

where $B(\alpha)$ is a normalising constant, expressed in terms of the Gamma distri-

bution. Using the Bayes rule:

$$\begin{aligned} P(\boldsymbol{p}|\mathcal{D}) &= \frac{P(\mathcal{D}|\boldsymbol{p})P(\boldsymbol{p})}{P(\mathcal{D})} \\ &\propto P(\mathcal{D}|\boldsymbol{p})P(\boldsymbol{p}) \\ &= \prod_{i=1}^{N} p_i^{x_i} \frac{1}{B(\alpha)} \prod_{i=1}^{N} p_i^{\alpha_i - 1} \\ &\propto \prod_{i=1}^{N} p_i^{x_i} \prod_{i=1}^{N} p_i^{\alpha_i - 1} \\ &= \prod_{i=1}^{N} p_i^{x_i} p_i^{\alpha_i - 1} \\ &= \prod_{i=1}^{N} p_i^{\alpha_i - 1 + x_i} \\ &= Dir(\boldsymbol{p}|\alpha_1 + x_1, \ldots, \alpha_N + x_N) \end{aligned} \tag{3.18}$$

Therefore, we have showed that the posterior $P(\boldsymbol{p}|\mathcal{D})$ is Dirichlet, which in turn proves that the Dirichlet is the conjugate prior of the Multinomial distribution.

What remains now is to compute the posterior predictive distribution for a single occurring n-gram w_i^n. This can be achieved by marginalising the distribution $P(W = w_i^n|\mathcal{D})$ over the parameters $\boldsymbol{p}$ as follows:

$$\begin{aligned} P(W = w_i^n|\mathcal{D}) &= \int P(W = w_i^n, \boldsymbol{p}|\mathcal{D}) d\boldsymbol{p} \\ &= \int P(W = w_i^n|\boldsymbol{p}) P(\boldsymbol{p}|\mathcal{D}) d\boldsymbol{p} \\ &= \int_{p_i} \int_{\boldsymbol{p_{!i}}} P(W = w_i^n|p_i) P(p_i, \boldsymbol{p_{!i}}|\mathcal{D}) d\boldsymbol{p_{!i}} dp_i \\ &= \int_{p_i} P(W = w_i^n|p_i) \left[\int_{\boldsymbol{p_{!i}}} P(p_i, \boldsymbol{p_{!i}}|\mathcal{D}) d\boldsymbol{p_{!i}} \right] dp_i \\ &= \int_{p_i} p_i P(p_i|\mathcal{D}) dp_i = \mathbb{E}[p_i|\mathcal{D}] = \frac{\alpha_i + x_i}{\sum_{k=1}^{N} (\alpha_k + x_k)} \end{aligned} \tag{3.19}$$

where $\boldsymbol{p_{!i}}$ denotes the vector of parameters such that each element $p_j \in \boldsymbol{p} \setminus \{p_i\}$ and the fact that parameters are independent of each other was used. Additionally, each w_i^n is dependent only on the parameter p_i. The inner integral on the penultimate step is replaced by $P(p_i|\mathcal{D})$ by the marginalisation rule, i.e., in the previous step, the distribution $P(p_i|\mathcal{D})$ had been marginalised by $\boldsymbol{p_{!i}}$.

3.2.6 Other Smoothing Methods and Comparisons

The list of presented methods is by no means exhaustive. From the methods that were not covered, it is worth mention the *Kneser-Ney Smoothing* [28], which is an extension of ADS. A variation of this last method, the *Modified Kneser-Ney*

[1] If the prior and the posterior have the same form, then the prior is referred as the *conjugate prior* for the corresponding likelihood [49].

Smoothing (MKNS), was proposed in [9], based on an improved discounting scheme.

An extensive experimental analysis of smoothing methods was performed in [9]. A first observation is that the MKNS method outperformed all the methods discussed in the previous sections and also some other methods not discussed here. The JMS and KM have shown consistent effectiveness across the experiments. As for ADS, it perform very poorly.

Zhai and Lafferty [68] analysed experimentally JMS, ADS and DS, in a information retrieval setting. For short sentences (*title queries*) DS outperformed the other three methods (significantly in some experiments) in terms of precision, with ADS outperforming JMS. For longer sentences JMS outperformed both DS and ADS methods. However, it was only marginally better than DS.

Another aspect discussed in this work is the computational efficiency of each method. For IR it is crucial that smoothing methods are fast, in order to be able to evaluate large collections quickly. Methods like KS and MKNS are more computationally expensive than ADS, DS and JMS. This is mainly due to the necessity of counting words with the same frequency in documents.

3.3 Discriminative Approaches

The LM formulation from section 2.1 follows a generative approach since it models the joint probability of the words in a sentence. An alternative approach is the discriminative approach in which instead of learning the same joint probability, boundaries separating data are learnt.

An advantage of this methods is that, given a specific application and under a supervised learning setting, one can directly optimise the error-rate, instead of attempting to learn the data distribution. An example of this approach in speech is presented in [58], in which the model is trained using acoustic sequences with their transcriptions. The fact that the approach is supervised is a drawback, due to a possible lack of annotated data.

Generative models are task-independent, since they focus on maximising the likelihood of the sentences from a given corpus. However, taking into account the implicit noise or data *uncleanness* (e.g. the existence of words with the same meaning), this may be sub-optimal for some tasks. Following this reasoning, in [33] a discriminative approach for machine translation is proposed. The perceptron (concretely, the Averaged Perceptron algorithm) discriminative classifier is trained with data in which reference translation sentences are available. It was observed that the discriminative was able to improve over a strong baseline.

One of the justifications for this effectiveness in specific tasks is that the models are trained to minimise directly the error inherent to that task. On the other hand, generative classifiers are task-independent and make explicit assumptions, while usually no assumptions are made (despite of the selected model) in the discriminative approach. If within the data at hand, a considerable amount of dependencies exist, due to the (sometimes hard) assumptions of generative models, they may not be able to capture those dependencies, deteriorating their performance.

Another aspect that is worth noting is that the number of parameters needed to be estimated for discriminative approaches is less than the generative counterpart. This has also a direct implication in the convergence rate of the

learning procedure [51].

Chapter 4

Neural Language Models

A major issue of statistical language modelling in general is the *curse of dimensionality*, i.e., the number of parameters involved when modelling the joint probability of the words of a sentence, given that realistic vocabularies are large, is huge[1]. *Neural Language models* (NLMs) are language models based on neural networks, that overcome this problem. To simplify the notation, the *bias* term used in neural networks models is omitted.

4.1 Neural Probabilistic Language model

Bengio et al. [4] proposed to learn distributed words representations (also known as *embeddings*), which are able to represent an exponential number of different words, such that similar words will have close representations (according to a given distance function), allowing for better generalisation over the high number of possible sentences in natural language. More formally, given a vocabulary $\mathcal{V}$ with dimension $|\mathcal{V}|$, each word will be represented as a real-valued vector in $\mathbb{R}^m$, in which in principle $m \ll |\mathcal{V}|$. The joint probability function $P(W_1 = w_1, \ldots, W_i = w_i, \ldots, W_N = w_N)$ (equation 2.1) of the sentence $s_k = (w_1, \ldots, w_i, \ldots, w_N)$ is then expressed in terms of the words w_i representations. Each conditional probability of equation 2.1 is learned with a *Multilayer Perceptron* [2] (MLP), which consists of a fully connected feedfoward artificial neural nework. On the architecture proposed by Bengio et al. [4] the word representations and the conditional probability of each word given the previous ones are learned simultaneously. This approach was named *Neural Probabilistic Language model* (NP-LM).

For every $s_k = (w_1, \ldots, w_i, \ldots, w_N)$, the objective is to learn a function:

$$f(s_k) = f(w_N, \ldots, w_i, \ldots, w_1) = P(W_i = w_i | W_{i-1} = w_{i-1}, \ldots, W_1 = w_1) \tag{4.1}$$

The function f is decomposed in two parts:

- A mapping $\phi(w_i) \in \mathbb{R}^m$ for every word $w_i \in \mathcal{V}$. Given all the words in $\mathcal{V}$, C is a $|\mathcal{V}| \times m$ parameter matrix such that $\phi(w_i) = C_i$ (the row i of C).

[1] Given a sentence of length 6 and a vocabulary size of 100,000, there are potentially $100000^6 - 1 = 10^30 - 1$ free parameters.

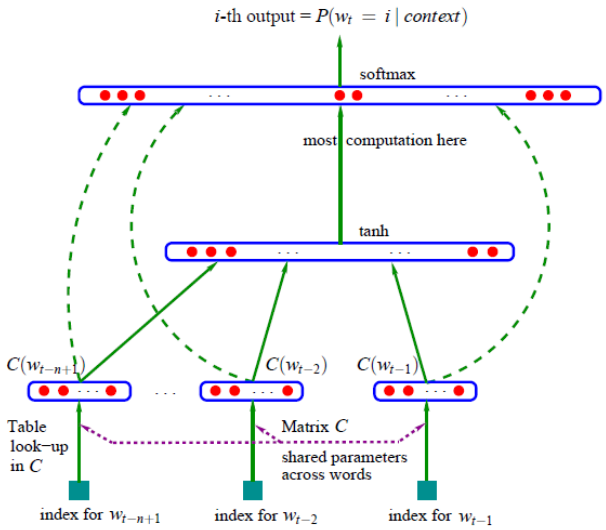

Figure 4.1: Neural Probabilistic Language Model Architecture. Source [4]

- A function g, taking as input the word representations $\phi(w_i)$ for each word w_i, and mapping the inputs to the conditional probability $P(W_i = w_i|W_{i-1} = w_{i-1}, \ldots, W_1 = w_1)$, for each $w_i \in \mathcal{V}$. Thus, the output of g is a vector whose i-th element corresponds to the probability $P(W_i = w_i|W_{i-1} = w_{i-1}, \ldots, W_1 = w_1)$.

In [4] the function g is materialised by a MLP. The complete architecture is depicted in figure 4.1. It includes one hidden-layer with all neurons using the *tanh* activation function. The output layer of the architecture uses the *softmax* activation function. The *softmax* is defined as:

$$softmax(y) = \frac{e^{y_i}}{\sum_{j=1}^{|\mathcal{V}|} e^{y_j}} \tag{4.2}$$

where y_i denotes the unnormalised log-probabilities of word i. The *softmax* enforces the constraint that outputs must lie between 0 and 1, and the sum of all output values is equal to 1, allowing the outputs of the network to be interpreted as probabilities. The *softmax* implements the probability distribution $P(W_i = w_i|W_{i-1} = w_{i-1}, \ldots, W_1 = w_1)$. Moreover, it should be noted that $softmax(y)$ assigns to every word, a probability greater than 0, for every y, which means that the probability distributions are already smoothed.

The parameters involved correspond to the parameters of the matrix C and to the parameters ω of the network. The network is trained using the *Backpropagation Algorithm* [2], and gradient-based methods, to maximise the penalised log-likelihood[2]:

$$L = \frac{1}{|\mathcal{V}|} \sum_{s_k} log\ f(s_k; C, \omega) + R(C, \omega) \tag{4.3}$$

where $R(C, \omega)$ is a regularisation term. A crucial property of this model is that

the number of parameters scales *linearly* with $\mathcal{V}$. A limitation of this model is that the length of the sentences (length of the input) must be fixed.

The authors evaluated the perplexity of the model just described, on a large dataset (16 million words and $|\mathcal{V}| = 17964$), and compared it with n-gram (concretely, 5-gram) models using the JMS and the MKNS, both discussed in section 3.2. The neural probabilistic language model outperformed all the n-grams in two different corpus. Although, it takes more time to train the NP-LM.

From an algorithmic perspective, the common Neural Language Model approach consists of the following steps:

1. Represent each word as a real-valued vector/embedding;

2. Train a neural network to learn the distribution $P(W_i = w_i | W_{i-1} = w_{i-1}, \ldots, W_1 = w_1)$, i.e., to predict the next word w_i given its context $(w_{i-1}, \ldots, w_1)$. The context is the set of word embeddings for each word in the context, that the network takes as input;

3. Learn projections $\phi(w_i)$ (by learning the parameters matrix C), that project each word w_i to a distributed representation, and the neural network weights ω jointly, by minimising a given cost function L.

Based on this canonical steps, several architectures and modifications were proposed, improving upon the first neural language model, the *NP-LM*, and introducing very interesting properties. In the next sections we will review some of these works.

The *softmax* function provides a probability distribution for the network outputs with every possible event being assigned a probability ≥ 0. Recently, an alternative function, the *sparsemax*, was proposed by [39], yielding a sparse probability distribution. This function is very interesting due to its sparse characteristic, what allows for filtering out some of the outputs. On the other hand, smoothness is lost. Nevertheless, it is still very promising. A possible application to LMs would be to learn clusters of candidates for each *next-word*, or more generally, candidates for a given context (not necessarily words). Another application is to limit the number of possible candidates, reducing the search space, for the next word to be emitted in a sentence generation task.

Based on the successes of deep learning, i.e., namely models with a large number of hidden layers, which in turn provides them the models the capability of learning more higher-level/abstract representations of the data, Arisoy et al. [1] proposed a deep neural language model. The proposed architecture, DNN LM (deep neural network LM) is based on the NP-LM proposed by Bengio et al. [4] described above, and adds more hidden layers to the architecture. In the experiments performed, it was observed that the DNN LM outperformed both a state-of-the-art n-gram model and a 1 hidden layer model, equivalent to the NP-LM. However, the model performed worse than a recurrent neural network model (discussed in section 4.3). Thus, increasing the depth of the model, in terms of hidden layers, may lead to performance improvements.

[2] Or to minimise the cross-entropy function with a regularisation term.

4.2 Log-Bilinear Language Model

Mnih and Hinton [46] proposed the Log-Bilinear Language model (LBL-LM), which consists of a NLM using a log-bilinear [3] energy function. Namely, given a word w_i and its context words $(w_{i-1}, \ldots, w_1)$, the distributed representation of $\hat{\phi}(w_i)$ of word w_i, is predicted as follows:

$$\hat{\phi}(w_i) = \sum_{j=1}^{i-1} T_j \phi(w_j) \tag{4.4}$$

where $\phi(w_j)$ denotes the embedding of word w_j defined just like for the $NP-LM$ and T_j is a $m \times m$ weight matrix associated with the context position i, transforming each context word representation. Thus, the number of C_i matrices is equal to the context size.

Then, the similarity between $\hat{\phi}(w_i)$ and the embedding of word w_i is computed using the inner product. To obtain a probability distribution over the next word, each similarity is exponentiated and normalised:

$$P(W_i = w_i | W_{i-1} = w_{i-1}, \ldots, W_1 = w_1) = \frac{e^{\left\langle \hat{\phi}(w_i), \phi(w_i) \right\rangle}}{\sum_{j=1}^{i-1} e^{\left\langle \hat{\phi}(w_j), \phi(w_j) \right\rangle}} \tag{4.5}$$

where the *bias* term has been omitted from the argument of the exponential in both the numerator and the denominator. This expression can be simplified by computing the log of the probability distribution:

$$\begin{aligned} &log\ P(W_i = w_i | W_{i-1} = w_{i-1}, \ldots, W_1 = w_1) = \\ &\quad log\ P(W_i = w_i | \phi(w_{i-1}), \ldots, \phi(w_1)) \propto \left\langle \hat{\phi}(w_i), \phi(w_i) \right\rangle \end{aligned} \tag{4.6}$$

In fact, the LBL-LM resembles a neural network with one hidden layer (equation 4.4) with no non-linearity and a *softmax* output layer (equation 4.5).

On the same dataset as the one used for the NP-LM experiments, the LBL-LM outperformed both the NP-LM and a n-gram model with Kneser-Ney smoothing.

However, the LBL-LM model has a large number of parameters. Concretely, given a context of size S and word vector representations of dimension m, each context matrix T_j will have $m * m$ parameters, yielding a total of $S * m * m$ parameters. Moreover, the number of parameters grows linearly as S increases. The LBL-LM has the same limitation as the NP-LM, regarding dealing with variable-length sentences.

An interesting aspect of all the previous analysed NLMs until this point is that the embeddings are learnt syntactically, i.e., they are learnt jointly with a LM which in turn is implicitly focused on the syntactics. Thus, it is likely that given for example the embedding of the words *wonderful* and *terrible*, they will be close (in the vector space). This is justified by the fact that both words have similar syntactic properties [35]. Ideally one wants to capture semantic similarities, i.e., learn a function $\phi(w_i)$ placing words with similar meanings close to each other, in the target vector space. The task described in the previous

[3] A bilinear model is a model that given two arguments, it combines them linearly. A log-bilinear is model is defined by the logarithm of a bilinear model.

sentence is still a research problem for which a wide range of techniques have been proposed [35, 43, 44, 32, 54]. This distributed word representations learnt lie in a vector space in which similar words are close. Usually, this similarity can be computed through *cosine similarity* (measures the cosine of the angle between two vectors) or by a distance metric (e.g. euclidean distance or L^2 norm).

Regarding the LBL-LM, Maas and Ng [35] proposed an extension to tackle the problem just described above by modelling words at the document-level, instead of sentence-level. The hypothesis is that given that a document represents a given topic, the word will be interpreted as belonging to that topic.

4.3 Recurrent Neural Networks

Recurrent neural networks [2, 17] (RNN) are a type of neural network which allows for cycles between connections of the network. Thus, it allows the network to exhibit dynamic temporal behaviour, or in other words, memory capabilities. This memory is responsible for allowing recurrent networks to deal with variable length sequences (e.g. sentences) efficiently. The network memory is created by the network internal state, which in turn is materialised by a matrix of parameters. These parameters are shared across the each *timestep* t[4], i.e., across all of the i-th element of the sequence, bounding the total number of parameters needed to process any sequence.

RNNs are a very interesting model for language modelling. Since they can capture dependencies within elements of a sequence efficiently, they allow one to learn a LM without performing any simplifying assumption to the model (e.g. Markov assumptions). This gives the model unlimited expressiveness. However, as it will be discussed, these networks have problems in retaining long-term dependencies.

Elman [13] introduced the first RNN model, the *simple recurrent network* or *Elman network*. This model resembles a MLP neural network with a directed cycle within a hidden layer, where each cycle iteration is referred as a *timestep*. Therefore, the output of the network at time t depends on the output at time $t-1$. The state of the network is defined by a hidden layer s_t, which in theory summarises all the information about the past. In practice, to implement the recurrence the network is unfolded across timesteps. Figure 4.2 depicts the model architecture and the unfolding procedure.

Formally, given a sequence $x = (x_1, \ldots, x_t, \ldots, x_{|x|})$ of length $|x|$, in each timestep t, the network takes as input the element x_t and produces an output o_t, taking into account the outputs of the timestep $t-1$. The state of the network s_t depends on the input x_t and on the previous state s_{t-1} of the network. Concretely, the model is defined by the following equations:

$$s_t = \sigma_s(Ux_t + W_{s_{t-1}}) \tag{4.7}$$

$$o_t = \sigma_o(Vs_t) \tag{4.8}$$

where U, W and V are weight matrices associated with the input x_t, the internal state, and the output of the network, respectively. Additionally, σ_s and σ_o are activation functions in the network state hidden layer and output layer, respectively. From equations 4.7 and 4.8 it can be verified that the network

[4]The time dimension must be discrete.

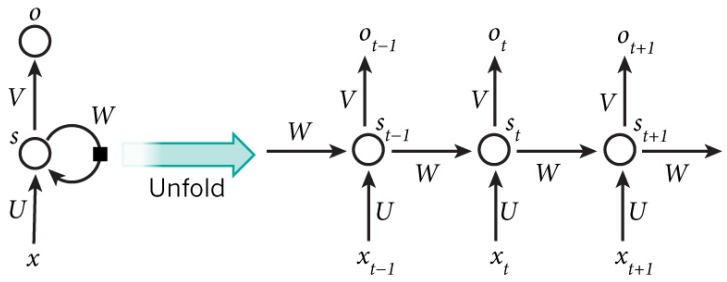

Figure 4.2: Recurrent Neural Network architecture and unfolding procedure. Source [6]

combines the input x_t with the previous state s_{t-1} through a sum operation. Usually σ_s is set as a nonlinear activation function (e.g. tanh, *sigmoid*, etc.) and σ_o as the *softmax* function. To train the network, the *Backpropagation Through time* algorithm [64] (BPTT) is used together with gradient-based techniques. BPTT is similar to the backpropagation algorithm, however the error is propagated across all the previous timesteps.

As already stated, RNNs have problems in retaining dependencies over medium/long-sized sequences due to the vanishing and exploding gradient problems [3, 53]. During the backpropagation phase, the errors are propagated backwards. The error propagation expressions at each timestep are obtained by using the derivative chain rule. As shown in [64] this expression is defined by a product of gradient expressions (w.r.t each timestep). The issue of vanishing and exploding gradients is originated by this product.

For both the tanh and *sigmoid* functions the derivative goes to 0 at both ends. When this occurs, this is referred as a saturated neuron. Thus, since the gradient is very close to 0, it will push gradients from previous layers towards 0. Given this fact, the previously mentioned gradients product will approach 0 exponentially, i.e., the gradients will be vanished after a few timesteps. The exploding gradients is reciprocal, it occurs when the gradient values are large. A possible approach to this last problem is to clip gradients at a certain threshold. For vanishing gradients the solution is not as straightforward.

A possible way to address this problem is to use different activation functions. Namely, the *Rectified Linear Unit* [30] (ReLU) has shown to yield faster training since it does not saturate as the *tanh* and *sigmoid* activation functions, i.e., it has a reduced likelihood of suffering from gradient vanishing. This is due to the fact that derivatives through the rectifier remain large whenever the unit (a neuron from a network layer) is active, unlike *sigmoid* and *tanh*. The ReLU also promotes sparsity within the network activations. However, when the value of the argument of the ReLU function is not positive, the gradient will be 0 and it will not be able to learn. Recently, the *Parametric ReLU* (PReLU) was proposed in [18] which consists in changing the slope of the ReLU function for inputs in $\mathbb{R}_{\leq 0}$, by multiplying it by a coefficient a_i. The coefficient a_i is treated as a learnable parameter. The final expression is:

$$f(x_i) = max(0, x_i) + a_i min(0, x_i) \tag{4.9}$$

With this expression, when the inputs of the function are in $\mathbb{R}_{\leq 0}$, the gradient

will not be 0.

Another approach is to use the truncated BPTT [2], i.e., instead of propagating the error through all the timesteps t truncate the propagation at timestep t_1, such that $t_1 \leq t$. Another approach is to use different RNN architectures. Namely, Long Short-Term Memory [20] (LSTM) networks introduce a set of modifications to the base RNNs architecture alleviating these problems. More recently, Gated Recurrent Units [10] (GRUs) were proposed whose performance is on par with LSTM, while being computationally more efficient.

In general, i.e., for both recurrent and non-recurrent neural networks (NLMs) techniques that reduce the training time have been proposed, namely the *Hierarchical softmax* [48] and *Noise Contrastive Estimation* [47].

4.3.1 RNN Language Models

Mikolov et al. [41] proposed a RNN-based LM for the task of speech recognition. Due to the RNN model characteristics, the proposed LM is able to model variable-length contexts, unlike the NP-LM or the LBL-LM previously described in sections 4.1 and 4.2, respectively. The RNN model was the *simple recurrent network* with σ_s being the *sigmoid* activation function and σ_o the *softmax* function. Given a sentence $s_k = (w_1, \ldots, w_t, \ldots, w_N)$, where N denotes the total length of the sentence as well as the total number of timesteps, the input x_t is the one *1-hot encoding*[5] of word w_t. The network is then trained to output the word w_{t+1}. The cost function used is the cross-entropy. It should be noted that the training algorithm consists of a truncated BPTT, with truncation being performed at timestep t ($t_{=}t$), i.e., weights are updated based on the error computed only for the current timestep.

The probability distribution of the next word given the previous words is computed as follows:

$$P(W_t = w_t | W_{t-1} = w_{t-1}, \ldots, W_1 = w_1) = o_t \tag{4.10}$$

In [41] an optimisation is performed, consisting of modelling the conditional distribution of each word given the preceding words for rare words (rarity is defined by given threshold), as uniform. Thus, for rare words, the output of the network is ignored and a uniform distribution is used. The justification for this optimisation is that the network does not have sufficient training examples to capture the context of rare words. Despite of its effectiveness, this method is somewhat ad-hoc. One could attempt to introduce in the LM, prior knowledge regarding each rare word.

The experiments shown that RNN outperformed significantly state-of-the-art n-gram models, even when being trained with less data. However, no experiments have been carried out comparing the proposed RNN LM with a feedfoward neural network LM.

Following the previous work, Mikolov et al. [42] trained the previous model with BPTT without truncation and compared its effectiveness with neural language models. At first, the authors assessed the influence of the truncation threshold, i.e., the number of timesteps from the current one in which backpropagation is performed. The experiment consisted of training different RNN models

[5] Given a a vocabulary $\mathcal{V}$ of size $|\mathcal{V}|$, the one-hot encoding of the i-th word of the vocabulary is a binary vector $\vec{v} \in \{0,1\}^{|\mathcal{V}|}$ with a 1 in the i-th position and 0's in all the other positions.

(different initialization and number of parameters) over different thresholds. For each threshold the different models were combined through linear interpolation, with the same weight for each model. The authors verified that setting the threshold to 4-5 steps is sufficient, by analysing the perplexity achieved by each configuration. Additionally, the RNN LM trained with $t_1 = 1$ (model a) and with BPTT (model b) with $t_1 > 1$ were compared to a MLP LM. Both model a and b outperformed the MLP LM in terms of perplexity, with model b being considerably better than model a. This observation supports the fact that BPTT does in fact yield better performance.

A different approach was taken by Kim et al. [25] in which the LM is constructed at character-level. The motivation is that approaches that work at the word-level end up without enough data such that models can actually learn the context of rare words. Additionally, word-level approaches are agnostic to morphemes. By working at the character-level and assuming that a given model can learn dependencies over several characters, then both morphemes and rare words can be handled. The model proposed in [25] consists of a RNN LM in which instead of using directly the words as input, the input is a character embedding, obtained from a Character-level Convolutional Neural Network [2] (CNN). Between the RNN and the CNN, a *Highway Network* [60], which essentially consists of a MLP network that adaptively carries some dimensions of the input directly to the output, is used. The proposed character-level performance (perplexity) was on par with the state-of-the-art results on the Peen Treebank (PTB) corpus, with a parameter reduction of $\approx 60\%$. Furthermore, for morphologically rich languages (Arabic, Czech, French, among others) the model outperformed three morphological LMs baselines (also using fewer parameters): a n-gram model with Kneser-Ney smoothing, a morphological LBL-LM [5] and a morphological LSTM (RNN) LM. This indicates that in fact the character-level is an effective approach, specially for morphologically rich languages. Another scenario in which the character-level approach is likely to be effective for LM learning is on social media text. This is due to the fact that social media text has a lot of abbreviations, mistakes and other non-linguistic elements (e.g. emoticons).

4.4 Multimodal Language Models

Multimodal LMs (MLMs) are defined based on elements of different modalities, like text and images for instance. Just like in traditional LM (based only on text), the idea is to capture regularities in and between different modalities. The model ends up learning correlations/alignments between different modalities and this is useful for a wide range of applications. The learning of MLMs is strongly related to the task of learning multimodal vector-spaces [57, 27, 14], as well as mappings from different modalities to vector representations in those spaces, such that semantically similar elements (or simply occurring in similar contexts) lie close to each other. It is worth noting that two general approaches can be used to achieve this purpose. One consists of learning a mapping (or projection) of a given modality embedding to the vector-space of other modality. An alternative, is to map both modalities embeddings to a common, new space, which is at an higher level of abstraction (e.g. semantically).

Although the focus in this work is on text and image modalities, the methods surveyed can be extended to other modalities given that one can obtain a

distributed representation (embedding) for that particular modality. Moreover, the focus is on Deep Learning methods, or more generally, neural network-based models. These methods are trained based on text-image pairs (there can be multiple sentences for an image and vice-versa), which are assumed to be correlated.

One of the first MLMs was recently proposed in [26]. The proposed method is based on LBL model to estimate a LM, and learns jointly embeddings for both images and text. The result is a LM that can be conditioned on images (can be generalised to other modality). The learnt MLM is then used to generate image descriptions, of variable-length. A Modality-Biased Log-Bilinear model (MB-LBL) is proposed which consists of extending the LBL LM model by adding a bias term incorporating a feature vector $\boldsymbol{x} \in \mathbb{R}^F$ of a given modality. This results in the following $\hat{\phi}(w_i)$ expression:

$$\hat{\phi}(w_i) = \sum_{j=1}^{i-1} T_j \phi(w_j) + T_F \boldsymbol{x} \tag{4.11}$$

where T_F is a $m \times F$ context matrix. Then, the LM is probability distribution is the same as the one from equation 4.5, however, the distribution is now also conditioned on the image, i.e., $P(W_i = w_i | W_{i-1} = w_{i-1}, \ldots, W_1 = w_1, \boldsymbol{x})$. The features $\boldsymbol{x}$ are obtained from a Convolutional Neural Network [2]. The proposed model can be used for image retrieval given a textual query, for retrieving text given an image query and to generate image descriptions. It is worth mentioned that the proposed model has the same limitation as the LBL LM model, the context size must be fixed (the authors used context size of 5 in the experiments). The experiments shown that the MB-LBL achieved lower perplexity, compared to a n-gram approach and higher BLEU scores. However, the model is very sensible to the features used for $\boldsymbol{x}$, different features yield significantly different results. An additional model, the Factored 3-way Log-Bilinear Model, was also proposed but was outperformed by the MB-LBL model for sentence generation. A major conclusion of the whole work is that incorporating images into the LM can help improving the performance of the models. This is a somewhat trivial aspect for humans (more "*viewpoints*" of the same element, can help capturing its meaning) but not always for machines.

Several works like [63, 23] follow a similar approaches between them to learn a MLM. It consists of obtaining a image representation vector $\boldsymbol{x}$ from a CNN and then incorporate that vector in a RNN. The RNN is then responsible for learning a LM but conditioned on $\boldsymbol{x}$. Some proposed strategies consist of giving the vector $\boldsymbol{x}$ as input to the RNN before the first word [63] (requires learning a mapping of both modalities into a common space) or, to set the bias term of the input of the RNN to $\boldsymbol{x}$ [23] (first the vector $\boldsymbol{x}$ is mapped by multiplying it by a weights matrix). Both strategies implement a distribution $P(W_i = w_i | W_{i-1} = w_{i-1}, \ldots, W_1 = w_1, I)$ where I denotes an image, and achieve not only very good perplexities (compared to the state-of-the-art at the time) but also very good qualitative results.

Mao et al. [38] proposed an architecture, the Multimodal Recurrent Neural Network (m-RNN), which is composed by three components: a vision component containing a CNN for image representation learning, a textual component containing a RNN that learns a textual LM, and a multimodal component consisting of a hidden layer that connects both the vision and textual components. This

architecture is particularly interesting because the whole model can be trained end-to-end, i.e., the errors can be propagated to each of the components. To train the architecture, the average log-likelihood cost function is used with a regularisation term (L^2 norm of the parameters). The model implements the probability distribution $P(W_i = w_i|W_{i-1} = w_{i-1}, \ldots, W_1 = w_1, I)$. Thus, to generate a sentence, a special start symbol or an arbitrary number of reference words is given to the network. Then, the output of the network i is used for the input of the word $i+1$ together with the image to be described, and this process is repeated until a stop symbol is emitted.

There are a couple aspects that distinguish this architecture. First, instead of learning word embeddings using one hidden layer, two hidden layers are used, what in principle allows for capturing more abstract word embeddings that impact directly the performance of the succeeding components. Unlike other works [23, 63], the RNN is used only for textual data, maximising the capacity of the network to capture dependencies.

On the experiments performed, the vision component weights were initialised using a pre-trained CNN and fixed during training due to a shortage of data. For the task of sentence generation, the model outperformed the state-of-the-art models at the time in terms of perplexity and BLEU scores.

As stated previously, all the works described in this section require pairs of text-image for training. This may be an issue if one wants very good descriptions for simple images, which are both truly correlated, although several quality datasets are available like the Flickr8k [56] which consists of 8000 images each with 5 descriptions, the Flickr30k [66] which is an extension to Flickr8K dataset comprising 31,783 each with 5 descriptions, the MS COCO [34] which consists of 82,783 images for training and 40,504 for validaton each with 5 descriptions, the YFCC-100M [61] dataset containing 100 million images with captions written by Flick users, among others.

It happens that from another perspective, these MLMs allow one to take advantage of all the data available in a wide variety of contexts like social-media, where users usually publish an image with some description, yielding a totally unsupervised method. However, this implies the assumption that correlation between the text and the image is verified (which may be wrong, specially in noisy data like the one present in social-media). Notwithstanding, with appropriate methods to clean this data this is a very promising approach not only for LM

but for capturing the semantics of the data in general.

Chapter 5

Conclusions

Throughout this work we reviewed and surveyed language models, from its foundations and more traditional models, to more complex methods extending and/or solving problems of more traditional models and even achieving *state-of-the-art* performance. The analysis of various methods and techniques was performed always with the task of sentence generation in mind, i.e., given each of the techniques, which characteristics of each method could actually be suitable or prohibitive for that task.

A strong focus was given to neural language models, which lately, together with the successes of deep learning techniques across a wide range of tasks, have achieved impressive results. These models are very interesting since they allow one to achieve a LM free of simplifying assumptions from a theoretical perspective, while also allowing for the development of architectures capable of being trained end-to-end.

We emphasised an application of these models regarding the estimation of LMs based on more than one modality (text and images). Recently several techniques have achieved exciting results that indicate that it is indeed a feasible approach that eventually, can lead us towards a good understanding of different forms of interpreting knowledge. Being able to truly abstract from different forms of knowledge (e.g. text, image, speech) is clearly an ambitious goal. Nevertheless, we believe that the LM approach, due to its properties, highlighted along this work, is very promising. Specially due to the fact that correlations between different modalities can be learned in a unsupervised manner.

A natural next step would be to perform experiments to assess how each LM technique deals with noisy text, namely social-media text (and images). The analysis performed over the works that tackle the task of learning common vector-spaces for different modalities could also be performed in more depth. Namely, it would be valuable to attempt to answer the question of how can this techniques be generalised to support different datasets. Being able to perform such generalisation would allow in principle to achieve much more rich abstractions and accommodate different "points of view" (or contexts) for each word or concept. The same question applies to neural language models.

Another interesting aspect worth researching is the incorporation of context in language models, which already has been explored in some works [40, 21]. This would allow one to condition or to influence the model towards a different

assignment of probability mass to a given word, based on prior knowledge.

Bibliography

[1] E. Arisoy, T. N. Sainath, B. Kingsbury, and B. Ramabhadran. Deep neural network language models. 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*, WLM '12, pages 20–28, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[2] I. G. Y. Bengio and A. Courville. Deep learning. Book in preparation for MIT Press, 2016.

[3] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, Mar. 1994. ISSN 1045-9227.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar. 2003. ISSN 1532-4435.

[5] J. A. Botha and P. Blunsom. Compositional morphology for word representations and language modelling. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1899–1907. JMLR.org, 2014.

[6] D. Britz. Recurrent neural networks tutorial, part 1 – introduction to rnns, 2016. URL `http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/`.

[7] P. F. Brown, V. J. D. Pietra, R. L. Mercer, S. A. D. Pietra, and J. C. Lai. An estimate of an upper bound for the entropy of english. *Comput. Linguist.*, 18(1):31–40, Mar. 1992. ISSN 0891-2017.

[8] C. Callison-Burch, M. Osborne, and P. Koehn. Re-evaluation the role of bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 249–256, 2006.

[9] 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*, ACL '96, pages 310–318, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

[10] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. Technical Report Arxiv report 1412.3555, Université de Montréal, 2014. Presented at the Deep Learning workshop at NIPS2014.

[11] A. Clark, C. Fox, and S. Lappin. *The Handbook of Computational Linguistics and Natural Language Processing*. Wiley-Blackwell, 2010.

[12] T. M. Cover and J. A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006. ISBN 0471241954.

[13] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[14] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. A. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2121–2129. Curran Associates, Inc., 2013.

[15] W. A. Gale and K. W. Church. What's wrong with adding one. In *Corpus-Based Research into Language. Rodolpi*, 1994.

[16] I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3 and 4):237–264, 1953.

[17] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012. ISBN 978-3-642-24796-5.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034, 2015.

[19] D. Hiemstra and W. Kraaij. Twenty-one at trec-7: Ad-hoc and cross-language track. In *In Proc. of Seventh Text REtrieval Conference (TREC-7*, pages 227–238, 1999.

[20] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. ISSN 0899-7667.

[21] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[22] F. Jelinek and R. L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *In Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, Amsterdam, The Netherlands: North-Holland, May 1980.

[23] A. Karpathy and F. Li. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3128–3137, 2015.

[24] S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Singal processing*, volume ASSP-35, pages 400–401, March 1987.

[25] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *AAAI 2016*, 2015.

[26] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Multimodal neural language models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 595–603. JMLR.org, 2014.

[27] R. Kiros, R. Salakhutdinov, and R. S. Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *CoRR*, abs/1411.2539, 2014.

[28] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1, pages 181–184 vol.1, May 1995. doi: 10.1109/ICASSP.1995.479394.

[29] P. Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010. ISBN 0521874157, 9780521874151.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems, NIPS 2012*, pages 1097–1105, 2012.

[31] A. Lavie and M. J. Denkowski. The meteor metric for automatic evaluation of machine translation. *Machine Translation*, 23(2-3):105–115, Sept. 2009. ISSN 0922-6567.

[32] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc., 2014.

[33] Z. Li and S. Khudanpur. Large-scale discriminative n-gram language models for statistical machine translation. In *In Proceedings of AMTA*, 2008.

[34] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, Zürich, 2014.

[35] A. L. Maas and A. Y. Ng. A probabilistic model for semantic word vectors. In *Workshop on Deep Learning and Unsupervised Feature Learning, NIPS*, volume 10, 2010.

[36] D. J. MacKay and L. C. B. Peto. A hierarchical dirichlet language model. *Natural Language Engineering*, 1:1–19, 1994.

[37] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

[38] J. Mao, W. Xu, Y. Yang, J. Wang, Z. Huang, and A. Yuille. Deep captioning with multimodal recurrent neural networks (m-rnn). *ICLR*, 2015.

[39] A. F. T. Martins and R. F. Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1614–1623. JMLR.org, 2016.

[40] T. Mikolov and G. Zweig. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pages 234–239, 2012.

[41] T. Mikolov, M. Karafiát, L. Burget, J. Cernocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010.

[42] T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur. Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531. IEEE, 2011. ISBN 978-1-4577-0539-7.

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[44] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[45] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38 (11):39–41, Nov. 1995. ISSN 0001-0782.

[46] A. Mnih and G. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 641–648, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3.

[47] A. Mnih and Y. W. Teh. A fast and simple algorithm for training neural probabilistic language models. In *In Proceedings of the International Conference on Machine Learning*, 2012.

[48] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *AISTATS'05*, pages 246–252, 2005.

[49] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

[50] H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38, 1994.

[51] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 841–848. MIT Press, 2002.

[52] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[53] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1310–1318, 2013.

[54] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[55] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281, New York, NY, USA, 1998. ACM. ISBN 1-58113-015-5.

[56] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 139–147, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[57] N. Rasiwasia, J. Costa Pereira, E. Coviello, G. Doyle, G. R. Lanckriet, R. Levy, and N. Vasconcelos. A new approach to cross-modal multimedia retrieval. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pages 251–260, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-933-6. doi: 10.1145/1873951.1873987.

[58] B. Roark and M. Saraclar. Discriminative language modeling with conditional random fields and the perceptron algorithm. In *In Proc. ACL*, pages 47–54, 2004.

[59] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? In *Proceedings of the IEEE*, volume 88, pages 1270–1278, 2000.

[60] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2377–2385. Curran Associates, Inc., 2015.

[61] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: The new data in multimedia research. *Commun. ACM*, 59(2):64–73, Jan. 2016. ISSN 0001-0782.

[62] R. Vedantam, C. L. Zitnick, and D. Parikh. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575. IEEE Computer Society, 2015. ISBN 978-1-4673-6964-0.

[63] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3156–3164, 2015.

[64] P. Werbos. Backpropagation through time: what does it do and how to do it. In *Proceedings of IEEE*, volume 78, pages 1550–1560, 1990.

[65] C. yew Lin. Rouge: a package for automatic evaluation of summaries. pages 25–26, 2004.

[66] P. Young, A. Lai, M. Hodosh, and J. Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2:67–78, 2014.

[67] C. Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2(3):137–213, Mar. 2008. ISSN 1554-0669.

[68] 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*, SIGIR '01, pages 334–342, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6.