Context-enhanced text recognition using semantic vector space models

Tom Gankema
September 16, 2016

Master's thesis
Artificial Intelligence
Department of Artificial Intelligence
University of Groningen, The Netherlands

Primary supervisor:
Prof. dr. L.R.B. Schomaker (Artificial Intelligence, University of Groningen)

Secondary supervisor:
Dr. M.A. Wiering (Artificial Intelligence, University of Groningen)

university of groningen / faculty of mathematics and natural sciences

Abstract

Contextual information is important to correctly recognize text. Several methods exist to use contextual information in text recognition systems, most notably statistical language models. However, these models have the disadvantage that they can not generalize over similar sequences of words, because they treat words as atomic units. Moreover has previous work failed to find a satisfiable solution to add semantic context information to text recognition systems. Several attempts have been made, however these often require a lot of manual work and do not scale to large vocabularies. In recent years, with the rise of deep learning techniques, substantial progress has been made to build semantic representations of words based on large text corpora with unsupervised methods. We propose a way to use these so called semantic vector space models to add semantic context information to text recognition systems. Two different neural network architectures are implemented and experiments show that they are able to make useful predictions about the meaning of unknown words based on the context in which these words occur. Experiments with a simulated text recognition system show that the proposed methods perform better than N-gram models when trained on the same data. The proposed methods can improve current text recognition systems, because they add semantic information, scale to large vocabularies and generalize over semantically similar words.

Contents

Chapter 1

Introduction

Humans perceive text in a meaningful way, for almost every word in a known language we know to which object, action or state of the world it refers to. If we read or hear a text and we perceive a yet unknown word, we can still attribute a meaning to this unknown word based on the context in which the word occurs (Nagy et al., 1987). Similarly, when we read an unclear handwriting or listen to a non-native language we can often still recognize the words by making use of syntactic and semantic information in the context (Becker, 1980; Bronkhorst et al., 1993).

For machines the use of context is much less natural. Pattern recognition is often focused on the pattern itself, rather than the context in which the pattern occurs. However, text recognition problems can not be satisfiably solved without limiting the search space with the use of context (Toussaint, 1978; Rose and Evett, 1995). Three types of textual context which are commonly referred to are lexical, syntactic and semantic context. Lexical context can limit the possible words to the words in a language, syntactic context can restrict the search space to words of the correct type for a given position in a sentence and finally semantic context can limit the possibilities to words which have a meaning which fits in the context.

Lexical context is used in practically all text recognition systems and several successful approaches for the use of syntactic context are known, most notably statistical language models (Vinciarelli et al., 2004; Marti and Bunke, 2001). However, despite the realization that semantic context is important (Toussaint, 1978), finding a successful and scalable approach to include semantic information has appeared to be a difficult problem (Rose and Evett, 1995; Erdogan et al., 2005).

This is partially because methods to extract such high-level, information from text corpora have been lacking. In recent years, substantial progress has been made in generating semantic word representations based on large amounts of texts (Turney and Pantel, 2010). In particular a set of algorithms referred to as word2vec (Mikolov et al., 2013a) has gained a lot of attention. This toolbox contains algorithms which calculate semantic vector representations on possibly billions of words very efficiently (Mikolov et al., 2013a; Levy et al., 2015). Moreover do these vectors possess another interesting property, linear combinations lead to intuitive and meaningful results. For example, in these $\mathbb{R}^n$ spaces the vector of 'Germany' and 'capital' can be added to get close to the vector of 'Berlin'.

The question arises whether these semantic word embeddings can be used to improve the understanding of text in machines and thereby improve text recognition, in

similar ways as humans use their understanding of the text to recognize words in unclear handwriting or non-native speech. This leads to the research question: can current text recognition systems be improved by using semantic vector space models to include contextual information in the recognition process?

This thesis elaborates on the theoretical background and recent developments in Part I. Then it proposes and details a method to incorporate semantic word representations in any kind of text recognition system and describes experiments to test the value of adding this information in Part II. Finally, it presents and discusses the experimental results and their implications in Part III.

Part I

Theoretical Background

Chapter 2

Context

2.1 Pattern recognition

The ability to distinguish patterns and make reasonable decisions about the categories of the patterns is essential for any intelligent system. Humans learn to recognize a lot of different patterns at a young age, for example characters are recognized regardless of size or orientation by most children at the age of five (Jain et al., 2000). We are unaware of exactly how we learn to recognize such a variety of patterns. However, research during the past century has taught us a lot about these processes. One of the things we have learned is that humans rely heavily on the context in which these patterns occur.

Toussaint (1978) already realizes the importance of context in the field of pattern recognition and gives several examples of patterns which have certain properties when viewed in isolation, which change when they are viewed in context, such as the well-known Müller-Lyer illusion (see Figure 2.1). He advocates a statistical approach to incorporate context in pattern recognition systems. This statistical approach has been the predominant approach to use context in pattern recognition for several decades. Examples of successful uses of statistical context models are compound decision theory derived from Bayes theory and Markovian models (Jain et al., 2000).

In recent years there has been a shift in pattern recognition. With the rise of deep learning techniques the usual process of hand-crafting features to differentiate between patterns can now often be automated by deep neural networks. In contrast to the old situ-

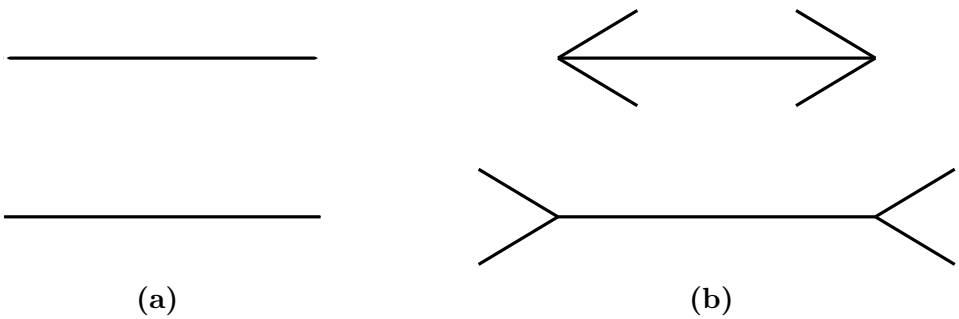


Figure 2.1: Müller-Lyer illusion. The lines in (a) appear to have the same length. But when context is added as in (b) the lower of the two lines appears to be longer than the top one.

ation in which different parts of the system were often "manually optimized, or sometimes trained, outside of its context" (Lecun et al., 1998), do deep networks learn an internal representation based on all relevant information to distinguish the presented patterns (LeCun et al., 2015). So if patterns are presented to the network with the context in which they occur, deep networks can also learn to utilize the contextual information. Different network architectures can exploit different types of contextual information, convolutional neural networks excel in utilizing spatial context (LeCun et al., 2015), whereas recurrent neural networks are better suited for exploiting temporal context (Graves et al., 2009).

2.2 Text recognition

Text recognition is the part of pattern recognition which focuses on recognizing textual patterns in the form of either writing or speech. Text can be viewed as a sequence of characters, phonemes or higher level sub-units such as words or even sentences. On each of the levels there are dependencies between subsequent patterns and humans appear to be aware of these dependencies. We know that one sequence of characters is more likely to occur than another and the same is true for sequences of phonemes, words and sentences.

The relations between these different concepts can roughly be divided into five categories. 1) Orthographic knowledge is the knowledge on character level and explains how characters form words, 2) phonetic knowledge is the knowledge on phoneme level and tells us how phonemes can be combined to form words, 3) lexical knowledge is the knowledge on word level and contains information about words in a language, 4) syntactic knowledge is knowledge on the sentence level and describes how words form sentences, and finally 5) semantic knowledge can transcend the sentence level and describes the meaning of a text. Humans make use of their knowledge about these different context levels when reading a text or listening to speech. Altmann and Steedman (1988) describe more thoroughly how humans make use of context during sentence processing.

Statistical methods are historically the most frequently used approach to include these different levels of contextual knowledge in a recognition system. Typically these methods first count co-occurrences of different units on a large corpus, these co-occurrences are also called N-grams (Vinciarelli et al., 2004). These co-occurrence statistics can then be used to rank the results of any kind of text recognizer. On character or lexical level this is a viable strategy, however to include syntactic information you need a corpus in which the words are tagged with their syntactical category, for example by part-of-speech tagging (Srihari, 1994). And for semantic knowledge this strategy becomes even more problematic, because it is hard to impossible to determine suitable semantic categories of words.

With the recent trend to use deep neural networks, these statistical methods have become less relevant. Recurrent neural networks can be trained to include contextual information in the recognition process. For text recognition the most successful neural network architecture is a bidirectional long short-term memory (BLSTM) network (Graves et al., 2009), this network processes the input sequence both forwards and backwards and can for that reason make use of context in both directions. Graves and Schmidhuber (2005) use a BLSTM network for phoneme classification and are one of the first to show that BLSTM networks are well suited for speech processing tasks, because of the way

Figure 2.2: Example in which semantic context is necessary to determine whether the last word is 'clip' or 'cup' (Toussaint, 1978).

they make use of context. Graves et al. (2009) discuss a BLSTM for character based unconstrained handwriting recognition and find after analysis that the network learns to use up to 120 time steps of context. More about BLSTM networks in Section 4.3. The presented work should be seen as part of this shift from a statistical to a neural network approach to include contextual information in a text recognition process.

2.3 Semantics

Words are not just discrete units, but refer to actions, objects or states of the world, in short; words have a meaning. Humans learn the meaning of words by reading or listening during interaction with the world. An important mechanism to learn the meaning of words is by deriving it from the context in which they occur. A single exposure to an unknown word can be enough to derive its meaning from the context, however several encounters of the same word in different contexts are needed to produce a deeper knowledge of the word (Nagy et al., 1987).

Part of the difficulty of learning the meaning of words is that words often have a slightly or sometimes completely different meaning in different contexts. Text recognition systems which try to exploit semantic context need to handle these ambiguities as well, which is one of the reasons why a satisfactory solution to include this information is a difficult problem. Semantic context is however essential to solve certain text recognition problems, take the example given by Toussaint (1978) shown in Figure 2.2. In the context of an office setting the last word should probably be 'clip', however in the setting of a 'picnic' the correct interpretation is probably 'cup'. Without adding more context it is impossible to determine which of these two words would be correct in this situation.

Although most text recognition systems do not include semantic information of words, several methods have been applied in the past. Some methods use the overlap between definitions of words in machine-readable dictionaries as a measure of their semantic similarity (Rose and Evett, 1993; Demetriou et al., 1997). Other methods label words with a semantic category and use them to predict the semantic category of the next word (Erdogan et al., 2005). This type of methods represent semantic information in a symbolic way, therefore we will refer to them as symbolic semantic methods.

However, all of these methods are supervised, as the semantic knowledge still needs to be programmed by humans. Therefore these methods are limited in use and do often not scale up from the domain in which they were developed (Rose and Evett, 1995). In contrast, humans learn most of their vocabulary knowledge without explicit instruction, but instead are able to derive meaning from context (Nagy et al., 1987). A machine learning approach which is able to derive meaning from context can solve the scalability issues of symbolic methods.

Based on the idea that words with similar meanings occur in similar contexts, such unsupervised methods have been developed and significantly improved in recent years

(Baroni and Lenci, 2010; Levy et al., 2015). This set of methods are called semantic vector space models, semantic embeddings or sometimes distributional semantic models. We will refer to them as semantic vector space models and use them to include semantic information in a text recognition process. Table 2.1 shows a comparison between the properties of different symbolic and vectorial methods to describe semantic information. More about semantic vector space models can be found in chapter 3.

Table 2.1: A comparison of symbolic and vectorial methods to describe semantic information of words. In this thesis we will focus on vectorial semantic methods.

Symbolic semantic models	**Semantic vector space models**
Machine-readable dictionaries (Rose and Evett, 1993; Demetriou et al., 1997)	Latent Semantic Indexing (Deerwester et al., 1990; Littman et al., 1998)
Conceptual graphs (Erdogan et al., 2005)	Neural networks (Elman, 1990; Mikolov et al., 2013a)
Explicit	Implicit
Atomistic	Distributed
Man-made	Learned
Supervised	Unsupervised
Static	Dynamic

Chapter 3

Neural network language models

3.1 Natural language processing

Traditionally the field of natural language processing tries to find rules underlying human languages. Until the 1980's this was done by manually establishing complex rules, after that statistical analysis on large text corpora became the main means of analysis and the current development appears to be towards neural network language models (Cambria and White, 2014; Hirschberg and Manning, 2015). Neural network language models are created by analyzing huge amounts of texts with a neural network. These networks are typically recurrent networks and are able to capture and generalize over the sequence information that is characteristic for text. In contrast to manually established rules or statistical patterns do these neural network models learn patterns in language more similar to the way the human brain processes natural language (Cambria and White, 2014).

One of the applications in which these neural network language models are currently the state-of-the-art is in machine translation (Sutskever et al., 2014). The reason why these models work so well is because they first encode a text sequence to a semantic representation and then decode this to a text sequence with a similar meaning as the initial sequence, but in another language. The key is that the neural network model is able to capture a part of the meaning of the text, which is fundamentally different as previously with statistical language models. LeCun et al. (2015) explain it very well: "N-grams treat each word as an atomic unit, so they cannot generalize across semantically related sequences of words, whereas neural network language models can because they associate each word with a vector of real valued features, and semantically related words end up close to each other in that vector space".

3.2 Semantic vector space models

The key of neural network language models is that they are able to capture semantic information of text in a vector space. In similar ways as it is possible to encode the meaning of a text in a vector, it is also possible to create semantic representations of single words, these are called semantic word embeddings. The idea behind this type of model is that words that share similar contexts tend to have similar meanings. This idea is called the distributional hypothesis and is the justification to use these models to

measure the semantic similarity of words (Turney and Pantel, 2010).

One of the first methods which use the idea of the distributional hypothesis to create meaningful representations of words is latent semantic analysis (Deerwester et al., 1990; Littman et al., 1998). Latent semantic analysis has been and still is a popular method to retrieve documents matching a query, for example in search engines. When leaving out important processing techniques like dimensionality reduction and smoothing, one can say that these models build a word-context matrix, in which every row corresponds with a word and every column with the counts of the context words. Because of the distributional hypothesis do words with similar meanings have similar context word counts and therefore end up close to each other in the vector space. These models are also called count-based models, because they are based on counts of context words.

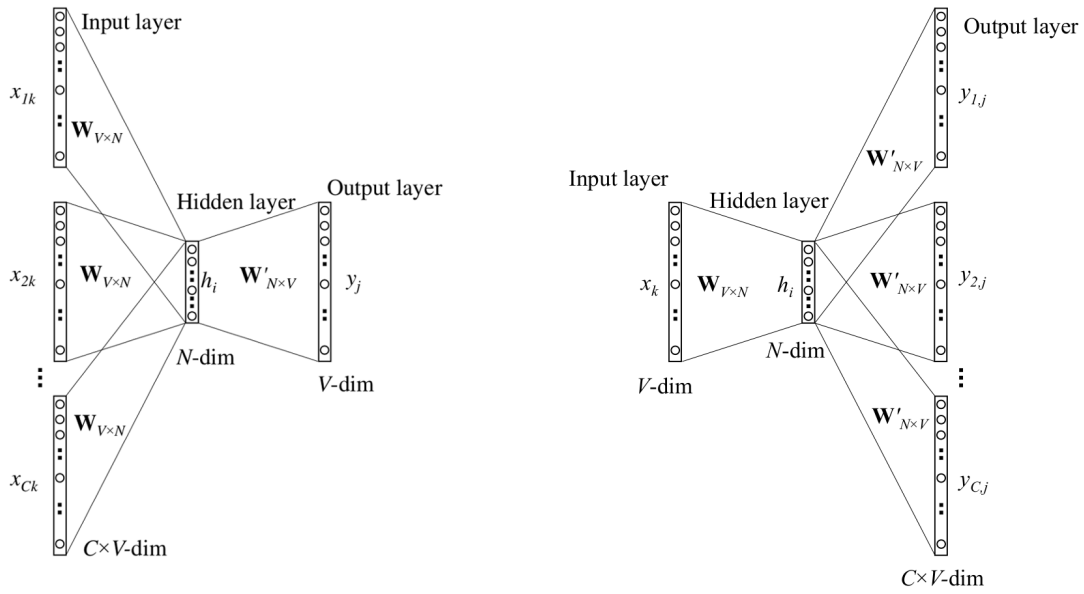
Elman (1990) is one of the first to use a neural network approach to create a semantic representation of words. He used a simple recurrent neural network architecture to predict the next word in a sequence and then was able to cluster semantically similar words using hierarchical clustering on a hidden representation in the network. Since then this idea has been developed further, primarily by using different neural network architectures. Some of these models try to predict the words in the context given another word in the sentence and others predict a word given the words in the context (Mikolov et al., 2013a).

For this reason these models are also called predict-based models. The predictions can be verified, an error measure can be calculated and the weights of the neural network can be updated to minimize the error and thereby improve the prediction (Bengio et al., 2003). However, it is not the prediction that is most interesting, but the representation of words in a hidden layer of the network. These hidden representations of words form a semantic space. Very similar to semantic spaces created by latent semantic analysis, do semantically similar words end up close to each other in this vector space and the similarity between words can be described by using a cosine similarity measure. Baroni et al. (2014) compared count-based and predict-based models and found to their surprise that predict-based models consistently outperform count-based models on a number of tasks. Currently the most popular implementation of such predict-based models is a set of algorithms known as word2vec (Mikolov et al., 2013a).

3.3 Word2vec

Word2vec* is a toolbox with algorithms to calculate semantic word embeddings. Specifically it contains two different algorithms, the continuous bag-of-words (CBOW) and skip-gram (SG) (Mikolov et al., 2013a). CBOW creates word embeddings through predictions of the current word based on the context. Both on the input and on the output of the network a 'bag-of-words' or also 'one-hot' representation is used to respectively represent the context and the word to predict. The hidden layer has a limited number of units corresponding with the desired dimensionality for the word vectors. After training the network, word vectors can be obtained by using a single word on the input. The word vector is then equal to the output of the hidden layer. SG creates word embeddings through predictions of the context based on another word in the sentence and therefore the network architecture is the reverse of the CBOW architecture, see Figure 3.1.

*https://code.google.com/archive/p/word2vec/



(a) Continuous bag-of-words **(b) Skip-gram**

Figure 3.1: The continuous bag-of-words (a) and skip-gram (b) neural network architectures used in word2vec (Rong, 2014).

One of the reasons word2vec has gained a lot of attention is because words with similar meanings are not just close to each other in the vector space, but have another special property. This property is that you can simply add or subtract word vectors and get meaningful results (Mikolov et al., 2013b). For example the vector of 'Germany' plus the vector of 'capital' is close to the vector of 'Berlin' and the vector of 'Germany' plus the vector of 'Berlin' minus the vector of 'France' is close to the vector of 'Paris', see also Figure 3.2. This shows that important parts of the meaning of the words is captured in the word vectors and they can be used for example to solve analogy questions (Mikolov et al., 2013b).

The question arises why the word vectors created by word2vec are so powerful. It has been shown that the skip-gram model implicitly calculates pointwise mutual information (PMI) (Levy and Goldberg, 2014; Levy et al., 2015), a concept known from natural language processing. However, word2vec is able to calculate the same information much more efficient than older methods and thereby makes it possible to calculate on data sets much larger than previously possible. Therefore resulting in embeddings with special semantic properties.

These semantic properties make word embeddings calculated by word2vec especially interesting to use for context based methods. Because a neural network model using these vectors can not just learn to generalize over words with similar meanings, but can also learn to generalize over similar relations between words in the context. For example, if such a model learns that 'king' is referred to by 'him', then it could theoretically derive that 'queen' should be referred to by 'her', even if that specific case does not occur in the training set. This is the reason why semantic word embeddings calculated by word2vec are used in the current work, although the proposed methods could be used with any type of word embedding.

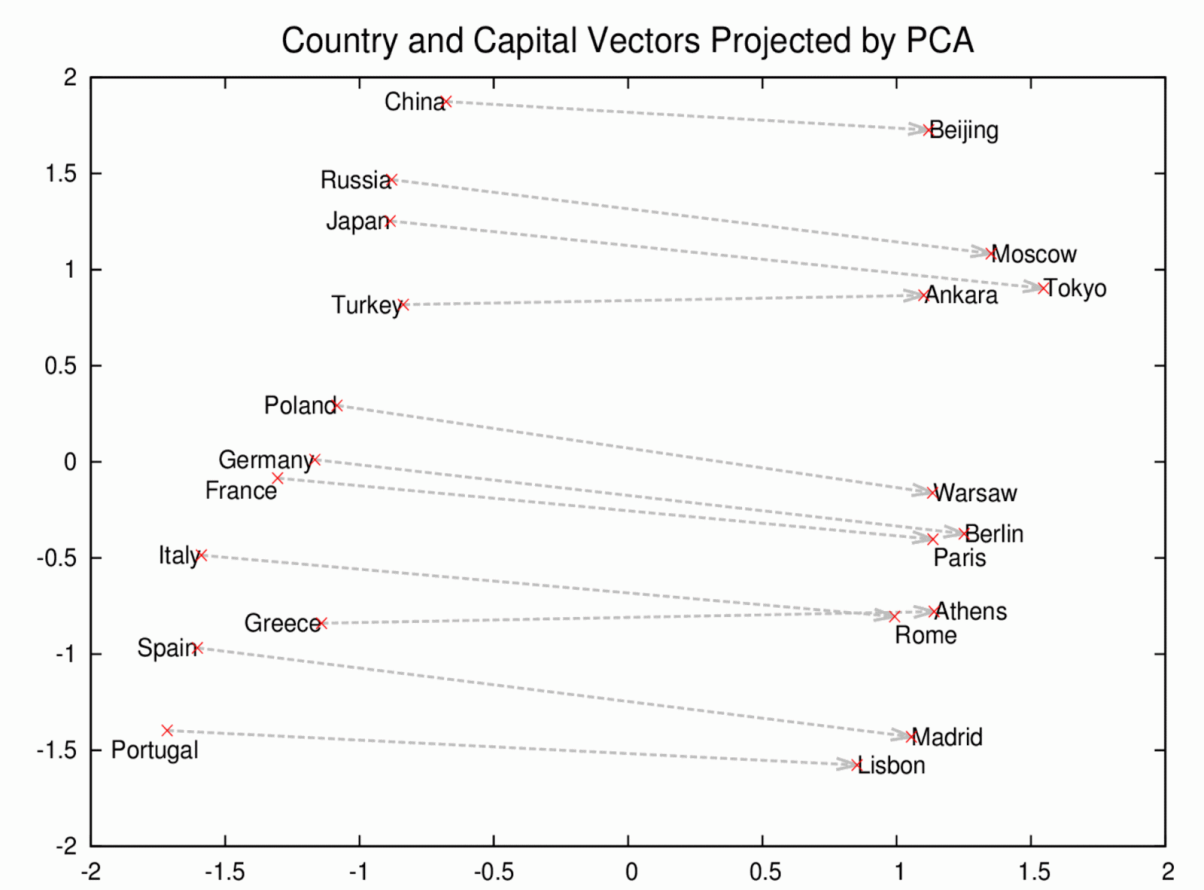


Figure 3.2: Two dimensional PCA projection of word vectors created by the SG algorithm. Not only are similar countries close to each other in the vector space, but their capital cities are also arranged in parallel (Mikolov et al., 2013b).

Chapter 4

Neural networks

4.1 Text in neural networks

Neural networks are used for a range of different regression and classification tasks, one of these tasks is text recognition. Text consists of words, which are discrete units, for neural networks these are often represented using a discrete representation containing only ones and zeros. However, in reality words are not just discrete units but have a meaning, in our perception some words are more similar than others. A discrete representation fails to capture these word similarities which make it hard to generalize over them. To overcome this problem word embeddings are sometimes used as input for neural networks (Bengio et al., 2003), which is the approach that is used in the presented work.

4.2 Multi-layer perceptron

A multi-layer perceptron (MLP) is a basic and well-known type of neural network, it has multiple layers with each layer fully connected to the next one. They can be trained to map sets of input values to appropriate sets of outputs, this is done by step-by-step minimizing an objective function which calculates the difference between the desired and the actual outputs (Rumelhart et al., 1988). A multi-layer perceptron has no memory of previous inputs and therefore every set of inputs is classified separately from previous inputs. To still include temporal information in the network it is possible to add previous inputs to the current input as well, such as with a context window as is used in the current research (see Section 5.5).

4.3 Bidirectional long short-term memory

Another approach to include temporal information in a neural network is by adding recurrent connections, through these connections previous information can be retained in the network for multiple time steps. These so called recurrent neural networks have been proven to be well-suited to learn sequence information (Graves et al., 2009). However, they often fail to correctly learn long-term dependencies due to the vanishing gradient problem (Bengio et al., 1994). To capture long-term dependencies a new type of neural network, called long short-term memory (LSTM) was created (Hochreiter and Schmidhuber, 1997).

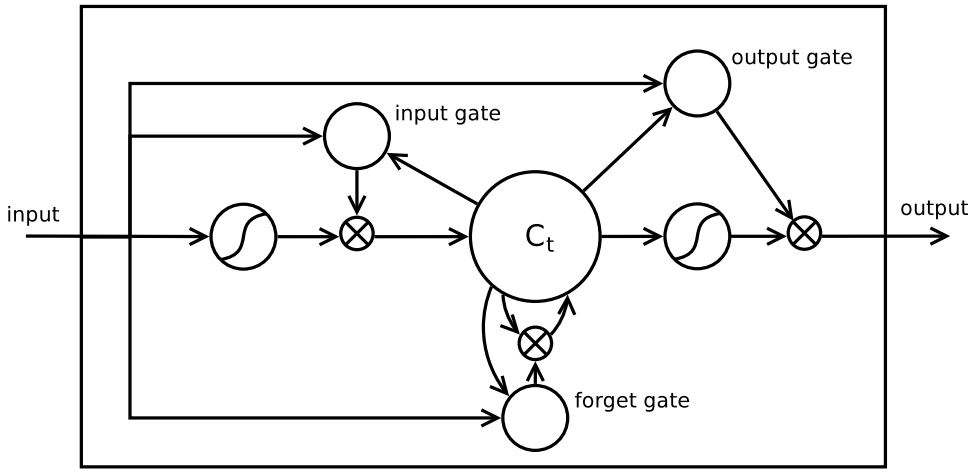


Figure 4.1: An LSTM cell showing how the forget gate, the input gate and the output gate interact with the cell state C_t.

An LSTM layer can be seen as a network consisting of four separate layers which interact in a special way. Central to the LSTM is the current cell state, which interacts with three gates. These gates typically consist of a sigmoid 'layer' and a pointwise multiplication. The sigmoid function determines how much information can pass through the gate, '0' means no information can pass through and '1' means everything gets through. A forget gate determines how much of the current cell state should be forgotten for the next time step, an input gate determines how much of the input should pass through to the next cell state and an output gate decides what portion of the cell state is returned as output, see Figure 4.1. The result is a structure which is very good at learning long-term dependencies, which is especially important when dealing with language such as with speech (Graves and Schmidhuber, 2005) and handwriting recognition (Graves et al., 2009).

A normal LSTM has the disadvantage that it can only use information from the past, in some situations, such as when reading text, future context is also important. The solution for this problem is to use a bidirectional LSTM (BLSTM) (Graves et al., 2009), which basically uses two LSTM layers, a forward layer which processes data from start to end and a backward layer to process the data in the opposite direction. After the entire input sequence is processed by both the forward and the backward layer, the output is calculated by combining the forward and backward predictions. LSTM and BLSTM networks can be used to return a single value or a sequence of values based on the inputs. In the presented work we use a BLSTM architecture which returns a sequence.

Part II

Model Implementation and Evaluation

Chapter 5

Methods

5.1 Overview

In this chapter we describe a method to include semantic context information in text recognition systems using semantic vector space models. An overview of the different components and how they interact is shown in the pipeline in Figure 5.1. First we will explain the task we try to solve using this method, in the sections that follow we will go through the different components of the pipeline.

5.2 Task description

Current text recognition systems are not perfect and generally leave gaps in a text because not all words can be recognized with a probability above a certain acceptation threshold. The words which can be recognized give a context for the rest of the words and in a second recognition iteration it is possible to exploit this information to improve the recognition further. Here we simulate a situation of an incomplete text classification. Specifically we assume that 70 percent of the text has been recognized correctly without the use of semantic context information, but in which 30 percent of the words in the text are still unknown. The question we are trying to answer is whether it is possible to fill in (some of) these unknown words using semantic information available in the context.

Different text recognition systems have varying percentages of words they are able to classify correctly. The higher the percentage of known words, the more semantic information is available and so the easier the task is of exploiting this information to fill in the unknown words. The 70 percent initial recognition rate is chosen somewhat arbitrarily, however we believe this provides both enough context to improve recognition as well as a realistic situation which currently can occur in a range of different text recognition systems.

As far as we know this is the first attempt at improving a text recognition using word vectors in this way. The described task probably shows the most similarities with a sentence completion task, such as the Microsoft sentence completion challenge (Zweig and Burges, 2011), in which the state-of-the-art method makes use of word2vec and predicts 59 percent (Mikolov et al., 2013a) of the words correctly. In that task short sentences are provided with a single word left out and five words are given to choose from to complete

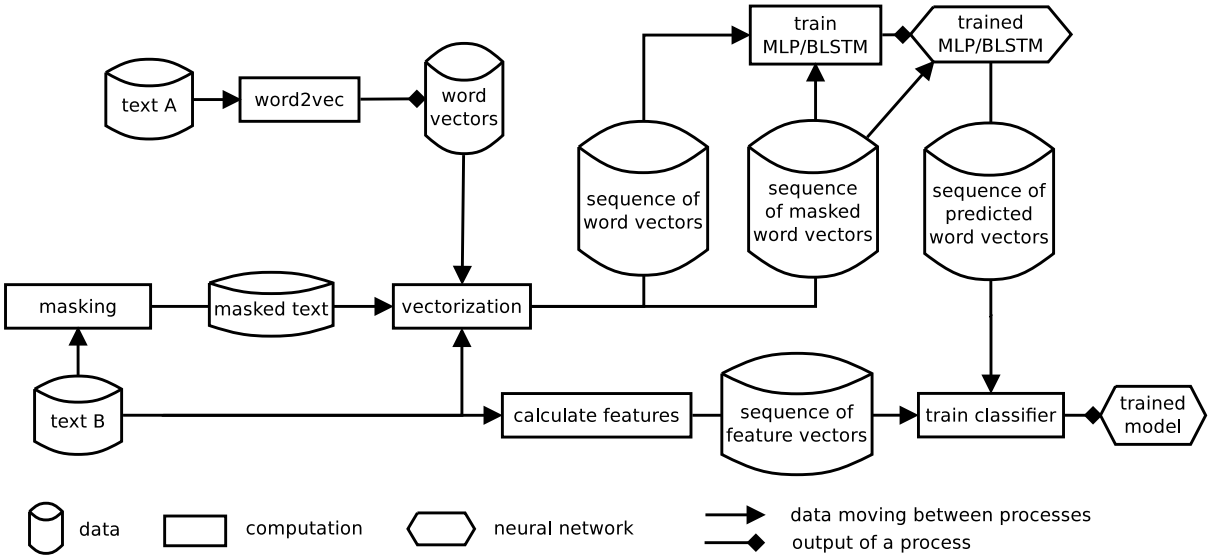


Figure 5.1: Pipeline showing the process of training the proposed semantic context-enhanced text recognition system.

the sentence, so the random baseline performance is 20 percent and human performance is reported at 91 percent. However, despite some similarities it is important to note that the presented task is different, because the texts are longer, but with only 70 percent of the words known. It is very well possible that multiple words after each other are unknown, this means that local context information may not always be available, however more global long-range context is available. On top of that are the number of options given not limited to pre-selected words, but equal to the vocabulary size, which makes the task considerably harder.

5.3 Data and preprocessing

Two separate data sets are used for training, one to calculate the word2vec word embeddings (text A) and the other to create a data set to train the neural networks on (text B). Text A contains 124 million words from 243,426 English Wikipedia articles, as well as 11 million words from 107 nineteenth century American novels from the Gutenberg project. Text B contains 11.7 million words, also from English Wikipedia articles.

We chose these texts because we needed large data sets and these are freely accessible on the internet. Wikipedia texts are generally very structured texts about a range of different subjects and therefore considered to be very suitable to train balanced word embeddings on. The Wikipedia texts are enriched by using texts from nineteenth century American novels. The reasoning behind this is that these older texts inevitably use a different vocabulary and therefore a relatively small amount of text can add a lot of extra information, this extra information was important to us because of a possible future use of these embeddings to improve handwriting recognition of texts from the nineteenth century. The current study however is limited to testing on texts from Wikipedia.

Both texts are preprocessed in the same way, according to the following rules:

- Punctuation and special characters are removed, except periods - these will be

treated as separate words

- Capitals are converted to lower-case characters
- Digits are spelled out - so '10' becomes 'one zero'

To simplify things, there are no indications of the start or end of an article or novel, all texts are simply concatenated to create one large sequence of words separated by spaces.

5.4 Word2vec

To calculate the word embeddings we use the continuous bag-of-words model in combination with the negative sampling training algorithm provided by the publicly available word2vec toolbox. The negative sampling algorithm is a relatively straight-forward way to allow for efficient training of the network, for a detailed description of the algorithm, see Mikolov et al. (2013b). This model is trained by making predictions of words based on the words in the context, as described in Section 3.3. The output of this training process is not only a trained neural network, but most importantly the word vectors of the words in the training set, which are obtained through the output of the hidden layer of the network. For a description of the training parameters, see Mikolov et al. (2013a).

Word vectors with 200 dimensions were trained in fifteen iterations using a context window of eight words in both directions. Frequent words were subsampled to occur less frequently in the training set according to the methods provided by the toolbox with a chosen threshold of 10^{-4} and words which occurred less than five times were discarded during training. The result was a vocabulary size of 224974 words for which word vectors were calculated.

To confirm the word vectors were trained correctly we evaluated the quality of the resulting word vectors using the methods described by Mikolov et al. (2013a). A list of semantic-syntactic word relationship questions as provided by the word2vec toolbox was used. Using the trained word vectors 72.45% of the 8869 semantic questions and 61.63% of the 10675 syntactic questions were answered correctly.

These word vectors are combined with text B by replacing the words in the text with their corresponding word vectors to create a sequence of word vectors, we call this vectorization. Words in the text for which no word vector has been calculated are replaced by a 200 dimensional vector of zeros, which is the representation that is used for 'masked' words. The resulting data set is used to train the neural network models.

5.5 Neural network models

Two different neural network models are implemented using the Keras library *, a multilayer perceptron (MLP) and a bidirectional long short-term memory (BLSTM) network. Both networks have the same goal: to approximate masked word vectors based on the context. To achieve this the networks use the previously calculated word vectors, both as input and as targets. The training process is kept the same as much as possible for both

*https://github.com/fchollet/keras

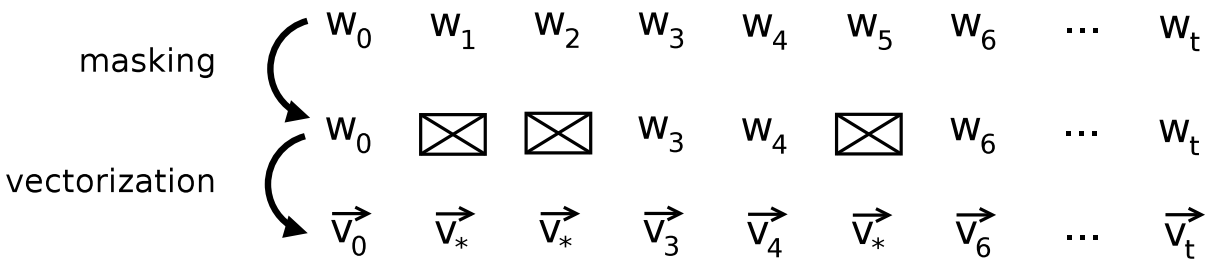


Figure 5.2: Example of the masking and vectorization processes at the start of each training epoch, in which you can see the masking of w_1, w_2 and w_5. $\vec{v}_t$ is the word vector corresponding with word w_t and $\vec{v}_* = [0., 0., ..., 0.]$.

neural network models. However, because of the fundamental differences between both network architectures, the sequence of word vectors is processed differently, the details of this are described below.

To include context information in the MLP, a context window is used. This means that a single input for the network contains all word vectors from within a context window of the target word vector. Therefore the number of nodes in the network depends on the width of the context window (W) and the word vector dimensionality (D). Because the current word is included in the context window, the network has $(W-1)*D$ inputs, a hidden layer with an equal amount of nodes and D outputs, both the hidden and output layer apply a hyperbolic tangent activation function, see the overview in Figure 5.3. Here we use vectors with 200 dimensions and the optimal context window size is determined in one of the experiments, see Section 6.3.

In contrast to the MLP, does the BLSTM network not need a fixed context size, instead it will process the word vectors sequentially and will implicitly learn the temporal properties of the data. For this reason the number of inputs of the network is equal to the dimensionality D of the word vectors and the number of nodes in both the forward and backward layer of the network are chosen to be identical to this. Both the forward and the backward layer apply a hyperbolic tangent activation function and the forget biases are initialized with ones, as recommended by Jozefowicz et al. (2015). The output is the average of the sequences returned by the forward and backward layers, see the overview in Figure 5.4.

To train the neural network models a random 30 percent of the word vectors is masked at the start of a training epoch by going through the masking and vectorization processes, see Figure 5.2. These masked word vectors are used to create the input of both networks as described above. For the output an unmasked version of the same data is used. Training consists of altering the neural network weights to minimize the cosine distance between the output and the target word vectors.

The neural network models are trained for ten epochs on the sequence of 11.7 million word vectors resulting from the combination of text B and the word vectors created by word2vec. To avoid over-fitting, 90% of this data is used as training set and 10% as validation set. The data is processed in batches of 10000. The gradient-based optimization algorithm that was used to train the networks is called Adam, for more information about this algorithm see Kingma and Ba (2015). Parameter values follow those suggested in the paper.

5.6 Classification

The output of the neural network models gives a prediction of the word vectors of the masked words, thereby approximating the meaning of these words. To select a word based on this prediction one could simply select the word vector with the smallest cosine distance to this prediction. However, this has the disadvantage that it is a computationally intensive method, because the word vectors of all the words in the dictionary need to be compared with the prediction. This is one of the reasons why we choose to train an extra MLP classifier which takes as input a prediction of a word vector and outputs a class corresponding with a word from the dictionary. This also has the advantage that it becomes easier to add additional features into the classification process.

This classifier is a fully connected neural network with D+F inputs and V outputs, in which D is the vector dimensionality, F an optional number of additional features and V the vocabulary size, see Figures 5.3 and 5.4. In the final layer a softmax activation function is used, to predict the probability distribution. All other nodes use a hyperbolic tangent activation function. In our experiments we either use no hidden layer or a single hidden layer with D nodes, depending on the experiment. For every different combination of inputs a new classifier was trained from scratch on data specific for that combination of inputs. So in case of a neural network model, the output of that neural network model on the training data is used to train the classifier. Training took place for five epochs using a categorical cross entropy objective, unless mentioned otherwise. During this training process the weights of the neural network models were not further updated.

The question we try to answer in the current study is to what extent it is possible to use semantic vector space models to improve current text recognition systems, for this reason we simulate a text recognition system by implementing four additional features. These can be used to train a separate classifier or can be added to the semantic context predictions to create one classifier trained on a combination of different features, more about this in Section 6.7. It is important to note that any kind of feature could be added into the process here, printed text features, speech features, features from another neural network model or even features extracted from images of the object or concept the word refers to. However, because the focus of the study is about semantic information, we only consider the following four shape features:

1. Number of ascending characters in a word (‘b’, ‘d’, ‘f’, ‘h’, ‘k’, ‘l’, ‘t’)
2. Number of descending characters in a word (‘g’, ‘j’, ‘p’, ‘q’, ‘y’)
3. Number of closed characters in a word (‘a’, ‘b’, ‘d’, ‘e’, ‘g’, ‘o’, ‘p’, ‘q’)
4. Number of non-closed characters in a word (‘c’, ‘f’, ‘h’, ‘i’, ‘j’, ‘k’, ‘l’, ‘m’, ‘n’, ‘r’, ‘s’, ‘t’, ‘u’, ‘v’, ‘w’, ‘x’, ‘y’, ‘z’)

These features are used, because they are simple to calculate and could also be used in a real text recognition system, for example in a handwriting recognition system. Next to that, should these features not be too powerful, because we can calculate them without any error and if it is possible to recognize most words using only these features, then a ceiling effect could occur which makes it hard to significantly improve on this text recognition system using a context model. The features mentioned here are always normalized using z-score normalization and are the features that are meant when we refer to the simulated text recognition system (STRS).

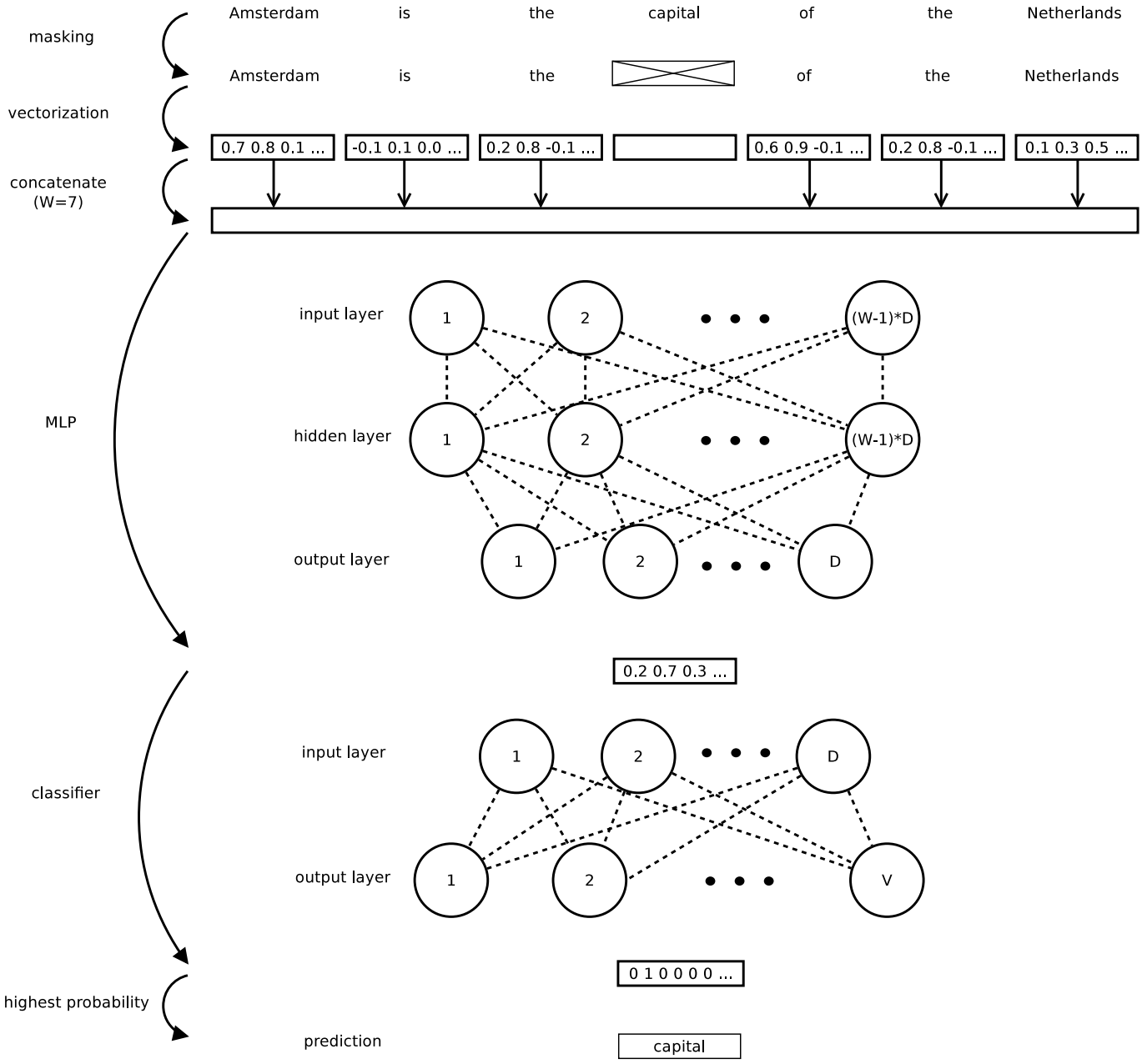


Figure 5.3: Overview of the context prediction model using the MLP, showing an example in which the word vector of the word 'capital' is predicted based on its context and then classified using the classifier. W indicates the context window size (in this example 7), D the dimensionality of the word vectors and V the vocabulary size. Through all experiments $D = 200$ and $V = 224974$. Dashed lines indicate learnable neural network connections.

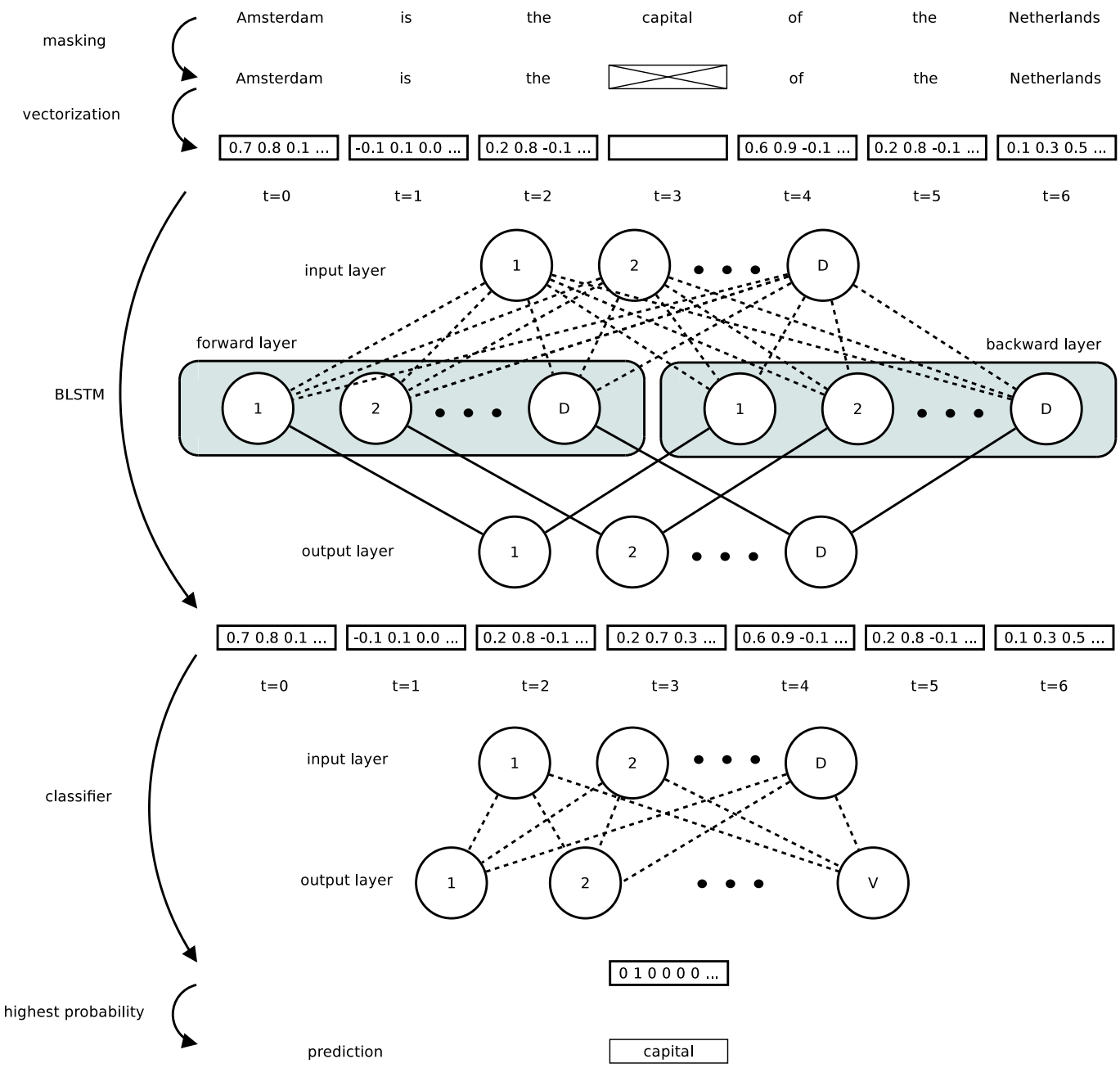


Figure 5.4: Overview of the context prediction model using the BLSTM, showing the processing of an example sentence through timesteps $t = 0$ to $t = 6$. The BLSTM returns the entire input sequence in which the masked vector of the word 'capital' is predicted based on the context. All output vectors of the BLSTM can be classified one by one using the classifier, however in practice we are only interested in the predictions of the word vectors which are masked on the input of the BLSTM, in this example of the word 'capital'. D indicates the dimensionality of the word vectors and V the vocabulary size. Through all experiments $D = 200$ and $V = 224974$. Dashed lines indicate learnable neural network connections.

Chapter 6

Experiments

6.1 Overview

In this chapter we will describe and discuss the experiments that were performed to inspect the performance of the discussed context prediction models. Part of this is a description of N-gram models that were implemented for comparison. Also different ways to combine the models are discussed in detail along with the corresponding experiments to compare these combination methods. For an overview of the data flow in the experiments see the pipeline in Figure 6.1.

For all experiments described in this chapter, the same text is used as test data to allow for fair comparisons. This text contains 20585 words from 58 English Wikipedia articles. Of this text the same 6175 random words were masked out for every experiment. We expect this data set is sufficient and has enough variety to show and compare the performance of the context prediction models. Next to that are the proposed methods computationally and memory intensive, which did not allow us the time to evaluate all the implemented models on a much larger text.

The performance was measured by the percentage of correctly predicted words. Apart from just top-1 performance, also a top-10 performance measure is reported. This top-10 performance shows the percentage of the predictions in which the correct word was among the ten most likely predictions. We report this because it is very hard to predict the correct word exactly based on just the semantic word embeddings. The most obvious reason for this is that two words can be semantically identical, synonyms, however only one will be considered correct and the other will be considered incorrect. Next to that do we believe that it is better to be almost correct than to give a completely wrong answer, because an almost correct answer is still likely to be meaningful. Therefore the top-10 performance measure will also contain interesting information about the results of the experiments.

6.2 Word-frequency compensated sampling

Word-frequency compensated sampling (WFCS), also called subsampling in word2vec, is a method that is frequently used when dealing with text in neural networks to make frequent words occur relatively less frequent in the training set. Here we test the effects of

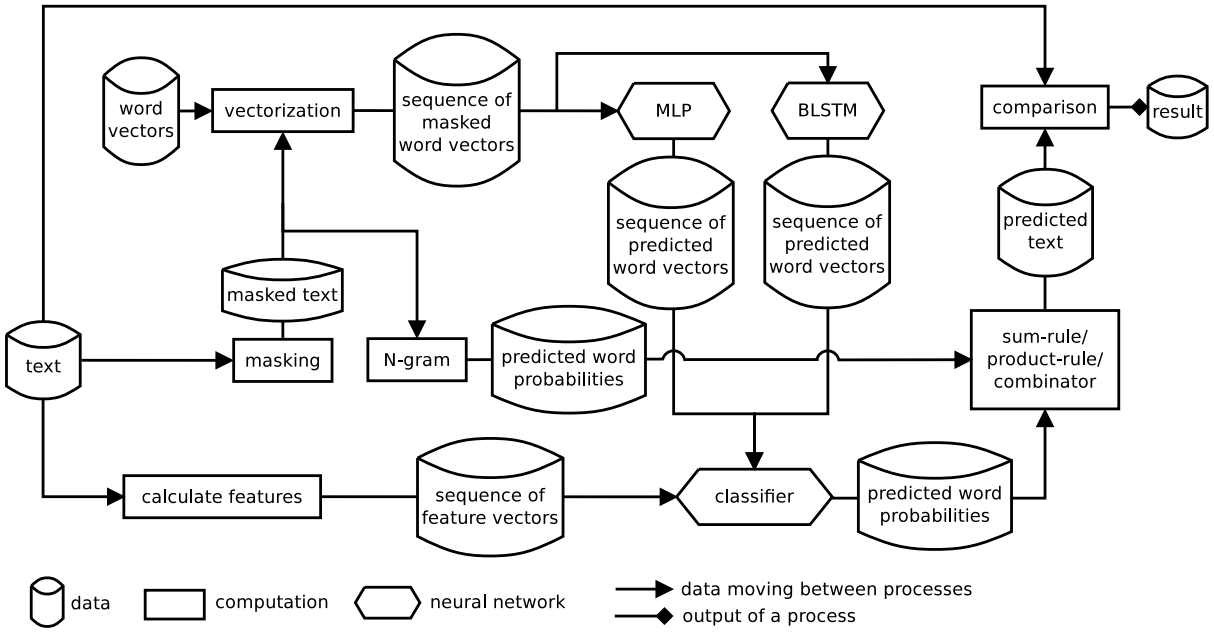


Figure 6.1: Pipeline showing the process of testing and evaluating the proposed semantic context-enhanced text recognition system.

WFCS on the performance of the proposed methods. The experiment consists of training two identical MLP and two identical BLSTM models, one of both trained on data without WFCS and the other with WFCS. Without WFCS the probability of a word to be masked is equal to 30 percent. However, when WFCS is used this probability is inversely proportional to the number of occurrences of that word in the current batch, while still making sure that eventually 30 percent of the words is masked. This way all words are masked more or less equally often over the entire training set. The same technique is also used when training the classifier part of the network. During these experiments the context window size for the MLP models is 15.

Next to the normal top-1 and top-10 performance measures described above, also a performance measure on infrequent words will be reported for this experiment. Infrequent words are defined as all words which are not numbers, prepositions, single character words or other words from the top-25 most frequent words such as 'is', 'have' and 'his'. When removing these frequent words from the test set, 3428 infrequent words remain of which the top-1 and top-10 performance measures will be reported.

We expect that WFCS will improve the predictions on infrequent words, because the number of training samples for these words will increase. The opposite happens to frequent words, so it is to be expected that the number of correct predictions on frequent words will drop. The idea behind implementing WFCS in this case is that infrequent words are generally more meaningful than frequent words. For that reason we are interested in the exact effects of the context prediction models on meaningful and less meaningful words when using WFCS as opposed to when not using WFCS. If the negative effects on the predictions of frequent words will be limited, this approach may be beneficial for the proposed methods.

6.3 Context size

The MLP context prediction model uses a context window. To determine the optimal size of this context window an experiment is performed in which the context window size is varied. For this experiment nine different neural network models are trained using context window sizes from three to nineteen words. Because the context window is defined as the current word plus an equal amount of words on either side of that word, models always have an odd numbered context size. For every neural network model a separate classifier is trained to optimally classify the context predictions of each model and allow a fair comparison between the models. These classifiers have no hidden layers and are in all ways identical, except for the context predictions on which they are trained.

Apart from an optimal context size for the MLP model, the experiment will also help to answer questions about how much context can still be relevant for the proposed semantic context predictions methods. This is important to know, especially when you want to apply the methods in situations in which long-range context may not always be available, such as when filling in words in a separate sentence without any other context.

We expect that larger context sizes will lead to a substantially higher percentage of correct predictions. However, there may be a trade-off, because choosing the context size too large will add mostly noise to the classification process and therefore no longer positively affect the performance. Despite this, we expect that long-range context will prove to be important for the proposed models.

6.4 N-gram comparison

To test if the semantic context models perform better than statistical language models, several N-gram models are calculated for comparison. First of all a bigram and a trigram model are calculated on the same data set that is used to train the neural network models and classifier, text B from Section 5.3. Secondly, another trigram model is calculated on a larger data set, which is the same as is used to train the word embeddings on, text A from Section 5.3. The performance of these statistical models is evaluated on the test set and compared to the performance of the implemented neural network models.

These N-gram models predict the next word probabilities based on previous context. In our case the text has 30 percent masked words, for this reason the trigrams use a special <unknown> token for these words. These act as a wild card, by taking the average over all N-grams with any word at the position of the <unknown> token. For a trigram model for example, a query for the tuple ('the', <unknown>) will return the average word probabilities over all trigrams starting with 'the'.

Our expectation is to show that the neural network models predict significantly more words correct than the statistical models. This would correspond with recent findings in natural language processing (Cambria and White, 2014; Hirschberg and Manning, 2015; LeCun et al., 2015). The trigram model will outperform the bigram model if the data on which the model is trained are sufficient and representative for the test data, which we believe to be the case. The trigram model trained on the extra large data set will naturally outperform the other N-gram models. Some may argue that this gives a more fair comparison to the neural network models, since the neural network models also benefit from the word embeddings which are trained on the same extra large data set. However,

we believe that the data on which the neural network models and classifier are trained are the deciding factor for the eventual performance of the neural network model, much more than the data on which the word embeddings are trained. Therefore we believe that the bigram and trigram models trained on the same data as the neural network models give the most fair comparison. However, for completeness and to get a better insight in the differences between the models we decided to also include a trigram model trained on a larger data set. Note that the neural network models could not be trained on this larger data set because of constraints to time and computational resources.

6.5 Comparison of MLP and BLSTM models

To inspect the differences between the MLP and BLSTM predictions, both are compared per vector dimension. This is done by looking at the root-mean-square error (RMSE) between the predictions and their targets of all 200 vector dimensions individually. To quantify the relation between the predictions of both models the Pearson product-moment correlation coefficient is calculated.

This is of interest because it gives an insight in the differences between both neural network models. Moreover is it an indication of whether a combination of both models is likely to further improve the results of the individual models. It could for example be the case that the MLP model is good at predicting other vector dimensions than the BLSTM model is good at. In that case, a classifier trained on a combination of both models is likely to significantly improve the results of each individual model.

To verify this idea, a classifier is trained by using as input a concatenation of the predictions of the MLP model with context size fifteen and the BLSTM model. Because the input of the classifier is a concatenation of the context predictions of both MLP and BLSTM models, the classifier has 400 inputs, it has no hidden layer and 224974 outputs, corresponding with the vocabulary size. The network is trained for a total of three epochs, and the prediction results of the classifier are compared with the prediction results of the individual models.

The expectation is that there is a correlation between how well the MLP and BLSTM models predict the different vector dimensions. If this correlation is small, it can be the case that a combination of both models will significantly increase the recognition rates compared to both individual models. However, this is unlikely to be the case when there is a high correlation between both models.

6.6 Combination of neural network and statistical models

The goal of the neural network models is to include semantic information to a text recognition process, however these models also capture statistical properties of the text. For this reason it is interesting to look at combinations of these neural network models with statistical models. If a combination of the semantic context predictions and statistical predictions lead to more correctly predicted words than the individual predictions, then both types of models contain complementary information.

Three different methods to combine the predictions of both models are compared here, a sum-rule, a product-rule and a neural network combiner. The sum-rule adds the predicted probabilities per word, the product-rule multiplies the predicted probabilities per word and the neural network combiner uses a neural network to combine the predicted probabilities per word. This means that in the case of combining two models, the combiner has two inputs, one for each model, and a single output and this network will be used to evaluate a final probability for all the words. This network has a single hidden layer of the same size as the input. All layers apply a sigmoid activation function. The goal of the combiner is to approximate the real probability of a word given the probability predictions of multiple models.

The combiner is trained on predictions of the specific models that it aims to combine. To do this a separate text is used, this text contains 68246 words from 63 English Wikipedia articles. As with the other texts, a random 30 percent of the words are masked. For every masked word all models make a probability prediction for all words in the vocabulary. The combiner is trained using these probabilities as input and either one or zero as output, depending on whether that vocabulary word was respectively the correct prediction or not. The network optimizes a binary cross entropy objective using the gradient based optimization algorithm Adam. Training takes place for 25 epochs, it was made sure that no over-training took place by testing the network after every epoch.

The different combination methods all have their own advantages and disadvantages. The advantage of the sum-rule is that it can still work if one of the classifiers gives a completely wrong prediction, as long as the prediction of most classifiers is correct. However, this can also be a disadvantage, because ideally you want to return a high probability only if both classifiers give a high probability for a word. This is what happens when you use the product-rule, which is also the most widely accepted way to combine probabilities, as it follows Bayes‘ theorem. However, if one of the classifiers generally is wrong this will have a huge impact on the predictions. Finally, the combiner has the advantage that it can weigh the predictions in a more advanced way. However, it requires extra training and a separate data set to train on, this also brings the problem of possibly under- or over-training the network, which will result in suboptimal performance.

Using the combination methods described above we combine the earlier described N-gram model predictions with both neural network model predictions individually. Then, we compare the performance of the combinations of these models with each other and with the performance of the individual N-gram and neural network models. We expect that a combination of N-gram and neural network models will perform slightly better than either of them alone. However, this effect will probably be small, because the neural network models are also able to capture statistical properties of the text.

6.7 Extending a text recognition system

The purpose of the current research is to study if existing text recognition systems can be enhanced with the use of semantic vector space models using the proposed methods. For this reason, we simulate a text recognition system. This simulated text recognition system (STRS) is a classifier which is trained on just the four basic text features described in Section 5.6. After it is trained the STRS is used to recognize the masked words in the

test set. This is very similar to how the performance of context prediction models are evaluated, with one important difference, the STRS uses features extracted from the word itself, whereas the context prediction models have no information about this word and just use information available in the context.

We aim to improve the STRS by combining it with the context predictions of the neural network models using the same combination strategies as described in Section 6.6. If a combination of the STRS with the neural network models leads to significantly more correctly recognized words than the STRS alone, then the proposed methods can add information to text recognition systems. However, in order to consider this an improvement over existing text recognition systems, the semantic context prediction models will also need to improve on existing methods to include contextual information to text recognition systems. Therefore, we will compare the combination of STRS and the neural network context prediction models with the combination of STRS and the implemented N-gram context prediction models.

Except the described methods to combine predictions, there is another way to combine the neural network context predictions with a text recognition system. A single classifier can be trained on a combination of semantic context predictions and the features of the STRS, this has briefly been discussed in section 5.6. We train and test two classifiers, one trained on a combination of MLP predictions and STRS features and another trained on a combination of BLSTM predictions and STRS features. Both classifiers are trained for 3 epochs. It has to be noted that a similar approach for the N-gram prediction models is not feasible, because the N-gram predictions are not condensed into a 200 dimensional vector, but instead give a probability for all vocabulary words. A fully connected neural network using these probabilities as inputs would take over a year to train per epoch, compared to around two days per epoch for the classifiers for the MLP or BLSTM models.

The MLP models used for these experiments use a context window size of 15. Two classifiers are trained for all combination of models, one without a hidden layer and one with a single hidden layer with the same size as the input. However, for brevity only the results of the classifier which achieved the highest recognition rate for every combination of models is reported.

We expect that the semantic context predictions and the STRS features complement each other well, because the semantic context predictions can limit the search space of the STRS to a semantic space in which the correct word is most likely to be. The N-gram models also limit the search space to words which are statistically most likely to occur. However, there is no generalization over this space, for that reason differences between co-occurrence counts of words in the training and test set will lead to suboptimal results. Therefore we expect that the proposed neural network methods can help to improve existing text recognition systems.

Part III

Results and Discussion

Chapter 7

Results

7.1 Word-frequency compensated sampling

A classifier trained on the MLP context predictions predicted 19.4% of the words correctly when trained without word-frequency compensated sampling (WFCS) enabled. The same model predicted only 7.0% of the words correctly with WFCS enabled. However, the opposite effect was visible on infrequent words. The model trained without WFCS predicted 3.3% of the infrequent words correctly, opposed to 6.2% correct predictions when using WFCS. This is a significant increase ($\chi^2 = 32.1$, df $= 1$, $p < 10^{-4}$).

Similar effects can be seen at the model which uses a BLSTM to make context predictions. The percentage of correctly predicted words decreased from 20.1% to 6.4% when enabling WFCS. However at the same time the percentage of correctly predicted infrequent words increased from 2.1% to 4.8% ($\chi^2 = 35.4$, df $= 1$, $p < 10^{-4}$). For the complete results see Table 7.1.

Table 7.1: Comparison of MLP and BLSTM models trained with and without word-frequency compensated sampling.

	parameters	All (N=6175)		Infrequent words (N=3428)	
		top-1	top-10	top-1	top-10
MLP	8403000	19.4%	45.4%	3.3%	7.6%
MLP with WFCS		7.0%	24.6%	6.2%	11.2%
BLSTM	641600	20.1%	44.9%	2.1%	5.7%
BLSTM with WFCS		6.4%	25.5%	4.8%	9.6%

7.2 Context size

The percentage of correctly predicted words increases when the context window size increases. With the lowest percentage of correctly predicted words of 16.0% measured at a context window size of three and the highest percentage of correctly predicted words of 19.7% measured at a context window size of fifteen words. The number of correct

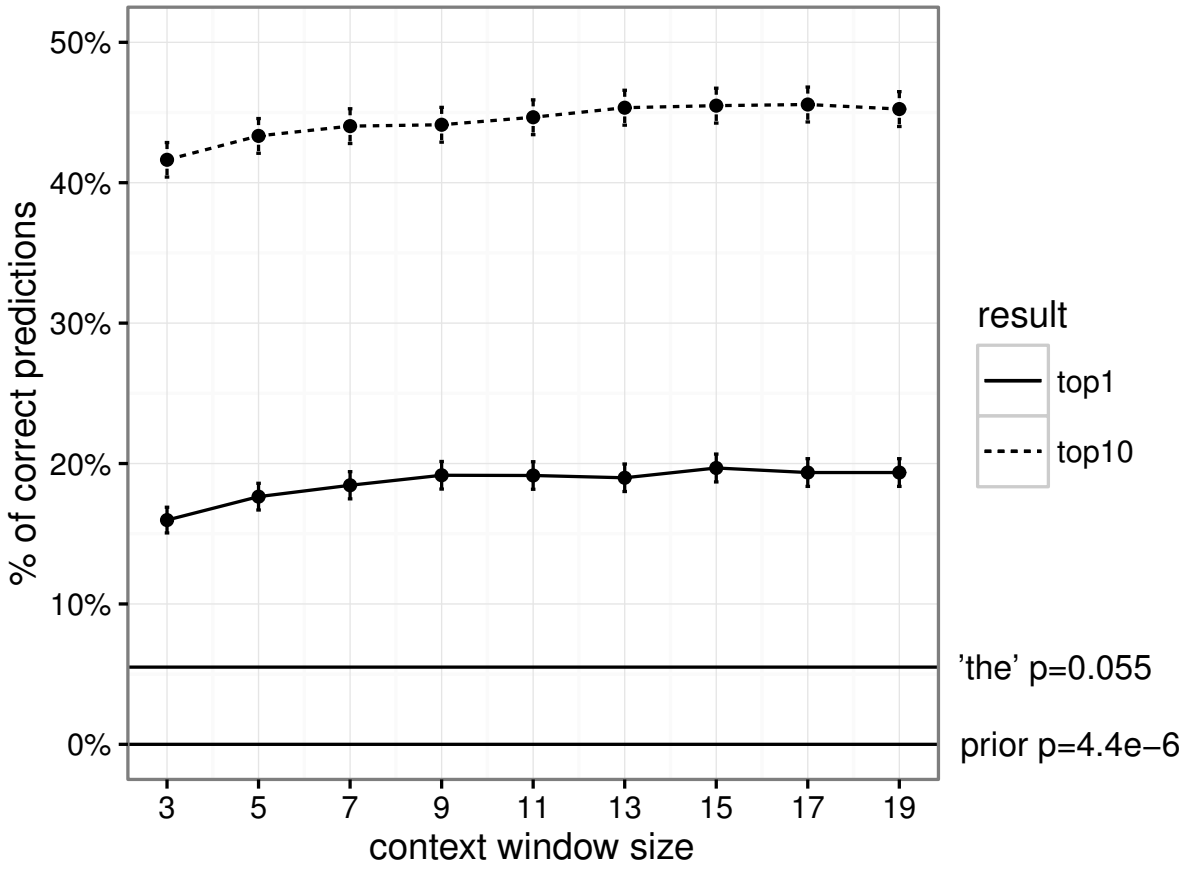


Figure 7.1: Effect of context window size for the MLP model on the prediction of words. Larger is better, but the differences are small. The prior is the expected percentage of correct predictions when making random guesses (1/224974) and 'the' indicates the expected percentage of correct predictions when always predicting 'the'. Error bars indicate the 95% confidence interval calculated using the t-statistic.

predictions decrease with context window sizes larger than fifteen words, however this decrease falls within the 95% confidence intervals.

The top-10 performance measure shows a very similar pattern. With the lowest percentage of correct words among the top-10 predicted words of 41.6% with a context window size of three and the highest percentage of correct words among the top-10 predicted words of 45.6% with a context window size of seventeen words. However, the differences between most of these models are small and fall within the 95% confidence intervals. For the complete results see Figure 7.1.

7.3 N-gram comparison

The bigram model predicted with 12.7% the least words correct of the different context models. The next best model was the trigram model which predicted 14.5% of the words correct, this is better than the bigram model ($\chi^2 = 8.3$, df $= 1$, $p = .004$). Both neural network models predicted more words correctly than the statistical models. The MLP

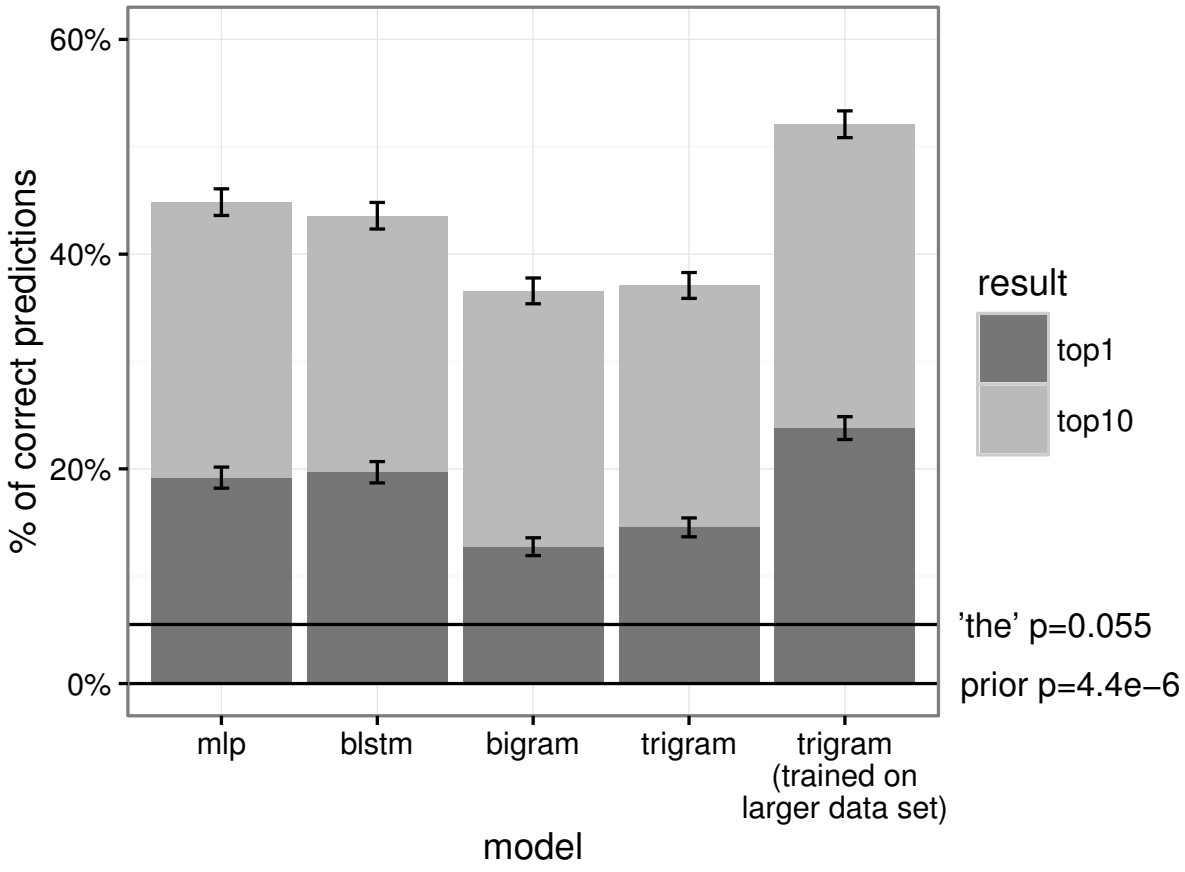


Figure 7.2: Comparison of percentages of correctly predicted words for the MLP, BLSTM, bigram and trigram models, including the trigram model trained on a larger data set. Larger values are better. The prior is the expected percentage of correct predictions when making random guesses (1/224974) and 'the' indicates the expected percentage of correct predictions when always predicting 'the'. Error bars indicate the 95% confidence interval calculated using the t-statistic.

predicted 19.2% of the words correct, which is more than the trigram model ($\chi^2 = 46.9$, df $= 1$, $p < 10^{-4}$), however does not differ from the BLSTM model which recognized 19.7% of the words ($\chi^2 = 0.466$, df $= 1$, $p = .495$). Finally, the trigram model trained on a larger data set than the other models recognized 23.8% of the words correctly, which is more than the BLSTM model ($\chi^2 = 30.4$, df $= 1$, $p < 10^{-4}$). For an overview of the results see Figure 7.2.

7.4 Comparison of MLP and BLSTM models

The root-mean-square error (RMSE) between the targets and the model predictions was calculated per vector dimension for both the MLP and BLSTM models, see Figure 7.3. A Pearson correlation coefficient was computed to assess the relationship between the vector dimension predictions of the MLP and BLSTM models. There was a clear positive correlation between MLP and BLSTM predictions ($r = 0.87, n = 200, p < 10^{-5}$).

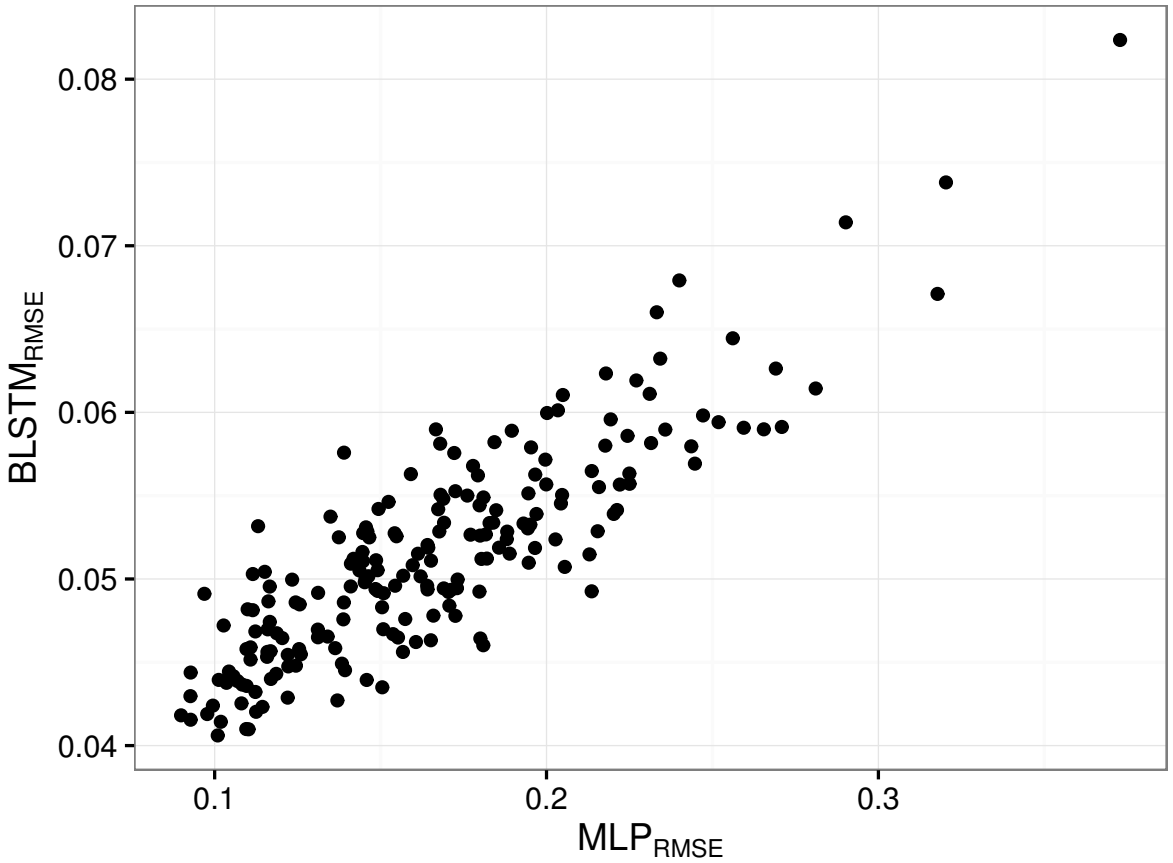


Figure 7.3: Root-mean-square error of predictions of all 200 vector dimensions shows a clear correlation between MLP and BLSTM model predictions. Dimensions with a low RMSE in predictions made by the MLP model tend to have a low RMSE in predictions made by the BLSTM model.

Another interesting part of the results is that the errors for the MLP model predictions are generally larger than for the BLSTM model predictions, for an explanation of this difference see Section 8.5.

A classifier was trained on a combination of the MLP and BLSTM models, this classifier recognized 20.6% of the words correctly. This does not differ from either the MLP ($\chi^2 = 2.8$, df $= 1$, $p = .092$) or the BLSTM model predictions ($\chi^2 = 0.51$, df $= 1$, $p = .475$). See Table 7.2 for the results.

Table 7.2: Comparison of a combination of MLP and BLSTM models (N=6175).

	parameters	correct	
		top-1	top-10
MLP	8403000	19.4%	45.4%
BLSTM	641600	20.1%	44.9%
MLP + BLSTM	9044600	20.6%	47.4%

7.5 Combination of neural network and statistical models

Six different combinations of neural network and N-gram models were tested and compared with the results of the individual models. Combinations of neural network models with either the bigram or trigram model did not predict more words correctly than the neural network models alone. However, a combination of the trigram model trained on a larger data set and the MLP model did lead to more recognized words in comparison to just the trigram model trained on a larger data set ($\chi^2 = 31.0$, df $= 1$, $p < 10^{-4}$). The same is true for the combination with the BLSTM model ($\chi^2 = 31.2$, df $= 1$, $p < 10^{-4}$).

Although the sum-rule is the optimal combination method in all situations except for the combination of the bigram and MLP models, the differences between the tested combination methods are negligible. See Figure 7.4 for a complete overview of the results.

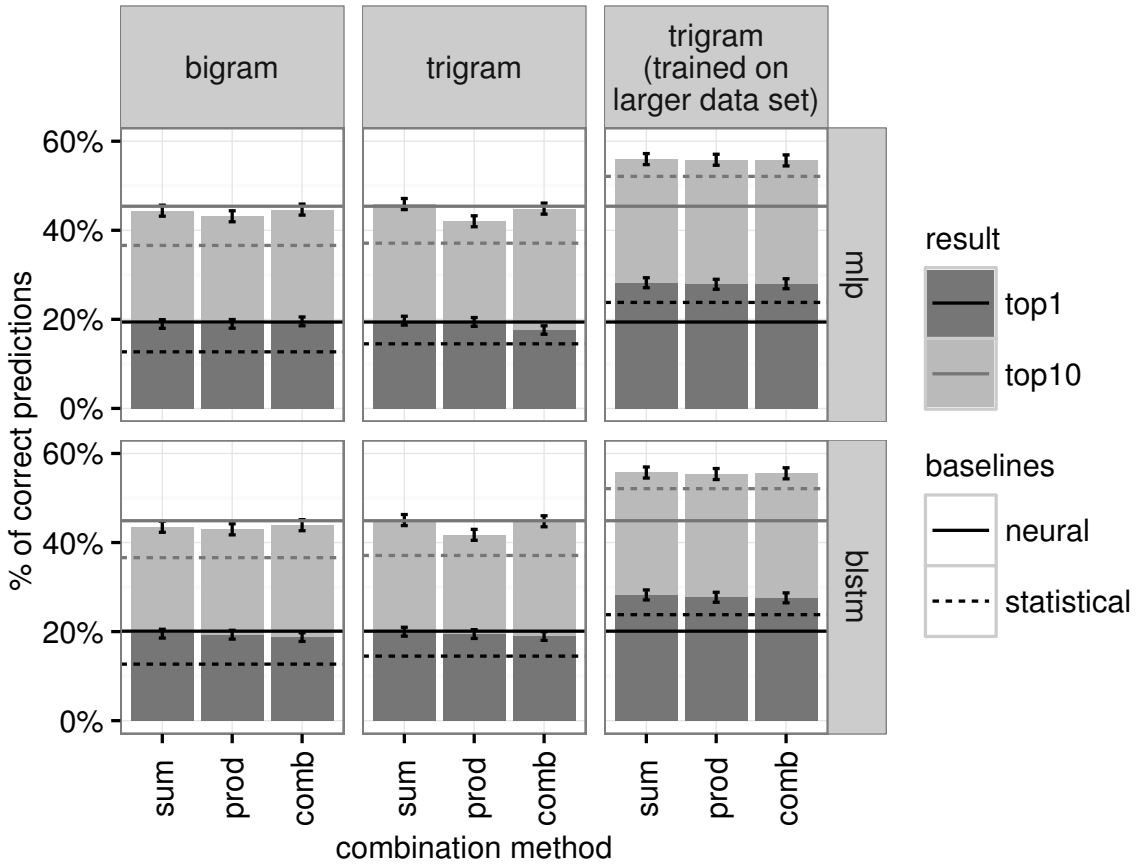


Figure 7.4: Comparison of the percentage of correctly predicted words for combinations of neural network (vertical) and statistical (horizontal) models. None of the combinations perform better than the best of the two individual models it is composed of (baselines), except for combinations with the trigram model which is trained on a larger data set. The differences between the sum-rule (sum), product-rule (prod) and combinator (comb) combination methods are relatively small. Error bars indicate the 95% confidence interval calculated using the t-statistic.

7.6 Extending a text recognition system

The simulated text recognition system (STRS) was able to recognize 36.8% of the words correctly, this was achieved using a classifier without hidden layer.

When the STRS predictions were combined with the MLP predictions using a sum-rule or combinator, no improvement was visible in comparison to just the STRS predictions. When combined using the product-rule the percentage of correctly predicted words increased to 41.0% ($\chi^2 = 22.7$, df $= 1$, $p < 10^{-4}$). And a very similar result can be seen for the combination of the STRS and BLSTM predictions, with an increase to 40.3% ($\chi^2 = 16.4$, df $= 1$, $p < 10^{-4}$) when using the product-rule. However, by far the most words are recognized correctly using a combination of the STRS and neural network models when they are combined in a single classifier. A classifier using as input a combination of the STRS features and MLP word vector prediction recognized 60.7% of the words correctly and a classifier trained on a combination of the STRS features and BLSTM word vector prediction recognized 60.3% of the words correctly. The best results were obtained using a classifier with a hidden layer of the same size as the input.

Combinations of the bigram or trigram predictions with the STRS predictions did not lead to a significant improvement over just the STRS predictions, regardless of the combination method that was used. However, this is not true for the trigram model which was trained on a larger data set. A combination of the STRS predictions with predictions from this model was able to recognize 47.3% of the words correctly when using a product-rule to combine the predictions. For an overview of all results see Figure 7.5.

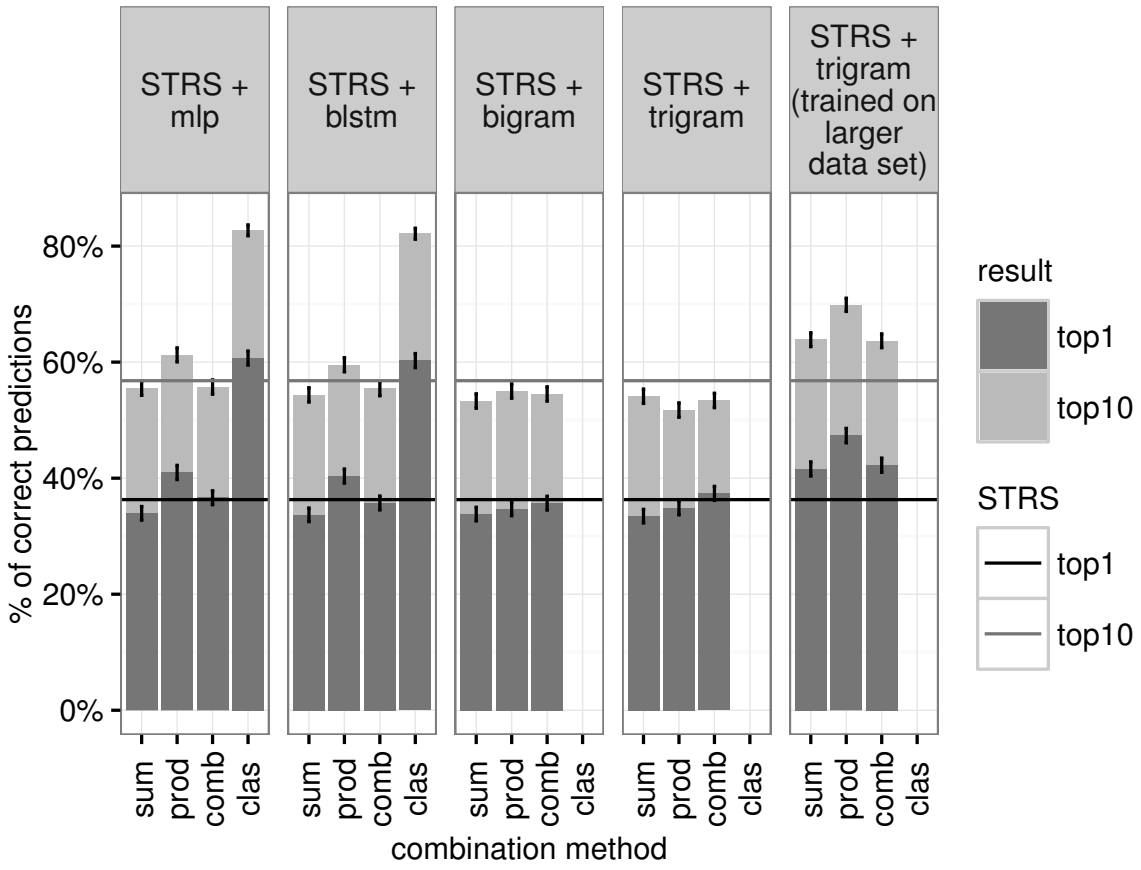


Figure 7.5: Comparison of the percentage of correctly predicted words for a simulated text recognition system (STRS) combined with different contextual models. When contextual models are combined with the STRS using a sum-rule (sum), product-rule (prod) or combinator (comb) they do not give any improvement or only give a relatively small improvement over just the STRS. The exception to this is the trigram model which is trained on a larger data set. However, the largest improvement over the STRS model was gained when it was combined with the neural network context predictions using a single classifier (clas). A similar approach for the statistical language models would be too computationally intensive to calculate. Error bars indicate the 95% confidence interval calculated using the t-statistic.

Chapter 8

Discussion

8.1 Interpretation of the results

This chapter will discuss the results of the performed experiments. First we will discuss the results in general and make some remarks about how these results should be interpreted. After this the individual experiments will be discussed one by one.

Both the MLP and BLSTM models are able to predict around 20% of the words correctly based on just the context. This may seem like a low percentage, however the vocabulary size is 224974 words so 20% is almost 45000 times higher than random guesses and also more than 3.5 times as high as always predicting the most frequent word ('the'). Next to that it is important to realize that the proposed methods were never meant to make perfect predictions, instead the goal was to show whether it is possible to improve on current text recognition systems by using these predictions. The results in Section 7.6 indicate that this is indeed possible.

To get an insight in the predictions of the MLP and BLSTM models, we will discuss three example sentences from the test set and their predictions. Figure 8.1a shows a sentence which is relatively accurately predicted by both neural network models, four out of seven words were predicted correctly. This example also shows that both the MLP and BLSTM models predict the same words correct and incorrect, which is something we saw through the entire test set.

In the example in Figure 8.1b the predictions of the MLP and BLSTM do not seem to make a lot of sense, because both models just return frequent words, such as 'the', 'of' and 'and'. The exception to this is the word 'school' predicted by the MLP model. Syntactically this prediction is clearly wrong, because the word 'school' follows as well, which creates an ungrammatical sequence. However, semantically it is not that strange, because of words like 'book', 'grammar', 'school' and 'learning' in the context. In terms of semantic vector space you could say that the model at least picks a word from the semantic space which corresponds with the context.

This becomes more intuitive by looking at the example in Figure 8.1c. In this example you see that both models often predict words which are semantically similar as the correct word. For example the BLSTM model predicts the word 'cities' instead of 'town' and 'america' instead of 'chile'. One may argue that a statistical language model can also give semantically sensible predictions, for example after the words 'in southern' a trigram model is likely to predict a country or area as well. This is true, however the difference

masked	...	[×]	was	born	[×]	the	surname	thynne	[×]	[×]
	the	spelling	thynn	in	[×]	[×]	seven	six	[×]	...
correct	...	he	was	born	with	the	surname	thynne	but	adopted
	the	spelling	thynn	in	one	nine	seven	six	<period>	...
MLP	...	he	was	born	in	the	surname	thynne	<period>	in
	the	spelling	thynn	in	one	nine	seven	six	<period>	...
BLSTM	...	he	was	born	by	the	surname	thynne	<period>	<period>
	the	spelling	thynn	in	one	nine	seven	six	<period>	...

(a) Both neural network models are able to make correct predictions and often predict the same words correct.

masked	...	the	book	[×]	[×]	grammar	[×]	school	[×]	learning
	[×]	basics	of	[×]	[×]	...				
correct	...	the	book	teaches	basic	grammar	to	school	students	learning
	the	basics	of	german	<period>	...				
MLP	...	the	book	of	the	grammar	school	school	of	learning
	<period>	basics	of	the	<period>	...				
BLSTM	...	the	book	of	the	grammar	and	school	of	learning
	the	basics	of	the	<period>	...				

(b) Both neural network models have a tendency to predict frequent words.

masked	...	parra	was	[×]	[×]	[×]	carlos	[×]	of	uble
	[×]	small	[×]	in	southern	[×]	<period>	...		
correct	...	parra	was	born	in	san	carlos	province	of	uble
	a	small	town	in	southern	chile	<period>	...		
MLP	...	parra	was	a	<period>	by	carlos	and	of	uble
	a	small	<period>	in	southern	europe	<period>	...		
BLSTM	...	parra	was	the	of	of	carlos	and	of	uble
	a	small	cities	in	southern	america	<period>	...		

(c) Both neural network models have a tendency to predict words which are semantically similar to the correct words.

Figure 8.1: Three example sentences from the test set and their predictions.

lies in the fact that if the exact sequence 'in southern' does not occur in the training set, a statistical model will fail to give a sensible prediction. Regardless of whether sequences like 'in northern' or 'in eastern' do occur in the training set. The approach chosen here, using semantic vector space models, can generalize over these semantically similar sequences and is therefore more likely to give correct predictions. Moreover, this is the reason that these models are able to take long-range context into account as well.

8.2 Word-frequency compensated sampling

Table 7.1 shows that the percentage of recognized frequent words is much higher than the percentage of recognized infrequent words. This difference can for a large part be explained by the fact that frequent words occur more often in the training set than infrequent words. This is confirmed by the fact that the difference almost completely disappears when both frequent and infrequent words occur about equally often in the training set due to WFCS. This also means that it is unlikely that the differences between frequent and infrequent words are caused by intrinsic properties of the word vectors. For example, because word vectors of frequent words would be close to most other words.

The percentage of correctly predicted frequent words decreases strongly when using WFCS compared to when not using WFCS. This is true for both neural network models. For this reason WFCS has not been applied in any of the other experiments. For applications in which the statistical properties of the text are not as important, but meaningful predictions are, the measured effects can be desirable. However, in other cases a less aggressive form of sampling frequent words in the training data can perhaps lead to a better balance between the prediction of frequent and infrequent words.

8.3 Context size

The percentage of correctly predicted words does increase when increasing the context window size of the MLP model, as can be seen in Figure 7.1. This corresponds with our expectations. However, the differences are smaller than we expected. Using a context window of three words, 16.0% of the words are recognized correctly compared to 19.7% with a context window of fifteen words. This small difference is probably caused by the fact that local context already gives a lot of the information needed to predict a word correctly, especially for frequent words such as 'the' and 'of'. A larger context window means that more semantic information is available for the recognition process. However, this is especially important for infrequent words which will stay hard to predict even with this extra information. This does also show the difficulty to increase the number of correct predictions, a lot of information needs to be added to get from 16.0% to 19.7% correct predictions.

Apart from an optimal context window size does the experiment also reveal something about contextual information in general. Words directly adjacent to a masked word are to be the best predictors for this word and words further away reveal increasingly less about it. This effect is caused by the variability of long-range context which is much larger than that of close-range context. Words further than four words before or after the current word do not seem to add a substantial amount of information about the current word anymore. However, it is possible that these boundaries will increase when a neural model is trained on a larger data set, as more data can help to explain an increasingly larger variability.

8.4 N-gram comparison

The results in Figure 7.2 show that both the MLP and BLSTM neural network models predict more words correctly than the bigram or trigram model. This corresponds with our expectations on the basis of recent findings in natural language processing, which have shown that language models using a neural network perform better than statistical language models on most natural language processing tasks. Only the trigram model which is trained on a data set more than ten times the size that is used to train the other models, is able to predict more words correctly than the neural network language models. This is not a surprise and an increase in the training data for the neural network models is likely to give a similar improvement. However, training these models on such a large data set will increase the training time even further, which is the reason we unfortunately have not been able to perform these tests.

An explanation for the differences between the neural network models and the statistical models can be the amount of context both models use for their predictions. For this reason it is interesting to compare the trigram model and the MLP model using a context window of three words. Both models have two words as context, the trigram model two words before the word which it tries to predict and the MLP model one word before and one word after the word it tries to predict. The former predicts 14.5% of the words correct and the latter 16.0%. Despite of the configurational differences between the two models, this still shows that if the neural network model only has a limited amount of context available its performance drops till around the levels of the statistical models. For this reason one may argue that N-gram models should include more contextual information, however these models have the disadvantage that the exact sequences of words need to be in the training set to give correct predictions. The number of combinations of words one can make increases exponentially with increasing N. Therefore the size of the training set needed to train an N-gram model which includes a large amount of context grows exponentially too, and no matter how large the training set there will always be a possible combination which it does not cover. Techniques such as smoothing can partially overcome these issues, however we believe that an approach which can generalize over similar sequences of words such as the proposed neural network models are fundamentally a better approach to this problem.

8.5 Comparison of MLP and BLSTM models

From the results in Figure 7.3 we can see that there is a very strong correlation between the predictions of the MLP and the BLSTM models. This means that all vector dimensions are more or less evenly hard to predict for both types of neural networks and is an indication that a combination of both models will not lead to further improvements. As a consequence it is not surprising that a classifier trained on a combination of both models does not lead to a significant higher prediction rate.

An interesting part of the results is that the RMSE of the BLSTM is smaller than the one of the MLP. This can be explained by the fact that both models minimize the cosine distance with the target vectors. A small cosine distance can be achieved by approximating the target values, this also leads to a very small RMSE. However, a small cosine distance can also be achieved by approximating the target values multiplied by a

constant and therefore leads to a larger RMSE. We assume it is easier for the BLSTM to approximate the target values, because the inputs are in the same value range and it is very easy for information to flow through unchanged. For the MLP this is much harder since the inputs are always multiplied by weights which initially have very small values. So although the RMSE is smaller for the BLSTM this does not mean that the predictions are any better.

Another difference between the MLP and BLSTM model is the amount of trainable parameters in both models. The MLP model has 8403000 trainable parameters (with a context window size of 15), whereas the BLSTM only has 641600 trainable parameters. Despite this, the performance of both models is very comparable. Generally a lower number of trainable parameters is preferred, because a small model can be trained with less data than a large model. For this reason and because no context window size needs to be specified upfront, the BLSTM is probably preferred in most situations.

8.6 Combination of neural network and statistical models

The results in Figure 7.4 show that a combination of neural network and statistical models which are trained on the same data set do not predict more words correctly than the neural network model alone, regardless of the combination method that is used. This means that the expected small improvement when both models are combined is not visible in the results and therefore we can not conclude that both models contain complementary information. However, a very interesting effect is that the combinations of neural network models and a statistical model trained on a larger data set does lead to more correctly predicted words than both the neural network and the statistical model alone. This is true for both the MLP and BLSTM models. So a neural network model which is trained on a relatively small data set still contains complementary information to a trigram model which is trained on a much larger data set.

Our explanation for these results is that the neural network models are able to capture statistical properties of the text just as well as the statistical models, but on top of that they also include long-range contextual information which the statistical models lack. So when a neural network model and a statistical model which are trained on the same data set are combined, the statistical model does not contain any information which the neural network model already captured and so the combination does not predict more words correctly than the neural network model alone. However, when a statistical model is trained on a much larger data set a neural network model can still add information to this, because it also uses long-range contextual information.

The effects of the specific combination method that is used to combine the predictions of neural network and statistical models are surprisingly small. It can not be the case that all three the combination methods can not effectively combine the results of these models, because if that was the case the combinations with the trigram model which is trained on a larger data set would not lead to improvements over the individual models. We can conclude that apparently the combination method to combine the predictions matter much less than the quality of both predictions.

8.7 Extending a text recognition system

The results in Figure 7.5 show that the proposed neural network context prediction models can add information to a text recognition system and improve text recognition systems which use an N-gram model to include context. By far the most promising result is the combination of the word vector predictions and the STRS features in a single fully connected neural network classifier. Using this method to add context information to the recognition process the percentage of recognized words increased from 36.8% without the use of any context models to 60.3% using BLSTM predictions and 60.7% using MLP predictions. None of the statistical models were able to improve the STRS close to this amount. However, it needs to noted that these differences come mostly from the combination method using a single classifier and not directly from the differences between neural network and statistical language models.

This combination method basically uses the MLP or BLSTM word vector predictions as a feature vector in the classification process. A similar approach is not possible for the statistical language models as they lack a relatively low-dimensional internal representation, such as the 200-dimensional word embeddings that are used for the neural network models. Instead the N-gram models immediately give probabilities for all words in the dictionary given the context. Theoretically these could be used as a feature vector in a neural network classifier as well, however for large vocabularies this will require a neural network which will be so large that it will take way too long to train such a network in practice.

If you disregard the differences because of the use of different combination methods, the neural network models are still preferred over the statistical models. A combination with the bigram or trigram predictions does not improve the STRS regardless of the combination method that is used. However, the combination with the neural network models lead to an increase from 36.8% when just using the STRS to 41.0% when this is combined with the MLP predictions using a product-rule and to 40.4% when combined with the BLSTM predictions using a product-rule. The successful combination of the STRS with the trigram model trained on a larger data set shows that it is definitely possible to improve a text recognition system using N-gram predictions. However, this is only possible if the N-gram predictions are of a sufficient quality, so that they add more information than noise to the classification process. This is not the case for the N-gram models trained on the same data set as the neural network models, but is the case if they are trained on a data set which is more than ten times as large. The predictions of the neural network models are just of sufficient quality to improve the STRS using a product-rule. These differences between the neural network and statistical models are expected based on the results of the previously discussed experiments which compared neural network and statistical models. These show that the neural network models are able to recognize more words correctly than the statistical models, the reasons for this are already discussed in the Sections 8.4 and 8.6.

Another question you can ask regarding the results is why the combination of the STRS features with the MLP or BLSTM predictions in a single classifier works so well. Our explanation for this is that the neural networks predict the meaning of a word in the form of a 200-dimensional word embedding and the STRS features look at how words look based on textual features. These two types of features complement each other very

well. The textual features limit the search space to all words which have the right textual features and the word embedding predictions limit the search space further to all words which are semantically close to the prediction. There are usually very few words which both have a similar meaning and do textually look the same. For this reason the chances that the system returns the correct word becomes much larger.

Summarizing, the results indicate that text recognition systems can be enhanced by adding contextual information using semantic vector space models. However, the described methods should be applied to a real text recognition system to verify that the proposed methods also work in real-world situations which use more advanced features than the relatively simple STRS features that are used here. Given this remark, we can still try to interpret the results in terms of a real text recognition system. The starting point is the assumption that 70% of the words has already been recognized with a certainty above the acceptation threshold by the text recognition system, the remaining 30% of the words are assumed to be unknown in the proposed methods. Of this 30%, 36.8% can be recognized using the STRS features, these words can be compared with words which are recognized by the text recognition system, but with a probability below the acceptation threshold. If this text recognition system is enhanced with the proposed semantic context predictions of the MLP or BLSTM models, respectively 60.7% or 60.4% of the remaining 30% of the words can be recognized. The text recognition system without any context model would have been able to recognize $70 + 30 * 0.368 = 81\%$ of the words. When the semantic context predictions are added to this text recognition system the recognition rate would increase to around $70 + 30 * 0.6 = 88\%$. Whether a similar improvement would also be possible for a real-world text recognition system which satisfies similar constraints, will need to be tested in a real-world situation.

Chapter 9

Conclusions and future work

The goal of this study was to show whether text recognition systems can be improved by adding semantic context information to the recognition process. A method was proposed to add this information using semantic vector space models. Two different neural network models were implemented to make predictions about unknown words in a text using only contextual information represented in a semantic space. It was shown that these neural network models can make meaningful predictions which can help to improve text recognition systems. Moreover, a comparison showed that these models have several advantages over statistical models to include contextual information.

These results correspond with recent findings that neural network language models perform better on many natural language processing tasks than statistical language models. The reason for this is that neural network language models are able to generalize over similar sequences of words, whereas statistical language models treat words as atomic units and can therefore not generalize over similar sequences. Another advantage of the proposed neural network models is that they can give a meaningful representation of words in a limited number of dimensions. This allows the use of these word embeddings as feature vectors which lead to the most promising results of the presented work.

Summarizing, these results indicate that any type of text recognition system can possibly be improved using the proposed methods. Therefore these results are relevant for research to both written and spoken text recognition. However, more research is needed to see if these results also hold for real-world text recognition systems which may not completely satisfy the constraints that were assumed in the simulated text recognition system which was used in the current work. This was outside of the scope of this research, but would be a logical research goal of a follow-up study.

Furthermore, we expect that the proposed methods can be improved in several ways. One of these improvements could be to use more advanced neural architectures. For example, the proposed methods would probably benefit from a single training phase. Instead of first training a neural network model to predict semantic word vectors and then use these predicted word vectors to train a classifier, a single deep neural network which can learn its own arbitrary internal representation may be more effective. Apart from these can other model improvements, such as stacking multiple BLSTM layers to create a deep-BLSTM, also help to improve context predictions.

Next to architectural improvements could the performance further be improved by using more data to train the models. This is not only true for the training of the neural

network models, but also for the word vectors. The larger the data set that is used to train the word vectors, the more accurate they become. Apart from more data could a larger dimensionality also help to improve the accuracy of the word vectors and thereby make the prediction and classification process easier.

Finally, additional sources of semantic information apart from just word vectors could be considered. It is clear that word vectors are able to capture a lot of the meaning of a word, however this method is still limited to text only. Humans learn the meaning of words not only by reading or listening, but also by other interactions with the world. For example, if we know what an object looks like, we can reason about ways to use it even if we have never read or heard about it before. So instead of just word vectors perhaps additional sources of information, such as images or sounds of words, can help to further enhance text understanding and thereby help to eventually solve text recognition problems completely.

Bibliography

G. Altmann and M. Steedman. Interaction with context during human sentence processing. *Cognition*, 30(3):191–238, 1988.

M. Baroni and A. Lenci. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721, 2010.

M. Baroni, G. Dinu, and G. Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 238–247, 2014.

C.A. Becker. Semantic context effects in visual word recognition: An analysis of semantic strategies. *Memory & Cognition*, 8(6):493–512, 1980.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

A.W. Bronkhorst, A.J. Bosman, and G.F. Smoorenburg. A model for context effects in speech recognition. *The Journal of the Acoustical Society of America*, 93(1):499–509, 1993.

E. Cambria and B. White. Jumping NLP curves: A review of natural language processing research. *IEEE Computational Intelligence Magazine*, 9(2):48–57, 2014.

S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

G. Demetriou, E. Atwell, and C. Souter. Large-scale lexical semantics for speech recognition support. In *Proceedings of the 5th European Conference on Speech Communication and Technology (EUROSPEECH'97)*, 1997.

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

H. Erdogan, R. Sarikaya, S. F. Chen, Y. Gao, and M. Picheny. Using semantic analysis to improve speech recognition performance. *Computer Speech & Language*, 19(3):321–343, 2005.

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(56):602–610, 2005.

A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):855–868, 2009.

J. Hirschberg and C.D. Manning. Advances in natural language processing. *Science*, 349 (6245):261–266, 2015.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37, 2000.

R. Jozefowicz, W. Zaremba, and I. Sutskever. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350, 2015.

D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, 2015. URL `http://arxiv.org/abs/1412.6980`.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

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. 2014.

O. Levy, Y. Goldberg, and I. Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

M.L. Littman, S.T. Dumais, and T.K. Landauer. *Automatic Cross-Language Information Retrieval Using Latent Semantic Indexing*, pages 51–62. Springer US, 1998.

U.V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):65–90, 2001.

T. Mikolov, K. Chen, G.S. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013a. URL `http://arxiv.org/abs/1301.3781`.

T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. 2013b.

W.E. Nagy, R.C. Anderson, and P.A. Herman. Learning word meanings from context during normal reading. *American Educational Research Journal*, 24(2):237–270, 1987.

X. Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014. URL `http://arxiv.org/abs/1411.2738`.

T.G. Rose and L.J. Evett. Semantic analysis for large vocabulary cursive script recognition. In *Proceedings of the Second International Conference on Document Analysis and Recognition, 1993*, pages 236–239, 1993.

T.G. Rose and L.J. Evett. The use of context in cursive script recognition. *Machine Vision and Applications*, 8(4):241–248, 1995.

D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Neurocomputing: Foundations of research. pages 673–695. MIT Press, 1988.

R.K. Srihari. Use of lexical and syntactic techniques in recognizing handwritten text. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 427–431, 1994.

I. Sutskever, O. Vinyals, and Q.V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014.

G.T. Toussaint. The use of context in pattern recognition. In *The Proceedings of the IEEE Computer Society Conference*, volume 10, pages 189–204, 1978.

P.D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1):141–188, 2010.

A. Vinciarelli, S. Bengio, and H. Bunke. Offline recognition of unconstrained handwritten texts using HMMs and statistical language models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):709–720, 2004.

G. Zweig and C.J.C. Burges. The Microsoft research sentence completion challenge. Technical report, Microsoft, 2011.