University of West Bohemia
Faculty of Applied Sciences

Deep Learning Methods for Dialogue Act Recognition using Visual Information

Ing. Jiří Martínek

Doctoral Thesis

Submitted in partial fulfillment of the requirements for a degree of Doctor of Philosophy in Computer Science and Engineering

Supervisor: Doc. Ing. Pavel Král, Ph.D
Department of Computer Science and Engineering

Pilsen, 2022

Západočeská univerzita v Plzni
Fakulta aplikovaných věd

Metody hlubokého učení pro rozpoznávání dialogových aktů s využitím vizuální informace

Ing. Jiří Martínek

Disertační práce

K získání akademického titulu doktor v oboru Informatika a výpočetní technika

Školitel: Doc. Ing. Pavel Král, Ph.D
Katedra: Katedra informatiky a výpočetní techniky

Plzeň, 2022

Abstract

Dialogue act (DA) recognition is an important step of dialogue management and understanding. This task is to automatically assign a label to an utterance (or its part) based on its function in a dialogue (e.g. statement, question, backchannel, etc.). Such utterance-level classification thus helps to model and identify the structure of spontaneous dialogues. Even though DA recognition is usually realized on audio data using an automatic speech recognition engine, the dialogues exist also in a form of images (e.g. comic books). This thesis deals with automatic dialogue act recognition from image documents. To the best of our knowledge, this is the first attempt to propose DA recognition approaches using the images as an input. For this task, it is necessary to extract the text from the images. Therefore, we employ algorithms from the field of computer vision and image processing such as image thresholding, text segmentation, and optical character recognition (OCR). The main contribution in this field is to design and implement a custom OCR model based on convolutional and recurrent neural networks. We also explore different strategies for training such a model, including synthetic data generation and data augmentation techniques. We achieve new state-of-the-art OCR results in the constraints when only a few training data are available. Summing up, our contribution is hence also presenting an overview of how to create an efficient OCR system with minimal costs.

We further deal with the multilinguality in the DA recognition field. We successfully employ one general model that was trained by data from all available languages, as well as several models that are trained on a single language, and cross-linguality is achieved by using semantic space transformations. Moreover, we explore transfer learning for DA recognition where there is a small number of annotated data available. We use word-level and utterance-level features and our models contain deep neural network architectures, including Transformers. We obtain new state-of-the-art results in multi- and cross-lingual DA regonition field.

For DA recognition from image documents, we propose and implement a novel multimodal model based on convolutional and recurrent neural network. This model combines text and image inputs. A text part is fed by text tokens from OCR, while the visual part extracts image features that are considered as an auxiliary input. Extracted text from dialogues is often erroneous and contains typos or other lexical errors. We show that the multimodal model deals with the erroneous text and visual information partially balance this loss of information.

Abstrakt

Rozpoznávání dialogových aktů (DA) je důležitým krokem v řízení a porozumění dialogu. Tato úloha spočívá v automatickém přiřazení třídy k výroku/promluvě (nebo jeho části) na základě jeho funkce v dialogu (např. prohlášení, otázka, potvrzení atd.). Takováto klasifikace pak pomáhá modelovat a identifikovat strukturu spontánních dialogů. I když je rozpoznávání DA obvykle realizováno na zvukovém signálu (řeči) pomocí modelů pro automatické rozpoznávání řeči, dialogy existují rovněž ve formě obrázků (např. komiksy). Tato práce se zabývá automatickým rozpoznáváním dialogových aktů z obrazových dokumentů. Dle nás se jedná o první pokus o navržení přístupu rozpoznávání DA využívající obrázky jako vstup. Pro tento úkol je nutné extrahovat text z obrázků. Využíváme proto algoritmy z oblasti počítačového vidění a zpracování obrazu, jako je prahování obrazu, segmentace textu a optické rozpoznávání znaků (OCR). Hlavním přínosem v této oblasti je návrh a implementace OCR modelu založeného na konvolučních a rekurentních neuronových sítích. Také prozkoumáváme různé strategie pro trénování tohoto modelu, včetně generování syntetických dat a technik rozšiřování dat (tzv. augmentace). Dosahujeme vynikajících výsledků OCR v případě, kdy je malé množství trénovacích dat. Mezi naše přínosy tedy patří to, jak vytvořit efektivní OCR systém s minimálními náklady na ruční anotaci.

Dále se zabýváme vícejazyčností v oblasti rozpoznávání DA. Úspěšně jsme použili a nasadili obecný model, který byl trénován všemi dostupnými jazyky, a také další modely, které byly trénovány pouze na jednom jazyce, a vícejazyčnosti je dosaženo pomocí transformací sémantického prostoru. Také zkoumáme techniku přenosu učení (tzv. transfer learning) pro tuto úlohu tam, kde je k dispozici malý počet anotovaných dat. Používáme příznaky jak na úrovni slov, tak i vět a naše modely hlubokých neuronových sítí (včetně architektury Transformer) dosáhly výborných výsledků v oblasti vícejazyčného rozpoznávání dialogových aktů.

Pro rozpoznávání DA z obrazových dokumentů navrhujeme nový multimodální model založený na konvoluční a rekurentní neuronové síti. Tento model kombinuje textové a obrazové vstupy. Textová část zpracovává text z OCR, zatímco vizuální část extrahuje obrazové příznaky, které tvoří další vstup do modelu. Text z OCR obsahuje často překlepy nebo jiné lexikální chyby. Demonstrujeme na experimentech, že tento multimodální model využívající dva vstupy dokáže částečně vyvážit ztrátu informace způsobenou chybovostí OCR systému.

Declaration

I submit this doctoral thesis for review and defense in fulfillment of the requirements for the degree of Doctor of Philosophy at the University of West Bohemia in Pilsen, Czech Republic. I declare that this doctoral thesis is completely my own work and that I used only the cited sources.

Plzeň, January 10, 2022

Jiří Martínek

Acknowledgement

Here, I would like to express my thanks to Doc. Ing. Pavel Král, Ph.D for his leadership, support, and valuable pieces of advice during my PhD studies. I would also like to thank my family for their continued support and understanding and, last but not least, to Ing. Ladislav Lenc, Ph.D for helpful consultations.

Contents

List of Figures

List of Tables

1 Introduction

"A journey of a thousand miles begins with a single step"
Stephen King

Dialogue act (DA) recognition is an important step of dialogue management and understanding. The goal of this natural language processing (NLP) task is to assign labels to a sequence of dialogue utterances (units of speech) that represent their function in the dialogue: *statement, question, thanking, commit, dialogue opening*, and others [50]. Even though this task is most often related to the processing of an audio (speech) signal within a dialog system, transcriptions of such dialogues in a text form are also available – usually using some automatic speech recognition (ASR) system. Spoken utterance classification is beneficial for dialogue understanding, especially for automatic chatbots. However, dialogues exist also in a form of images (e.g. comic books). This thesis primarily deals with automatic DA recognition from image documents. To the best of our knowledge, this is the first attempt to propose DA recognition approaches using the images as an input.

Since the input is an image, first, it is necessary to segment and recognize the text using optical character recognition (OCR). For this task, we design and employ a deep neural network model based on convolutional and recurrent neural networks as an OCR engine. We conduct a set of experiments with different training strategies for training, including synthetic data generation and data augmentation. Furthermore, we present an overview of how to create an efficient OCR system with minimal costs.

Our next subtask in the DA recognition field is the multilinguality. For all NLP tasks, it is currently very important to have models that are able to be easily adaptable to other languages (ideally for languages with a lack of training samples as well). So the goal is to design a model with acceptable success rates while minimizing human efforts (i.e. manual annotations) for an adaption to other languages. For DA recognition, we propose one general model that is trained by data from all available languages, as well as several models that are trained on a single language, and cross-linguality is achieved by using semantic space transformations. Moreover, we explore transfer learning for DA recognition where there is a small number of annotated data available. We use word-level and utterance-level features and our models contain deep neural network architectures, including Transformers.

For DA recognition from image documents, we propose and implement a novel multimodal model based on convolutional and recurrent neural network that combines text and image inputs. Mul-

timodal approaches in general are currently very widespread and have become state-of-the-art in image document classification and understanding. In such a model, a text part is fed by recognized text from OCR, while the visual part extracts image features that are considered as an auxiliary input. The recognized text is often erroneous and contains typos or other lexical errors. We show that the multimodal model deals with the erroneous text and visual information partially balances this loss of information. Our experiments are made based on the image version of the Verbmobil DA dataset and this chapter is considered as our main contribution presented by this doctoral thesis.

The main contribution of this thesis lies in the exploration of multimodal approaches for dialogue act (DA) recognition based on image documents. We utilize OCR and computer vision algorithms for extracting text. To minimize the impact of potential OCR errors, we use visual information in our models. The motivation for this task is the automatic processing of comic books where dialogues in the written form occur (speech balloons).

In past couple of years, there have been efforts to develop methods to address NLP tasks (including DA recognition) in more languages. Tasks are relatively easy once we find an appropriate dataset. However if such a dataset is not available or it is not large enough, it is necessary to apply different approaches (e.g. cross-lingual models based on semantic spaces or machine translation). Such a case occurs in DA recognition where there is a lack of datasets in more languages and even if they exist there is often a different labeling policy. Therefore we also put our attention on multi- and cross-lingual DA recognition approaches.

1.1 Thesis Contributions

The first goal of this thesis is to provide theoretical knowledge about machine learning approaches with a particular focus on deep neural networks. We present the most popular models with their origins and historical development. The main scientific contributions of this work are summarized below.

1. **Proposing new approaches for DA recognition while focusing on multi- and cross-lingual scenarios**. We created both multi- and cross-lingual models. For cross-lingual scenario we rely on semantic space transformations and transfer learning approaches. Besides, we design a multilingual model that is trained on all languages available;

2. **Design of novel multi-modal deep learning methods for DA recognition in image documents**. We combine text and image inputs and show that using visual information as an auxiliary input is beneficial and improves an overall success rate. To the best of our knowledge, this is the first attempt for DA recognition using images as input;

3. **Proposing an approach to create an efficient OCR system with minimal costs (i.e. minimal human effort during annotation process as well as the minimal time required to train models)**. This includes the proposition of the whole processing pipeline from text segmentation to the final OCR and the exploration of the strategies for training such an OCR model.

1.2 Thesis Structure

The thesis is organized as follows. The theoretical part is covered by Chapters 2 and 3. We first start with the theory of text processing and NLP where we gradually guide you through several text representations and popular neural network models to address related tasks. In Chapter 3, we describe the convolutional neural network (the most popular model for image processing) as we deal with image inputs and we provide the OCR review including our research and contributions.

Chapter 4 provides a summary of our research specialized in historical document analysis, OCR, and, above all, DA Recognition. Within this chapter, we present our relevant conference and journal papers. We highlight the crucial section of this chapter: *Dialogue Act Recognition from Image Documents* where we connect image and text area together by presenting the paper **Dialogue Act Recognition using Visual Information** including the experiments of our multimodal approach. We consider this section to be the main and primary contribution of this thesis. Chapter 5 concludes the whole doctoral thesis.

Last, but not least, we want to emphasize that the structure of this thesis follows the course of a doctoral studies. We progressed from relatively simple isolated tasks to the final task where we employed everything that we have designed, evaluated, and published so far.

"There must be a beginning of any great matter, but the continuing unto the end until it be thoroughly finished yields the true glory."
Sir Francis Drake

2 Deep Learning Survey in Text Processing

"All models are wrong, but some are useful."
George E. P. Box

2.1 Natural Language Processing – Introduction

The area of NLP is a set of computational techniques combining linguistics, computer science, and artificial intelligence that aim to enable computers to analyze, represent and reproduce human language.

The goal is to allow computers to perform a number of human-language-related tasks such as categorization, machine translation, text summarization, information retrieval, or text generation. Tasks such as named entity recognition, text classification, neural machine translation have been well tackled by tools and models which are described below. However, there are some challenges and tasks where the required success has not yet been achieved (e.g. fake news or hoax detection, idiom understanding).

A deep neural network (DNN), mostly composed of recurrent layers, was the best choice for NLP tasks for a long time. However, the current state-of-the-art models make use of **Transformers** which have been recently proposed as an alternative to the recurrent layers to deal with NLP tasks.

In the following text, we gradually summarize the development of the models used in the NLP field (from the first RNNs to the Transformers). Before getting to that, though, we outline common text representations and ways how a text model works with text.

2.2 Text Representation

For most NLP tasks, a raw text must be pre-processed into a suitable form, since machine learning algorithms, including neural networks, require a specific input (numbers, vectors, or matrices).

Early methods were based on simple word representations such as bag-of-words (BOW). These representations were subsequently used in traditional supervised classification algorithms including decision trees, naive Bayes classifier, maximum entropy classifier, or support vector machines (SVM) [64].

2.2.1 Bag-of-words

BOW is a very simple representation that takes into consideration words and their occurrence. Often a limited fixed-sized vocabulary V is produced, and for each sentence, a vector of dimension $|V|$ is created (the dimension is equal to the number of words in the vocabulary). Each component is related to a particular word. Once a word is present in a sentence, a specific component of a vector is one, otherwise zero. Such a binary vector, though, cannot express the frequency of words – so it doesn't distinguish between rare and frequent words. Hence, the frequency (number of occurrences) or TF-IDF (Term Frequency – Inverse Document Frequency) might be calculated and used instead.

To create BOW represenation, following pre-processing steps are usually performed:

1. Split document into sentences;
2. Tokenize sentences into words:
   - Remove punctuation;
   - Lowercasing;
   - Filtering out stop-words – the most common words in a language that bears no information;
   - Other possible operations (e.g. stemming or lemmatization) according to the language we deal with;
3. Create the frequency distribution of words.

Despite the fact that BOW is a very simple text representation method, for simple NLP tasks such as binary text classification (e.g. spam vs. ham), they perform very well.

The main drawback is that once we work with huge training data, the vocabulary size will be enormous and it implies that particular vectors will be very sparse because most of the elements are equal to zero. Another disadvantage is that it doesn't take into consideration the order in which the words appear, which may be desirable for some tasks.

Furthermore, we also have neither information about the context nor the semantic features of the words. Therefore, the development of more sophisticated feature extraction methods emerged – e.g. word embeddings (WE) that have a close relationship to the distributional semantics.

2.2.2 Distributional Semantics

Distrubitional semantics allows to derive the meaning of words only from text where these words occurred. In other words: similar words should occur in similar contexts and similarly distributed words should have similar meaning. The context is based on the size of the "window" (how many surrounding words are considered) [12].

A very simple example of this distributional hypothesis is *Hyperspace Analogue to Language* (HAL) [63] which typically uses local context with a relatively small windows size (typically 4 words to both sides). HAL distinguishes between left and right contexts so the model is a matrix $\mathbb{M} = |W| \times 2|W|$. Besides the simple number of occurrences, the weighting based on word distance is computed because close words have a greater influence on semantics.

2.2.3 Word Embeddings

The goal is to encode the word into a vector of real numbers while embedding high-quality features extracted from the text, including semantic information. Word embeddings (WE) have been increasingly popular because of the continuous performance improvements that their use has demonstrated in a wide range of basic NLP tasks (e.g. [76, 58, 16]).

The dimension of the word vectors is not strictly determined. It depends on the model and is not related anyhow to the vocabulary size. Nevertheless, the vectors with a dimension between 100 and 1000 are commonly used. The assumption that the larger the dimension, the more information we are able to encode and therefore better results, has not been proven [16]. It is possible to achieve reasonable results even with a relatively low dimension (e.g. 100) [16]. Too high dimensions (several thousand), except for the greater complexity of the calculation don't bring any significant improvements. A thorough study about understanding and robustness of word vector dimensionality has been presented by Yin and Shen in [122].

Word embeddings are able to attribute semantic similarities and word analogies, so vectors for semantically close words should be located, in the terms of Euclidean or Cosine distance, close to each other [12].

Mathematically, WE have a close relation to the semantic space. According to Brychcin [12], a semantic space S is a function that projects word w from vocabulary $\mathbf{V}$ into Euclidean space with dimension d.

$$S : \mathbf{V} \mapsto \mathbb{R}^d \tag{2.1}$$

The meaning of the word w is represented as a real-valued vector $S(w)$. Figure 2.1 shows the example of projected word vectors. We have here an interesting observation. In the leftmost image, there is a relationship between words that express male and female words. Similarly, the middle image shows a verb tense (or gramatic) dependencies. Finally, the last image shows dependencies between countries and their capital cities.

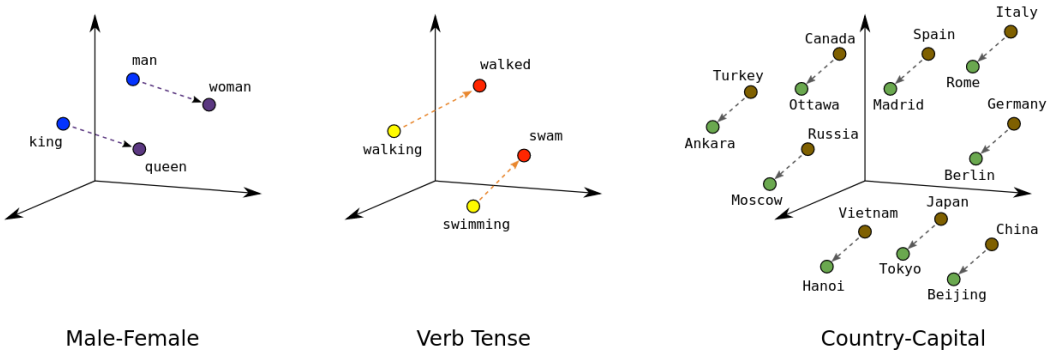


Figure 2.1: Word embeddings analogies and relations[1]

This allows to use calculus with words. So it is possible to add, subtract and multiply individual words as follows:

$$\overrightarrow{\mathbf{v}}_{king} + (\overrightarrow{\mathbf{v}}_{woman} - \overrightarrow{\mathbf{v}}_{man}) \approx \overrightarrow{\mathbf{v}}_{queen} \tag{2.2}$$

[1] https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space

$$\overrightarrow{\mathbf{v}}_{Canada} + (\overrightarrow{\mathbf{v}}_{Moscow} - \overrightarrow{\mathbf{v}}_{Russia}) \approx \overrightarrow{\mathbf{v}}_{Ottawa} \tag{2.3}$$

This sort of meaningful space gives your machine learning system opportunities to detect patterns that may help with the learning task. Word embeddings are most often produced using neural networks and we can divide them into 2 main types:

1. Static;
2. Dynamic (contextualized).

Static WEs are usually generated from pre-trained models (separately from the task-specific model) and a look-up table is required for their usage. One of the first attempts to achieve static continuous representation of words that captures syntactic and semantic created Bengio et al. [7]. Their neural network probabilistic language model used one-hot vectors as an input and they try to learn parameters shared across words to create continuous word vectors. Very slow training though and difficulties to find optimal hyper-parameters were the main drawbacks. Now the most common static representatives are *word2vec* [75], *Glove* [79] or *FastText* [42].

To summarize and compare, let's see the following Table 2.1. The contextual WEs have a very close connection to a model (actually they are a model per se). Static WE can be simply stored because there is only one vector for one word. In the contextual vector representation though, there is $1 : N$ relation between a word and its representations (based on the particular context). So we would need to store all possible representations based on a particular context and this would be very cumbersome even impossible. Instead, the whole model is stored and then load when it is necessary.

Static	**Contextualized**
A static window around each word is used.	The whole context (e.g. a sentence) is used.
Only specific word embeddings are generated.	Both a trained model and word embeddings can be produced.
Different contextual meanings to a single word cannot be achieved.	The polysemy based on different contexts is targeted.
Mostly used only as input.	Embeddings are basically a model and vice versa.
Vectors can be directly stored.	A whole model must be stored.

Table 2.1: Comparison of static and contextualized word embeddings

Word2Vec

The number one representative of static WE model is word2vec [75, 77] proposed by Mikolov et al. This model is based on a fundamental idea: simple models trained on huge data perform better than complex models trained on small data. So the goal was to train a new neural network language model, provide huge data, and as a result get better continuous representations of words. Mikolov et al. [75] proposed two models that share similar components:

1. **CBOW – Continuous Bag-of-words**;
2. **Skip-gram**.

The learnable weight matrix contains the semantic vectors and this component of both models use as a look-up table (see Figure 2.2 and "input" and "projection" components). In the CBOW model, the distributed representations of context (or surrounding words) are combined to predict the word in the middle. In the Skip-gram model, the distributed representation of

the input word is used to predict the context. We can affect the quality of a word2vec model by using a corpus with different domains, bigger training data, CBOW or Skip-gram, increasing the window size of words, and other parameters.

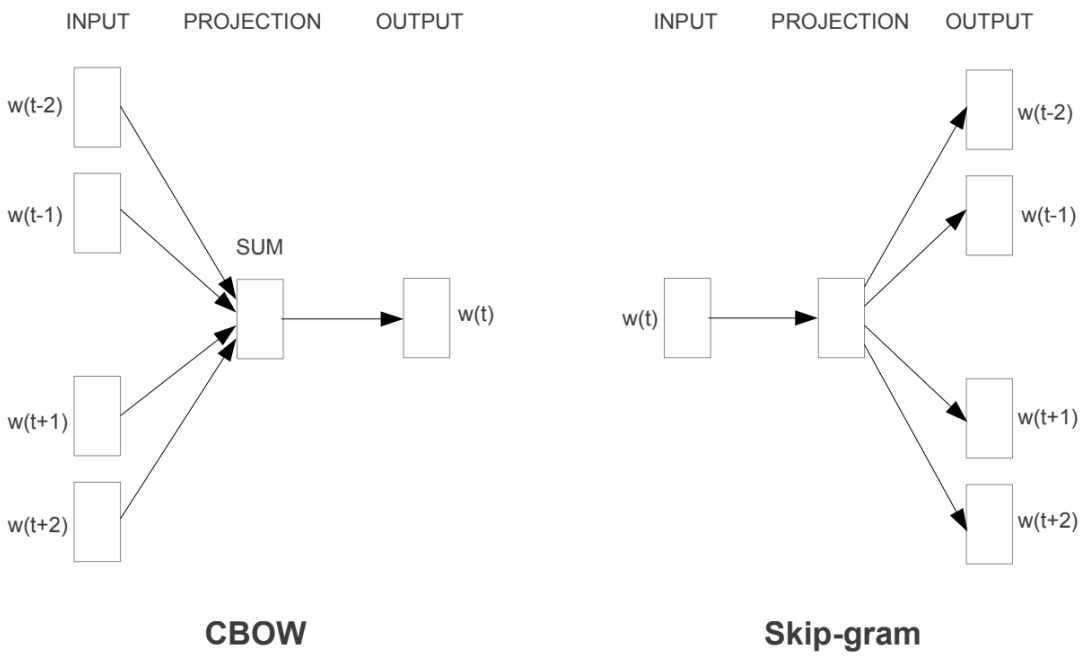


Figure 2.2: Graphical representation of the CBOW model and Skip-gram model. [76]

It is worth mentioning that the authors propose **Negative Sampling**, which causes each training sample to update only a small percentage of the model's weights and as a consequence, it speeds up the training process. The main idea is to train a binary classifier for a true pair (the center word and word in its context windows) versus several noise pairs (picked randomly).

Word2vec superiority against its predecessors can be attributed to the significant computational complexity reduction and its ability to take additional context into account. An example of projected word embedding is depicted in Figure 2.3.

For the word "Berlin", we can see other German cities very close in the space. There are also some other capital cities, predominantly in Europe. Even some German words can be found.

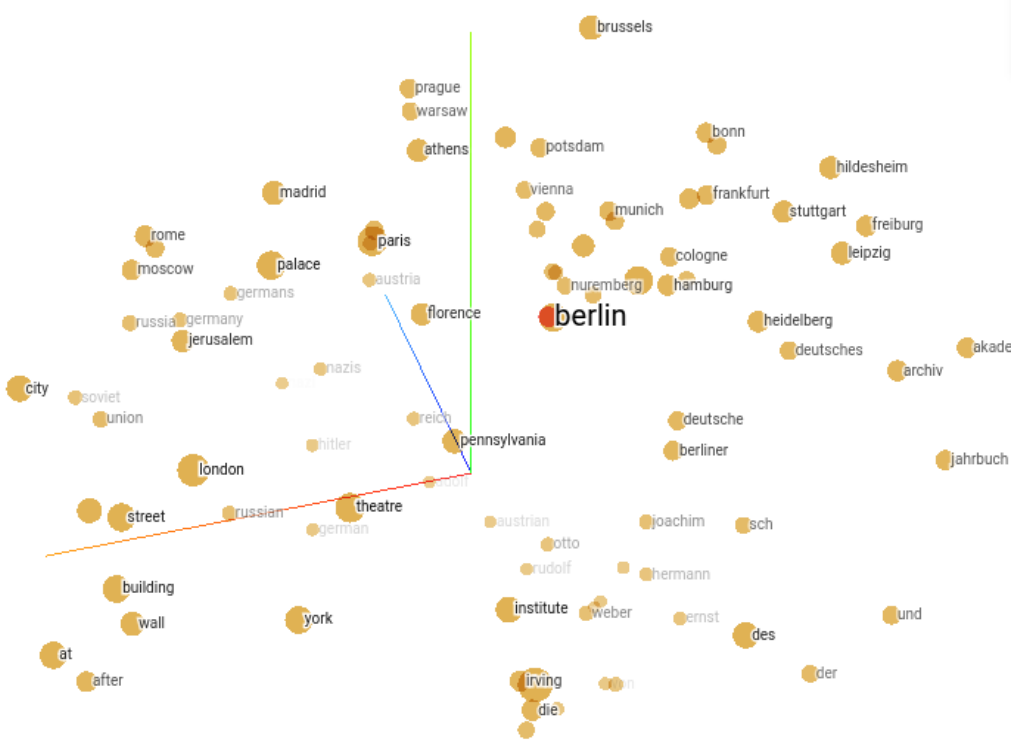


Figure 2.3: Example of projected word2vec word vectors. Only top N closest words are filtered[2]

GloVe

Another worth-mentioning word vector learning technique is Global Vectors – *GloVe* [79]. GloVe, unlike word2vec, takes into account also global statistics (word co-occurrence) by creating a word co-occurrence counts matrix X. The matrix X contains information how many times the word j occurs in the context of word i. This information is then considered in the loss function during training (see term $f(X_{ij})$ in Equation 2.4).

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - logX_{ij})^2 \tag{2.4}$$

Symbol V indicates the size of the vocabulary. The authors state that the weighting function f should obey several properties bearing in mind that rare and frequent co-occurences should not be overweighted. As a result they proposed following.

$$f(x) = \begin{cases} (\frac{x}{x_{max}})^{\alpha}, & \text{if } x < x_{max} \\ 1, & \text{otherwise} \end{cases} \tag{2.5}$$

According to the paper, the cutoff x_{max} is set to 100 and hyperparameter $\alpha = 0.75$. For any more details and further explanation see the paper [79].

[2] https://projector.tensorflow.org/

FastText

In a nutshell, FastText method presented by Joulin et al. in [42], can be seen as an extension of the word2vec model with character *n-grams*, e.g.: `apple = <ap, app, ppl, ple, le>, <apple>`. The goal of this subword (character level) information is to deal with the issue when a particular word is out of vocabulary and also to understand suffixes and prefixes. FastText is able to produce a model for a text classification task directly once pieces of text with associated labels are available. The model maximizes the probability of a correct label given the input text. Due to the fact that is computationally expensive to calculate scores between a piece of text and all labels, the hierarchical softmax is used as an approximation to speed up the training process.

2.2.4 Dynamic Word Embeddings

As we indicated above, the problem with static WEs are, among others, polysemous words (e.g. bank, mouse). Regardless of the context, only one vector is produced. Nonetheless, words are not present in isolation, they are part of sentences. The dynamic (contextualized) word vector representation depends on the entire input sequence, therefore they are able to express a context to a certain extent. The paradigm is outlined in Figure 2.4.

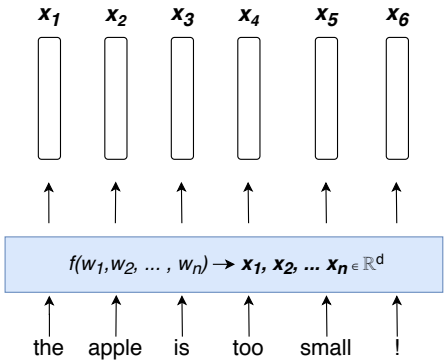


Figure 2.4: Example of a dynamic WE

Formally, to compute contextual vector we use the following formula:

$$\mathbf{x}_k = f(w_k|w_1, w_2, ..., w_n) \in \mathbb{R}^d \tag{2.6}$$

So in the sequel, following two word vectors will be different:

$$\texttt{f(bank | I have built the biggest bank in the world)} \rightarrow \mathbf{x}_{bank}$$
$$\neq$$
$$\texttt{f(bank | A river bank is such a beatiful place)} \rightarrow \mathbf{x}_{bank}$$

CoVe

The first significant contextualised WEs came from Context Vectors (CoVe) [73]. With a development of seq2seq for machine translation and the Attention mechanism [5], there were naturally

efforts to use such architectures for producing WEs. In this approach, there is an LSTM encoder [37] (in the paper labeled as MT-LSTM) which is fed by GloVe WEs [79] (see the following Equation 2.7).

$$CoVe(w) = \text{MT-LSTM}(GloVe(w)) \tag{2.7}$$

The $GloVe$ is a function that produces the corresponding sequence of word vectors based on a sequence of words w. As a result, a sequence of context vectors is produced that is used in an Attention-based decoder. Additionally, the output of the MT-LSTM encoder may be used in NLP downstream tasks (e.g. classification or question answering) as indicated in the paper.

ELMo

In 2018, Embeddings from Language Models (ELMo) [80] have been presented. The centerpiece of the ELMo architecture is two-layer bidirectional Long Short-Term Memory – LSTM [37] which is pre-trained on a large text corpus. Furthermore, there is also a character-level convolutional neural network (CNN) which produces word representations (tokens) for the LSTM. The main reason is to catch the structure of words by producing the sub-word units through convolutions. The main difference between CoVe and ELMo is that CoVe uses a supervised task – machine translation while ELMo uses an unsupervised task – language modeling.

Other Dynamic Word Embeddings

Among other representatives, let us mention ULMFiT [38] and Transformer-based models GPT [85] and BERT [24]. The Transformer [110], as well as BERT, are crucial parts of the modern NLP, and we will devote a whole section for both later. To conclude this section, we present the summary of word embeddings categorization that is depicted in Figure 2.5.

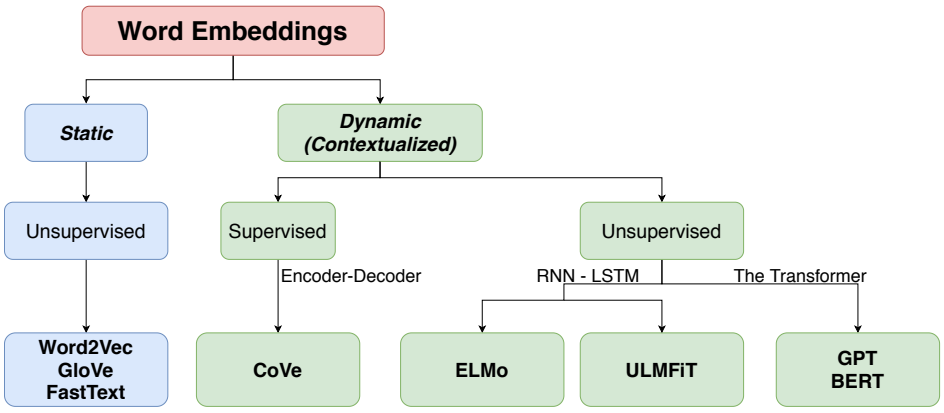


Figure 2.5: WEs – categorization

2.2.5 Multilingual Text Representations

Just before getting to the next section, let us briefly outline the possibilities of multilinguality in terms of word embeddings. Unfortunately, it is very hard to create a representation of the words throughout more languages. Two same sets of words in two different languages might have different semantic spaces due to the different language characteristics. One of the possible ways to unify the semantic spaces is to use a linear transformation.

In a nutshell, linear transformation can be expressed by a transformation matrix $\mathbb{T}$. By the multiplication of the vectors with this matrix, the scaling, translation or rotation can be performed, which results in a different space. See for example Brychcin in [12] where he explores unsupervised techniques to achieve the unified semantic space for different languages.

Since the creation of models based on the Transformer architecture in 2017 (models with self-attention mechanism), there have been efforts to create a huge general model even for multiple languages. Such models are trained unsupervised on large Wikipedia corpus with powerful GPUs for a very long time. They are intended to use in a way of fine-tuning on a specific NLP task. Bellow, we dedicate a whole section to the attention mechanism, the Transformer architecture, and the BERT model as the most famous model from this family of architectures. Last but not least, the multilinguality can be achieved by using two separate semantic spaces and use machine translation to deal with more languages.

2.3 NLP Approaches

This section describes all popular models and architectures related to NLP. We begin with recurrent neural networks with a particular focus on *Long Short-Term Memory* – LSTM [37] and *Gated Reccurent Unit* – GRU [19]. Then we move to the sequence-to-sequence models, the Transformer architecture, and the Attention mechanism. In the last part, we will mention the BERT – currently one of the most popular state-of-the-art models for most of NLP tasks.

2.3.1 Recurrent Neural Network

The architecture of a recurrent neural network (RNN) is developed to handle time-dependent inputs. Text (i.e. sequence of words) is one example of such input. In image processing tasks, these inputs can be columns of values in consecutive pixels, but there are more appropriate approaches such as convolutional neural networks (CNN) that will be discussed later.

The illustration of the RNN is depicted in Figure 2.6. Each input x_t at time-step t influences the hidden state h_t of the RNN which is the output of the current time-step. At the same time, the output h_t is part of an input to the next step (see the unrolled form of the RNN on the right side). At the end of a sequence, the final hidden state contains information based on the last part of a sequence as well the inputs that have been previously seen and processed.

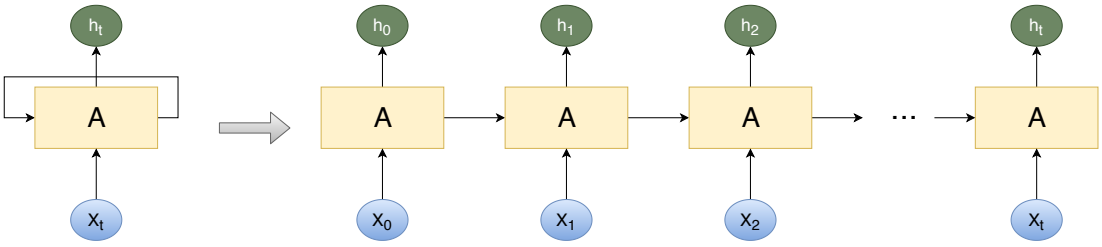


Figure 2.6: Illustration of RNN

Thus, this model can be formally expressed as the following recurrent formula:

$$h_t = f_{\mathbf{W}}(h_{t-1}, x_t) \tag{2.8}$$

The current hidden state h_t is a function of previously seen hidden state h_{t-1} and the current input x_t. The weight matrix W contains the parameters of the neural network model.

The usage of hidden states depends on the type of task. We can either use only the final hidden state (e.g. for text classification) or the sequence of all hidden states which the RNN generates throughout the processing of the whole sequence (e.g. for word embeddings). The main problem with such architecture is that it is very hard to learn preserving information over many time time-steps (i.e. long sequences) because the hidden state is constantly (partially or completely) rewritten.

This kind of network is also prone to the vanishing gradient problem [29]. The gradient is calculated using the backpropagation algorithm and an optimizer (e.g. stochastic gradient descent – SGD) is used to update weights in the network. If the weights in the earlier layers are about to update and the back-propagated gradient has a small (vanishing) value, the weights are barely updated.

The bottom line is that the RNN may forget the initial inputs with the entrance of the new ones and decay of information through time occurs. So the main idea to solve this issue was to come up with a new architecture with separated memory – another component that controls the flow of information and is able to store, modify or delete it as needed.

2.3.2 Long Short-Term Memory

In 1997, Hochreiter and Schmidhuber proposed a type of RNN – Long Short-term memory (LSTM) [37] as a solution to avoid the vanishing gradient problem.

Similarly as standard RNNs, there is also a hidden state h_t for each time-step t, but in addition there is the **cell state** c_t for keeping long-term information. LSTM can control this cell state by adding, modifying, or deleting information. This behaviour is controlled by three corresponding **gates**, namely **input gate** (i), **output gate** (o) and **forget gate** (f). We can imagine the gate mechanism as a valve which is opened, closed, or somewhere in-between. Formally, the gates are vectors and the regulation of information is performed by pointwise multiplications. The values of the gates are not constant, their value is influenced by the current input (context). See the following set of Equations:

$$\begin{aligned} \mathbf{f}_t &= \sigma_g(\mathbf{W_f}\mathbf{x}_t + \mathbf{U_f}\mathbf{h}_{t-1} + \mathbf{b}_f) \\ \mathbf{i}_t &= \sigma_g(\mathbf{W_i}\mathbf{x}_t + \mathbf{U_i}\mathbf{h}_{t-1} + \mathbf{b}_i) \\ \mathbf{o}_t &= \sigma_g(\mathbf{W_o}\mathbf{x}_t + \mathbf{U_o}\mathbf{h}_{t-1} + \mathbf{b}_o) \end{aligned} \tag{2.9}$$

Sigmoid function (σ_g) represents an activation function of each gate, so the value between 0 and 1 expresses how much information is released or kept. Each gate has its own weight matrix ($\mathbf{W_f}$, $\mathbf{W_i}$, $\mathbf{W_o}$) and biases ($\mathbf{b_f}$, $\mathbf{b_i}$, $\mathbf{b_o}$). The matrix $\mathbf{U}$ in each gate represents the recurrent connection between the previous and the current steps. The current input (at timestamp t) is $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ is the previous output (at timestamp $t-1$).

The forget gate controls what is kept (and forgotten) from the previous cell state. The input to this gate are $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$. The first part of the new cell state is computed by pointwise multiplication (a.k.a. the Hadamard product) of the output of the forget gate and the cell state in the previous step: $(f_t \circ c_{t-1})$.

The second gate is the input gate that controls how much information from the current input will influence the current cell state. The input gate values close to zero means not important parts of the input, while the values close to one means very important inputs. Just like the forget gate, the input to this gate are $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$. In order to compute the new cell state, the previous hidden state and the current intput need to be combined to create the new potential cell content $\widetilde{c_t}$ which is computed by the following formula:

$$\widetilde{\mathbf{c_t}} = tanh(\mathbf{W_c}\mathbf{x}_t + \mathbf{U_c}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{2.10}$$

This new cell content $\widetilde{c_t}$ is combined with the input and forget gates resulting into new cell state c_t:

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i_t} \circ \widetilde{\mathbf{c}_\mathbf{t}} \tag{2.11}$$

Finally, the output gate controls how big part of the cell state influence the new hidden state. So the new hidden state is computed as the Hadamard product of the output gate $\mathbf{o_t}$ and the new cell state $\mathbf{c_t}$ with *tanh* function.

$$\mathbf{h_t} = \mathbf{o_t} \circ \tanh(\mathbf{c_t}) \tag{2.12}$$

All vectors used in previous equations have the same dimension. The network architecture is depicted in Figure 2.7

It must be admitted that LSTM per se cannot guarantee to solve the vanishing gradient problem, it provides only better learning of long-distance dependencies. Between 2013 and 2015, LSTMs were the state of the art in most NLP tasks, and they became the dominant text processing tool.

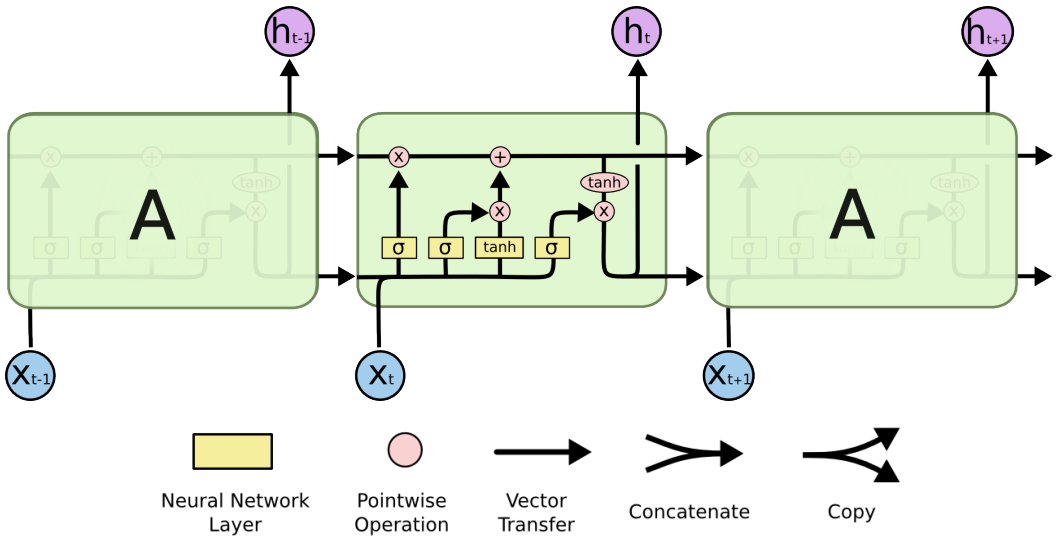


Figure 2.7: LSTM architecture[3]

2.3.3 Gated Recurrent Unit

In 2014 Cho et al. [19] proposed Gated Recurrent Unit (GRU) as an alternative to LSTM. Although GRU is designed more simply than LSTM and generally achieves comparable results, it stands a bit in the shadow of LSTM. In principle, GRU is very similar to LSTM. We have on each time-step t an input $\mathbf{x_t}$ and a hidden state $\mathbf{h_t}$, but there is no cell state.

GRU utilizes only two gates: the **update gate** that controls what parts of the hidden state are updated or preserved and the **reset gate** which decides what parts of the previous hidden

[3] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

state are used to compute new content. GRU is described by the following equations:

$$\begin{aligned} \mathbf{u}_t &= \sigma_g(\mathbf{W_u}\mathbf{x}_t + \mathbf{U_u}\mathbf{h}_{t-1} + \mathbf{b}_u) \\ \mathbf{r}_t &= \sigma_g(\mathbf{W_r}\mathbf{x}_t + \mathbf{U_r}\mathbf{h}_{t-1} + \mathbf{b}_r) \end{aligned} \tag{2.13}$$

$$\begin{aligned} \widetilde{\mathbf{h}}_\mathbf{t} &= tanh(\mathbf{U_h}(\mathbf{r}_t \circ \mathbf{h}_{t-1}) + \mathbf{W_h}\mathbf{x_t} + \mathbf{b_h}) \\ \mathbf{h_t} &= (1 - \mathbf{u_t}) \circ \mathbf{h}_{t-1} + \mathbf{u}_t \circ \widetilde{\mathbf{h}}_\mathbf{t} \end{aligned} \tag{2.14}$$

The reset gate selects useful parts of the previous hidden state and together with the current input, they compute the new hidden content $\widetilde{\mathbf{h}}_\mathbf{t}$. Then simultaneously the update gate decides what is about to kept from the previous hidden state and what will be updated in a potential new hidden state content. See GRU architecture in Figure 2.8.

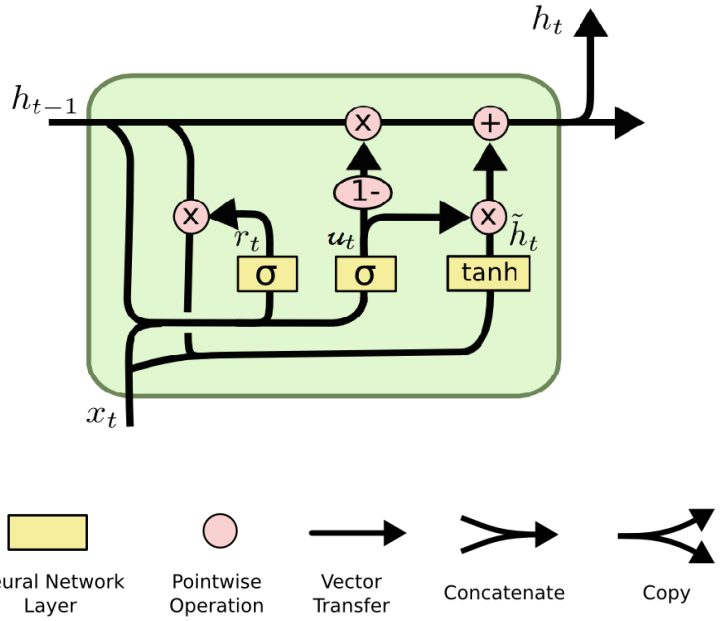


Figure 2.8: GRU architecture[4]

Just like LSTM, GRU is able to retain long-term information (e.g. setting update gate close to zero). During learning, the GRU performs fewer operations on fewer parameters than LSTM which should result in faster training.

Besides LSTM and GRU, let's mention some other RNNs: e.g. Clockwork RNN – 2014, Koutnik et al. [49] and Depth Gated RNN – 2015, Yao et al. [120]. The fact remains, though, that the most widely used RNN for text processing is LSTM.

Last but not least, we must mention bidirectional RNNs. They usually have more layers, and they are employed to deal with both left and right contexts. Naturally, due to the left and right contexts, the bidirectional RNN is only applicable when the entire input sequence is available. An example of general bidirectional architecture is depicted in Figure 2.9.

[4] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

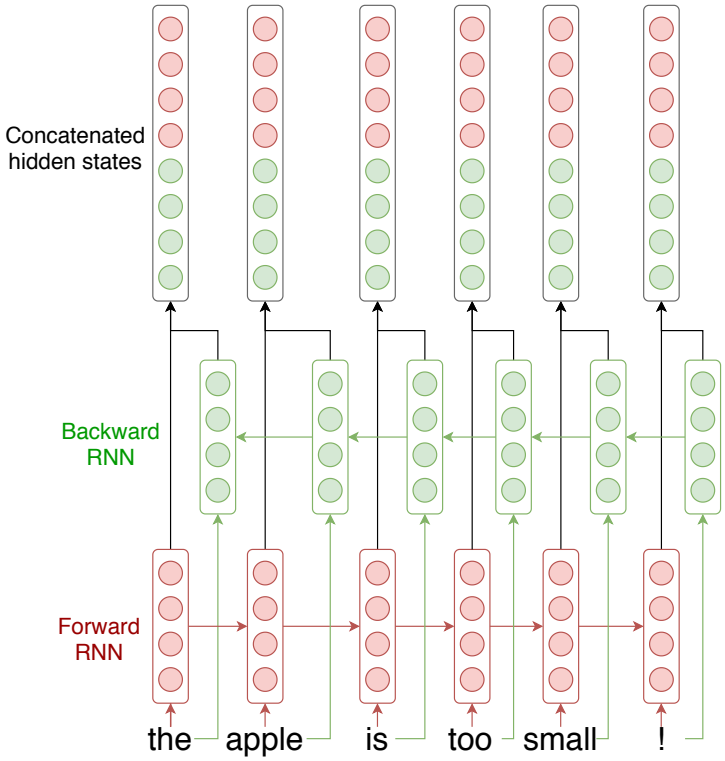


Figure 2.9: Architecture of Bidirectional RNN

2.3.4 Encoder-Decoder and Seq2Seq Models

Next worth-mentioning models in the area of NLP are encoder-decoder and seq2seq. Motivated by the machine translation, the goal is to model a function $f(x)$, where x is the input sequence (e.g. list of text tokens). As a result another output sequence is generated.

Usually the network consits of two components: **encoder** and **decoder** which were originally RNNs (e.g. LSTM). The purpose of the encoder is to produce a latent state in the form of fixed-length encoded vectors based on the input sequence. The decoder learns to generate a target state in time t given the previous targets and the input sequence.

In 2013-2014, several **Neural Machine Translation** (NMT) models were proposed as a way to perform machine translation with a single neural network [5, 106, 18]. The architecture is outlined in Figure 2.10.

For training such a model model it is necessary to provide an appropriate corpus. The encoder RNN produces an encoding vector of the source sentence (the final rightmost red hidden state). This is the input (initial hidden state) for the decoder that can be seen as a language model that generates a target sentence taking mainly into account the encoded vector (i.e. the final hidden state from the encoder RNN). Despite the fact that Bahdanau et al. [5] consider the fix-length vector as a limitation and bottleneck, this architecture has paved the way for many other models and innovations in the area of NLP.

Although machine translation is probably the most common task for seq2seq models, there are some other possible applications (e.g. summarization: long text → short text or dialogues: utterance → next utterance, or any other text generating tasks).

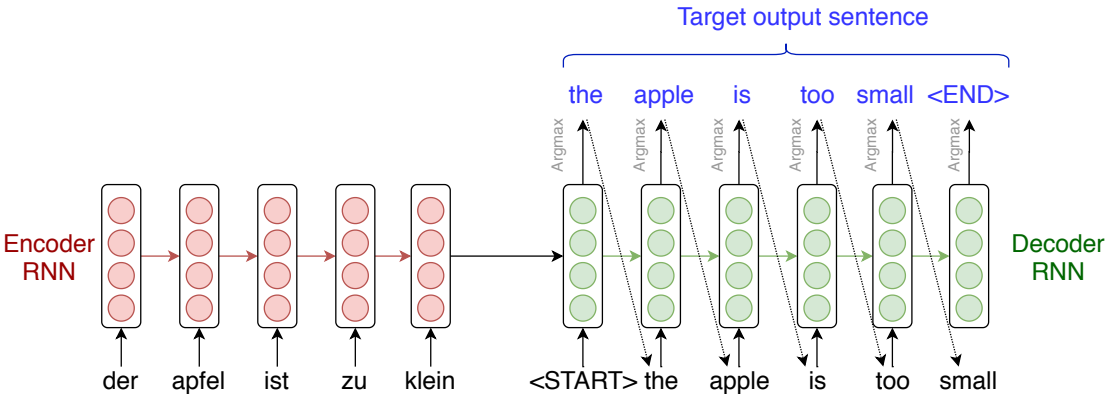


Figure 2.10: Seq2Seq model for neural machine translation

2.3.5 Attention Mechanism

The attention mechanism is an absolutely fundamental improvement of seq2seq models that has been proposed by Bahdanau et al. [5]. A standard seq2seq architecture (see Figure 2.10) has a bottleneck in the input to the decoder because it contains information about the whole input sequence, and some information (especially from the beginning) might be forgot or lost. Moreover, the dimension of the encoder output is fixed regardless the length of an input sequence.

The attention is an "interface" between an encoder and a decoder that provides the decoder with information from every encoder's hidden state (see Figure 2.11). Therefore, it is possible to selective focus on useful parts of the input sequence that is important in the case of longer sequences. In other words, the attention highlights relevant features of the input.

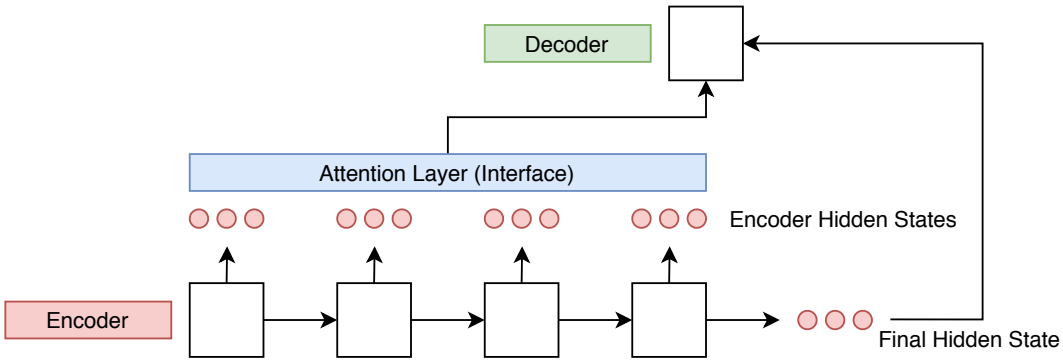


Figure 2.11: Attention interface

The attention mechanism can be divided into four steps as follows:

1. **Prepare the current decoder hidden state at time t and calculate a score for every encoder hidden state**[5].

2. **All scores are fed into a softmax layer.**
   Softmax allows to normalize the score between zero and one, so the high attention scores will be close to one.

3. **Multiplication of each encoder hidden state by its softmaxed score.**
   This step allows to suppress the influence of the given low-scored encoder hidden state and at the same time support those with higher attention scores.

4. **Calculate a context vector and fed it into the decoder.**
   A context vector is an aggregation of the previously computed alignment vectors.

The illustrated attention mechanism in detail shows Figure 2.12. Input to each decoder time-step t is the predicted output from decoder time-step $t-1$ and the context vector produced by the attention mechanism (layer). During training, the input to each decoder time-step t is our ground truth output from decoder time step $t-1$.

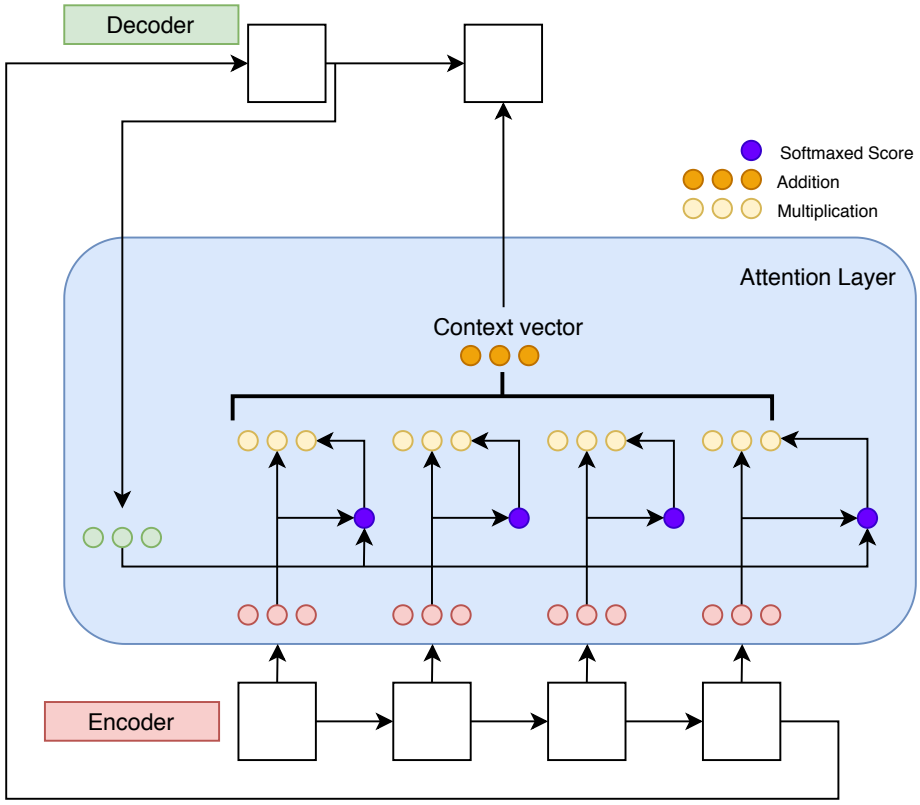


Figure 2.12: Attention mechanism in detail.

[5]Since the dimension of all hidden states are equal, the dot product is often used to express the attention score for given encoder hidden state.

Allow us to illustrate the operations on the following example:

Decoder hidden state: $[10, 5, 10]$
Encoder hidden states: $[0, 1, 1], [5, 0, 1], [1, 1, 0], [0, 5, 1]$

Perform dot product: Scores for each encoder hidden states:
$[10, 5, 10] \bullet [0, 1, 1] = 15$
$[10, 5, 10] \bullet [5, 0, 1] = 60$ (high attention score)
$[10, 5, 10] \bullet [1, 1, 0] = 15$
$[10, 5, 10] \bullet [0, 5, 1] = 35$

Perform Softmax:
$[15, 60, 15, 35] \rightarrow [\sim 0, \sim 1, \sim 0, \sim 0]$

Aligment vectors:
$[0, 1, 1] \cdot \sim 0 \approx [0, 0, 0]$
$[5, 0, 1] \cdot \sim 1 \approx [5, 0, 1]$
$[0, 1, 1] \cdot \sim 0 \approx [0, 0, 0]$
$[0, 1, 1] \cdot \sim 0 \approx [0, 0, 0]$

Aggregation to the context vector:
$[(0 + 5 + 0 + 0), (0 + 0 + 0 + 0), (0 + 1 + 0 + 0)] \approx [5, 0, 1]$

From the example above, we can provide two conclusions:

1. All encoder hidden states except [5,0,1] are surpassed;
2. Next output of the decoder (after this state with high attention score) is going to be heavily influenced by this state.

To conclude this section, let us highlight some important points. The attention has significantly improved the performance of seq2seq models, by focusing on certain parts of the input sequence. Furthermore, it is possible to get an implicitly-learned alignment between the input and output sequences. Moreover, the attention is not only applicable for NMT, but also for other NLP or even image processing tasks (see for instance the RNN with the attention over the image by Xu et al. [119]).

2.3.6 Self-Attention and Transformers

The motivation for the following section is that if the attention is powerful enough to significantly improve the seq2seq model, perhaps we could rely only on the attention and exclude recurrent layers.

An absolutely crucial article for this section is "Attention Is All You Need" from 2017 written by Vaswani et al [110]. Within this paper, they propose new architecture called **the Transformer** for NMT. This model is based solely on the attention mechanism, omitting any recurrent layers entirely. Moreover, it allows much more parallelism than standard RNNs resulting in less time to train.

Before getting to the description of the Transformer, we will briefly introduce the **Self-Attention** which is one of the crucial components of the Transformer.

Self-Attention

The Self-Attention (intra-attention) shares the same concept and many mathematical operations with the attention mechanism that has already been described. On top of that, the Self-Attention allows the inputs to interact with each other (that's why "self"). So without the current decoder hidden state, it is possible to obtain better input representation.

Every input has three other representations – the **key**, the **query** and the **value** that are initialized randomly and its values are learned during a training process. Figure 2.13 shows the summary of all necessary operations.

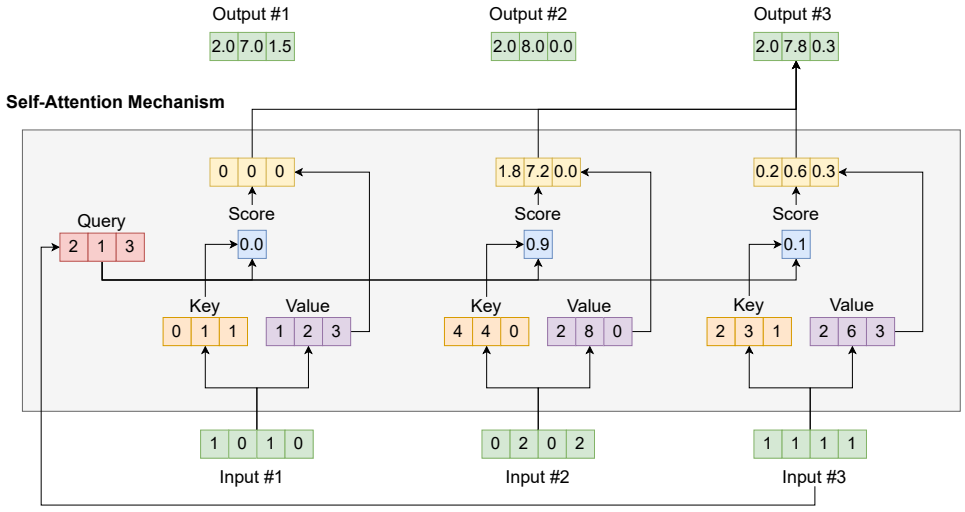


Figure 2.13: Self-attention in detail

The dot product between the **query** (Q) and all **keys** (K) is used for the calculation of the self-attention score (blue square). Softmax scores are multiplied with all **values** (V) resulting in alignment vectors (yellow squares on the top). All alignment vectors are then summed to get the output. The output dimension corresponds to the value dimension.

Allow us to demonstrate the operations on the example according to the Figure 2.13. The dimensionality of the input is 4 whereas the dimensionality of the K,Q, and V representations equal to 3.

The K, Q, and V are randomly initialized matrices:

$$K: \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

$$V: \begin{pmatrix} 0 & 2 & 0 \\ 0 & 3 & 0 \\ 1 & 0 & 3 \\ 1 & 1 & 0 \end{pmatrix}$$

$$Q: \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

Then particular W_K, W_V, and W_Q representations might be derived by matrix multiplication with the "input matrix":

$$W_K = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 \\ 4 & 4 & 0 \\ 2 & 3 & 1 \end{pmatrix}$$

$$W_V = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 2 & 0 \\ 0 & 3 & 0 \\ 1 & 0 & 3 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 8 & 0 \\ 2 & 6 & 3 \end{pmatrix}$$

$$W_Q = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 2 \\ 2 & 2 & 2 \\ 2 & 1 & 3 \end{pmatrix}$$

The matrices W_K, W_V, and W_Q are adjusted during training a network (backpropagation). So by applying this mechanism, given an embedding input x, it learns to produce three separate vectors from it.

The next step is to calculate the attention scores. For the "Input #3" we use the transposed third row of W_q and calculate the dot product with a particular transposed row of matrix W_k.

$$s_1 = \begin{pmatrix} 2 & 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = 4$$

$$s_2 = \begin{pmatrix} 2 & 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 4 \\ 0 \end{pmatrix} = 12$$

$$s_3 = \begin{pmatrix} 2 & 1 & 3 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} = 10$$

Then we perform softmax normalization.

$$softmax(s_i) = \frac{e^{s_i}}{\sum_{j=1}^{len(\mathbf{s})} e_j^s} \tag{2.15}$$

$$softmax[4, 12, 10] \approx [0.0, 0.9, 0.1] \tag{2.16}$$

These scores are the values in blue squares in Figure 2.13. The softmax scores are multiplied with particular rows from W_v:

$$\begin{aligned} 0.0 \cdot [1, 2, 3] &= [0, 0, 0] \\ 0.9 \cdot [2, 8, 0] &= [1.8, 7.2, 0] \\ 0.1 \cdot [2, 6, 3] &= [0.2, 0.6, 0.3] \end{aligned} \tag{2.17}$$

These three alignment vectors are summed to create a final output vector.

$$output = [0, 0, 0] + [1.8, 7.2, 0] + [0.2, 0.6, 0.3] = [\mathbf{2.0}, \mathbf{7.8}, \mathbf{0.3}] \tag{2.18}$$

The final output vector corresponds to the green "Output #3" in Figure 2.13. This can be interpreted as the query representation from the third input, interacting with all keys. The same operations are performed for all inputs available.

Summing up, this mechanism is able to enrich the input with other potentially important information about the interaction of individual parts of the sequence.

Transformer

We remind that the main idea behind the Transformers was to omit recurrent layers and create non-recurrent seq2seq (encoder-decoder) model with self-attention allowing the maximum possible degree of parallelism.

The major component of the Transformer is **multi-head self-attention** [110] that uses multiple **keys**, **queries** and **values** resulting into better representation that takes into account more contexts (i.e. multi-head). Experiments in [110] focus on the machine translation task with excellent results. Figure 2.14 shows the architecture of the Transformer. The architecture reflects the task of machine translation, so for the training, we need a parallel corpus. On the left side, there is the encoder which consists of the multi-head self-attention mechanism and feed-forward layers. The right part is the decoder which purpose is to generate next word based on all inputs and one word before the current one (since the outputs are shifted right).

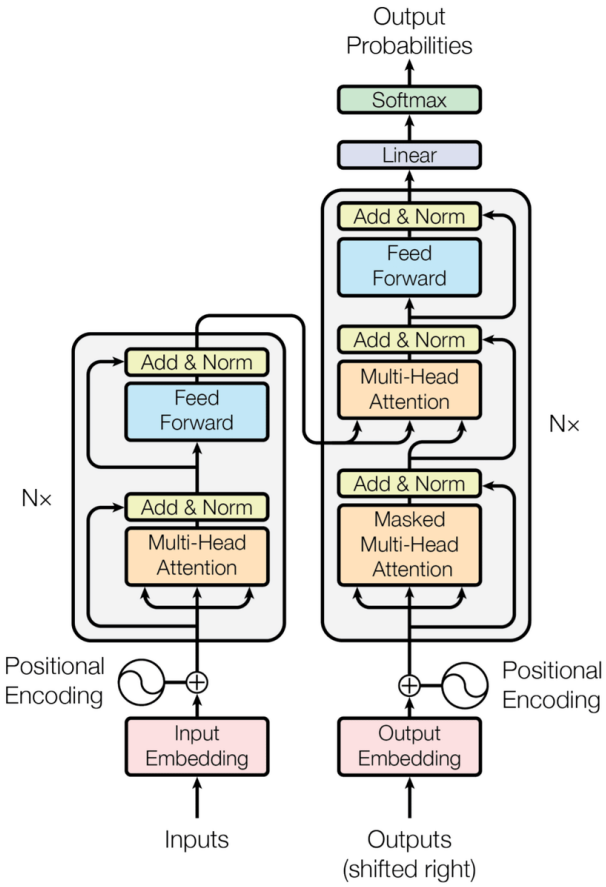


Figure 2.14: The Transformer model architecture [110]

The inputs are transformed into vectors by the positional encoding. As there is no component whatsoever that deals with the sequential nature of the input data, it is necessary to provide information about the relative or absolute position of the tokens of an input sequence. The effect of the positional encoding is tackled by the sine and cosine of different frequencies based on the token position and dimension of the embeddings.

This type of architecture paved the way for greater parallelization than standard recurrent ANNs, resulting in training-time reduction. Moreover, it enables training with much larger datasets.

The first application of transformers in NLP was the Generative Pre-Training model (GPT) [84, 85] that achieved state-of-the-art performances in text classification tasks, as well as other 8 NLP tasks.

A later application of transformers led to BERT [24], which obtained new state-of-the-art results on 11 NLP tasks, text classification included. BERT has been especially influential for the next generation transformer-based models, because of its efficiency, performance, and ease of use in various NLP tasks.

Most of the transformer-based models for NLP tasks contain a large number of trainable parameters which are pre-trained as LMs on large text corpora like Wikipedia. These pre-trained models, like BERT, are available online.

2.3.7 BERT

Bidirectional Encoder Representations for Transformers (BERT) is probably the most significant deep learning model that has made an enormous impact on the recent NLP. It was proposed by Devlin et al. in 2018 [24]. They have reached state-of-the-art results in a wide spectrum of NLP tasks.

The model is pre-trained on Wikipedia and requires further task-specific fine-tuning. In contrast to classical unidirectional language models, BERT generates a bidirectional language model which results in, among other things, the powerful text representation because words can "see themselves" by merging both directional pieces of information.

Devlin et. al came up with an idea to train a language model by predicting masked words in a sentence. See the following example:

```
the man went to the store to buy a gallon of milk.
the man went to the [MASK] to buy a [MASK] of milk.
```

The goal is to mask out $k\%$ of the words for each input sequence, and the training objective is to predict these words based on the context (originally $k = 15\%$).

Naturally, there is a trade-off between the amount of masked and unmasked words. If the ratio of the masked words is low, the model would be too expensive to train. On the other hand, too much masking reduces the context.

The next sentence prediction (NSP) is another training objective that focuses on learning relationships between sentences. See the following example:

```
Sentence A: The man went to the store.
Sentence B: He bought a gallon of milk.
Label = IsNextSentence

Sentence A: The man went to the store.
Sentence B: Penguins are flightless.
Label = NotNextSentence
```

The model creates a binary classifier and learns to predict whether the second sentence is the subsequent sentence in the original text. Two special tokens are inserted. The token $[CLS]$ labels the beginning of the first sentence while the token $[SEP]$ marks the end of each sentence. Furthermore, the positional embedding from the Transformer and sentence embeddings are combined to create an input (see Figure 2.15). The sentence embedding distinguishes between sentence A and sentence B. The top line shows the final input to BERT model including masked words and special tokens.

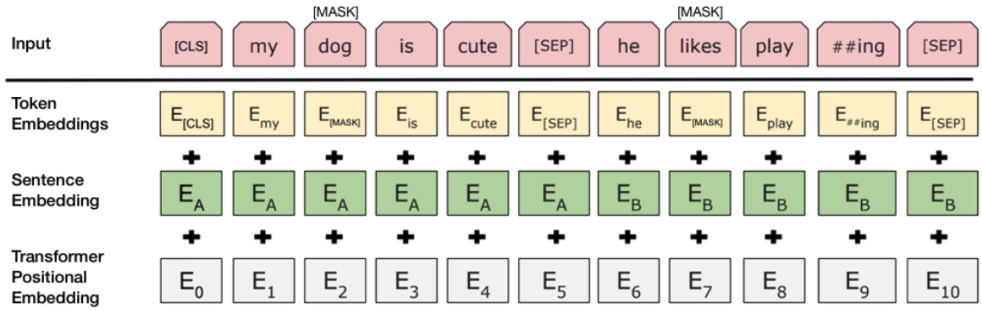


Figure 2.15: BERT input representation [24]

Like we outlined above, the training of BERT model is twofold. The masked language model and next sentence prediction are trained simultaneously to minimize combine loss functions. This process is depicted in the left part of Figure 2.16. The right part of the figure shows an application of the pre-trained "core" to particular tasks (e.g. named entity recognition on Figure 2.17). This fine-tuning phase consists in replacing classification heads (i.e. extra layers at the end of the model that are suited for different use-cases).

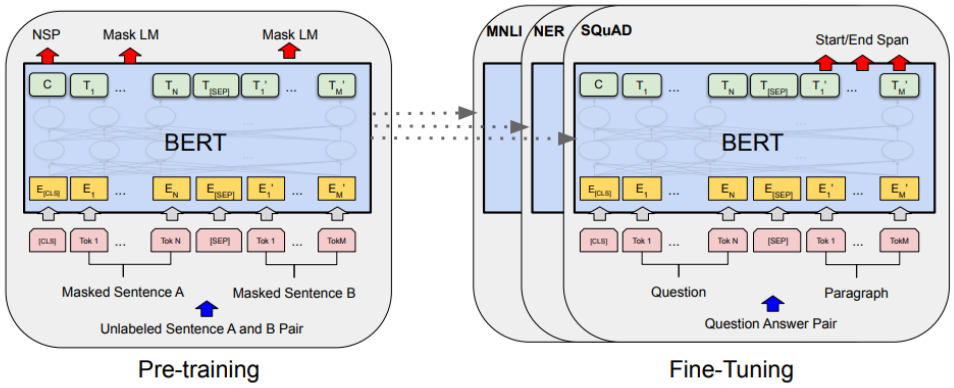


Figure 2.16: BERT pre-trainining (left part) and examples of fine-tuning phase (right part) [24]

As an example, we briefly present named entity recognition (NER) fine-tuning. We remind that the goal of NER is to locate and classify named entities (e.g. names, geografical locations, organizations and others) in unstructured text. So basically we model a function which maps each word (token) to one of desired NER labels.

As a first step, we need to define a set of pre-defined NER labels and provide a dataset. In the case of NER, the usual file format is CONLL file, where each line starts with a word and

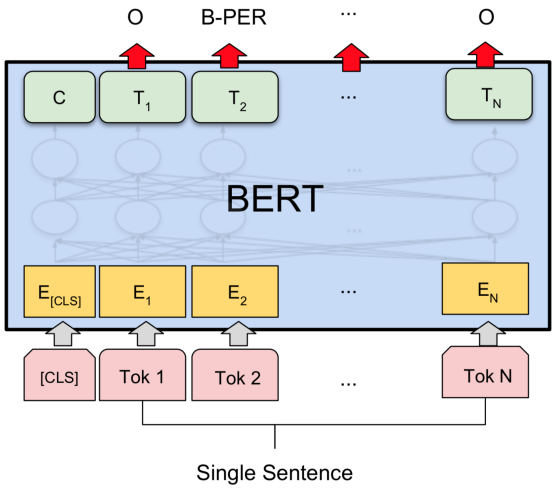


Figure 2.17: BERT NER [24]

next to it is a NER category or "O" if a word is not a named entity.

Wolff	B-PER
,	O
currently	O
a	O
journalist	O
in	O
Argentina	B-LOC
,	O
played	O
with	O
Del	B-PER
Bosque	I-PER

Table 2.2: Example of the CoNLL-2002 NER dataset [93]

Then it is necessary to choose a desired pre-trained model (e.g. bert-base-multilingual-uncased) and build a pipeline for the NER task (i.e. set the NER classification head). After providing a model with training data, we can carry out the fine-tuning phase when we utilize a language model and input representation from the pre-trained model and we optimize the weights of the NER classification head.

2.4 Dialogue Act Recognition Review

Our main NLP-related task in this thesis is DA recognition, hence the last section of this chapter is dedicated to the thorough summary of related work. We remind that DA recognition is an utterance-level classification task whose purpose is to assign appropriate labels depending on their function in a dialogue. Bunt in his paper *Context and Dialogue Control* [13] defined dialogue acts as functional units used by speakers to change the context. DAs are defined by several taxonomies [104] (e.g. questions, statements, backchannels, etc). The DA tag-set including decription potential problems will be outlined in Section 2.4.2. First, we present the most popular relevant datasets and then the methods.

2.4.1 Datasets

An excellent review of the state of the art of the domain is summarised by Ribeiro et al. in [89], where the models typically reach 80% of accuracy on common DA datasets. In the paper, there are descriptions of all relevant datasets including tables with label distributions and results of their character-level approach for DA recognition. The most common datasets are listed below.

- **Switchboard Dialog Act Corpus (SwDA)** [30] SwDA consists of about 2400 telephone conversations among over than 500 American English speakers. The final dataset is a subset of this corpus with manually transcribed conversations resulting into more than 220,000 utterances with about 200 unique tags, but usually for experiments 42-label version is used.
- **Meeting Recorder Dialogue Act (MRDA)** [99] MRDA corpus contains over than 100,000 hand annotated DA labels that come from about 72 hours of speech from 75 naturally-occurring meetings. Moreover, the corpus contains three levels of annotations: *basic label*, *general label* and *specific label*.
- **Verbmobil Dialogue Acts corpus** [40, 2]. It has been used in the past as a representative of the multilingual corpus (see e.g. Reithinger and Klesen [87], Samuel et al. [92] or [67]) It is composed of English, German as well as Japanese dialogues.
- **DIHANA** [6] This corpus consists of 900 Spanish dialogues between 225 human speakers. The total number of manually transcribed and annotated utterances is approximately 23,000. Since the data is based on telephonic train information system, dialogues are very similar to each other. Besides DA labels, DIHANA contains two more levels of labels with detailed information about the particular conversation (e.g. *departure time, day, origin, destination or arrival time*).
- **LEGO corpus** [95] The LEGO is an annotated subset of 347 calls from the Carnegie Mellon University (CMU)' s Let's Go Bus Information System recorded during 2006. It contains approximately 14,000 utterances.

Finally, we mention other DA recognition datasets:
Czech Railways corpus [52], **Mastodon** [15], **DialogBank** [14], or Chinese corpora **CASIA-CASSIL** [123].

2.4.2 Tag-sets

Before we delve into the methods, let's outline potential issues with the variety of DA labels. It is very difficult to define a general DAs tag-set for several reasons:

- DAs labels should be generic enough to tackle different domains;
- DAs labels must be separable as much as possible due to the agreement between the human annotators;
- In some domains, there is a request for having labels to be more specific (e.g. the utterance represents `feedback`, but information about positive or negative might be required like depicted in Figure 2.18);
- Declarative questions are relatively common in dialogues (i.e. a question in the grammar form of a statement and the question is emphasized by the intonation). This fact makes the annotation process also difficult since the utterance is lexically built as a statement, but there is a request for some information so in terms of the semantic, it is a question.

The above-mentioned reasons implies that human annotation process is cumbersome and once a new dataset is published, it is usually, derived from the existing scheme.

It is noteworthy that the variety of DA labels in datasets is an obstacle for effective multilingual and multi-dataset research. Several interesting research efforts have thus emerged to define and exploit generic dialogue acts [74]. However, in practice, the specific requirements of most target tasks prevent a widespread usage of such standards, like we indicated above.

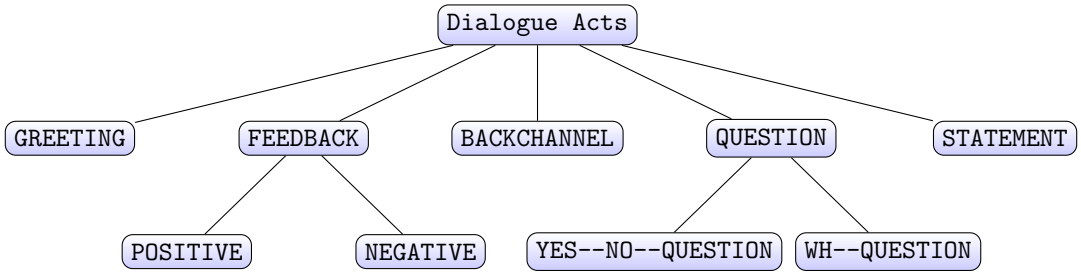


Figure 2.18: Example of the possible structure of dialogue act labels

2.4.3 Approaches

The standard DA recognition input is a speech signal which is usually converted into textual representation using an automatic speech recognition (ASR) system [27]. The combination of the following information sources is often considered for recognition [16]:

- lexical (words in the sentence);
- prosodic (sentence intonation);
- dialogue history (previous sequence of DAs).

Several lexical models are proposed including Bayesian approaches such as n-gram language models [87, 104, 3] and also non-Bayesian methods – semantic classification trees [72], transformation-based learning (TBL) [92] or memory-based learning [91]. Syntactic features that are created using a full parse tree are considered for instance in [51].

Prosodic information is often used to provide additional clues to recognize DAs as presented for example in [100]. Dialogue history can be used to predict the most probable next dialogue act and it can be modelled for example by hidden Markov models [104] or Bayesian networks [45].

The above-mentioned approaches require the feature engineering and knowledge of linguistic characterization. Nowadays, DA recognition is often tackled with modern deep learning architectures which are able to create their own language representation and features.

Ji et al. [41] proposed a hybrid architecture that combines a recurrent neural network language model with a latent variable model that treats the relations between adjacent utterances (DAs). For the task of DA recognition their model is trained to maximize the conditional probability of a sequence of DAs given a sequence of utterances (segments). The evalution on SwDA dataset achieved 77% accuracy.

Khanpour et al. [46] and Duran et al. [25] experiments with LSTM architecture and try to find the best utterance representation. While Khanpour et al. compare word2vec and Glove as a word-level representation, Duran et al. propose new features called "probabilistic word embeddings" which are based on word distribution across DA-set. The experiments show that these features perform slightly better than word2vec.

A comparison of LSTM model with convolutional neural network based word2vec embeddings has been presented by Lee and Dernoncourt in [56]. The authors experimented with different embedding sizes and network hyper-parameters. The models provide context information based on the preceding segments (utterances) and the approach achieved 84% accuracy on MRDA dataset and 71% on SwDA.

In 2020, Shang et al. [96] presented experiments with a deep BiLSTM-CRF architecture with an additional extra input representing speaker-change information. Seq2seq architectures or Transformer based model for DA recognition have also been proposed. Dai et al. [22] fine-tune BERT to classify a single utterance with quite good results, while Wu et al. [117] propose task-oriented dialogue BERT. Colombo et al. [21] proposed a *seq2seq* deep learning model with the attention mechanism and achieved excellent results that are comparable or even better than current state-of-the-art results.

Recently, in 2021 Wu et al. [118] and Wang et al. [111] have highlighted the problem of imbalanced distribution of labels in DA recognition datasets. Wang et al. propose a Hierarchical Label Structured Network with multi-head attention over the input utterances. As an utterance representation, they use composed word-level embeddings based on Glove and also fine-tuned BERT as an utterance-level encoder.

The centerpiece of work by Wu et al. is the two-branch model based on the Bidirectional LSTM model with the self-attention mechanism. An interesting aspect of their model is the interaction between both branches when one branch contributes to the performance of the other based on the composition of the values of loss functions. Furthermore, it uses also human prior knowledge about hierarchy label structure which had been injected.

Although both papers present interesting results while showing to a certain extent robustness to low-frequent DA labels, a thorough comparison based on multiple datasets is not possible. The main reason is that summaries of results on SwDA, MRDA, or Verbmobil (e.g. Ribeiro et al. [89]) are in terms of accuracy while the main metric in both papers [118, 111] is F1-score that is more relevant in the case of imbalanced datasets.

"The best weapon against an enemy is another enemy."
Friedrich Nietzsche

3 Optical Character Recognition

"Those who wish to succeed must ask the right preliminary questions."
Aristotle

This chapter provides theoretical knowledge related to the OCR. We describe the convolutional neural network (CNN) that is broadly used for image classification or object detection which are the most common computer vision tasks. However, they are also very commonly used in the NLP field (as we will describe in Section 3.6). We start with a theory behind image filtering and convolution and then we briefly describe the history of evolving CNN as well as a summary of the most popular models. Then, we provide the OCR review and present our research while highlighting the main contributions:

1. **The exploration of the strategies for training CRNN and their influence on the success rate of OCR** (see our papers *Training strategies for OCR systems for historical documents* [69] and *Hybrid training data for historical text OCR* [71]).
2. **The proposition of the whole processing pipeline from text segmentation to the final OCR with minimal human effort during the annotation process** (see our two journal papers *HDPA: Historical document processing and analysis framework* [59] and *Building an efficient OCR system for historical documents with little training data* [70]).

3.1 Image Filtering and Convolution

The image filtering operators can be used to filter images in order to add soft blur, sharpen details, accentuate edges, or remove noise [108]. We often use the linear combination of neighboring pixels to produce an output value. Such filtering is called the *Linear filtering*.

In this context, it is necessary to mention the concepts of *correlation* and *convolution*. The formula for correlation is as follows:

$$g(i,j) = \sum_{k,l} f(i+k, j+l) \cdot h(k,l) \tag{3.1}$$

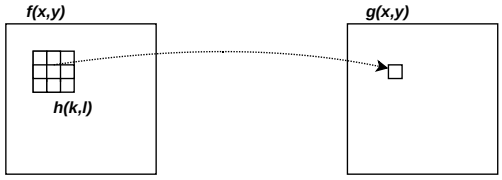


Figure 3.1: Image operators using surrounding pixels

where $h(k,l)$ is the *kernel* (or the *mask*). Nonetheless, the convolution operator is much more frequent and can be expressed as:

$$g(i,j) = \sum_{k,l} f(i-k, j-l) \cdot h(k,l) = \sum_{k,l} f(k,l) \cdot h(i-k, j-l) \tag{3.2}$$

The sign for this operation is $*$, so it is possible to use the shortened version of the equation. Figure 3.2 shows an example of discrete convolution of an image f with a kernel h and the resulting image g.

$$g(i,j) = f * h \tag{3.3}$$

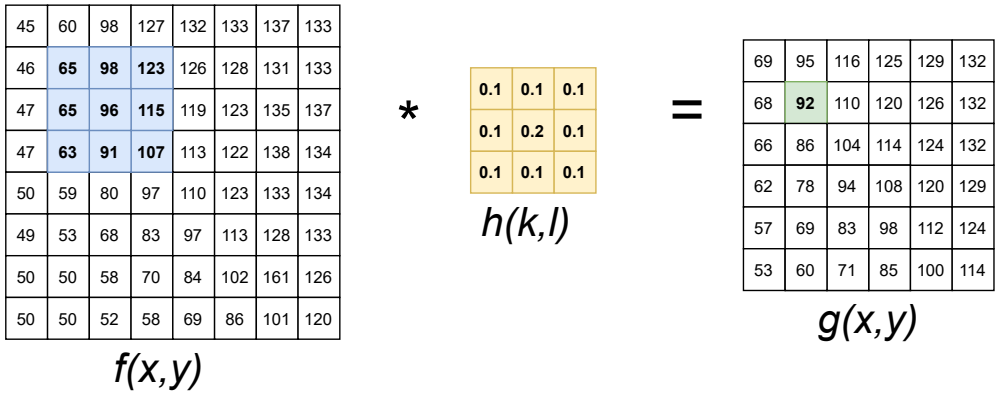


Figure 3.2: Neighborhood filtering (convolution): The image on the left is convolved with the filter in the middle to yield the image on the right. The blue pixels indicate the source neighborhood for the light green destination pixel [108].

The significant representative of image filtering operators is the *Gaussian Filter* that is very effective in removing noise from the image (in particular, the Gaussian noise which the probability density function reminds the normal distribution). Last but not least, there are also non-linear filtering methods (e.g. Median or Bilateral filtering) that are worth mentioning. For a detailed description see relevant literature ([31, 81]).

3.2 LeNet and Origins of CNN

CNNs were first introduced in the late 1980s by Yann LeCun. Their first application was the task of handwritten digit recognition [54] and at that time it was called LeNet, after the author's name. Within his work, it has been shown that for 2D shape recognition, CNN eliminates the

need for hand-crafted feature extractors. Therefore, it was possible to feed the network with raw inputs (normalized images) and to rely on backpropagation to turn the first few layers into an appropriate feature extractor.

Once a feature has been detected, its exact location becomes less important, as long as its approximate position relative to other features is preserved. Therefore, each convolutional layer is followed by an additional layer that performs sub-sampling (average or max pooling layer), reducing the resolution of the feature map, and reducing the sensitivity of the output to shifts and distortions [55].

Furthermore, two basic properties: shared weights and subsampling bring invariance with respect to small geometric transformations or distortions and also they reduce the number of network parameters. Last, but not least the network could be trained on a low-level representation of data that has minimal preprocessing [54] (as opposed to elaborate feature extraction).

Figure 3.3 shows an example of CNN architecture (LeNet). Typically, there is a number of convolutional layers with convolution kernels (filters) followed by pooling (subsampling) layers. As a final layer, it is usually used a fully connected layer. The filters are the base of local connections that are convoluted with the input and share the same parameters to generate feature maps. In the next two sections, the convolution operation and subsampling techniques will be explained.

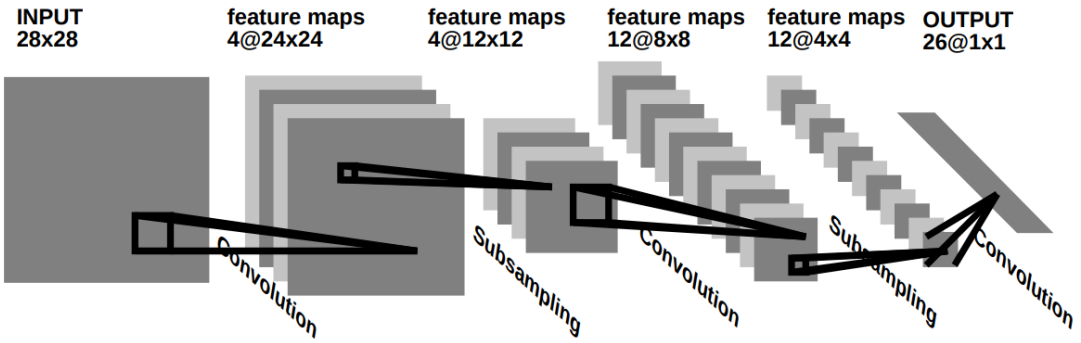


Figure 3.3: Example of the CNN for image processing [55]

3.3 Convolution operation

The convolution is a mathematical operation of two functions (f and h) that produces another function $y = (f * h)$ and it is defined by the following integral.

$$(f * h)(t) = \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau \tag{3.4}$$

In a continuous domain, we can intuitively imagine two signals (functions) when one is shifting and overlapping over another. The greater the overlap of f and h while shifting from $-\infty$ to ∞, the greater the magnitude of the convolution as depicted in Figure 3.4

In a discrete domain (e.g. image processing), the convolution operation is defined as:

$$(f * h)[m,n] = \sum_j \sum_k h[j,k]f[m-j,n-k] \tag{3.5}$$

where f is the input image and h is the convolutional kernel. Indices m and n are concerned with the image matrices while j and k are the kernel indices. Similarly, the kernel (h) slices over the image (f) and it creates a feature map. The equation is very similar to the formula that has been described earlier while describing correlation and convolutional operators.

The convolutional layer computes a dot product between the weights (kernel) and its inputs in the way depicted in Figure 3.5. It is noteworthy that we are not limited strictly to the 2D domain. We can use kernels and inputs with arbitrary dimensions with regard to data available.

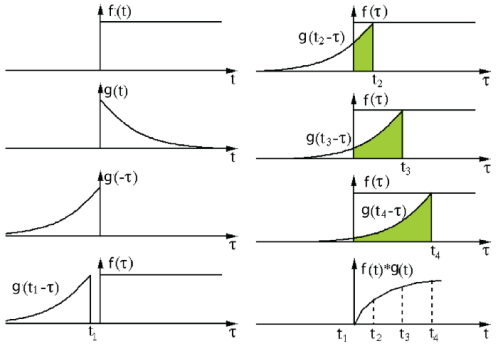


Figure 3.4: Visualized convolution of two signals (functions)[1]

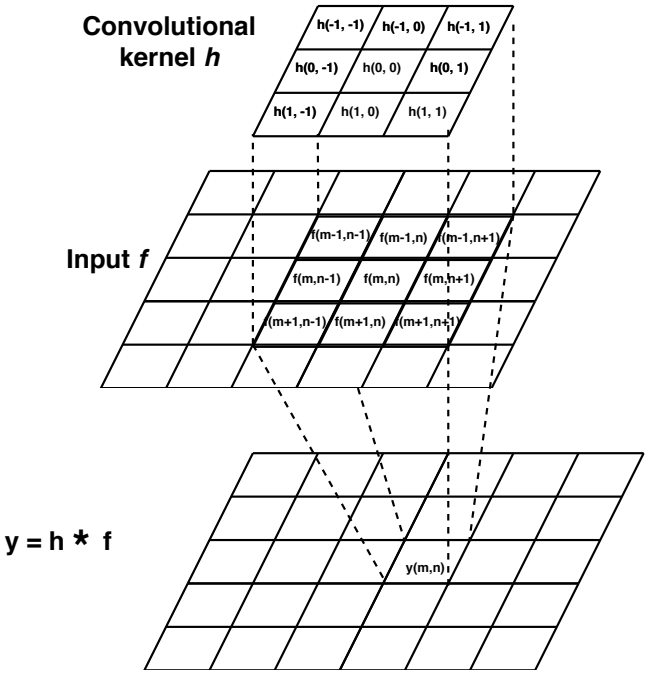


Figure 3.5: Visualized digital convolution with a kernel 3×3

[1] http://fourier.eng.hmc.edu/e161/lectures/convolution/index.html

3.4 Subsampling – Pooling

The purpose of the subsampling layer is to downsample a feature map in order to decrease the number of parameters. The natural property of the CNN is the reduction of dimensionality because with each increasing layer with fewer neurons, the number of parameters decreases and the dimension as well. In the final consequence, it reduces the risk of overfitting.

Figure 3.6 shows two main subsampling methods. In general, max-pooling extracts the most important features like edges. Average pooling takes basically all, even features that might not be important for object detection.

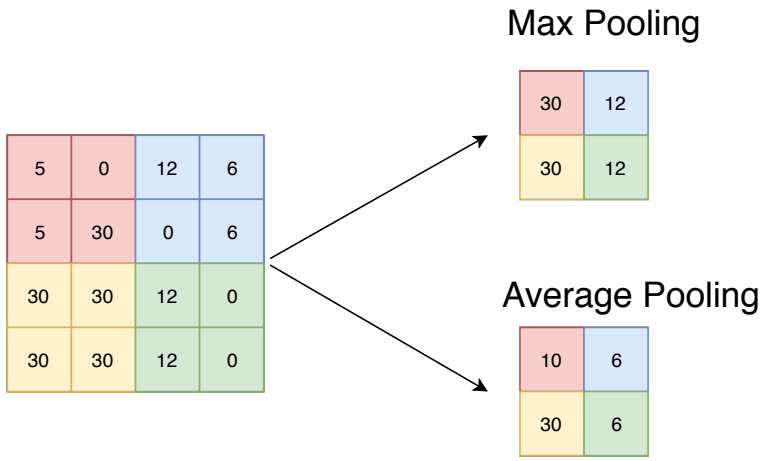


Figure 3.6: Illustration of the max and average pooling

3.4.1 Padding and Strides

Similarly as other layers, the convolutional and subsampling layer has set of parameters (weights) and an activation function. In addition, there are a number of other parameters, of which the `padding` and the `strides` are the most important.

The stride parameter specifies in which way the kernel is moving over the input. More precisely it expresses the number of strides of the operation (convolution or subsampling) along the height and width. By default the convolutional strides step is one for each dimension. In a subsampling layer (e.g. MaxPooling), it specifies how far the pooling window moves for each pooling step – by default it is according to the pool size (without overlapping values).

The padding defines a way of creating the resulting feature map especially in the edges of the input where the convolution/pooling is carried out in a slightly different way. Basically, there are two options:

1. Applying padding to the input image so that the input image gets fully covered by the filter and specified stride. This option is called **same** since, for stride 1, the output dimension will be equal to the input dimension.

2. Applying no padding, resulting in a smaller output dimension.

Figures 3.7 and 3.8 shows both options.

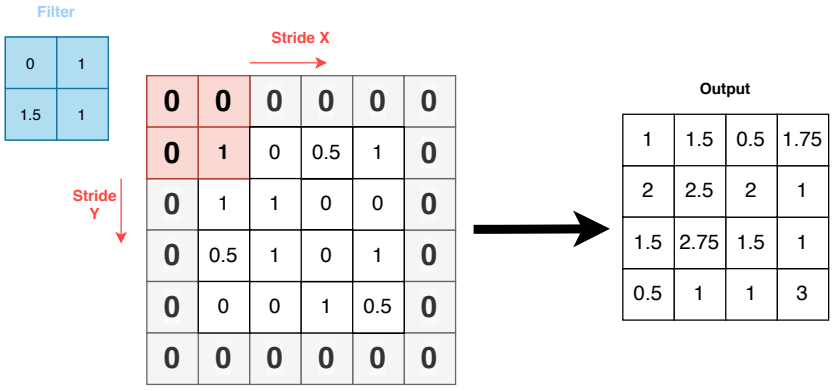


Figure 3.7: Illustration of zero padding (`padding=same`) with stride = 1 in both directions

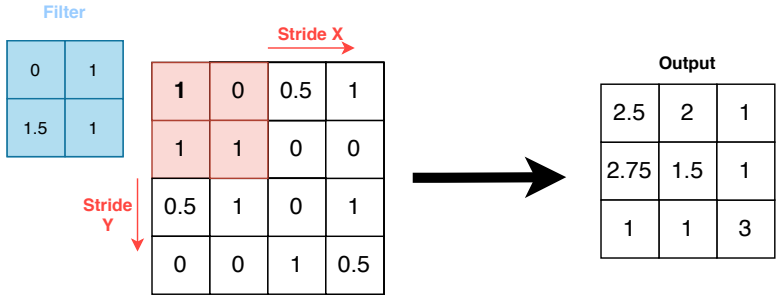


Figure 3.8: Illustration of no padding (`padding=valid`) with stride = 1 in both directions

3.5 Further Development and Modern Architectures

The problem that CNNs encountered was scaling [53]. It was very difficult (nearly impossible) to train a network capable of efficiently processing large images. They were only applicable to images with relatively low resolution.

With the development of GPUs and deep learning and availability of large datasets (e.g. *ImageNet*) with a huge amount of annotated pictures, there were efforts to create complex CNNs capable of tackling computer vision and image processing tasks that were earlier very hard to perform.

3.5.1 AlexNet

In 2012, Alex Krizhevsky et. al [53] trained a large, deep convolutional neural network (the *AlexNet*) to classify the 1.2 million high-resolution images in the ImageNet Large-Scale Visual Recognition Challenge (LSVRC) contest into the 1000 different classes.

The architecture consists of eight layers with ReLU activations (the first five convolutional layers and the last three fully-connected layers). See Figure 3.9.

The LSVRC dataset consists of variable-resolution images. Their model though requires a constant input dimensionality. Therefore, a down-sampling of the images to a fixed resolution of 256 ×256 was conducted.

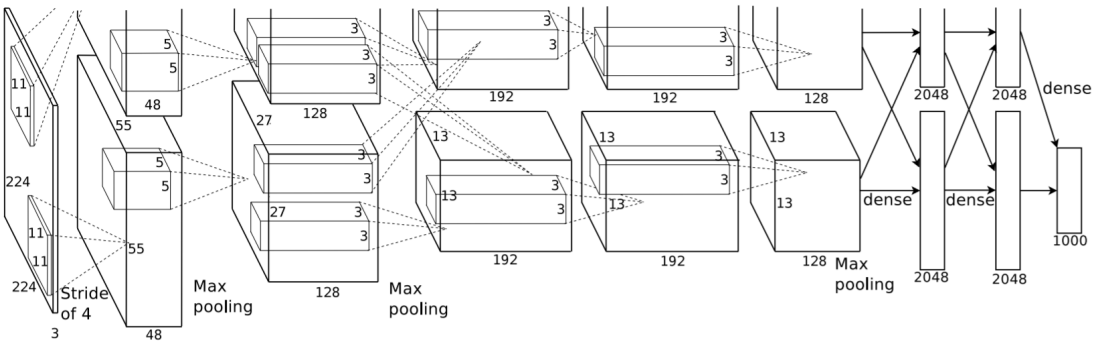


Figure 3.9: AlexNet architecture [53]

3.5.2 GoogLeNet

Two years later (2014) in paper "Going deeper with convolutions" [107], there has been proposed the *GoogLeNet* – a 22 layers deep neural network with 4 mil. parameters. They become the new state of the art for image classification and detection.

Compared to *AlexNet*, there was a significant reduction in network parameters, among other things thanks to the so-called inception modules [107]. The input image size is 224×224.

3.5.3 VGGNet

Another significant example of modern CNN is the *VGGNet* presented in [102]. The authors used huge number of very small filters 3×3 resulting into 138 mil. parameters. Similarly to *AlexNet* and *GoogLeNet* the input image dimension is fixed 224×224. The original version has 16 layers *(VGG-16)* but there is also the version with 19 layers (*VGG-19*).

3.5.4 ResNet

Last but not least, we mention the *ResNet* from 2015 [36]. It utilizes residual connections – skip connections allowing to ease a training, trying to tackle vanishing gradient problem. In a nutshell, they are shortcuts, which enable the error backpropagation and identity propagation even for very deep structures as shown in Figure 3.10. There are also more versions of ResNet (ResNet-18, ResNet-34 ResNet-50) depending on the depth of the network.

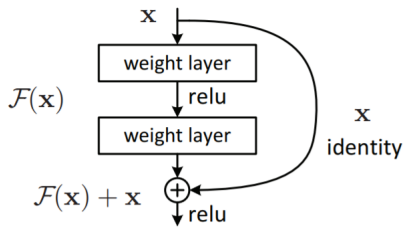


Figure 3.10: Residual learning – a building block [36]

3.6 CNN in NLP Tasks

CNNs have other successful applications than image processing. CNN models have proven to be an effective approach for some NLP tasks with excellent results.

While processing a text with a CNN, the most common text representation is the usage of the vector space. Every text element (characters, words, sentences, or even whole documents) are encoded into a vector of dimension d that embeds desired features. Once we have a vector for each element, such representation makes an input to the network.

In Figure 3.11 there is a sentence representation with a list of word embeddings that creates a matrix. Such matrix can be seen as a certain form of a raster image. The y coordinate is time (i.e. the order of the word within a sentence) and the x coordinate expresses a vector dimension.

Such vector representation encodes semantic information based on the context (surroundings elements). Convolutional operations with multiple filters can extract features that can be used for another processing (e.g. text classification).

Kim [47] trained a simple CNN with one layer of convolution on top of word vectors obtained from an unsupervised neural language model (word2vec [75]). It uses three sizes of convolutional kernels - $(3, d)$, $(4, d)$ and $(5, d)$ where d is the embedding dimensionality. There are 100 kernels of each size that are computed simultaneously and their outputs are merged and fed into a fully connected layer. The architecture is depicted in Figure 3.11.

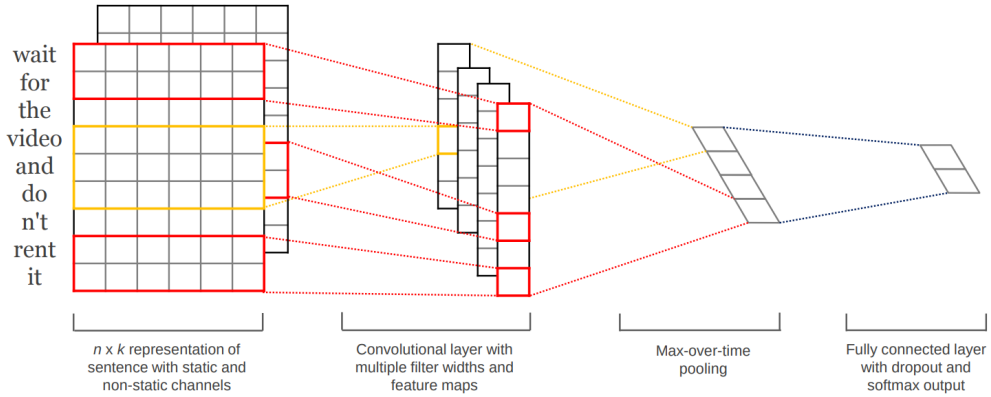


Figure 3.11: Kim's architecture for sentence classification task [47]

As we can see, the shape of the convolutional kernel is designed to catch both dimensions (i.e. time dimension as well as the embedding dimension). Kim used a pre-trained word2vec model and use its word embeddings as an embedding layer and made a set of comparison experiments.

Broadly speaking, in all but a few cases, it has been proven that the usage of pre-trained word embeddings is beneficial when they are allowed to train and even if they are kept static they are better than trained random-initialized ones. Despite little tuning of hyperparameters, this simple model achieves excellent results [47].

From other CNN applications in NLP tasks, we can mention for example semantic parsing [121] or semantic information retrieval [97], and sentence modeling [43].

3.7 Fully Convolutional Neural Network

We further mention the **Fully Convolutional Neural Network** (FCNN) which is also very popular in the image processing area. Its deployment and purpose lie in the semantic segmentation (i.e. detection of regions of interest – ROI). In Figure 3.12, there is an example of such segmentation.

Nowadays, popular representatives of such networks are: U-Net [90], the HP-FCN [113] and last but not least the ARU-Net [34]. Such networks have been successfully deployed in the medical image processing or in historical document analysis and text segmentation and some of them are used within our research and will be described later.

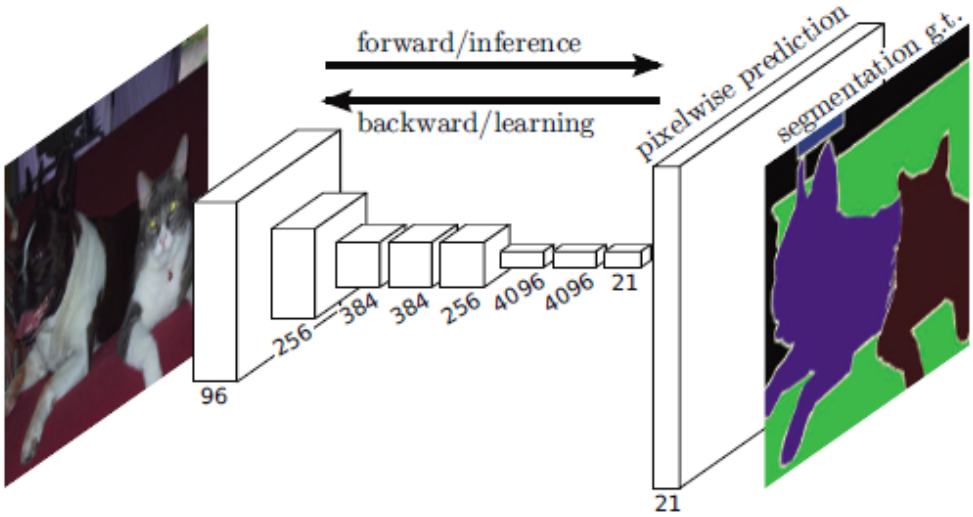


Figure 3.12: FCN for semantic segmentation task [62]

The architecture of FCNs consists of series of convolutional layers together with max-pooling followed by up-sampling techniques that result in a final layer with the same dimension as an input image. The values of these "output pixels" lead to the pixelwise prediction and in the consequence to detection/segmentation of ROI.

3.8 Optical Character Recognition Review and Contribution

Optical Character Recognition is the crucial component for our dialogue act recognition approach based on images. Hence, we provide the review of OCR methods withing this section. We start with the historical development of OCR and the summarization of the related work. Next, we present our research of the analysis of historical German newspapers where we used convolutional recurrent neural network (CRNN) model for OCR.

3.8.1 Motivation and Historical Development

Many digitized documents exist only in a form of scanned images. Effective information retrieval and other text-based automatic processing are very hard without proper pre-processing: efficient text segmentation and OCR. A significant part of such scanned materials are historical documents (e.g. newspaper, chronicles, or maps).

The efforts to preserve the cultural heritage of historical documents have become a significant task. As the number of digitized historical documents has increased rapidly during the last few decades, it is important to make the data accessible. Methods for automatic processing of such documents are strongly dependent on OCR[2]. which needs to be implemented with a respect to the historical domain. In such a domain we often struggle with the quality of the historical documents and with a lack of annotated data as well.

Once a text is obtained its purpose is mostly to access the full-text searching or any other form of knowledge extraction based on a desired task.

OCR algorithms in different systems may vary, but in general, they mostly require preprocessed images i.e. a thresholded image with a minimal amount of noise and other undesirable effects such as rotations or skewing. We outline the most popular OCR Engines and tools in Section 3.8.3.

3.8.2 Related Work

The first OCR methods were based on the images of individual characters and there was a limitation to the font given. Nevertheless, the first successful solutions were developed and it pioneered the implementation of applications such as machine reading addresses and names from the envelopes, and automatic processing of passports or price tags [54].

Since then, great progress has been made, both in terms of universality (multiple fonts, languages, handwritten text) and in philosophy and approach to particular solutions. The current trend in the OCR field is to utilize neural networks which often process whole text lines instead of individual characters. The whole text line approach has been chosen as a promising way because it uses a context of other characters/words. Nowadays, OCR is available even online as a cloud-based service, accessible by mobile applications.

Although there are some papers that try to solve OCR by algorithms other than neural networks, e.g. support vector machine (SVM) [23, 109], the field of OCR is dominantly tackled by deep neural networks.

In the last decade, a great progress has been made in terms of training custom OCR models because training examples consist of only images of text-lines with corresponding transcription. So there is no need to label particular image frames or column of pixels onto individual characters because the alignment is automatically learned. This way of training and annotation is possible

[2]In the case of hand-written documents (that are very common in historical domain) methods of hand-written text recognition (HTR) are empoloyed.

thanks to Graves et al. [32] because they proposed Connectionist Temporal Classification (CTC). This approach was first used for automatic speech recognition, however, it can be directly applied for text recognition. CTC has significantly facilitated supervised training of OCR as can be seen for example in Breuel et al. [11]. Breuel's approach was based on LSTM which has become a part of the OCRopus system [9].

Usage LSTMs and CTC loss is considered as a big milestone which paved the way for further improvement of OCR models based on deep neural networks. First, models used relatively shallow neural network whose centerpiece was LSTM. Later though, architectures have been extended by convolutional neural networks (CNN), estimating new state-of-the-art and reaching excellent results in terms of recognition accuracy [10, 115]. These enhancements have become particularly useful when dealing with historical printed/handwritten documents.

Another approach that combines CNN and RNN is proposed in Rawls et al. [86]. The system utilizes a CNN for feature extraction and an LSTM is then used for sequence modeling. The model is evaluated on handwritten and printed data where such a model performs well for both data types. This work also presents a weighted finite-state transducer (WFST) that supplies a language model to the decoding procedure.

Another example of the successful application of the combination of CNN and RNN presented Shi et al. [98]. They conducted experiments with the convolutional recurrent neural network (CRNN) model on scene text recognition and achieved excellent results. Elagouni et al. [26] used similar approach to perform text segmentation in videos.

Since 2017, Transformers have been achieving excellent results in a wide range of NLP tasks, and recently, OCR/HTR transformer-based approaches have also emerged. In the paper *Pay Attention to What You Read: Non-recurrent Handwritten Text-Line Recognition* [44] written by Kang et al. there is the first attempt to employ Transformers for the HTR task. They propose a non-recurrent model for HTR based on ResNet50 [36] and multi-head self-attention layers. Their approach achieved state-of-the-art performance in IAM dataset [125].

Wick et al. [116] propose a model for handwritten text recognition (HTR) composed of convolutional neural network, as a feature extractor, and bidirectional Transformer-based encoder/decoder. Although Transformer needs a larger dataset than usual (contrary to the CRNN for example), still it is able to learn a good language model and outperform other approaches.

Another Transformer-based approach has been proposed by Li et al. [61]. They proposed Transformer-based Optical Character Recognition (TrOCR). In the pre-training phase, they provided model 684M text-lines. Withing the second phase, the model is fine-tuned for both printed and/or handwritten text recognition tasks. Unlike other approaches, there are no convolution layers within a model and solely the Transformer model is used as a visual encoder.

There are also efforts to train the model with as little human effort as possible (e.g. just a few hundred annotated text-lines) while keeping the best possible performance. So there are papers which motivation is to optimize performance of OCR for limited ground truths. For example recent paper *Transformer for Handwritten Text Recognition Using Bidirectional Post-decoding* [114], written by Wick and Reul in 2021, addresses this motivation by proposing a model which uses ensemble learning. Basically, ensemble learning is a multiple classifier system which combines multiple learners to solve a learning problem (also called committee-based learning) [124]. Wick and Reul employed joint training of n individual CNN/LSTM models (so-called *voters*) which used combined CTC loss function. The final prediction/recognition is provided by all voters because an input image is processed by all models independently combining and averaging resulting confidence matrices.

For achieving even better success rate, several training strategies have been proposed. Above all, we mention the synthetic data generation and data augmentation or so-called voting when several models are combined to achieve ensemble-learning (see e.g. [88, 115, 114]).

3.8.3 OCR Tools and Engines

This section describes popular OCR tools and engines that are available. Although there are some cloud-based OCR Engines (e.g. Google Cloud Vision or Microsoft Azure Computer Vision), below we limit ourselves only to the description of free OCR tools that is possible to easily customize and train.

Tesseract

Tesseract[3] [103] is an open-source OCR system which was originaly developed by the Hewlett-Packard company at the end of the 20-th century. Tesseract is one of the best OCR engines in terms of integrated language support and recognition scores. The current stable version (4.1.1) uses a powerful LSTM based OCR engine and integrates models for 116 additional languages. It is also possible to carry out training of the Tesseract engine and create a custom `tessdata` file which is fine-tuned on particular font and data.

Kraken & Calamari

Kraken is an open-source OCR system that is optimized for use on historical documents. According ot the authors the crucial features are, among others, fully trainable layout analysis and character recognition or support for tackling right-to-left text.

Calamari OCR is another representative of open-source OCR system. This OCR engine based on OCRopy and Kraken is written in python. It is designed to both be easy to use from the command line but also be modular to be integrated and customized from other python scripts[4]. It requires, though, the segmentation tool because the input must be a segmented line of text.

Other OCR Engines

Among other OCR and layout analysis systems, we can mention Aletheia [20]. It is a full document image analysis system, which allows to annotate documents for layout evaluation (page element localization) and OCR in the form of XML file – e.g. PAGE XML [82]. There is also a web version of Aletheia[5].

OCRopus [9] is an efficient document analysis and OCR system. This system has a modular architecture and it is possible to use it freely for any purpose. The main components are dedicated to the analysis of document layout, OCR, and the use of statistical language models that support OCR. Moreover the OCRopus contains a tool "ocropus-linegen" that allows rendering text lines images, usable for training an OCR Engine.

There are a number of tools, for rendering text lines (or synthetic data generation). Besides OCRopus we can mention a tool for Arabic OCR [65] or Aletheia that deal with synthetic data generation and annotation. Finally, there are also other standalone tools such as *TextRecognitionDataGenerator*[6] that are very useful in the training of an OCR model.

Last but not least, we mention the Transkribus [105, 57] which is another complex platform for the analysis of historical documents which covers many research areas as for instance layout analysis, handwritten text recognition, etc. It includes also OCR using ABBYY Finereader Engine 11[7].

[3] https://github.com/tesseract-ocr/
[4] https://github.com/Calamari-OCR/calamari
[5] https://github.com/PRImA-Research-Lab/prima-aletheia-web
[6] https://github.com/Belval/TextRecognitionDataGenerator
[7] https://www.abbyy.com/

3.8.4 Historical Newspaper Analysis

This section presents our results of the historical newspaper analysis that we tackled within the Czech-Bavarian Project. Before we get to the description of the models and experiment, we remind the main contributions of our work. We used most of the following for DA recognition using visual information:

- The creation of a text segmentation method based on Fully Convolutional Neural Networks that solves a specific newspaper layout;
- Proposing an OCR method based on CRNN to address historical newspaper text recognition;
- Giving an overview of alternative training strategies with a small amount of annotated data available Proposing text region and text line segmentation approaches based on fully convolutional networks using transfer-learning for efficient training with a low number of training examples;
- Building a novel dataset from the real historical data available on the *Porta fontium* portal dedicated to the evaluation of OCR systems;
- **Giving an overview of how to create an efficient OCR system with minimal costs (i.e. minimal human effort during annotation process as well as the minimal time required to train models).**

In general, we try to determine optimal conditions for training a CRNN based OCR system with a limited number of real annotated data to achieve results that are competitive with state of the art.

Text Segmentation

The OCR results depend on the layout analysis and text segmentation. Based on the layout analysis, we can perform image segmentation into smaller logical units as individual text blocks and lines which are the input of our OCR engine. The whole process is decomposed into three main tasks: block segmentation, line segmentation, and optical character recognition itself as depicted in Figure 3.13.

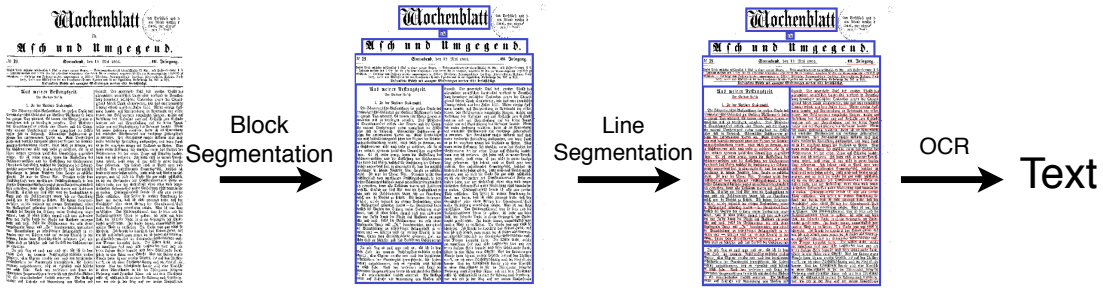


Figure 3.13: Whole process pipeline: source image (left), text region segmentation (blue color; middle) and individual lines segmentation (red color; right).

Block Segmentation

The goal of block segmentation is to detect and extract text blocks. For this task, we utilize the well-known fully convolutional network U-Net [90]. Although U-Net has been developed for semantic segmentation of medical images, it is possible to apply this architecture to a different segmentation task such as text block extraction.

The FCN networks are composed of encoder and decoder parts. The encoder comprises a set of convolutional and pooling layers. In the decoder part, deconvolutions are used to up-sample the image to the original size. The architecture of the U-Net is depicted in Figure 3.14.

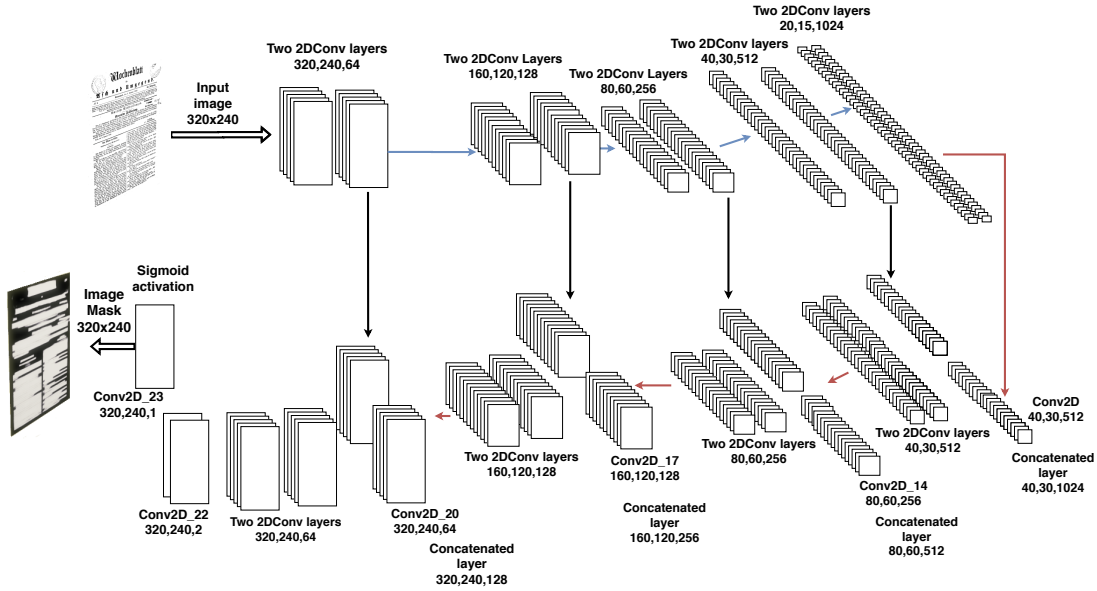


Figure 3.14: U-Net architecture. Except for the last layer, ReLU activation functions are used. From the input (320×240 pixel) image, an output image mask is produced. Blue arrows indicate max-pooling with the pool size (2, 2) and the red arrows show up-sampling with the size (2, 2).

The next task is segmenting the regions into individual text lines. Although there are some algorithms that can solve the text line segmentation from the whole page in one step, we prefer using the two-step approach, which allows determining logical text units and simplifies determining the reading order whose preservation is often desired. The subsequent segmentation into lines becomes then significantly easier.

Documents we usually process have mostly a two-column layout. In the case of well-separated columns, a one-step approach would be sufficient. However, there are also more complicated pages with irregularities where the determination of reading order from coordinates of single lines is complicated. The one-step approach can also merge lines across column separators, which can jeopardize the reading order as well. The second benefit of using the two-step approach is the possibility to filter out some types of noise, such as pictographic illustrations or various decorative elements.

Summing up, our segmentation method is as follows. The input page is first pre-processed which includes binarization and page rotation. The next step is block segmentation. The page is first processed by the U-Net which predicts a mask indicating positions of the text regions. Based on the thresholded predicted mask, we extract individual regions. Since the U-Net output

is a binarized image (due to the thresholding) we can use a simple connected component analysis algorithm. This algorithm provides a list of connected components (i.e. areas with the same pixel intensities) and several properties that can be used to crop connected component regions from the image.

The list of extracted text regions is the input to the line segmentation process. In this step, we apply a line segmentation method in order to obtain images of individual text lines. After necessary post-processing which removes noise and parts of surrounding text lines we can feed the resulting text line images directly to the OCR engine (see Figure 3.15). As an alternative, we also employed a modification of the U-Net that has been proposed by Wick and Puppe [113]. The whole architecture of this network is much simpler and the number of parameters is lower. Yet the final segmentation results are comparable.

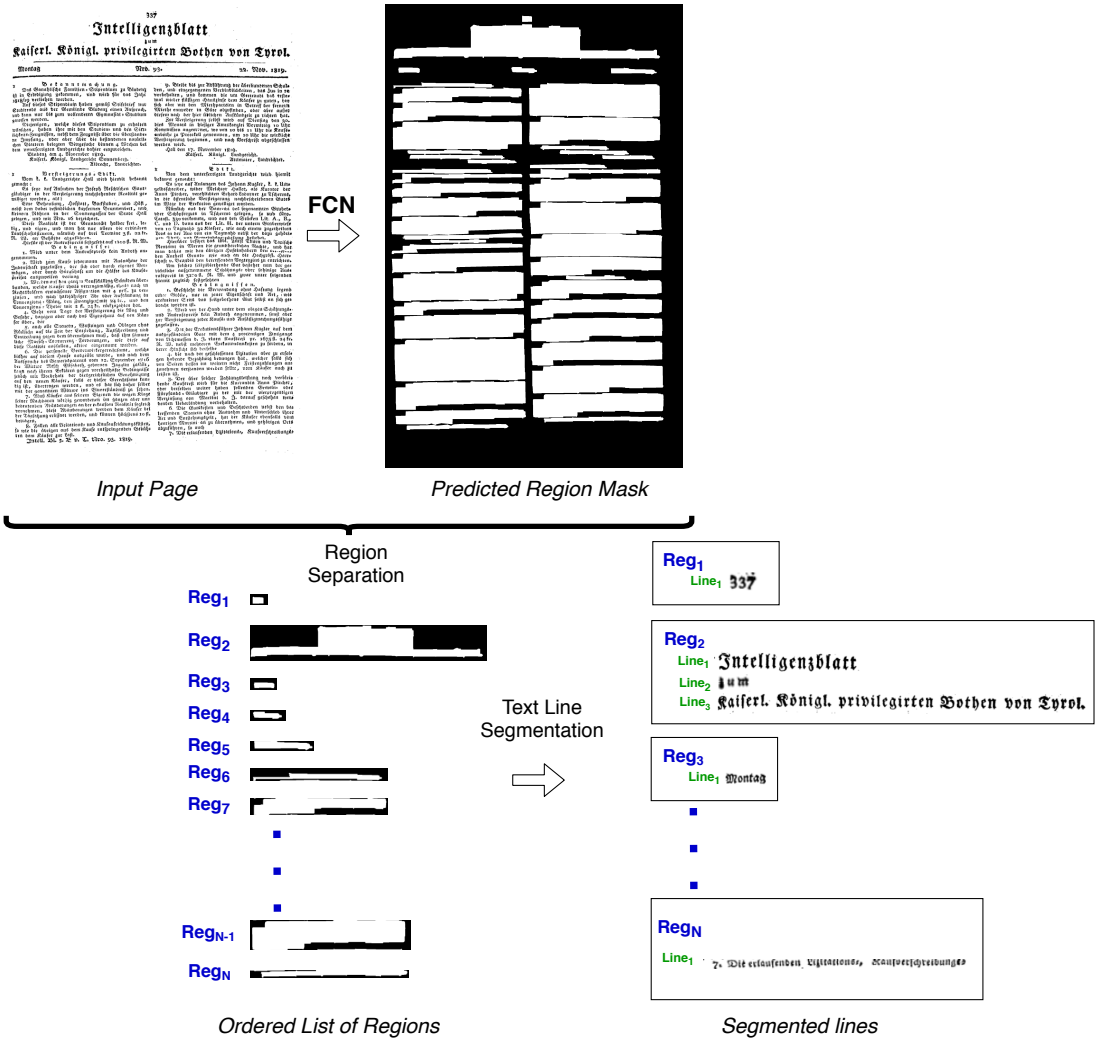


Figure 3.15: Region and line segmentation tasks scheme

Line Segmentation

For the line detection and segmentation, we use another representative of FCNs the ARU-Net, a deep neural network presented by Gruning et al. [34]. It was designed to detect baselines (i.e. the line upon which the letters sit) in handwritten historical documents. The ARU-Net extends the U-Net and it should provide a better line detection in pages with variable font size. It includes also an attention mechanism that allows the ARU-Net to focus on image content at different positions and scales.

The usual way of training an FCN is to provide the ground truth image. Mostly, it is a one-channel binarized image – the mask, where the white color represents the regions of interest (ROI). The whole training process is basically a pixel labeling problem, so the network sees many examples of the desired output from a single image. That is one of the reasons that only a few dozens of training samples are enough to achieve promising and reasonable results. Naturally, the dimensions of the image are a well-posed problem. If an image has a high resolution (e.g. 5000 x 7500 px), the training of such a big image would be cumbersome and limiting in terms of memory and time. The solution is either to rescale the image to lower dimensions (but we risk a possible loss of information) or we can split the image into smaller regions – so-called patches, and by cropping them we create more training samples from one image. Thereafter, the patches are processed one by one and the final output is a composition of these smaller units.

Anyway, we must either use and deploy a directly previously-trained model or we must provide ground truth label images and carry out the training process. Figure 3.16 shows, the example of the ground truth image for line segmentation training. Similarly, the block ground truth label is just a bounding box around text paragraphs that is filled with white color.

Figure 3.16: Text block and its ground truth

3.8.5 CRNN Model

A combination of deep convolutional and recurrent neural networks (CRNN) is still utilized as a line-based OCR (see e.g. [10, 98, 101, 33]). A quite common representative of the recurrent neural network within the CRNN model is the LSTM [37].

Figure 3.17 shows the CRNN Model architecture. The inputs of our network are binarized line images (that were previously segmented) with a dimension of 1300×40 pixels. We resize all input images so that their height is 40 pixels. The width is set to the maximum image width occurring in the training set. We keep the aspect ratio of the images and pad the rest of the image with white space. We apply two convolutional layers with 40 kernels with a shape of 3×3 and $MaxPooling$ layers to the input layer. After this first phase, we obtain 40 feature maps and we also reduce dimensionality by scaling down the input to 325×10.

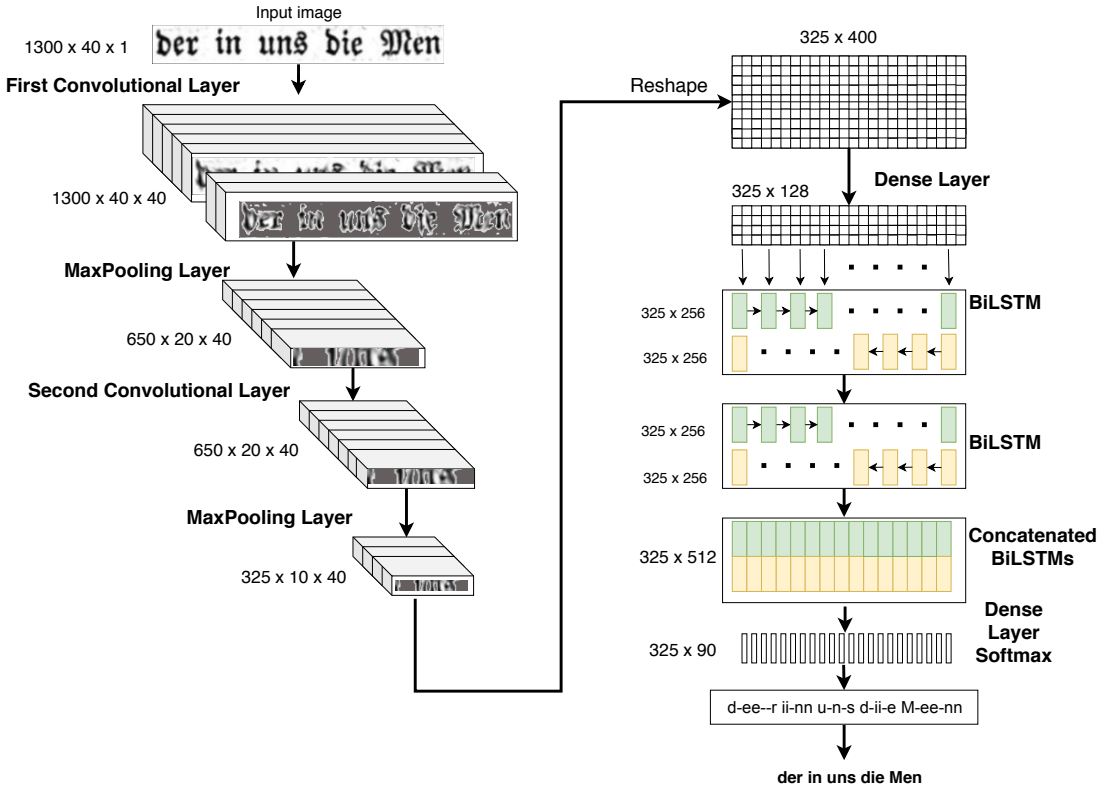


Figure 3.17: CRNN architecture

The CNN creates feature vectors and through a reshaping mechanism, we create a dense layer that is fed into two Bi-directional LSTM layers. Each Bi-LSTM layer comprises two LSTM layers that process the input from opposite sides. One LSTM layer contains 256 units. The outputs of the LSTMs from the first Bi-LSTM layer are merged by addition operation, while the outputs of the second pair of LSTMs are concatenated. We use the ReLU activation function after each layer.

The output of the Bi-LSTM layer is passed to a set of dense layers followed by the softmax activation function. It is a representation of probability distributions of individual symbols per each time frame. Let $\mathcal{A}$ ($|\mathcal{A}| = 90$) be a set of all symbols. It includes out of vocabulary (OOV)

and *blank symbol*. The most probable symbol $\hat{a}_t$ of each time frame t is determined as:

$$\hat{a}_t = \underset{a_i \in \mathcal{A}}{\operatorname{argmax}}\, p_t^{a_i} \tag{3.6}$$

where $p_t^{a_i}$ is a probability of observing character a_i at a given time t. At each time t the sum of the probabilities of all symbols is equal to 1.

$$\sum_{i=1}^{|\mathcal{A}|} p_t^{a_i} = 1 \tag{3.7}$$

The last part of the classifier is a transcription layer which decodes the predictions for each frame into an output sequence. To be able to distinguish each individual character, the blank symbol (-) is present. It is also necessary to deduplicate the sequences of the same symbols (see Figure 3.18).

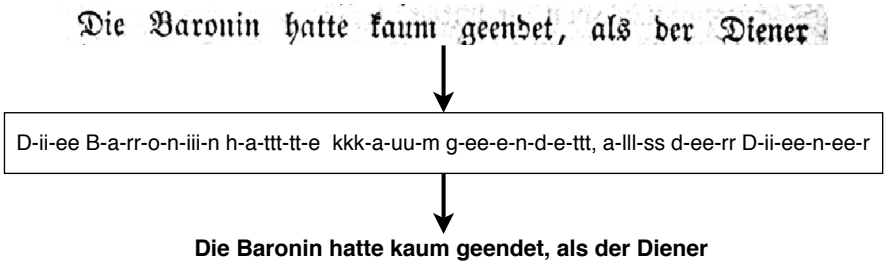


Figure 3.18: Input image and transcription layer for obtaining text

We use a connectionist temporal classification (CTC) output layer with related CTC loss [32]. We dedicate the following subsection to the CTC Loss since it is an important and key component of the CRNN.

CTC Loss

In the beginning, let us highlight the main advantage of using the CTC Loss. If we are creating an OCR model, we annotate data for training and the only thing we define is the target text. We don't have to specify either the precise character position in the image or bounding box coordinates. We let the network learn the alignment per se.

The origins of this way of training began in 2006 when Alex Graves et al. [32] designed a method for training RNNs. The problem was that RNNs can only be trained to make a series of independent label classification (e.g. in the case of speech recognition we must have provided correct labels at specific time frames). This means that the training data must be previously pre-segmented, and we must perform some post-processing methods to give the final output.

The main contribution of Graves et al. [32] is that they found a way of training RNNs that removes the need for pre-segmented training data and post-processed outputs, and models all aspects of the sequence within a single network architecture. Graves used this approach for speech recognition and outperformed other approaches such as Hidden Markov models (HMM) [83] without requiring any task-specific knowledge.

The basic idea is to interpret the network outputs as a probability distribution over all possible label sequences, conditioned on a given input sequence. Given this distribution, an objective

[8] https://stats.stackexchange.com/questions/320868/what-is-connectionist-temporal-classification-ctc

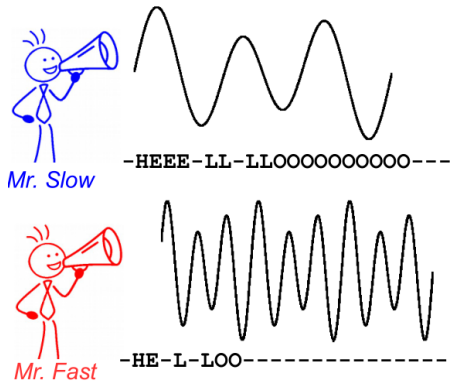


Figure 3.19: Decoding of two different phoneme sequences provides the same text for both speakers, even though the alignment and positions of the phonemes differ. The blank symbol is '-'.[8]

function can be derived that directly maximizes the probabilities of the correct labelings. Since the objective function is differentiable, the network can then be trained with the backpropagation algorithm [32]. To be clear about the terms used by Graves, the *connectionist temporal classification* refers to the labeling of unsegmented data while the *framewise classification* is the independent labeling of each time-step in accordance with the literature and the Graves paper.

Now we will follow the [32] to outline the mathematical background and formulas. Let S be a set of training examples from the fixed distribution $D_{X\times Z}$. X is the input space (set of all sequences of real-valued vectors) and Z is the set of labels (over the finite alphabet L). We also define the S' as a set of testing examples (which is disjoint from S)

So each sample in S and/or S' is basically a tuple of sice two (i.e. pair of sequences ($\mathbf{x}$,$\mathbf{z}$)). Furthermore, there is a condition that the target sequence $\mathbf{z}$ is at most as long as the input sequence $\mathbf{x}$. Formally, if $\mathbf{z} = (z_1, z_2, ..., z_U)$ and $\mathbf{x} = (x_1, x_2, ..., x_T)$, then $U \leq T$. This implies that it is not possible to align these two sequences a priori, since as they have different lengths in general.

However, it is possible to use S to train a classifier $h : X \mapsto Z$ to classify previously unseen input sequences in a way that minimizes some task-specific error measure – Label Error Rate (LER).

$$LER(h, S') = \frac{1}{|S'|} \sum_{(\mathbf{x},\mathbf{z})\in S'} \frac{ED(h(\mathbf{x}), \mathbf{z})}{|\mathbf{z}|} \tag{3.8}$$

where ED is the edit distance between two sequences[9]. The aim is to minimize the rate of transcription mistakes.

When we use the CTC in a neural network, the crucial step is to transform the network outputs into a conditional probability distribution over label sequences. The network can then be used as a classifier by selecting the most probable labeling for a given input sequence.

The activations of the first $|L|$ units are interpreted as the probabilities of observing the corresponding labels at particular times. It is important to note, that the set of target labels L has one extra unit – the blank symbol (a.k.a. "no label"). We can imagine this as a gap between characters in the image (or the pause between individual phonemes in speech).

[9] ED($\mathbf{p}$,$\mathbf{q}$) is the minimum number of insertions, substitutions, and deletions required to change $\mathbf{p}$ into $\mathbf{q}$.

When we consider a recurrent neural network, let y_k^t be the activation of an output unit k at time t. We can interpret this as the probability of observing label k at time t, which defines a distribution over the set L'^T of length T sequences over the alphabet L. The elements of L'^T are *paths* and they are denoted with symbol π. Then the probability of observation π given the input sequence $\mathbf{x}$ is as follows:

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in L'^T \tag{3.9}$$

Next thing, we need to define is a "collapsing" function $\mathcal{B}$ which is basically "many-to-one mapping": $\mathcal{B}$: $L'^T \rightarrow L^{\leq T}$. The $L^{\leq T}$ is a set of labels without blank symbols and repeated labels (see following two examples).

$$\mathcal{B}(-A--AA-AA-BBB-CCCC) = AABC \tag{3.10}$$

$$\mathcal{B}(-A----AA-BBB-CC) = AABC \tag{3.11}$$

So we are able to get an output label sequence based on any path π from L'^T.

Another mapping function $\mathcal{B}^-1$ is used to map a label sequence $\mathbf{z}$ to the set of all possible paths π that "collapse" to $\mathbf{z}$. So it is "one-to-many mapping".

So the probability (the likelihood) of a target label $\mathbf{l}$ given the input sequence $\mathbf{x}$ is the of the probabilities of all the paths π corresponding to a given labeling $\mathbf{l}$.

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}) \tag{3.12}$$

Figure 3.20 shows the visual representation of the Equation 3.12

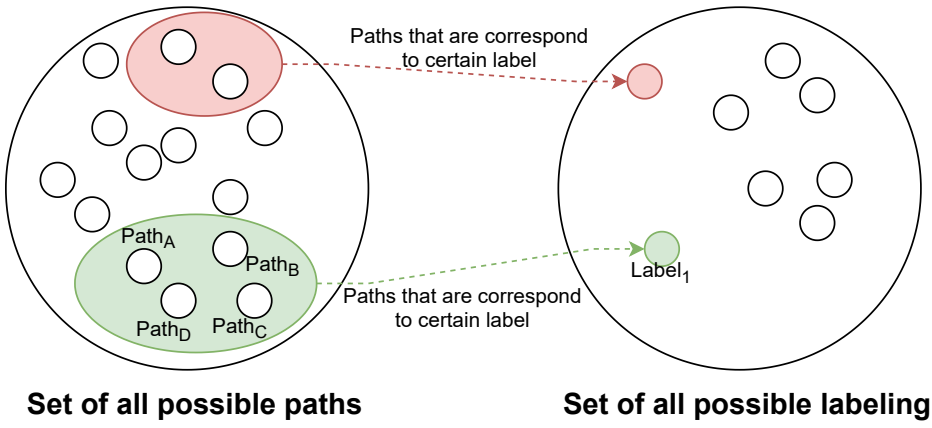


Figure 3.20: Illustration of paths and corresponding labelings

So according to the image the probability of *Label_a* based on a given set of possible paths is as follows:

$$p(Label_1) = p(Path_A) + p(Path_B) + p(Path_C) + p(Path_D) \tag{3.13}$$

The crucial information that is depicted in Figure 3.20 is the "accumulation" of probabilities of all paths that are mapped to a certain label.

The problem is how to compute these probabilities because the complexity is very high (it grows exponentially with the input sequence lengths). Fortunately, Graves et al. came up with the CTC Forward-Backward algorithm which allows calculating the approximation of conditional probabilities efficiently using dynamic programming. Since the product of probabilities can lead to very small numbers and, as a sequel, to the underflowing and losing information. To avoid this we use the sum of logarithms.

We remind that the main objective of the classifier is to pick the most probable labeling for the input sequence:

$$h(\mathbf{x}) = \underset{\mathbf{l} \in L^{\leq T}}{\operatorname{argmax}} p(\mathbf{l}|\mathbf{x}) \tag{3.14}$$

and the training objective is to minimize the following loss function J:

$$J = -\sum_{(\mathbf{x},\mathbf{z}) \in S} ln(p(\mathbf{z}|\mathbf{x})) \tag{3.15}$$

where S is a training set[10]. For the final evaluation, the LER value is used to measure the error rate. See [32] for any details.

Summing up, the CTC contributions are as follows:

- The training of a neural network can be achieved only by specifying the pairs of sequences $(\mathbf{x}, \mathbf{z})$. For example a line image with rendered text with corresponding text ground truth;
- No need of post-processing the output. A CTC automatically transforms the neural network output into the final sequence (see above-mentioned $\mathcal{B}$ mapping function);
- Considering the speech recognition, the CTC successfully deals with situations when some people speak more slowly than others (Figure 3.19);
- Considering the OCR (or HTR), the CTC handles the gap difference between individual characters without any problem.

3.8.6 Dataset Enlarging and Training Strategies

The purpose of this section is twofold. First, we describe the data we use to train the CRNN model for OCR, including techniques for its enlarging – synthetic data creation and data augmentation. Second, we present the various training strategies and scenarios to adjust the CRNN model for the best possible results.

OCR methods utilizing recurrent neural networks require a significant amount of annotated data. The best way to obtain such data is the annotation of real-world examples. However, it is often a costly process since both the cropped images of text lines and their transcriptions are required. Therefore, the methods of enlarging our dataset are considered to allow the better training of neural network models.

Data Augmentation

The main motivation to facilitate the data augmentation is the lack of annotated data (especially when the manual annotation is labour-intensive and expensive process). The prerequisite to carry out data augmentation is at least few annotated samples. Data augmentation techniques are much utilized in the image classification task.

We outline the most popular image transformations to achieve augmented samples.

[10] In Grave's paper the loss (objective) function is labeled as O

- Flipping and Rotation;
- Scaling and Cropping;
- Filtering (blurring) and artificial noise.

Figure 3.21 shows an example of data augmentation. From one real example, we can create several "new examples" by one or more above-mentioned transformations that are randomly realized[11].

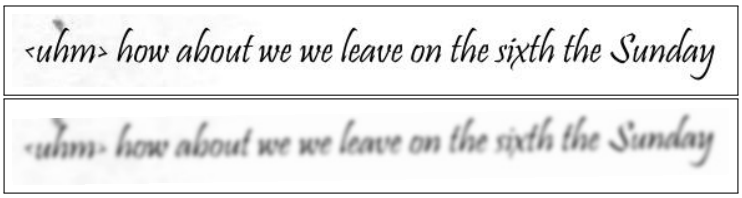


Figure 3.21: Example of data augmentation. The top image shows the original image of a text line. The bottom image is the augmented variant of the original image – a slight rotation and blurring with the Gaussian filter.

Wang and Perez in [112] explore, among other things, the usage of Generative Adversarial Networks (GANs) to achieve neural augmentation because they provide unsupervised generation of new images for training. They also provide experiments and discussion with conclusions that a combination of traditional augmentation followed by neural augmentation would lead to the significant improvement of classification tasks.

Figure 3.22: Example of traditional data augmentation [112]

Synthetic Data

Unlike the previous data augmentation method, in this case, we do not transform the real annotated examples. Instead, we create completely new data samples as similar to real examples as possible. Since we create samples from scratch, the wide and universal usage of this method is strongly limited (i.e. it is near impossible to render programmatically new images/photos which would be similar to the real ones – Figure 3.22). However, it is very simple to render text on the images. This is particularly useful for enlarging a dataset for OCR training.

To achieve this we can use some existing tools for synthetic text-line generation (for example: OCRopy–linegen [9]). Such tools have several configuration parameters. We outline the most common ones:

- Font (handwritten or printed);

[11] In the case of the OCR, the flipping is not an option since we require text-line images. On contrary adding noise is very useful.

- Source of text (random or text from a desired domain);
- Data augmentation (noise, rotations, scaling, cropping etc.);
- Background (plain white or any custom);
- Image width, height, and number of samples.

Creating synthetic documents for Arabic OCR was introduced by Margner and Pechwitz [65]. The whole process started by typesetting Arabic pages. Then the bitmap representation and corresponding ground truth were generated resulting in the OCR dataset.

Another synthetic data generation approach was described by Gaur et al. [28]. It aimed at handwritten Indian texts, which were created from fonts that are similar to handwriting. Various distortions were applied to enhance the script's appearance.

Jaderberg et al. [39] discussed methods for synthetic data generation for natural scene text recognition. They generated images with three layers: background, foreground, and an optional border/shadow layer. The fonts were randomly selected from a large catalog to ensure variability. The noise was also added so that the images were more realistic.

German historical newspapers are printed with German Fraktur font which can be easily generated using a modern text processor. As a text source for synthetic images of text lines, we use historical German corpora from which we picked 25,000 sentences. The main contribution of this synthetic data generation research is to balance the lack of real annotated and provide enough data to initialize a good language model.

We created two types of synthetic data. The first version is based on manually-cropped character images. We provided several images of each character that occurs in our documents. Then, we composed text-line images according to the prepared sentences from the historical corpora. Each character of the text line is picked randomly from available samples to achieve diversity. We created three versions of such data according to the gaps between individual characters:

1. Constant space (CS) - fixed sized gap of 4 pixels;
2. Random space (RS) - random value between 1 and 5 pixels;
3. Precise space (PS) - a value computed from the annotated real examples.

The second version was created automatically by *TextRecognitionDataGenerator*[12]. We provided a font (German Fraktur) and a historical German dataset. Figures 3.23 and 3.24 show the examples of synthetic data, while the real examples are depicted in Figure 3.25.

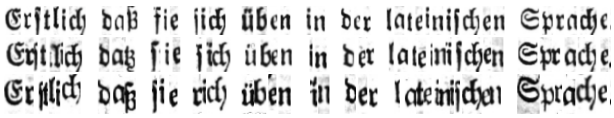


Figure 3.23: Examples of synthetic data generated by composing character images: Constant space (CS), Random space (RS), Precise space (PS) (top to bottom).

So in total, we created five versions of synthetic data. Two versions have been created by an external tool. We will present experiments about the impact of synthetic data for OCR training in the next section. Before getting to that, though, we first outline the training strategies.

[12] https://github.com/Belval/TextRecognitionDataGenerator

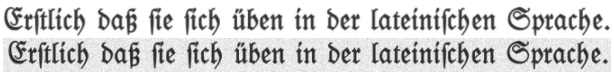


Figure 3.24: Examples of generated data by *TextRecognitionDataGenerator*: white background (top), background with Gaussian noise (bottom).

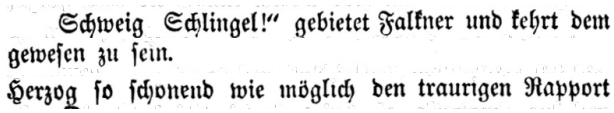


Figure 3.25: Three line examples from the real dataset which are annotated manually

Training Strategies

Based on our dataset we present gradually three training strategies.
Strategy 1: use only the training part of the annotated real corpus for training (training from scratch).
Strategy 2: use only synthetic data for training (no real annotated samples).
Strategy 3: use synthetic data first and then real annotated data. (initial training and thereafter fine-tuning).

The first strategy is straightforward and does not use synthetic data. The second one is basically a comparison of the qualities of the synthetic data without the influence of the real dataset. Finally, the last strategy combines the synthetic and the annotated data and tries to find optimal settings.

3.8.7 Experiments and Results

We conducted series of experiments. We evaluate all training strategies that we mentioned previously and provide success rates. We emphasize that validation and test data are real annotated examples from the *Porta fontium* dataset.

Dataset and Evaluation Metrics

In the *Ponta Fontium* dataset, we have 10 annotated newspaper pages available which consist of 1368 text line images in total. The height of the segmented text-line images is approximately 40 pixels, while the width is variable with respect to the text content. All images are binarized using Otsu's method [78].

We split our 10 pages into three non-overlapping sets. Two pages serve as a test set, one is used for validation and the remaining seven pages are used for training. Table 3.1 shows the statistics of the *Porta fontium* dataset.

	# pages	# lines	# words	# chars
Train	7	955	7653	50 426
Val	1	138	1084	6 669
Test	2	275	2163	13 828

Table 3.1: Statistics of the *Porta fontium* dataset

For presenting the OCR results, we use the following evaluation metrics. First, we present the average accuracy (Avg ACC) results. In our case, the average accuracy indicates how many text line images were recognized completely correctly from all text line images:

$$Avg\ ACC = \frac{c}{n} \tag{3.16}$$

where c is the number of correctly recognized text lines and n is the number of the text lines in the test dataset.

We further measure how many corrections in the text line must be done by the user to obtain the correct output. This metric is reported by Levenshtein edit distance (ED) [60], which also indicates the number of insertions, deletions, and substitutions:

$$Avg\ ED = \frac{1}{n}\sum_{i=1}^{n} ED(pr, gt)_i \tag{3.17}$$

n refers to the number of total text lines in the test dataset (pr is the predicted output and gt is the ground truth).

We report also character error rate (CER) and word error rate (WER):

$$Avg\ CER = \frac{1}{n}\sum_{i=1}^{n} \frac{S_i + D_i + I_i}{N} \qquad Avg\ WER = \frac{1}{n}\sum_{i=1}^{n} \frac{S_i + D_i + I_i}{N} \tag{3.18}$$

where S refers to the number of substitutions, D is the number of deletions and I is the number of insertions in each text line (i) from the test dataset, N is the number of characters (words) in each text line and n is the number of total test lines.

Annotated Dataset Experiment – Strategy 1

The first experiment analyzes whether it is possible to train our CRNN model from scratch solely on the real data we have at our disposal (955 train and 138 validation text-line images).

Figure 3.26 shows that training configuration is quite slow with respect to the number of epochs. However, the CRNN is still able to learn from this amount of data.

For the final evaluation, we chose the optimal number of epochs equal to 80 since after crossing these values the CRNN model begins to stagnate and further training brings no improvement. The overall performance of the first strategy shows Table 3.2.

WER	CER	Edit distance
0.068	0.014	0.464

Table 3.2: Strategy 1 – overall evaluation (test dataset) after 80 epochs of training

As we can notice, the OCR results are very good, despite the fact that the CRNN was trained only with 955 text-line images. This shows a very strong positive impact of the CTC because the CRNN model is able to successfully recognize images even with previously unseen text.

Synthetic Dataset Experiment – Strategy 2

The goal of the second set of experiments is to examine the behaviour of the CRNN when trained only on different kinds of synthetic data (five sets of 25,000 synthetic text-line images).

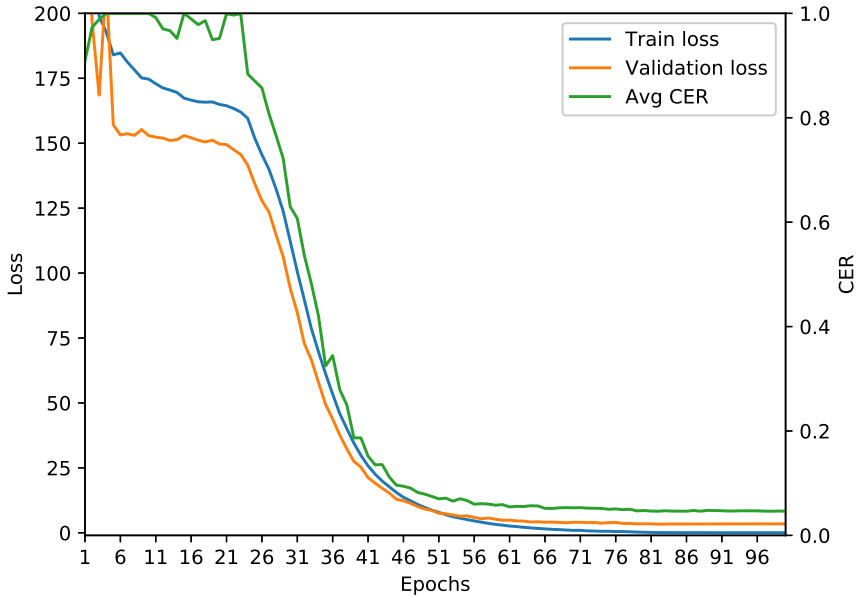
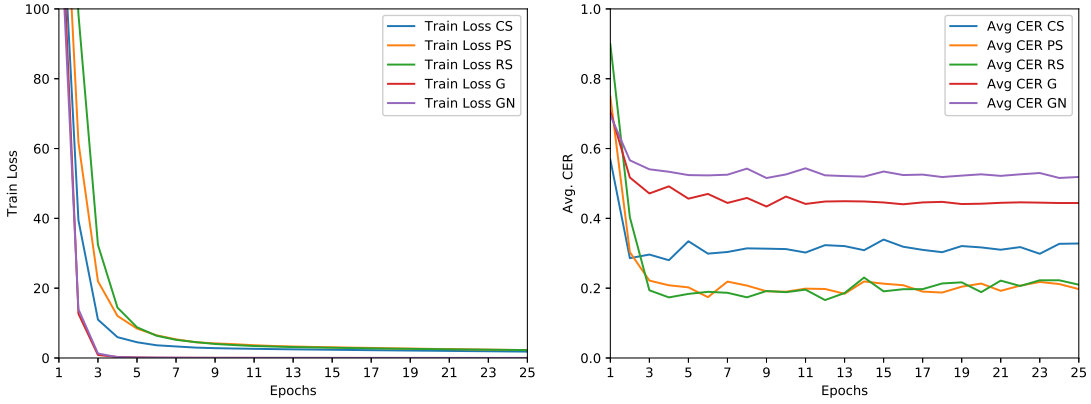


Figure 3.26: Strategy 1 – training visualization

In all cases, we train the network only on particular synthetic data and evaluate it on the real validation set.

Figure 3.27 shows the comparison of all five synthetic datasets. The left part of the image shows the training progress (train loss values), while the right one shows the validation results (average CER) after each epoch.



(a) Training progress on different kinds of synthetic data

(b) Average CER curves on different kinds of synthetic data

Figure 3.27: Strategy 2 – training visualization

We can observe that the training loss decreases very rapidly in all cases (contrary to the previous experiment – training from scratch). This experiment further shows that the training on generated data using *TextRecognitionDataGenerator* (G and GN) is faster and the loss value is lower than for hybrid data by the other methods (CS, RS, PS). This behavior is expected since there are 25,000 samples to iterate per one epoch while in the previous training strategy there are only 955 training samples.

Figure 3.27 (b) shows that in terms of validation CER, the models trained on the data created by our hybrid generation technique (CS, RS, PS) perform significantly better than the models learned on the data generated by the standard tool (G, GN). The best-obtained CER value is around 18% for PS and RS while the best CER for generated data by *TextRecognitionDataGenerator* is only 46%. Furthermore, variable-sized gaps (RS and PS) perform better than the constant ones (CS). Therefore, for the final evaluation, we chose synthetic data with random spaces (RS). On the basis of the visualization, we set the target number of epochs equal to 10. Table 3.3) presents the evaluation on test data.

WER	CER	Edit distance
0.655	0.237	10.732

Table 3.3: Strategy 2 – overall evaluation (test dataset) after 10 epochs of training

The results are significantly worse than the previously presented first strategy. This indicates that the CRNN model isn't able to train based solely on synthetic data. Hence, it is necessary to obtain some number of real examples for reaching a reasonable OCR performance.

Fine-tuning Experiment – Strategy 3

This section describes the third strategy to train the CRNN model. In the case of the second strategy, we showed that training the CRNN model longer than 10 epochs is not profitable. As a starting point, we thus picked the CRNN model trained with random spaces synthetic data for 10 epochs. Then, we applied 955 training samples for another 25 epochs.

This strategy turned out to be the best option. Nonetheless, using solely real examples has reached competitive results.

We find the third strategy the most appropriate since the model learns the better language model (almost 26,000 versus only 955 text lines). In other words, the output labels space is significantly bigger and the CRNN model saw many more different examples. See Table 3.4 for the overall evaluation of all training strategies that we have presented.

Training Strategy	WER	CER	Edit distance
Strategy 1	0.068	0.014	0.464
Strategy 2	0.655	0.237	10.732
Strategy 3	0.065	0.012	0.428

Table 3.4: Overall evaluation of all training strategies

For any details about the CRNN model, training strategies, or experiments, see our paper *Hybrid Training Data for Historical Text OCR* [71] or *Training Strategies for OCR Systems for Historical Documents* [69]. There is also the comparison with other OCR engines (*Transkribus*, *Tesseract* and *OCRopus*).

The overall description of the *Historical Newspaper Analysis*, including a whole pipeline from the pre-processing and text segmentation to the OCR, is reported in our journal paper *Building an Efficient OCR System for Historical Documents with Little Training Data* [70].

3.9 Conclusion

In this chapter, first, we described the theory of CNNs and presented the development of this architecture. Second, we provided a review of OCR with related work and existing popular OCR tools. Finally, we presented the topic of historical newspaper analysis that was our main task within the Czech-Bavarian project. We will use most of the presented approaches and techniques for the task of DA recognition using visual information.

As a main result, we show that it is possible to use a small amount of real annotated data for training and achieve very good results. We guided through the whole process of building an efficient OCR system for historical documents from the pre-processing steps through seeking an optimal strategy for training the OCR system.

In the case of historical documents, we often struggle with a lack of OCR methods that are adapted to such a domain and they usually need a huge training dataset.

We also created a set of synthetic data with respect to the language of the given era. We compared several methods for synthetic data preparation and their influence on the final results. Synthetic data creation is an interesting way of enlarging a dataset and its great asset lies in sparing a tremendous amount of time by manual annotation of real examples (both cropped images and their transcriptions are necessary to provide).

Furthermore, we have created a *Porta fontium* historical dataset that can be used for segmentation experiments as well as for the OCR evaluation.

Another contribution lies in the focus on the minimal costs needed for the system training. We presented and evaluated several scenarios on how to train the best possible models with the limited annotated dataset.

Although we have presented several contributions, we would like to highlight the most important one, that is, the possibility to create an efficient OCR system for historical documents even with a small amount of real annotated training data.

"Trust in Allah, but tie up your camel."
Arabian proverb

4 Dialogue Act Recognition

"I will either find a way or make one."
Hannibal Barca

This chapter presents our research of the task of DA Recognition. Related work has already been outlined in Chapter 2. In our case, we use only dialogue transcriptions in text files[1]. The structure of this chapter consists of three sections. First two sections deal with multi- and cross-lingual DA recognition. In the last section, we focus on DA recognition from image documents. We highlight the following main contributions:

1. **The proposition of cross-lingual DA recognition models based on semantic transformations and transfer learning and comparing with a big multilingual model which is trained by all languages available** (see our paper *Multi-lingual Dialogue Act Recognition with Deep Learning Methods* [67]);

2. **The proposition of a new multimodal model for DA recognition from image documents. To the best of our knowledge, this is the first attempt to use a strategy of using two inputs (images and text) to complement each other and enhance the overall success rate, especially when the quality of documents is poor** (see our paper *Dialogue Act Recognition Using Visual Information* [66]).

4.1 Multi- and Cross-lingual DA Recognition

Many researchers have proposed multilingual approaches based on neural networks for a wide spectrum of NLP tasks, including document classification [48], named entity recognition [1] and semantic role labeling [8]. Unfortunately, research in the multilingual automatic DA recognition field is scarce. As already been stated, DA recognition is an important step in dialogue understanding and it plays a pivotal role in dialogue management.

The presented methods will utilize deep neural networks and word2vec embeddings for word representation. We employed the multilingual Verbmobil corpus [2] and proposed two approaches for DA recognition in German and English.

The first approach trains one general model on annotated DAs from all available languages. This model is thus able to perform DA recognition in multiple languages simultaneously. The

[1] It is worth-mentioning that in some corpora, there are sound recordings provided together with transcriptions

second method trains the model only on one language and cross-linguality is achieved by a linear semantic space transformation.

4.1.1 Multilingual Model

We start by describing the multilingual model in a formal way. Let $\mathbb{L} = \{L_1, L2, ..., L_M\}$ be a set of languages with available annotated DAs and $\mathbb{T}_{L_i}$ be the set of DAs for language L_i. Pooling together all of these labels into a single set $\mathbb{T} = \bigcup_{i=1}^{M} \mathbb{T}_{L_i}$ enables to train a multilingual classifier that assigns to any input text in any language $L_i \in \mathbb{L}$ a single label from $\mathbb{T}$. Such a model is able to recognize DAs in arbitrarily many languages but it is necessary to retrain the model when a new language is added.

4.1.2 Cross-lingual Model

The cross-lingual model relies on a semantic space transformation. It is indeed possible to transform the lexical semantic space of any language so that word representations of similar concepts in different languages are close.

To achieve a semantic space transformation, we chose the canonical correlation analysis (CCA) [12] method. It is a technique for multivariate data analysis and dimensionality reduction, which quantifies the linear associations between a pair of random vectors. It can be used for a transformation of one semantic space to another.

The DA recognition model is trained on a single pivot language. The test examples from any language are then projected into the target pivot language. It thus allows classifying DAs in any language from within the transformed semantic space. The significant advantage is that retraining the model is not necessary when a new language is considered.

4.1.3 DA Representation

Word2vec embeddings are used to encode word semantics. For the cross-lingual scenario, we create a vocabulary V of the $|V|$ most frequent words in the pivot language used for training. In the multilingual case, the vocabulary is shared and consists of the union of the vocabularies of all available languages.

The input to each proposed neural network model is either a sequence of W vocabulary indexes when the word embedding matrix is considered as part of the model's parameters; or directly a sequence of W embedding vectors when these embeddings are considered as constant.

The advantage of the former input is the possibility to fine-tune the word vectors, while the latter option allows us to use the transformed semantic spaces seamlessly.

4.1.4 Neural Network Topologies

Convolutional Neural Networks

We use two CNN networks with different configurations. The first one is the model presented in [68] where it was used for document classification. We have modified the size of the convolutional kernels to adapt them to the dialogue acts domain. In such a domain, we usually work with much shorter inputs so we use a smaller kernel – $(4, 1)$ as shown in Figure 4.1.

We use 40 convolutional kernels with $ReLU$ activation. A final fully-connected layer after the convolutional one consists of 256 neurons, which are further concatenated with the previous vector when we consider the history (the previous DA). We use categorical cross-entropy as

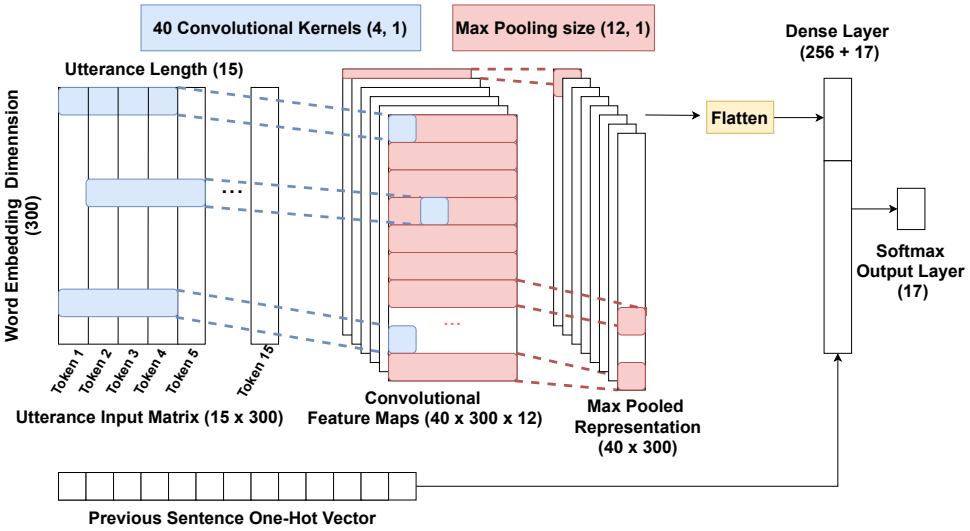


Figure 4.1: CNN_1 architecture

a loss function and softmax activation in the output layer. We refer to this architecture as CNN_1.

The second configuration follows Kim [47]. It uses three sizes of convolutional kernels – $(3, EMB)$, $(4, EMB)$ and $(5, EMB)$ where EMB is the embedding dimensionality. 100 kernels of each size are computed simultaneously and their outputs are merged and fed into a fully connected layer. The final layer is the same as in the previous case. The differences of the two networks are illustrated (see Figures 3.3 and 4.2). The Kim's original architecture is depicted in Figure 4.3. We refer to this model as CNN_2.

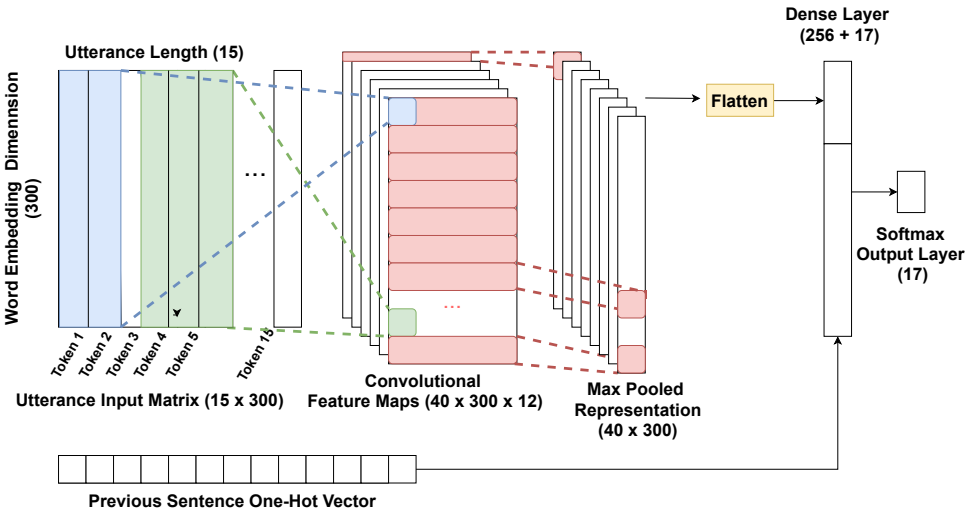


Figure 4.2: Kim's CNN architecture for DA recognition

In both CNN architectures, we have a possibility to use the information about the previous

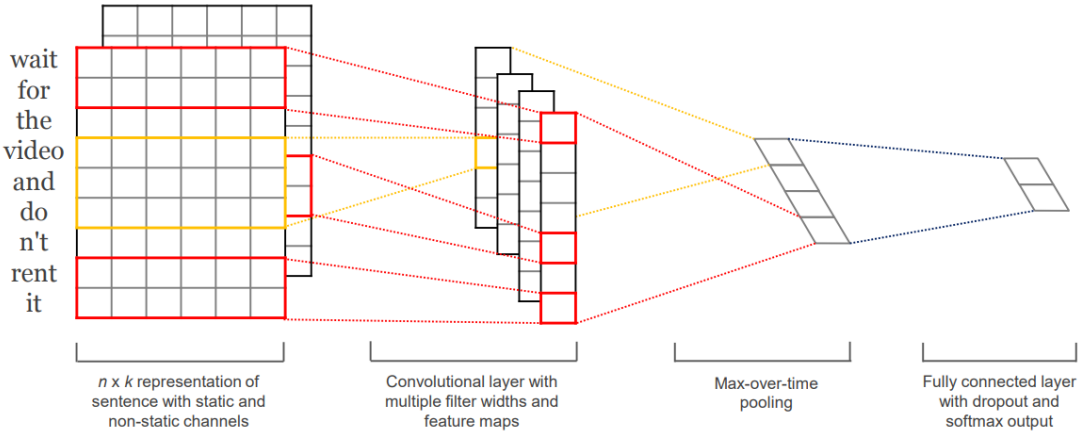


Figure 4.3: Kim's CNN architecture for sentence classification [47]

utterance. To doing so, we concatenate the one-hot vector of a predicted DA class of the previous sentence to the penultimate dense layer.

Bidirectional Long Short-Term Memory

The second approach exploits a Bidirectional LSTM layer. The representation of the input and the embedding layer is the same as for the CNNs. The core of this model is the Bi-LSTM layer with 100 units (i.e. 200 units in total for both directions).

The word embedding representation of the input (with 15×300 size) is fed into the Bi-LSTM layer, which outputs a single vector of 200 dimensions. This vector is then concatenated with the predicted dialogue act class of the previous sentence encoded in the form of a one-hot-vector (as in the CNN models). If the dialogue act has no history (e.g. initial dialogue act), a zero vector is used. The output layer has a softmax activation function. Figure 4.4 shows this model's architecture.

4.1.5 Experiments

Multilingual Verbmobil Corpus

This corpus [2] was created within the Verbmobil project the goal of which was the development of a mobile application for translation of spontaneous dialogues.

It is composed of English, German as well as Japanese dialogues, however, our version downloaded from LDC[2] contains only English and German utterances annotated with DA labels. Therefore, we evaluate the proposed approaches in English and German. Statistical information about this corpus is depicted in Table 4.1.

This dataset is annotated with 42 dialogue acts, which are grouped into the 16 following classes: *feedback*, *greet*, *inform*, *suggest*, *init*, *close*, *request*, *deliberate*, *bye*, *commit*, *thank*, *politeness_formula*, *backchannel*, *introduce*, *defer* and *offer*.

The corpus is very unbalanced. In both languages, there are four dominant DAs (namely *feedback*, *suggest*, *inform*, *request*) which represent almost 80% of the corpus size.

[2] https://www.ldc.upenn.edu/

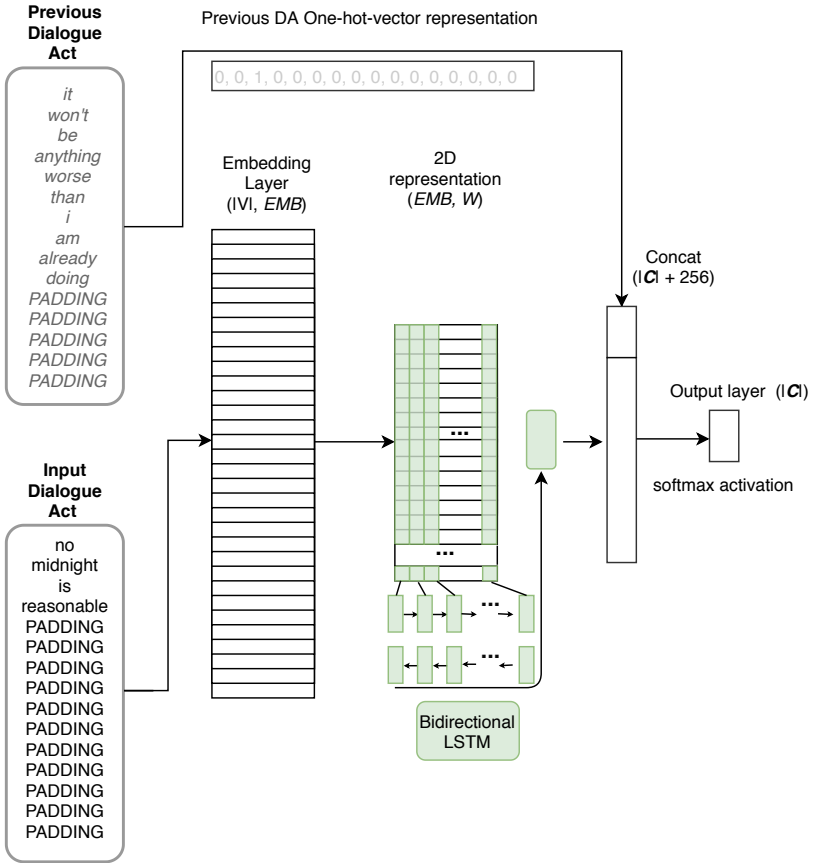


Figure 4.4: Bidirectional LSTM architecture

	English		German	
unit	Training	Testing	Training	Testing
dialogue #	6 485	940	15 513	622
DA #	9 599	1 420	32 269	1 460
word #	79 506	11 086	297 089	14 819

Table 4.1: Corpus statistical information

Experimental Set-up

We use word2vec vectors trained on the English and German Wikipedia to initialize the word embeddings in our models. Sentences are truncated or padded to 15 words in all experiments. The vocabulary size is set to 10,000.

We evaluate all models with and without information from the dialogue history, which consists of the dialogue act that has been predicted in the previous sentence.

Although most related works use the accuracy measure, we further compute the macro F1-score, because the corpus is unbalanced and therefore the F1-score is more relevant. We run all experiments 10 times and the results are averaged.

Multilingual Model Results

This series of experiments shows the results of the multilingual model. Table 4.2 reports the performance of the models with static word2vec embeddings while Table 4.3 presents the results with fine-tuned embeddings. These tables show that, generally, fine-tuning word2vec embeddings does not bring any improvement for DA recognition.

The relatively high differences between the accuracy and macro F1-score are caused by the significant corpus unbalances. Another interesting observation is that the dialogue history helps for DA recognition in all but a few cases and that the best neural classifier is the Bi-LSTM network.

		CNN_1				**CNN_2**				**Bi-LSTM**			
		With Hist.		**No Hist.**		**With Hist.**		**No Hist.**		**With Hist.**		**No Hist.**	
Train	Test	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
en	en	72.1	58.9	72.2	58.4	74.5	**65.8**	74.3	65.1	**74.9**	60.5	74.1	59.9
de	de	72.5	**60.8**	71.8	58.2	71.9	57.5	70.8	56.6	**74.3**	59.3	73.6	59.4
en+de	de	72.0	59.9	71.2	57.7	70.9	57.5	71.1	54.6	**74.3**	**61.7**	73.2	60.6
en+de	en	70.3	55.1	70.0	55.5	71.4	57.1	70.7	**58.6**	**72.8**	58.5	72.6	57.4

Table 4.2: Multilingual DA recognition with static word2vec embeddings

		CNN_1				**CNN_2**				**Bi-LSTM**			
		With Hist.		**No Hist.**		**With Hist.**		**No Hist.**		**With Hist.**		**No Hist.**	
Train	Test	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
en	en	72.2	**69.2**	72.2	68.4	**73.7**	68.4	72.1	59.1	73.5	67.2	72.7	67.2
de	de	72.7	59.2	71.7	57.7	72.1	59.1	72.6	**60.4**	**74.9**	57.2	74.3	59.1
en+de	de	71.8	**60.8**	70.8	58.9	70.8	58.4	71.7	58.8	**72.7**	58.2	71.4	58.3
en+de	en	**69.2**	61.2	68.6	58.6	69.6	61.2	68.7	60.2	68.5	60.1	**69.2**	**63.1**

Table 4.3: Multilingual DA recognition with fine-tuned word2vec embeddings

		CNN_1				**CNN_2**				**Bi-LSTM**			
		With Hist.		**No Hist.**		**With Hist.**		**No Hist.**		**With Hist.**		**No Hist.**	
Train	Test	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
en	de	30.7	11.9	**34.3**	13.9	31.5	14.5	31.2	15.4	34.0	16.4	34.0	**17.0**
de	en	55.1	26.4	54.4	25.5	53.9	28.3	53.0	27.4	**58.6**	**37.1**	57.5	33.7

Table 4.4: Cross-lingual DA recognition based on CCA transformation

4.1.6 Cross-lingual Model Results

Table 4.4 shows the results of the cross-lingual model. The scores of the cross-lingual model are significantly lower than the scores of the multilingual methods reported in Tables 4.2 and 4.3. The best reported accuracy is obtained by the Bi-LSTM network and it is close to 60% when we use the German part of the corpus for training (pivot language) and the English dataset for testing. Low F1 score occurred because of the poor results of infrequent DAs, which do not impact much the accuracy values. The lower results for the English → German direction can be explained by the significantly smaller corpus size for training. This table further shows that dialogue history slightly helps for DA recognition and that the Bi-LSTM significantly outperforms the two other CNN models.

4.1.7 Comparison with Related Work

Table 4.5 compares the performances of the proposed models with several state-of-the-art systems.

Table 4.5: Comparison with the state of the art [accuracy in %].

Method	Acc %
n-grams + complex features [87]	74.7
TBL + complex features [92]	71.2
ME + BoW features	49.1
LSTM + w2vec features [17]	74.0
CNN + w2vec features (proposed)	74.5
Bi-LSTM + w2vec features (proposed)	**74.9**

We only consider in these experiments our monolingual English models, because we have not found any cross- or multilingual results in the literature about dialog act recognition to compare with. First, we report the results of traditional feature-engineering methods, which combine a rich set of handcrafted features with dialogue history using Bayesian n-gram [87] or TBL classifier [92]. These methods have obtained the best score on the Verbmobil corpus so far.

We further implemented another baseline that uses a maximum entropy (ME) classifier with simple bag of words (BoW) features. Then, we show the results of our previous LSTM system [17], which uses only simple word-level features and word tokens from the previous dialogue act. The results of our approaches are presented in the last two lines of this table.

Although we have done our best to replicate the same experimental set-up as in the related works, some doubts subsist, because the training/testing splits are not available. Therefore, the reported results of the first two methods may not be precisely compared with the others. However, we can still conclude that the performance of our methods is comparable with the state of the art.

4.2 Cross-lingual Transfer Learning for DA Recognition

Developing DA-recognition models in other languages than English requires the costly annotation of large enough corpora. In this section, we describe our research where we exploit cross-lingual models to enable DA recognition for specific tasks (language) with a small number of annotations.

Transfer learning aims at reusing knowledge gained from a large corpus to improve the performances of small models trained on a related task with low resources. We investigate in this work two sources of information from which we transfer knowledge: pre-trained English BERT sentence (vectors) embedding and English corpus annotated with dialogue acts.

We consider low-resource target tasks – German and French DA recognition. Both datasets are small and the amount of annotated DA is limited to a few hundred samples that may be annotated by one application developer within a few hours.

Two target conditions are also considered: either with the same set of dialogue acts in German and in English or with a different set of dialogue acts in French.

In addition to the relatively large resources available in English, we further assume the availability of a relatively good machine translation system; we will use for this purpose Google's

translation. Hence, our transfer learning strategy consists first in translating every target training and test material to English and then fine-tuning our initial "large" English model to recognize these translated dialogues.

4.2.1 Models

English DA Classifier

Our initial English dialogue act recognition model trained on the English dataset annotated with dialogue acts is a multi-layer perceptron (MLP) with BERT embeddings as inputs. We have tested various topologies for this initial model, and chosen this one because of its good performances and fast training times. Each speaker turn (utterance), composed of a variable number of words, is first encoded into a single pre-trained 1024-dimensional sentence embedding vector with BERT Large[3].

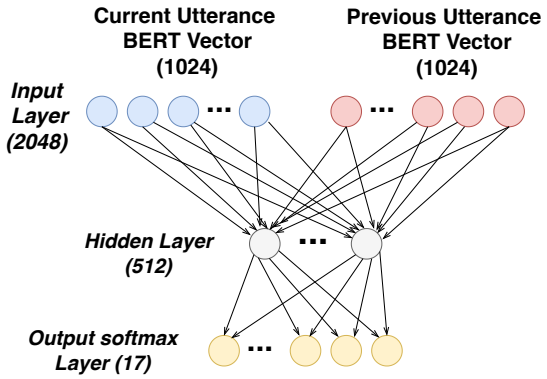


Figure 4.5: English MLP model

As shown in Figure 4.5, two such vectors are actually computed, respectively for the previous and the current speaker turns and concatenated as an input to the MLP. The MLP outputs 17 tags, which correspond to the dialogue act labels of the Verbmobil-EN corpus. This MLP has been trained on the training part of the Verbmobil-EN corpus.

Speaker Turn Embeddings

In our MLP model described previously, variable-length sentences are encoded into a unique speaker turn embedding vector with a pre-trained BERT model. We have further used two other models to compute the speaker turn embeddings: a convolutional neural network (CNN) and a multi-head self-attention (MH-SAtt) model.

The CNN model, shown in Figure 4.6, is derived from our model (see [67]). It takes as an input a word sequence truncated/padded to 15 words that always include the last two words of the sequence, following [16]. These 15 words are encoded into either random embeddings or W2V vectors. The CNN outputs a 256-dimensional vector for the current speaker turn.

The MH-SAtt model transforms each input random word embedding with the standard scaled dot-product multi-head self-attention module [110] [4]. A global max-pooling operation is

[3]from https://github.com/google-research/bert#pre-trained-models – BERT-Large, Uncased (Whole Word Masking): 24-layer, 1024-hidden, 16-heads, 340M parameters

[4]https://github.com/CyberZHG/keras-multi-head

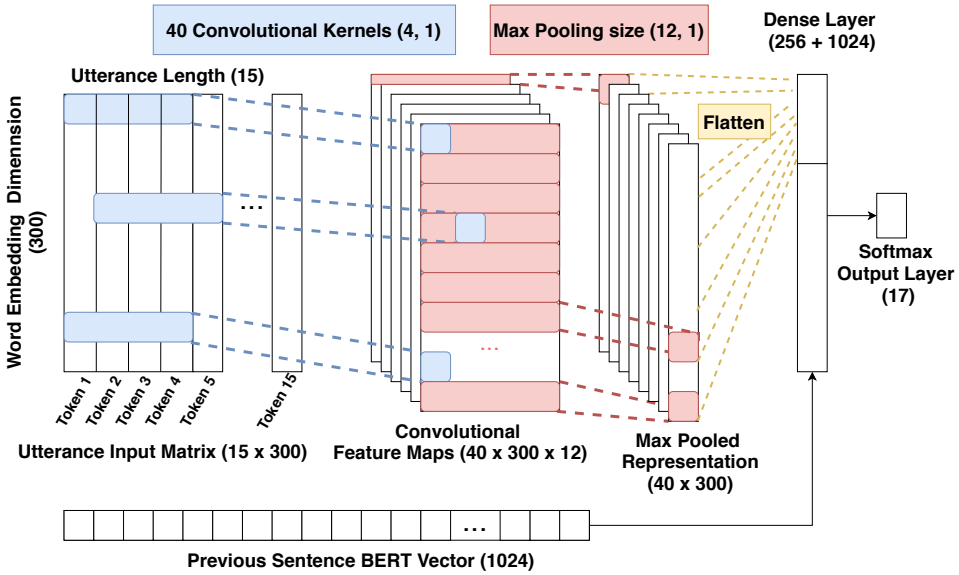


Figure 4.6: CNN model for DA recognition

then applied to compute the speaker turn embedding. Figure 4.7 shows how this model is used in our experiments.

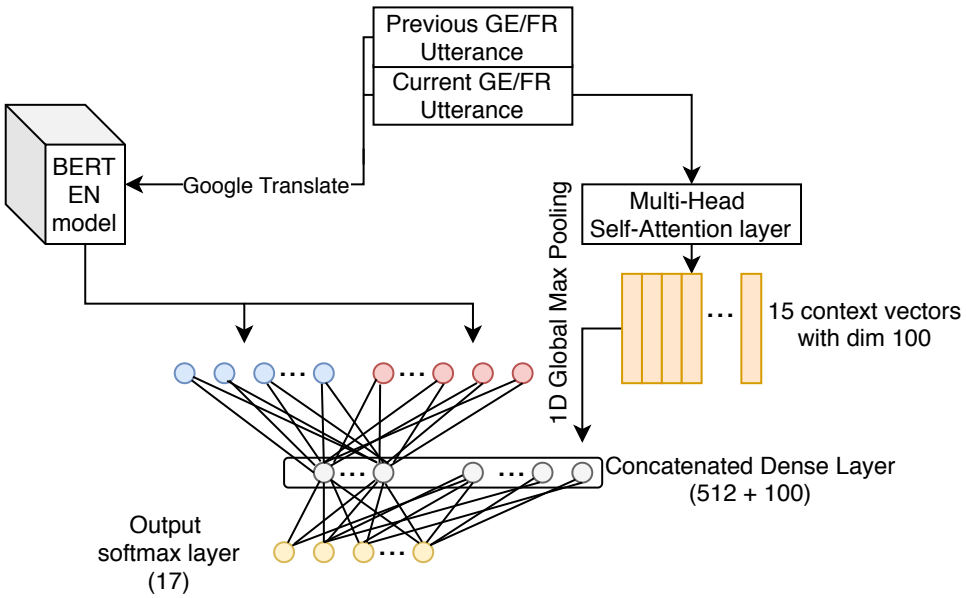


Figure 4.7: Multi-head self-attention model

In all our models, the previous speaker turn is also encoded into a 1024-dimensional vector with BERT and injected into the classification step: this is an easy way to take into account the previous dialogue act, without requiring a costly RNN, CRF, and/or beam search procedure.

4.2.2 Transfer Learning Approach

Pre-trained BERT is famous for transferring general English lexical information into a large variety of downstream NLP tasks. We propose an approach to exploit them for cross-lingual dialogue act recognition through automatic language translation. Similarly, we evaluate pre-trained word2vec English vectors in this context.

We further investigate the option to stack the translated inputs with the original foreign representations in the MH-SAtt model in order to increase the robustness of the model to translation errors.

In all our experiments, we only use English pre-trained W2V and BERT embeddings and assume that no such embeddings exist in the target language. Of course, such pre-trained embeddings do exist both in German and French, but we do not exploit them as this is not the case in all languages, and their quality is not always comparable.

Assuming that the application developer has manually designed and annotated a few hundred samples for his target non-English task, we then evaluate the benefit from fine-tuning the English DA classifier on this target small dataset. Our global transfer learning approach thus consists of two phases:

1. **Initial phase**
   The purpose of this phase is to train the original English model from which we will transfer the parameters.

2. **Fine-tuning phase**
   First, all foreign sentences are automatically translated into English with Google Translate (but any other translation system may also be used), in order to be able to reuse our original English DA classifier.

   Then, the final classification layer (from Figures 4.5, 4.6, 4.7) is adapted to accommodate for a potentially different number of outputs, and its parameters are randomized.

   Finally, the model parameters are trained for a few epochs on the small target corpus translated into English.

The diagram shown in Figure 4.8 summarises the two phases for both target languages, with a common initial phase.

4.2.3 Experiments

The objective of the following experiments is to validate our cross-lingual transfer learning proposal for dialogue act recognition on low-resource application domains. We apply transfer learning from English to German and French languages. Every experiment is run 10 times and the results are averaged. The standard deviation is also computed.

German Data

We build a low-resource German dialogue act corpus by randomly sampling 100 utterances from the Verbmobil-GE training corpus with the same dialogue acts distribution as in the complete training corpus, see Table 4.6. The model trained on this small corpus will be tested on the complete (1460 utterances), standard Verbmobil-GE test corpus.

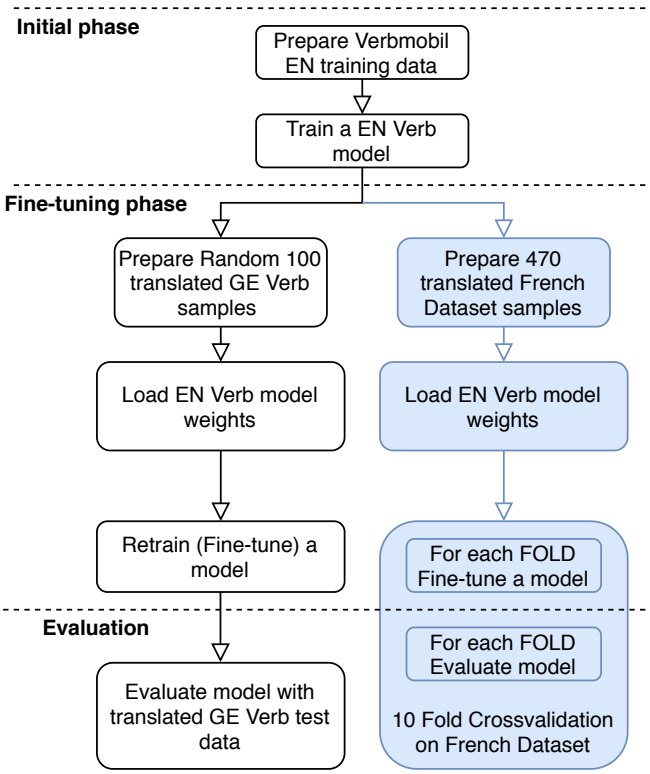


Figure 4.8: Proposed fine-tuning transfer learning

Label	Occurence
FEEDBACK	28%
SUGGEST	19%
INFORM	18%
REQUEST	9%
GREET	4%
BYE	4%
INIT	4%
BACKCHANNEL	3%

Label	Occurence
DELIBERATE	3%
INTRODUCE	2%
COMMIT	1%
CLOSE	1%
POLIT. FORM.	1%
THANK	1%
DEFER	1%
OFFER	1%

Table 4.6: Distribution of DAs in Verbmobil-GE corpus

French Data

We manually annotate 470 turns from the TCOF corpus [4] with dialogue acts. This corpus contains manual transcriptions of spoken dialogues in French recorded by linguists in real-life situations involving volunteer citizens. The types of dialogue in our 470 turns are very different from the ones found in standard DA corpora: they involve two friends trying to find a suitable gift, three students talking about their courses while chatting with someone else on their smartphone and an adult talking with a young girl who is drawing. Hence, the French corpus is annotated with a slightly different set of labels, which distribution is shown in Table 4.7. This French corpus is freely distributed with a CC BY-NC-SA license[5].

[5]https://github.com/cerisara/TCOFDA

Label	Occurence
INFORM	26%
AGREE	13%
BACKCHANNEL	10%
Y/N QUESTION	10%
OPEN QUESTION	9%
PERFORMATIVE	8%

Label	Occurence
OPEN ANSWER	7%
DISAGREE	5%
YES ANSWER	5%
NO ANSWER	5%
OTHER ANSWERS	1%
GREETINGS	$< 1\%$

Table 4.7: Distribution of DAs in the French corpus

Initial Phase: English Model

The initial English MLP model is trained on the full Verbmobil-EN corpus. The hyper-parameters of this model are tuned on the English corpus: this model is thus trained for 200 epochs with a learning rate of 0.002. Table 4.8 compares its accuracy when trained either from BERT embeddings or from speaker turn embeddings obtained with the CNN model. The CNN may use initial random or pre-trained W2V word embeddings. Every experiment is run 10 times and the results are averaged.

Model	Embeddings	Epochs	Test Acc	Std. Dev.
MLP	BERT	200	0.734	0.010
CNN	W2V	200	0.704	0.009
CNN	Random	200	0.702	0.008

Table 4.8: Initial Phase – English MLP Model trained on Verbmobil-EN (9599 turns) and tested on Verbmobil-EN (1460 turns). The **Embeddings** column indicates how the word embeddings are initialised.

Fine-Tuning Phase

Fine-tuning consists of training the last classification layers of the models presented that have been presented previously. on the small foreign corpus that has been translated into English. This fine-tuning process thus involves some additional hyper-parameters, such as the learning rate and the fixed number of epochs, which are tuned on a German development corpus composed of 10,000 utterances randomly sampled from the Verbmobil-GE training corpus. The same hyperparameter values found on this German corpus are also used for the French model[6].

Baseline Approaches

Three baseline classifiers are shown in the first part of Table 4.9:

1. **Majority Class Classifier (MC)**: it always predicts the most common DA in the training dataset;

2. **Training from Scratch**: Instead of transferring the model parameters from the initial English MLP model, the parameters (without the embeddings) are randomly initialized in this baseline;

[6]All hyperparameter values along with the source code are distributed with an open-source license in https://github.com/cerisara/TCOFDA

3. **No-Fine-Tuning**: The model parameters are simply transferred from the initial English MLP model, and no further training is done.

Fine-Tuning Experiments – German

Table 4.9 shows the results of our models on the Verbmobil-GE test set. All these models process only translated English sentences, except for the last MH-SAtt model that further includes the original German words, as shown in Figure 4.7.

Model	Embeddings	Epochs	Test Acc	Std. Dev.
Baseline MC		–	0.279	–
From scratch				
CNN	W2V	15	0.410	0.012
MLP	BERT	25	0.468	0.008
No-Fine-tuning				
CNN	W2V	–	0.380	
MLP	BERT	–	0.479	
Fine-Tuning				
CNN	W2V	15	0.463	0.008
MLP	BERT	25	0.484	0.025
MH-SAtt	BERT + GE	50	0.502	0.012

Table 4.9: Accuracy on the Test Verbmobil-GE corpus

With regard to transfer from pre-trained models, BERT is consistently better than W2V. Simply reusing the English models without fine-tuning gives the worst results, while fine-tuning the English models on the small target corpus systematically improves the results. The best performances are obtained with the MH-SAtt model that combines both BERT pre-trained vectors and fine-tuning transfer learning with the original and translated word sequences.

Fine-Tuning Experiments – French

We carried out 10-fold cross-validation on the 470 speaker turns of the French corpus. As we do not have enough data to create a development corpus in French, we share the same hyperparameters as found on the German corpus. Table 4.10 show the results.

Model	Embeddings	Epochs	Acc	Std. Dev.
Baseline MC		–	0.210	–
From scratch				
CNN	W2V	15	0.403	0.006
MLP	BERT	25	0.421	0.005
Fine-Tuning				
CNN	W2V	15	0.400	0.009
MLP	BERT	25	0.430	0.008
SelfAtt	BERT+FR	50	0.436	0.007

Table 4.10: French cross-validation experiments

The relative contributions of the various sources of information and models are similar to the

German experiments. However, the differences between the results are much smaller, which is likely due to the fact that the hyperparameters have not been tuned on French but on German.

4.3 Dialogue Act Recognition from Image Documents

In previous sections, we presented models and experiments for DA recognition while focusing strictly on text representation of dialogues and presented several approaches for tackling multi- and cross-linguality in this task.

This section describes DA recognition using visual information. We have conducted research on the multimodal approaches for DA recognition in a printed form.

The goal of this section is, among others, to show how image processing methods are connected with NLP. We demonstrate this connection on the DA recognition task. Before we delve into further explanation, we summarize techniques that we utilize for this task.

- **Image pre-processing:**
  - Binarization;
  - Morphological operations;
  - Connected Components Analysis.
- **Segmentation of the image:**
  - Machine learning (or computer vision) approaches;
  - Detection and segmentation of text-line images.
- **OCR methods**
- **Text classification:**
  - DA recognition.

All above-mentioned techniques will be described in detail in the text further. We first outline the task by providing a brief introduction and next we present models and experiments. Then, we highlight the main contributions and provide the future work.

A dialogue system standard input is a speech signal which can be converted into textual representation using an automatic speech recognition (ASR) system [27]. However, dialogues are also available in a written form (e.g. books and comics), and their automatic analysis is also beneficial.

Similarly, as in the DA recognition from the audio signal, we first convert the images into a lexical representation using OCR methods. We assume that the image form (as the speech signal) contains some additional information.

Therefore, the main contribution of this work lies in the usage of visual information for automatic DA recognition from printed dialogues. To the best of our knowledge, there is no prior work that focuses on DA recognition from printed / handwritten documents.

For evaluation, we create a novel image-based DA recognition dataset from written dialogues. This corpus is based on the dialogues from the Verbmobil corpus [40].

We further assume that with the decreasing quality of the printed documents, the importance of the visual text representation will play a more important role for DA recognition, since a recognized text contains a greater amount of OCR errors. We will evaluate this hypothesis using four different image quality in the corpus.

For visual DA recognition, we use a similar model as we presented in the previous chapter – the Convolutional Recurrent Neural Network (CRNN). To verify the benefits of visual information, another text-only model – BiLSTM is used for comparison.

4.3.1 Model Architectures

We describe gradually three models we use for the DA recognition. First of all, we present the visual model that we use for DA recognition based only on image features. Next, we describe our text model and, finally, the joint model that combines both image and text inputs.

Visual Model

The key component in this model is the Convolutional Recurrent Neural Network (CRNN) that has been successfully utilized for OCR (e.g. [98, 71]) and also for image classification [35]. For the visual DA recognition, the input is the image of an entire page of a dialogue where each text line represents an utterance. This page is processed by the *Utterance Segmentation* module that produces segmented images of text lines. These images are fed into the CRNN that maps each utterance to the predicted label. The scheme of this approach is depicted in Figure 4.9.

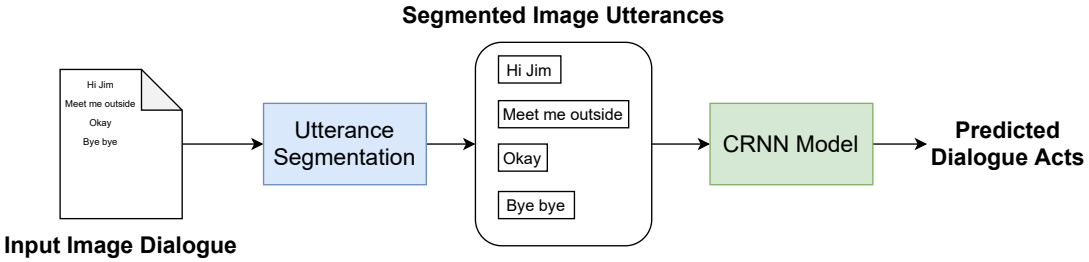


Figure 4.9: Visual DA recognition model

Convolutional layers within CRNN create feature maps with relation to the specific receptive fields in the input image. Due to the pooling layers, dimensionality is reduced, and significant image features are extracted, which are further processed by recurrent layers. The recurrent layers are fed by feature sequences (the feature vectors in particular frames in the image). The CRNN model is depicted in Figure 4.10 in two forms: the original model proposed by Shi et al. [98] for OCR and our adapted version for DA recognition.

The inputs of both models are segmented images of utterances. The activation function of convolutional and recurrent layers is ReLU and we employed the *Adamax* optimizer.

The crucial part of the OCR model is the connectionist temporal classification (CTC) loss function which has been presented by Graves et al. [32]. As we already explained, the CTC is designed to create an alignment between the labels and specific image frames. It allows using a simple form of annotation, for example, image and annotation text without the necessity of providing the precise character positions in the image. The output of the BiLSTM is given to the output which represents a probability distribution of characters per image frame.

The right part of Figure 4.10 visualizes our modified version for the image-based DA recognition. It doesn't utilize the CTC loss function but we use the *categorical cross-entropy* since the output is a vector of probabilities indicating the membership in the particular DA class. The size of the output layer corresponds to the number of recognized DA categories.

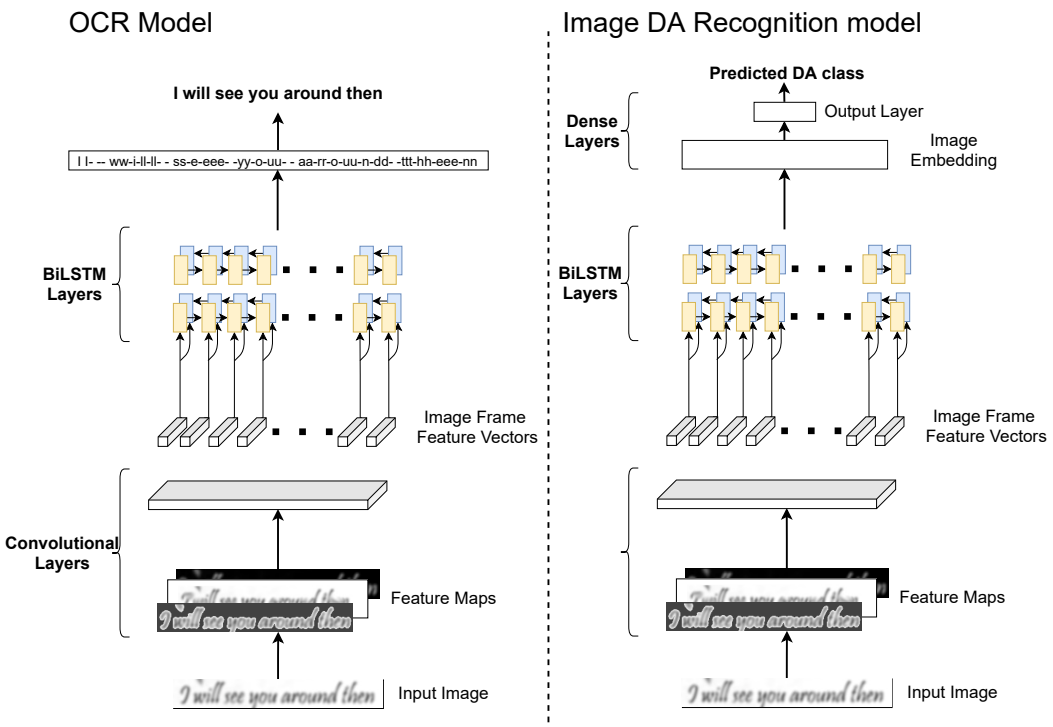


Figure 4.10: CRNN models: OCR model proposed by Shi et al. [98] (left); our modified version used for visual DA recognition (right)

Text Model

The centerpiece of this model is the Bidirectional Long Short-Term memory [37] (BiLSTM). The input utterance is aligned to 15 words, so the utterances with less than 15 words are padded with a special token while the longer ones are shortened. We chose *word2vec* [75] embeddings as a representation of the input text. The word vectors (with the dimension equal to 300) are fed into the BiLSTMlayer and the final states of both LSTMs are connected to a dense layer with size 400. Then a DA label is predicted through the softmaxed output layer. The model is depicted in Figure 4.11.

Joint Model

The second employment of the CRNN is in the combination with the text model presented in the previous section. The objective is to create a joint model that takes multimodal input (segmented utterance image and simultaneously the text of an utterance). Figure 4.12 shows the joint model with both inputs.

Since the input text doesn't have to be well-recognized, some words which are out of vocabulary might appear resulting in a worse performance of the text model. In such a case, the image embedding input should help to balance this loss of text information.

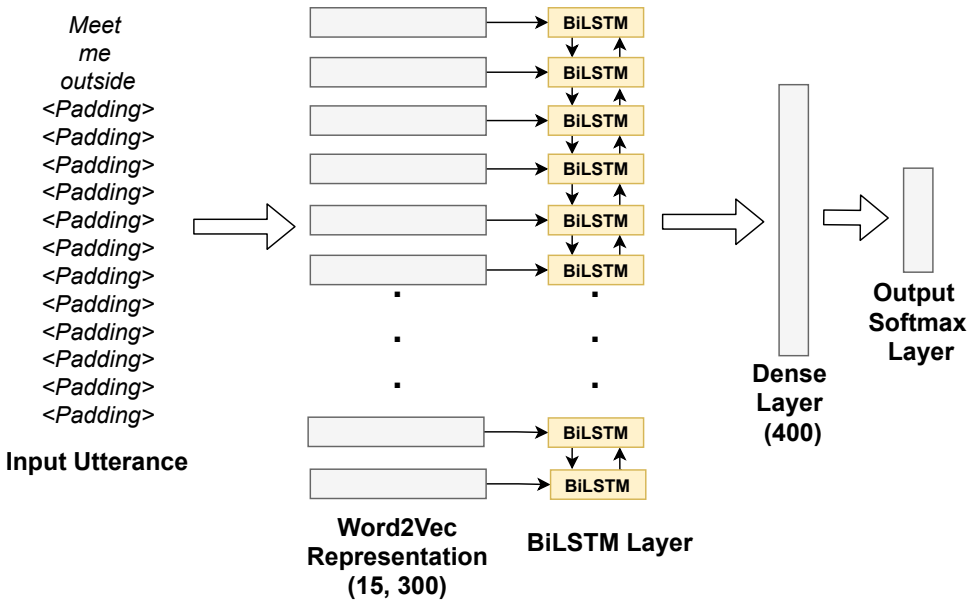


Figure 4.11: Text DA recognition model

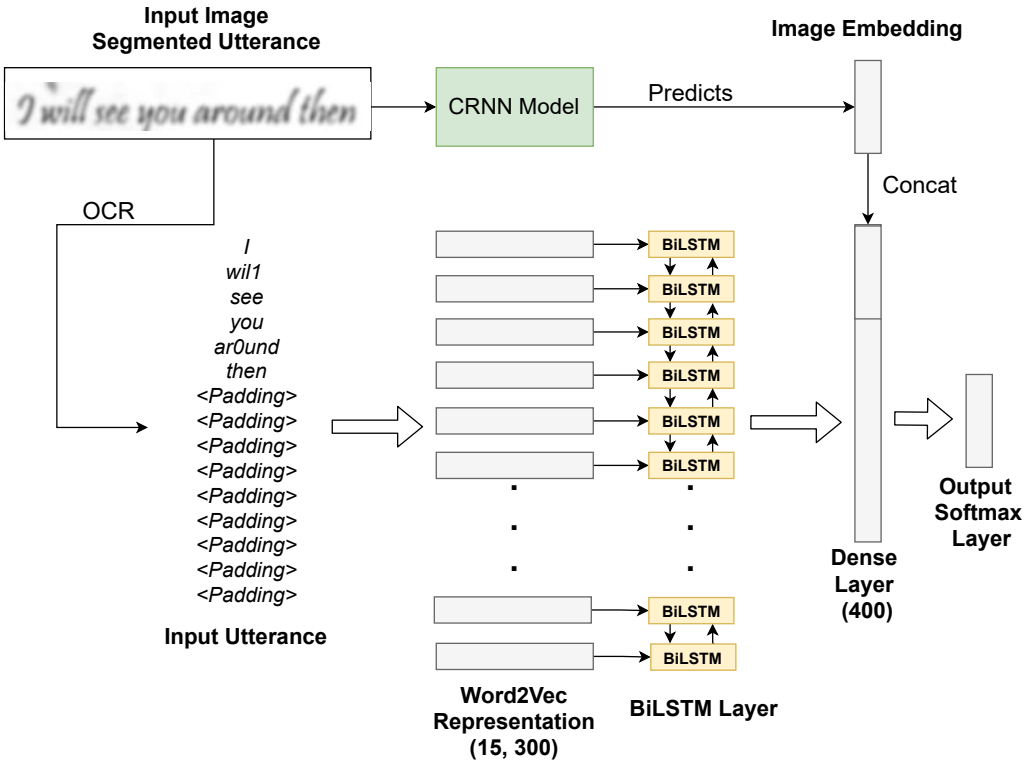


Figure 4.12: Joint DA recognition model

4.3.2 Multimodal Dataset

The multilingual Verbmobil dataset [40, 2] contains English and German dialogues, but we limit ourselves only to the English part. The dataset is very unbalanced. The most frequent labels are `FEEDBACK` (34%) and `INFORM` (24%) while the eight least frequent labels occur in only 1% of utterances or less.

Table 4.11 shows the distribution of Verbmobil EN labels in the training part of the corpus.

Label	Occurrence	Label	Occurrence
FEEDBACK	34%	GREET	1%
INFORM	24%	DEFER	1%
SUGGEST	14%	COMMIT	1%
REQUEST	11%	POLIT. FORM.	1%
BACKCHANNEL	4%	INTRODUCE	$< 1\%$
DELIBERATE	4%	THANK	$< 1\%$
INIT	2%	CLOSE	$< 1\%$
BYE	2%	OFFER	$< 1\%$

Table 4.11: Distribution of DAs in Verbmobil-EN train dataset

The Verbmobil data are already split into training and testing parts and stored in `CONLL` format. We created validation data by taking the last 468 dialogues from the training part. To summarize, we have 8921 utterances in the training part, 667 utterances as validation data, and, finally, 1420 utterances serve as our test dataset.

Image Dataset Acquisition

For each dialogue, we have created four pages with image backgrounds of different noise levels and programmatically rendered the utterances.

The first background (*noise_0*) contains no noise (perfectly scanned blank piece of paper) while the fourth level (*noise_3*) contains a significant amount of noise[7].

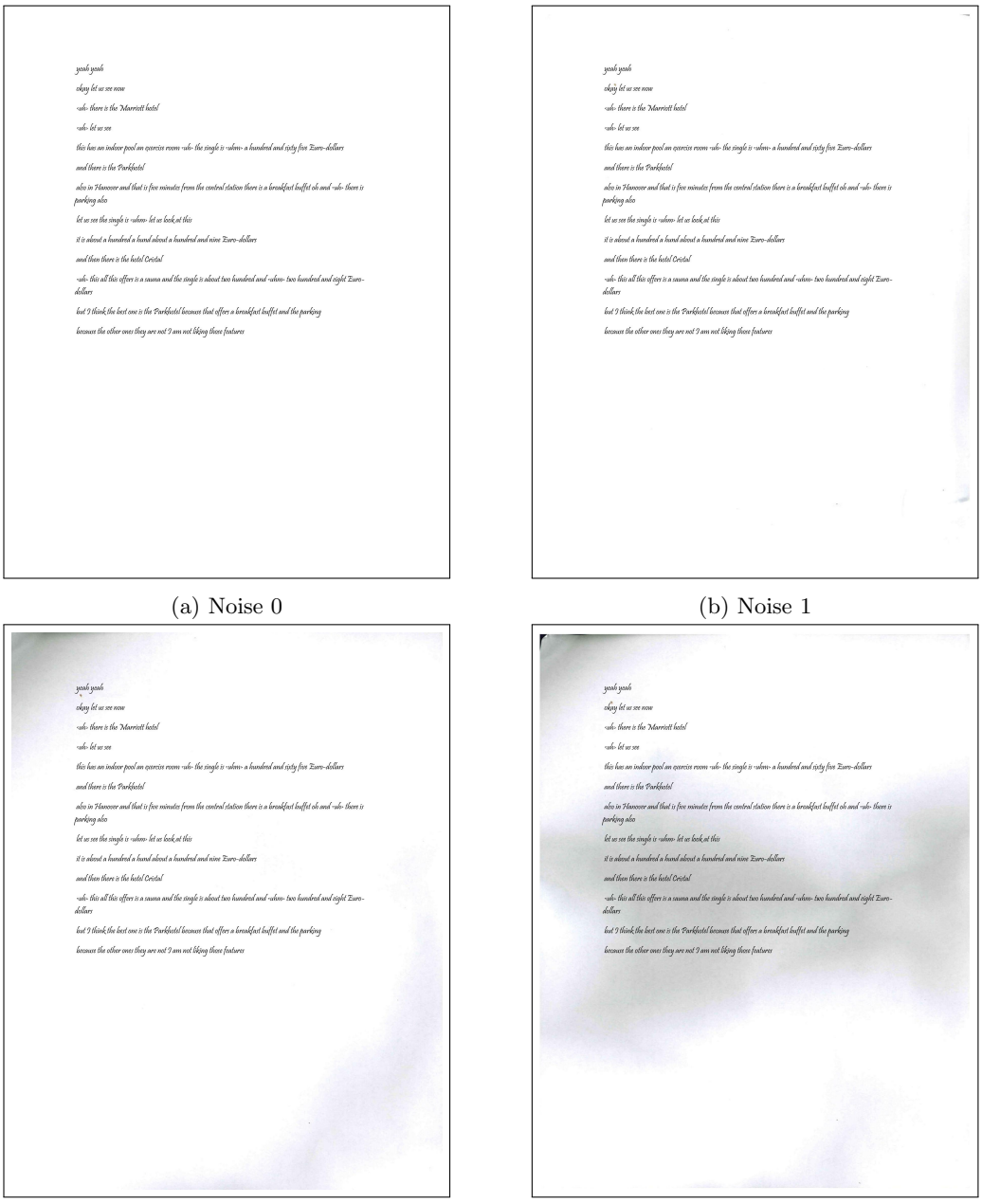
Each rendered utterance is considered as a paragraph. We must take into account, though, the utterances that are too long to fit the page width. In such a case, it continues on the next line and we would struggle with the situation where the beginning of the next utterance and continuing of the current utterance would be indistinguishable. Therefore, we increased the vertical space between paragraphs and we employed the intending of the first line of paragraphs. These two precautions together solve the above-mentioned potential problem and make the segmentation easier. Another parameter that can be used to adjust the dataset difficulty is the font. We chose the *Pristina Font* which is a hybrid between printed and handwritten font.

Summing up, four steps of the acquisition of the image dataset are as follows:

1. Split original Verbmobil `CONLL` files to the individual dialogues;
2. Create the realistic scanned noisy background;
3. Choose a font;
4. Render the dialogues according to the above-mentioned scenario.

Figure 4.13 shows the examples of each dataset.

[7]The noise is not artificial (i.e. we didn't perform any image transformation), but we have created the noise by real usage of the scanner. We put a blank piece of paper in the scanner and we changed the scanning quality by different scanning options and the amount of light.



(a) Noise 0

(b) Noise 1

(c) Noise 2

(d) Noise 3

Figure 4.13: Examples of the pages in all four datasets

We have artificially applied random image transformations (rotation, blurring, and scaling) to create a second version of each of the four datasets. These transformations significantly increase the difficulty of our task because the segmentation and OCR will become harder to perform. We call this version the *transformed dataset* and in the following text, it will be labeled as follows: (*noise_0_trans*, *noise_1_trans*, *noise_2_trans*, *noise_3_trans*). So in total, we

have eight datasets of different noise levels and difficulties.

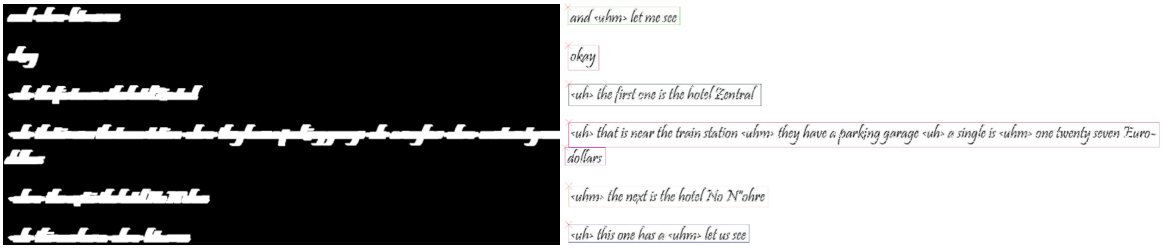
Utterance Segmentation

This section describes the algorithm we used for segmentation of the entire page into individual text line images – utterances. We utilized a simple segmentation algorithm based on the analysis of connected components.

We first employed the Sauvola thresholding [94] to binarize the input image that is a necessary step to perform the analysis of the connected components. Before getting to that, though, we carry out the morphological dilation to merge small neighboring components that represent fragments of words or individual characters (see Figure 4.14). The ideal case is if one text line is one connected component.

Figure 4.14: Example of the morphological dilation with kernel $(2, 10)$

Thereafter, the analysis of the connected components is conducted. Figure 4.15 shows the output of this algorithm. The left part of the image shows the binarized image after morphological dilation while the right part depicts the bounding boxes detected by the analysis of connected components.



(a) Dilated binary image

(b) Detected bounding boxes

Figure 4.15: Utterance segmentation

Once the bounding boxes are obtained, we crop these regions from the image and resize them to the common shape (1475×50). To maintain the image quality of narrow images, we perform the image expansion to the desired width by padding with a white background.

4.3.3 Experiments

Within this section, we first present the comparison with state-of-the-art (SoTA) results and then we quantify the difficulties of our datasets by measuring the OCR performance. The next experiment presents results with various sizes of image embedding within the visual model and their influences on the overall success rate.

We split the remaining experiments into three scopes to investigate the impact of the visual information in the DA recognition task. The first scope is "image-only" and its goal is to verify the performance of the visual model presented in Section 4.3.1. The second scope is called "text-only" and similarly to the first scope, the goal is to evaluate our text model. The purpose

of the final scope is to find the best joint model which is robust enough to be able to respond to the deteriorating quality of text input.

For all experiments, we employed the *Early Stopping* that checks the value of the validation loss to avoid over-fitting. We ran every experiment 5 times and we present average accuracy, macro F1-score, and also standard deviation of each run evaluated on the testing part of each dataset.

Comparison with SoTA

Table 4.12 compares the results of our text model with state-of-the-art approaches on the testing part of the English Verbmobil dataset. This table shows that our results are comparable, but we need to take into account that some approaches in the table utilize the information about the label of the previous utterance. In this work, we did not use this information, since the utterance segmentation from the image is not perfect. Some utterances may be skipped or merged that results in jeopardizing the continuity of the dialogue.

Method	Accuracy
n-grams + complex features [87]	74.7
TBL + complex features [92]	71.2
LSTM + Word2vec features [16]	74.0
CNN + Word2vec features [67]	74.5
Bi-LSTM + Word2vec features [67]	**74.9**
Text model (proposed)	73.9

Table 4.12: Comparison with the state of the art [accuracy in %].

OCR Experiment

We use Tesseract as the OCR engine within this work. We measured the OCR performance by calculating the Word Error Rate (WER) and Character Error Rate (CER) against ground truth text in `CONLL` files.

Tesseract was employed on the testing part (1420 utterances) of each 8 datasets. The results are presented in Table 4.13 and depicted in Figure 4.16. We can conclude that with the increasing noise and difficulty, the WER and CER values are increasing as expected.

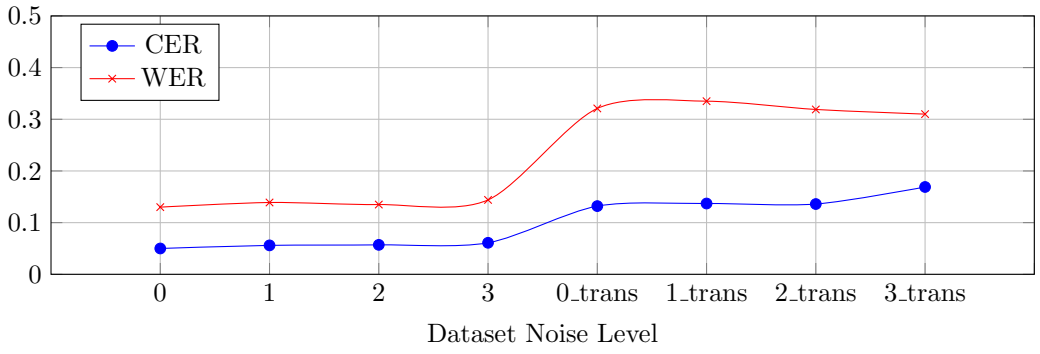


Figure 4.16: Average Word Error Rate (WER) and Character Error Rate (CER) of all datasets

	Dataset Noise Level							
	0	**1**	**2**	**3**	**0_trans**	**1_trans**	**2_trans**	**3_trans**
WER	0.132	0.149	0.132	0.143	0.319	0.322	0.306	0.325
CER	0.049	0.053	0.049	0.053	0.131	0.128	0.128	0.168

Table 4.13: OCR experiment – Word Error Rate (WER) and Character Error Rate (CER) over all datasets

Image Embedding Dimension

The goal of this experiment is to find the optimal dimension of the image embedding (the size of the penultimate dense layer in the Visual model).

For this purpose, we use only the dataset with the poorest quality (*noise_3_trans*). We started at a dimension equal to 100 and this value was gradually increased by 100. Within each run, a new model with a particular embedding size was trained and evaluated. Figure 4.17 shows the results of this experiment. We present accuracy as the evaluation metric. The number of epochs that are needed for training was in the interval 8 – 16 depending on the early stopping.

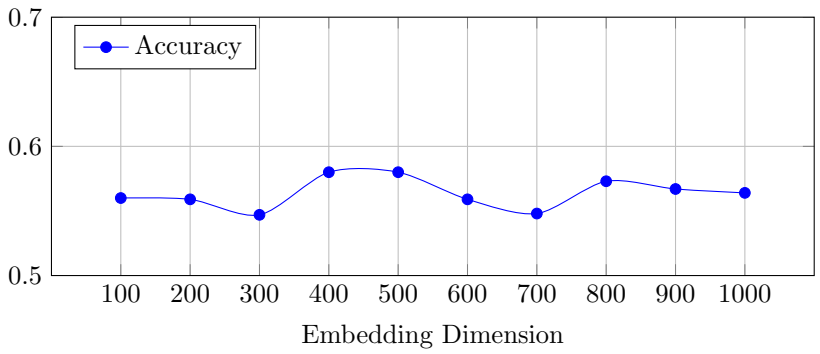


Figure 4.17: Experiment to determine the optimal image embedding dimension. The standard deviation did not exceed the 0.006 for all runs.

We have here an interesting observation that the amount of information is not increasing with the higher dimension. The best results were obtained with values 400 and 500. So for the next set of experiments, we chose the value of the image embedding dimension equal to 500.

Visual Model Experiment

Table 4.14 shows the performance of the Visual model. This table illustrates that the results are relatively consistent for all given datasets.

	Dataset Noise Level							
	0	**1**	**2**	**3**	**0_trans**	**1_trans**	**2_trans**	**3_trans**
F1	46.8	54.5	47.2	43.7	47.3	43.7	43.2	40.2
Acc.	56.6	54.9	57.8	54.6	59.1	56.5	59.4	55.9

Table 4.14: Visual model DA recognition results [in %]

Text Model Experiment

Within a training phase, the model is fed with a text from the Verbmobil dataset while in the evaluation (prediction) phase the input utterances are provided by the OCR. Our intention is to create a real situation where only images with rendered text will be available and the only way to acquire the text itself is to use OCR methods. Table 4.15 shows accuracies and macro F1-scores for all datasets.

	Dataset Noise Level								
	0	**1**	**2**	**3**	**0_trans**	**1_trans**	**2_trans**	**3_trans**	**GT**
F1	59.2	59.9	54.8	59.6	50.9	54.2	56.6	52.2	61.6
Acc.	71.9	70.4	71.8	71.2	56.2	56.4	58.1	56.0	73.9

Table 4.15: Text model DA recognition results [in %]

The left part of the table presents the results on not transformed datasets (*Noise_0* – *Noise_3*). For these datasets, the OCR results turned out well (see Section 4.3.3 – the average CER value around 0.05), which corresponds to accuracy exceeding 0.7.

The results on transformed datasets (*Noise_0_trans* – *Noise_3_trans*) are presented in the right part of the table. The OCR performed significantly worse (average CER in range 0.12 – 0.16). Hence, the results are worse as well.

For completeness and comparison, the rightmost column of Table 4.15 shows the results when the perfect ground truth text (from the `CONNL` Verbmobil files) is used instead of the recognized text. This accuracy is used for the comparison with state of the art (Section 4.3.3).

Last but not least, for transformed datasets, in terms of accuracy, the Image and Text model performed similarly. For datasets without transformation, the Text model was significantly better, primarily due to the fewer amount of recognition errors.

Joint Model Experiment

The fact that it is possible to successfully train a Visual DA recognition model based solely on images with reasonable results brought us to the idea to use learned image features in combination with text to create the joint model. We assume that it might have better adaption to recognized text with a significant amount of errors.

Similar to the text model, to simulate the real situation, the ground truth text from the Verbmobil dataset is used to train the model while a recognized text from the OCR is used to test the model to verify its generalization. As long as the very same text is used in both text and joint model, it is very easy to verify and measure the positive impact and the contribution of the visual information.

Our final experiment shows, among other things, the impact of the information which was embedded into a single image feature vector (image embedding) by the CRNN model. Based on the preliminary experiment, we chose the dimension of embedding equal to 500.

We have eight stored CRNN models that have been trained separately on particular datasets. We remind that the training of the joint model was carried out in the same way as the training of the text model. The only difference from the previous Text Model experiment is the usage of an auxiliary image input which is predicted by the CRNN model as depicted in Figure 4.12. We present the results in Table 4.16.

As you can notice, the results no longer oscillate so much across all datasets. Another important observation is that some transformed dataset results outperformed results based on the not transformed datasets (compare *Noise_0* and *Noise_2* with their transformed versions).

	Dataset Noise Level							
	0	**1**	**2**	**3**	**0_trans**	**1_trans**	**2_trans**	**3_trans**
F1	49.6	56.8	50.3	51.9	48.3	46.8	54.2	49.3
Acc.	70.3	69.3	70.1	68.8	61.2	59.9	66.4	60.1

Table 4.16: DA recognition results with Joint model [in %]

The help of auxiliary image input has a bigger impact on transformed datasets where the amount of noise is massive and vice versa.

Figure 4.18 shows the visual comparison of all models we used in our experiments. The blue curve shows visual Model results, the red line represents text Model results and the green line depicts the performance of the joint model (text and image input).

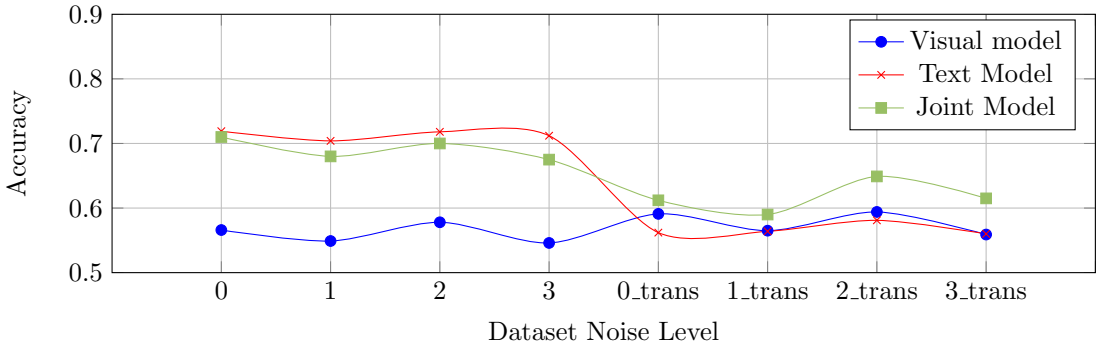


Figure 4.18: Depicted results and comparison of all models

As expected, in the case of a better quality of recognized text (*Noise_0* – *Noise_3*), the performance of the text model is the best. However, if the quality of the recognized text is low (*Noise_0_trans* – *Noise_3_trans*), accuracy and macro F1-score decrease.

4.4 Conclusion

We presented our multi- and cross-lingual approaches for DA recognition in first two sections of this chapter. The third section detailed our research in multi-modal dialogue act recognition area.

First, we focused on a multilingual strategy, where we have compared and evaluated two different CNN configurations and one Bi-LSTM on the Verbmobil corpus in English and German. We have shown that the multilingual model significantly outperforms the cross-lingual approach based on semantic space transformations. The multilingual model is less flexible and may not scale easily to many languages, because retraining is necessary when adding new languages. We have confirmed that the dialogue history is beneficial for DA recognition in almost all cases and that the Bi-LSTM model outperformed CNN. We have also compared our approaches with several state-of-the-art methods in the monolingual scenario and concluded that the performance of our methods is comparable with the state of the art.

Then, we presented the methods for cross-lingual transfer learning. We have investigated two types of transfer learning: pre-trained word embeddings and classifier fine-tuning. Three types of sentence representations, namely BERT, CNN, and multi-head self-attention were utilized. The

objective was to port a reference DA recognition model trained on English to another language and dialogue context with only a limited amount of annotated resources. We have validated these approaches on two target languages, German and French, and developed a dedicated French DA corpus with real-life dialogues recorded in quite different conditions than the existing standard DA corpora.

The main conclusion is that all available sources of information are required to obtain the best results, but also that some data should be reserved for proper hyperparameter tuning. Another conclusion is that automatic translation introduces some bias and mistakes that need to be compensated with fine-tuning. The reduced success of transfer learning for French is likely due to the greater mismatch between the corresponding corpus types, and transfer learning does not totally compensate for the lack of training data in the target domain. Nevertheless, the proposed transfer learning architecture gives good enough results to open the way to future research in transferring rich English DA systems to other languages and less explored domains.

The last section presented the task of dialogue act recognition in the written form using a model with multimodal inputs. First, we have successfully employed the CRNN model as the visual model for image-based DA recognition. We have shown that despite employing only visual features it is possible to obtain reasonable results in the task that is dominantly text-based.

Second, we have carried out a set of experiments where we have used the same CRNN model as an image feature extractor and we have combined it with the BiLSTM text model for handling both text input (obtained by OCR) and image input. We have successfully extracted the hidden layer representation of the CRNN model (image embedding) and together with the text model we have created the joint model. For poor-quality datasets, where the OCR success rate is low, we have outperformed the text model that uses solely text input.

Hereby, we have shown that the visual information is beneficial and the loss of the text information is partially compensated. The impact of such image features results in improving accuracy (4% – 10%) depending on the noise level in the particular dataset.

"The greatest teacher, failure is."
Master Yoda

5 Conclusion

"There are no facts, only interpretations."
Friedrich Nietzsche

This thesis dealt primarily with dialogue act (DA) recognition task. We first explored multi and cross-lingual dialogue acts recognition. We used convolutional and recurrent neural network models but we also employed modern Transformer-based architectures as well (Multi-Head Self-Attention and BERT-like models). For cross-lingual approaches, we explored semantic space transformations and transfer learning approaches.

As the main contribution, we presented DA recognition from image documents since dialogues do not occur only in speech but also in a written form (e.g. comic books). Such DA recognition has not been covered by research yet. Therefore, we designed and employed multimodal model that takes into account text inputs from OCR as well as images of text regions. We showed that our multimodal model deals with the erroneous text and visual information partially balance this loss of information.

Since one of the main components for DA recognition from image documents is OCR, our second topic in this thesis is related to the OCR and the analysis of image documents in general. We conducted experiments with historical German documents (printed newspaper in Fraktur from the end of the 19-th century). In such a domain, there is often lack of annotated data and we struggle with bad quality documents. Hence, we experimented with different training strategies for training an OCR model with a lack of training data.

A crucial note which we want to highlight is the fact that we were able to connect computer vision techniques together with OCR with DA recognition from image documents. The conclusions and experiments are promising and are a good basis for the following dialogue detection and DA recognition from comic books that are an excellent example of image dialogue documents.

The goals which were set at the beginning of this thesis have been fulfilled. The whole research related to the OCR was beneficial and obtained knowledge has been successfully employed in our paper "Dialogue Act Recognition using Visual Information". We showed possible connections between computer vision and NLP and with the help of the above-mentioned paper, we created a "bridge" between computer vision and NLP.

List of Acronyms

ASR	Automatic Speech Recognition
BERT	Bidirectional Encoder Representations
BOW	Bag of Words
CCA	Canonical Correlation Analysis
CER	Character Error Rate
CNN	Convolutional Neural Network
CoVe	Context Vectors
CRF	Conditional Random Field
CRNN	Convolutional Recurrent Neural Network
CTC	Connectionist Temporal Classification
DA	Dialogue Acts
DAR	Dialogue Acts Recognition
DNN	Deep Neural Network
ELMo	Embeddings from Language Models
FCNN	Fully Convolutional Neural Network
GAN	Generative Adversarial Networks
GPT	Generative Pre-Training
GRU	Gated Recurrent Unit
HAL	Hyperspace Analogue to Language
HMM	Hidden Markov Model
HTR	Handwritten Recognition
LER	Label Error Rate
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory
MLP	Multi-layer Perceptron
MRDA	Meeting Recorder Dialogue Acts
NLP	Natural Language Processing
NMT	Neural Machine Translation
OCR	Optical Character Recognition
RNN	Recurrent Neural Network
ROI	Region of Interest
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
SVM	Support Vector Machines
WE	Word Embedding
WER	Word Error Rate
XML	Extensible Markup Language

Author's Publications

Journal Publications

[1] J. Martinek, L. Lenc and P. Kral, **Building an Efficient OCR System for Historical Documents with Little Training Data**, in *Neural Computing and Applications*, **IF: 4.66**, Received: 25 December 2019, Accepted: 6 April 2020, Online: 9 May 2020, Special Issue: Emerging Applications of Deep Learning and Spiking Ann, pp. 1-19, Springer, ISSN: 0941-0643, doi: 10.1007/s00521-020-04910-x

[2] L. Lenc, J. Martinek, P. Kral, A. Nicolao and V. Christlein, **HDPA: Historical Document Processing and Analysis Framework**, in *Evolving Systems*, **IF: 2.07**, Received: 20 December 2019, Accepted: 23 April 2020, Online: 20 May 2020, pp. 1-14, Springer, ISSN: 1868-6478, doi: 10.1007/s12530-020-09343-4

[3] Maltoudoglou, L., Paisios, A., Lenc, L., Martinek, J., Kral, P., & Papadopoulos, H. (2022). **Well-calibrated confidence measures for multi-label text classification with a large number of labels** Pattern Recognition, **IF: 7.74**, 122, 108271

Conference Publications Indexed in Scopus

[1] J. Martinek, P. Kral and L. Lenc, **Dialogue Act Recognition Using Visual Information**, in 16th International Conference on Document Analysis and Recognition (ICDAR 2021), Lausanne, Switzerland, 5-10 September 2021, pp. 793-807, Springer, ISBN: 978-3-030-86330-2, doi: https://doi.org/10.1007/978-3-030-86331-9_51, **(rank "A" based on the CORE portal)**

[2] J. Martinek, C. Cerisara, P. Kral and L. Lenc, **Cross-Lingual Approaches for Task-Specific Dialogue Act Recognition**, in 17th International Conference on Artificial Intelligence Applications and Innovations (AIAI 2021), on-line, 25-27 June 2021, pp. 232-242, Springer, ISBN: 978-3-030-79149-0, doi: 10.1007/978-3-030-79150-6_19

[3] J. Martinek, L. Lenc, P. Kral, A. Nicolaou, V. Christlein, **Hybrid Training Data for Historical Text OCR**, in 15th International Conference on Document Analysis and Recognition

(ICDAR 2019), Sydney, Australia, 20-25 September 2019, pp. 565-570, IEEE, ISBN: 978-1-7281-2861-0, doi: 10.1109/ICDAR.2019.00096, **(rank "A" based on the CORE portal)**

[4] J. Martinek, P. Kral, L. Lenc, C. Cerisara, **Multi-lingual Dialogue Act Recognition with Deep Learning Methods**, in Interspeech 2019, Graz, Austria, *15-19 September 2019*, pp. 1463-1467, ISCA, ISSN: 2308-457X, doi: 10.21437/Interspeech.2019-1691 **(rank "A" based on the CORE portal)**

[5] L. Lenc, J. Martinek, P. Kral, **Tools for Semi-automatic Preparation of Training Data for OCR**, in 15th *International Conference on Artificial Intelligence Applications and Innovations* (AIAI 2019), Crete, Greece, 24-26 May 2019, pp. 351-361, Springer, e-ISBN: 978-3-030-19823-7, doi: 10.1007/978-3-030-19823-7_29

[6] J. Martinek, L. Lenc, P. Kral, **Training Strategies for OCR Systems for Historical Documents**, in *15th International Conference on Artificial Intelligence Applications and Innovations* (AIAI 2019), Crete, Greece, 24-26 May 2019, pp. 362–373, Springer, e-ISBN: 978-3-030-19823-7, doi: 10.1007/978-3-030-19823-7_30

[7] J. Martinek, L. Lenc and P. Kral, **Neural Networks for Multi-lingual Multi-label Document Classification**, in *27th International Conference on Artificial Neural Networks* (ICANN 2018), Rhodes, Greece, 4-7 October 2018, pp. 73-83, Springer, ISBN: 978-3-030-01417-9, doi: 10.1007/978-3-030-01418-6_8

[8] J. Martinek, L. Lenc and P. Kral, **Semantic Space Transformations for Cross-Lingual Document Classification**, in *27th International Conference on Artificial Neural Networks* (ICANN 2018), Rhodes, Greece, 4-7 October 2018, pp. 608-616, Springer, ISBN: 978-3-030-01417-9, doi: 10.1007/978-3-030-01418-6_60

Remaining Conference Papers

[1] P. Kral, K. Halla, R. Siroky, L. Lenc, J. Martinek, **Představení projektu: Moderní zpřístupnění historických pramenů**, in Data a znalosti & WIKT 2018, Brno, Czech Republic, 11-12 October 2018, pp. 173-176, ISBN: 978-80-214-5679-2

Bibliography

[1] R. Agerri and G. Rigau. Robust multilingual named entity recognition with shallow semi-supervised features. *Artificial Intelligence*, 238:63–82, 2016.

[2] J. Alexandersson, B. Buschbeck-Wolf, T. Fujinami, M. Kipp, S. Koch, E. Maier, N. Reithinger, B. Schmitz, and M. Siegel. *Dialogue acts in Verbmobil 2*. DFKI Saarbrücken, 1998.

[3] J. Ang, Y. Liu, and E. Shriberg. Automatic dialog act segmentation and classification in multiparty meetings. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 1, pages I–1061. IEEE, 2005.

[4] ATILF. Tcof : Traitement de corpus oraux en français, 2018. ORTOLANG (Open Resources and TOols for LANGuage) –www.ortolang.fr.

[5] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[6] J.-M. Benedı, E. Lleida, A. Varona, M.-J. Castro, I. Galiano, R. Justo, I. López, and A. Miguel. Design and acquisition of a telephone spontaneous speech dialogue corpus in spanish: Dihana. In *Fifth International Conference on Language Resources and Evaluation (LREC)*, pages 1636–1639, 2006.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[8] A. Björkelund, L. Hafdell, and P. Nugues. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48. Association for Computational Linguistics, 2009.

[9] T. M. Breuel. The ocropus open source ocr system. In *Document Recognition and Retrieval XV*, volume 6815, page 68150F. International Society for Optics and Photonics, 2008.

[10] T. M. Breuel. High performance text recognition using a hybrid convolutional-lstm implementation. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 11–16. IEEE, 2017.

[11] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait. High-performance ocr for printed english and fraktur using lstm networks. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 683–687. IEEE, 2013.

[12] T. Brychcín. Linear transformations for cross-lingual semantic textual similarity. *arXiv preprint arXiv:1807.04172*, 2018.

[13] H. Bunt. Context and dialogue control. *Think Quarterly*, 3(1):19–31, 1994.

[14] H. Bunt, V. Petukhova, A. Malchanau, K. Wijnhoven, and A. Fang. The dialogbank. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3151–3158, 2016.

[15] C. Cerisara, S. Jafaritazehjani, A. Oluokun, and H. Le. Multi-task dialog act and sentiment recognition on mastodon. *arXiv preprint arXiv:1807.05013*, 2018.

[16] C. Cerisara, P. Král, and L. Lenc. On the effects of using word2vec representations in neural networks for dialogue act recognition. *Computer Speech and Language*, 47:175–193, 2018.

[17] C. Cerisara, P. Kral, and L. Lenc. On the effects of using word2vec representations in neural networks for dialogue act recognition. *Computer Speech & Language*, 47:175–193, 2018.

[18] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[19] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[20] C. Clausner, S. Pletschacher, and A. Antonacopoulos. Efficient ocr training data generation with aletheia. *Proceedings of the International Association for Pattern Recognition (IAPR), Tours, France*, pages 7–10, 2014.

[21] P. Colombo, E. Chapuis, M. Manica, E. Vignon, G. Varni, and C. Clavel. Guiding attention in sequence-to-sequence models for dialogue act prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7594–7601, 2020.

[22] Z. Dai, J. Fu, Q. Zhu, H. Cui, Y. Qi, et al. Local contextual attention with hierarchical structure for dialogue act recognition. *arXiv preprint arXiv:2003.06044*, 2020.

[23] N. Das, B. Das, R. Sarkar, S. Basu, M. Kundu, and M. Nasipuri. Handwritten bangla basic and compound character recognition using mlp and svm classifier. *arXiv preprint arXiv:1002.4040*, 2010.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[25] N. Duran and S. Battle. Probabilistic word association for dialogue act classification with recurrent neural networks. In *International Conference on Engineering Applications of Neural Networks*, pages 229–239. Springer, 2018.

[26] K. Elagouni, C. Garcia, F. Mamalet, and P. Sébillot. Text recognition in videos using a recurrent connectionist approach. In *International Conference on Artificial Neural Networks*, pages 172–179. Springer, 2012.

[27] J. Frankel and S. King. Asr-articulatory speech recognition. In *Seventh European Conference on Speech Communication and Technology*, 2001.

[28] S. Gaur, S. Sonkar, and P. P. Roy. Generation of synthetic training data for handwritten indic script recognition. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 491–495. IEEE, 2015.

[29] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[30] J. J. Godfrey, E. C. Holliman, and J. McDaniel. Switchboard: Telephone speech corpus for research and development. In *Proceedings of the 1992 IEEE International Conference on Acoustics, Speech and Signal Processing - Volume 1*, ICASSP'92, page 517–520, USA, 1992. IEEE Computer Society.

[31] R. C. Gonzalez, R. E. Woods, et al. Digital image processing, 2002.

[32] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[33] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[34] T. Grüning, G. Leifert, T. Strauß, J. Michael, and R. Labahn. A two-stage method for text line detection in historical documents. *International Journal on Document Analysis and Recognition (IJDAR)*, 22(3):285–302, 2019.

[35] L. Han and M. R. Kamdar. Mri to mgmt: predicting methylation status in glioblastoma patients using convolutional recurrent neural networks. 2017.

[36] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[38] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018.

[39] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.

[40] S. Jekat, A. Klein, E. Maier, I. Maleck, M. Mast, and J. J. Quantz. Dialogue acts in verbmobil. 1995.

[41] Y. Ji, G. Haffari, and J. Eisenstein. A latent variable recurrent neural network for discourse relation language models. *arXiv preprint arXiv:1603.01913*, 2016.

[42] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov. Fasttext.zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[43] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[44] L. Kang, P. Riba, M. Rusiñol, A. Fornés, and M. Villegas. Pay attention to what you read: Non-recurrent handwritten text-line recognition. *arXiv preprint arXiv:2005.13044*, 2020.

[45] S. Keizer, A. R., and A. Nijholt. Dialogue act recognition with Bayesian networks for Dutch dialogues. In *3rd ACL/SIGdial Workshop on Discourse and Dialogue*, pages 88–94, Philadelphia, USA, July 2002.

[46] H. Khanpour, N. Guntakandla, and R. Nielsen. Dialogue act classification in domain-independent conversations using a deep recurrent neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2012–2021, 2016.

[47] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[48] A. Klementiev, I. Titov, and B. Bhattarai. Inducing crosslingual distributed representations of words. *Proceedings of COLING 2012*, pages 1459–1474, 2012.

[49] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber. A clockwork rnn. *arXiv preprint arXiv:1402.3511*, 2014.

[50] P. Kral. *Automatic Recognition of Dialogue Acts.* Theses, Université Henri Poincaré - Nancy 1, Nov. 2007.

[51] P. Král and C. Cerisara. Automatic dialogue act recognition with syntactic features. *Language resources and evaluation*, 48(3):419–441, 2014.

[52] P. Král, C. Cerisara, and J. Klečková. Combination of classifiers for automatic recognition of dialog acts. In *Interspeech'2005*, pages 825–828, Lisboa, Portugal, September 2005. ISCA.

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

[54] Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, 27(11):41–46, 1989.

[55] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[56] J. Y. Lee and F. Dernoncourt. Sequential short-text classification with recurrent and convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 515–520. Association for Computational Linguistics, 2016.

[57] G. Leifert, T. Strauss, T. Grüning, and R. Labahn. Citlab argus for historical handwritten documents, 2016.

[58] L. Lenc and P. Král. Word embeddings for multi-label document classification. In *International Conference Recent Advances in Natural Language Processing (RANLP 2017)*, pages 431–437, Varna, Bulgaria, 4-6 September 2017. INCOMA Ltd.

[59] L. Lenc, J. Martínek, P. Král, A. Nicolao, and V. Christlein. HDPA: Historical document processing and analysis framework. *Evolving Systems*, pages 1–14, 2020. Received: 20 December 2019, Accepted: 23 April 2020, Published: 20 May 2020.

[60] V. Levenshtein. Binary codes capable of correcting spurious insertions and deletions of ones. *Russian Problemy Peredachi Informatsii*, 1:12–25, 1965.

[61] M. Li, T. Lv, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei. Trocr: Transformer-based optical character recognition with pre-trained models. *arXiv preprint arXiv:2109.10282*, 2021.

[62] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[63] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers*, 28(2):203–208, 1996.

[64] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 1 edition, July 2008.

[65] V. Margner and M. Pechwitz. Synthetic data for arabic ocr system development. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 1159–1163. IEEE, 2001.

[66] J. Martínek, P. Král, and L. Lenc. Dialogue act recognition using visual information. In *International Conference on Document Analysis and Recognition*, pages 793–807. Springer, 2021.

[67] J. Martínek, P. Král, L. Lenc, and C. Cerisara. Multi-lingual dialogue act recognition with deep learning methods. *arXiv preprint arXiv:1904.05606*, 2019.

[68] J. Martínek, L. Lenc, and P. Král. Semantic space transformations for cross-lingual document classification. In *27th International Conference on Artificial Neural Networks (ICANN 2018)*, volume 11139 LNCS, pages 608–616, Rhodes, Greece, October 4-7 2018. Springer International Publishing.

[69] J. Martínek, L. Lenc, and P. Král. Training strategies for ocr systems for historical documents. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 362–373. Springer, 2019.

[70] J. Martínek, L. Lenc, and P. Král. Building an efficient OCR system for historical documents with little training data. *Neural Computing and Applications*, pages 1–19, 2020. Received: 25 December 2019, Accepted: 06 April 2020, Published: 09 May 2020.

[71] J. Martínek, L. Lenc, P. Král, A. Nicolaou, and V. Christlein. Hybrid training data for historical text ocr. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 565–570. IEEE, 2019.

[72] M. Mast, H. Niemann, E. Nöth, and E. G. Schukat-Talamazzini. Automatic classification of dialog acts with semantic classification trees and polygrams. In *International Joint Conference on Artificial Intelligence*, pages 217–229. Springer, 1995.

[73] B. McCann, J. Bradbury, C. Xiong, and R. Socher. Learned in translation: Contextualized word vectors, 2017.

[74] S. Mezza, A. Cervone, E. Stepanov, G. Tortoreto, and G. Riccardi. Iso-standard domain-independent dialogue act tagging for conversational agents. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3539–3551, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics.

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

[76] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[77] 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*, pages 3111–3119, 2013.

[78] N. Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

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

[80] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

[81] M. Petrou, P. Bosgogianni, and P. Bosdogianni. *Image processing*. Wiley Online Library, 1999.

[82] S. Pletschacher and A. Antonacopoulos. The page (page analysis and ground-truth elements) format framework. In *2010 20th International Conference on Pattern Recognition*, pages 257–260. IEEE, 2010.

[83] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[84] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. In *preprint*, 2018.

[85] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[86] S. Rawls, H. Cao, S. Kumar, and P. Natarajan. Combining convolutional neural networks and lstms for segmentation-free ocr. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, volume 1, pages 155–160. IEEE, 2017.

[87] N. Reithinger and M. Klesen. Dialogue act classification using language models. In *Fifth European Conference on Speech Communication and Technology*, 1997.

[88] C. Reul, U. Springmann, C. Wick, and F. Puppe. Improving ocr accuracy on early printed books by utilizing cross fold training and voting. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 423–428. IEEE, 2018.

[89] E. Ribeiro, R. Ribeiro, and D. M. de Matos. A multilingual and multidomain study on dialog act recognition using character-level tokenization. *Information*, 10(3):94, 2019.

[90] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Miccai*, pages 234–241, 2015.

[91] M. Rotaru. Dialog Act Tagging using Memory-Based Learning. Technical report, University of Pittsburgh, Spring 2002. Term Project in. Dialog Systems.

[92] K. Samuel, S. Carberry, and K. Vijay-Shanker. Dialogue act tagging with transformation-based learning. *arXiv preprint cmp-lg/9806006*, 1998.

[93] E. F. Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.

[94] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000.

[95] A. Schmitt, S. Ultes, and W. Minker. A parameterized and annotated spoken dialog corpus of the cmu let's go bus information system. In *LREC*, pages 3369–3373, 2012.

[96] G. Shang, A. J.-P. Tixier, M. Vazirgiannis, and J.-P. Lorré. Speaker-change aware crf for dialogue act classification. *arXiv preprint arXiv:2004.02913*, 2020.

[97] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd international conference on world wide web*, pages 373–374, 2014.

[98] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2017.

[99] E. Shriberg, R. Dhillon, S. Bhagat, J. Ang, and H. Carvey. The icsi meeting recorder dialog act (mrda) corpus. Technical report, International Computer Science Inst Berkely CA, 2004.

[100] E. Shriberg *et al.* Can prosody aid the automatic classification of dialog acts in conversational speech? In *Language and Speech*, volume 41, pages 439–487, 1998.

[101] F. Simistira, A. Ul-Hassan, V. Papavassiliou, B. Gatos, V. Katsouros, and M. Liwicki. Recognition of historical greek polytonic scripts using lstm networks. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 766–770. IEEE, 2015.

[102] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[103] R. Smith. An overview of the tesseract ocr engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

[104] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. V. Ess-Dykema, and M. Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373, 2000.

[105] T. Strauss, M. Weidemann, J. Michael, G. Leifert, T. Grüning, and R. Labahn. System description of citlab's recognition & retrieval engine for icdar2017 competition on information extraction in historical handwritten records, 2018.

[106] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks, 2014.

[107] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[108] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[109] S. S. Tabrizi and N. Cavus. A hybrid knn-svm model for iranian license plate recognition. *Procedia Computer Science*, 102:588–594, 2016.

[110] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[111] D. Wang, Z. Li, D. Sheng, H.-T. Zheng, and Y. Shen. Balance the labels: Hierarchical label structured network for dialogue act recognition. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[112] J. Wang, L. Perez, et al. The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit*, 11, 2017.

[113] C. Wick and F. Puppe. Fully convolutional neural networks for page segmentation of historical document images. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 287–292. IEEE, 2018.

[114] C. Wick and C. Reul. One-model ensemble-learning for text recognition of historical printings. In *International Conference on Document Analysis and Recognition*, pages 385–399. Springer, 2021.

[115] C. Wick, C. Reul, and F. Puppe. Comparison of ocr accuracy on early printed books using the open source engines calamari and ocropus. *J. Lang. Technol. Comput. Linguistics*, 33(1):79–96, 2018.

[116] C. Wick, J. Zöllner, and T. Grüning. Transformer for handwritten text recognition using bidirectional post-decoding. In *International Conference on Document Analysis and Recognition*, pages 112–126. Springer, 2021.

[117] C.-S. Wu, S. Hoi, R. Socher, and C. Xiong. Tod-bert: Pre-trained natural language understanding for task-oriented dialogues. *arXiv preprint arXiv:2004.06871*, 2020.

[118] M. Wu, L. Wang, Y. Si, and J. Dang. Dialogue act recognition using branch architecture with attention mechanism for imbalanced data. In *2021 12th International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pages 1–5. IEEE, 2021.

[119] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

[120] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer. Depth-gated recurrent neural networks. *arXiv preprint arXiv:1508.03790*, 9, 2015.

[121] W.-t. Yih, X. He, and C. Meek. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648, 2014.

[122] Z. Yin and Y. Shen. On the dimensionality of word embedding. In *Advances in Neural Information Processing Systems*, pages 887–898, 2018.

[123] K. Zhou, A. Li, Z. Yin, and C. Zong. Casia-cassil: a chinese telephone conversation corpus in real scenarios with multi-leveled annotation. In *LREC*, 2010.

[124] Z.-H. Zhou. Ensemble learning. In *Machine Learning*, pages 181–210. Springer, 2021.

[125] M. Zimmermann and H. Bunke. Automatic segmentation of the iam off-line database for handwritten english text. In *Object recognition supported by user interaction for service robots*, volume 4, pages 35–39. IEEE, 2002.