Sentiment Classification and Medical Health Record Analysis using Convolutional Neural Networks

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Sentiment Classification and Medical Health Record Analysis using Convolutional Neural Networks

Master Thesis

Maurice Gonzenbach

May 24, 2016

Advisors: Dr. Martin Jaggi, Dr. Valeria De Luca
Department of Computer Science, ETH Zürich

Acknowledgements

I would like to thank the following people for supporting me during the course of this thesis:

Dr. Martin Jaggi and Dr. Valeria De Luca for suggesting this fascinating topic, providing constant advice and granting me leeway with regards to the implementation.

Prof. Dr. Thomas Hofmann and Prof. Dr. Gunnar Rätsch for offering me the opportunity to work in their labs and discussing ideas and opportunities.

Stephanie Hyland and Stephan Stark for preparing the medical datasets and giving counsel on how to make best use of it.

Dr. med. Roger Gonzenbach for reviewing the predictions of numerous medical phrases with regards to their plausibility and helping me understand their meaning.

Abstract

This thesis tackles two problems in the area of natural language processing (NLP) using convolutional neural networks (CNNs).
The first one is the prediction of the polarity for English language Twitter messages. Our approach is based on the shallow CNN framework in [31], extended to two convolutional layers. A corpus of 90M tweets is employed to generate unsupervised word-embeddings. The neural network is pre-trained on a set of 60M tweets, annotated automatically based on the emoticons they contain. Subsequently, the system is fine-tuned on hand labelled samples, obtained from the *SemEval* competition [28]. The impact of various hyper-parameters is analyzed and discussed. Our solution is part of the winning ensemble for task 4a of the *2016 SemEval* contest [24].
The second part of this work deals with predicting ICD-9 codes from the text of electronic health records. Here we developed two kinds of labels: *(i)* the most represented top-level parent categories of the codes (8-class problem) and *(ii)* a one-vs-all approach for a specific code (binary problem). The documents are scanned for specific sections, which serve as input. Slightly different neural network configurations are evaluated, while a support vector machine classifier based on a bag-of-words model is used as baseline. The performance is discussed with respect to the input sections, the classifiers and the labels. In order to verify that the network actually looks for meaningful patterns and does not base its predictions on spurious correlations, a qualitative evaluation of the sentences predicted with high certainty was performed by a clinician.

Contents

Chapter 1

Introduction

Inferring high-level abstractions and conclusions from text is a task performed by humans dozens of times a day, often even unconsciously, but an enormously challenging one for computers. Although written texts (should) follow some well-defined grammatical and syntactic rules, in machine learning terms these are highly unstructured. Making sense of natural language is a process which does not follow any strict path, but often requires extensive knowledge about all present entities and their respective relationships with each other. Even without the use of rhetorical devices such as sarcasm and irony, the complexity of interpreting language is thus staggering. This thesis tackles two problems in the area of natural language processing (NLP) with the help of convolutional neural networks, namely sentiment analysis in tweets and classification of medical health records.

1.1 Twitter Sentiment Classification

Sentiment analysis is a well-established task in NLP, with the goal of assessing the polarity (i.e. 'positive', 'negative', 'neutral') of a given document. Such analysis is commonly performed on Twitter messages, due to the broad availability of datasets and previous works, which offer a solid baseline for performance assessment. Tweets have the advantage of being limited to 140 characters and hence imposing constraints on the amount of information that can be conveyed in one message. Yet the main challenges of tweets are that they often exhibit bad grammar, worse spelling, sentence fragments, contextualization (being part of a conversation), abbreviations and the use of special language features such as hashtags. The polarity of tweets can only be determined unequivocally up to a certain degree and hence there remains a portion of subjectivity to the classification. Consider the following tweet which was part of one of the datasets:

> "WTO barred China on Thursday from imposing duties on certain U.S. steel exports, siding with U.S. Pres. Barack Obama."

Without a very strict definition of sentiment, this tweet could be interpreted by the reader as positive for various reasons, e.g. for his favoring of free markets, his geopolitical views, his business interests or his support of president Obama, to name just a few. Readers with another stance on the mentioned issues would probably strongly object any positivity in the statement. Even when being unbiased towards these subjects – should this be possible at all – the answer depends on what subject we refer to, which here could be China, the U.S., the steel industry, etc. The third option would be to label it as neutral, for which there is arguably the strongest case, as the tweet *itself* does not favor any side but rather reports facts in a way, very typical for news agencies. In our datasets, this specific tweet was labeled as 'negative'. It is also one of the samples, which the proposed algorithms, described in the following chapters, have often failed to classify. This example highlights the sense of ambiguity which also humans experience when tasked with estimating the polarity. As described in the *SemEval 2015* Sentiment Analysis problem description [28], the degree to which humans agree on such questions is between 65% and 85%.

1.2 Medical Health Record Analysis

If analyzed using NPL, electronic health records (EHRs) could offer many insights, which have not been exploited yet. NPL can help scan the vast amounts of notes and literature for hidden correlations, aid in clustering and categorizing these documents or build the fundaments of systems assisting doctors with diagnosis and suggesting treatment plans. Previous works have dealt with visualizing time series and correlations of events in order to reveal patterns unknown to physicians [26]. Other authors have clustered patients based on their clinical notes and highlighted correlations between these clusters and gene mutations in those patients' tumors [4]. In this work, we aim to predict the *International Classification of Diseases, Revision 9* (ICD-9) code(s) – or its (their) derivatives – from the raw text records. Examples of analyzed EHRs are initial consultation reports, treatment plans, lab or other diagnostic results and notes on follow-up meetings. These vary drastically in length and quality and exhibit a plethora of different abbreviations and structures, adding to the existing challenges of developing a global classifier. The patients' documents and codes stand in a many-to-many relationship, joined by a timestamp. However, as the average of codes per document is between 1 and 2, the problem was designed as *single label multiple class*, thus duplicating all training samples with more than one code. A convolutional neural network was employed for this task.

1.3 Approach

After cleaning and tokenizing the text a vocabulary is compiled, consisting of all unique words that occur 5 or more times. For all words in the vocabulary, amounting to over 0.5M for each of the two problems, embeddings are constructed through the *GloVe* software [25]. The details of these steps are described in Section 2.1.2 on page 8 for the Twitter sentiment prediction and in Section 2.2.2 on page 13 for the medical health record analysis. As architecture we suggest a convolutional neural network with one to two layers and intertwined max-pooling, similarly to *Severyn et al. 2015* [31]. Every convolutional layer is run through a nonlinear activation function with added bias. Ultimately, a fully connected soft-max classifier is optimized in terms of negative log likelihood. A dropout rate of 0.5 is applied to the output of the penultimate layer to avoid feature co-adaptation. Overtraining is counteracted by adding an L_2 regularization term to the objective function and halting training when no improvement on the validation set is achieved for a predefined number of epochs. During training the weights of the convolutional filters, biases, soft-max classifier and also the word-embeddings are updated though the *AdaDelta* [37] algorithm. This results in the order of 90M trainable parameters. The various components of the network and their mathematical definitions are described in Section 2.3 on page 22, while the concrete architectures employed for the Twitter sentiment and medical health record analysis are detailed in Section 2.1.3 on page 10 and 2.2.3 on page 18 respectively. The main contributions of this work are the suggestion of a 2-layer layout for polarity classification, yielding cutting-edge results and establishing a solid baseline for the prediction of ICD-9 codes.

1.4 Related Work

Neural networks have been widely adopted for various computer vision tasks, including digit recognition [20, 8, 7], object detection [21, 7] and facial recognition [6]. In recent years they have also become popular for a variety of NLP problems, starting with learning of word representations [2] and being extended to information retrieval [1, 32], generating language models [23], machine translation [12], speech recognition [16] and sentiment analysis [31, 35].

In particular, shallow convolutional neural networks (CNNs) have recently improved the state-of-the-art in text polarity classification by a significant increase in accuracy [19, 18, 31, 31, 17, 29]. F1 scores achieved range between 65 and 90, depending on the tested dataset. The main idea of these methods is to use supervised training on a network with convolutional filters, acting as a sliding window over the sequence of given words, followed by max-pooling. Such architectures share a lot of similarity with CNN methods used in computer vision, although the latter typically consist in many more layers than those in currently successful NLP applications. Deeper networks increase the representation power of the system, but also the number of parameters and hence require larger amounts of (labelled) training data. This is a major impediment that partly explains why the use of deep architectures in NLP applications has been very limited so far. An exception is the 6-layer CNN proposed by *Zhang et al. 2015* [39] for character-level text classification. More sophisticated alternatives are recursive neural networks, exhibited among others by *Dong et al. 2014* [10], although the best-performing approaches of the *SemEval Twitter sentiment analysis* contest of recent years mostly featured simpler architectures [24, 28].

Limitations of these kind of setups include their requirement of large amounts of training data and that many algorithms base their prediction on short (contiguous) phrases, but struggle when information between far-apart word groups has to be combined.

Typically, common word-embedding techniques are used to represent each input word [22, 25]. Word-embeddings map words to fixed-sized vectors [30], encoding semantical and syntactical closeness between them. These embeddings are created in an unsupervised manner from a large text corpus and serve as inputs to the networks. There has been considerable progress in recent years in the quality of vectors these methods produce and the dataset size they can employ (billions of words) [22]. State of the art algorithms mostly rely on neural networks [22] (predictive model) or factorization of the co-occurrence matrix [25] (count-based model) in order to generate these dense vectors. The idea of embeddings can also be extended to encode sentences or entire paragraphs. For this, techniques similar to the ones used for word-embeddings, like skip-gram models, have been demonstrated on very short phrases [22]. Other approaches made use of context-aware recursive

neural networks, working with sentences of a length up to 15 [34].

As CNNs for NLP exhibit up to millions of trainable parameters and the number of hand-annotated training samples is often limited, *Read 2005* [27] suggested labelling large quantities of tweets automatically based on the emoticons they contain, in order to enlarge the training set. This so called semi-supervised or distant-supervised learning technique has been an active research direction in machine learning and in particular NLP applications. It has been empirically shown that unsupervised pre-training can be beneficial for supervised machine learning tasks [11]. Distant pre-training can be seen as an improved approach compared to unsupervised pre-training of neutral networks. Here the main idea is to infer weak labels from data without manual labeling. This approach has also been used for text polarity classification where significantly larger training sets were generated from texts containing emoticons [15, 31]. These methods have shown that training a CNN on such larger datasets (often by one to three orders of magnitude), followed by additional classical supervised training on an a smaller set of manually annotated labels, yields improved performance. Nevertheless, there is no systematic method known for enabling transfer learning in this way, and so most applications of distant pre-training currently follow empirical rules.

Chapter 2

Materials and Methods

2.1 Twitter Sentiment Classification

This section tackles the problem of predicting the sentiment polarity of tweets. The reasoning for using tweets is the availability of extensive, standardized datasets, with which the results can be compared to other state of the art techniques. Our proposed method is based on convolutional neural networks, pre-trained with distant-labelled datasets and refined on samples with hand-annotated labels. Word-embeddings generated in an unsupervised manner are built beforehand.

Geared towards the task 'Sentiment Analysis in Twitter' (4a) of the *SemEval* workshop 2016 [24], our approach is a refinement of the method employed by last year's runner-up of the same challenge (see *Severyn et al. 2015* [31]).

2.1.1 Datasets

The hand-annotated datasets were obtained from the *SemEval* competition [24]. They include a little over 60k tweets, either labelled as 'positive', 'negative' or 'neutral', of which around 18k could be used for training purposes according to the rules of *SemEval*. The various datasets and their content are listed in Table 2.1 on the next page.

In addition, a large number of tweets (around 2B) were collected through the Twitter-API, referred to as '*Sheffield*' tweets. Subsets thereof are employed for unsupervised learning of word-embeddings (see Section 2.1.2 on page 9) and to pre-train the sentiment classification network with weak labels (see Section 2.1.3 on page 11).

Table 2.1: Tweet datasets by *SemEval* [24], with training (top) and test sets (bottom).

Dataset	Total	Positive.	Negative	Neutral
Train 2013 (Tweets)	8224	3058	1210	3956
Dev 2013 (Tweets)	1417	494	286	637
Train 2016 (Tweets)	5355	2749	762	1844
Dev 2016 (Tweets)	1269	568	214	487
DevTest 2016 (Tweets)	1779	883	276	620
Test: *Twitter2016*	20632	7059	3231	10342
Test: *Twitter2015*	2390	1038	365	987
Test: *Twitter2014*	1853	982	202	669
Test: *Twitter2013*	3813	1572	601	1640
Test: *SMS2013*	2093	492	394	1207
Test: *LiveJournal2014*	1142	427	304	411
Test: *Tw2014Sarcasm*	86	33	40	13

2.1.2 Preprocessing

Before feeding the tweets to the neural network, a few steps are required which are crucial to the performance of the system:

Tokenization

Splitting a sentence into meaningful words seems a simple task for humans. However, it can be quite challenging for a machine learning system. It involves a number of design-decisions on how to treat hyphenations, punctuation, contractions, emoticons and names. For tweets this step is further complicated by the remarkably bad orthography demonstrated by many users. An often cited example which a naïve tokenizer fails to split is 'New York-based', which should correctly be divided between 'New York' and 'based'. The most solid option would probably be to use a dictionary as primary decision resource, but for performance and simplicity reasons a heuristic approach is used in this system. It is based on the *CMU ARK Twitter Part-of-Speech Tagger* [14] and further modified to suit this work. In the following, examples of design-decisions are listed:

- Keeping emoticons together and replacing them with special tokens:

  `:-)` ⇒ `<smile>`

  `:-(((` ⇒ `<sadface>`

- Splitting at ampersands and slashes:

  `Jack&Jones` ⇒ `jack & jones`

- Keeping contractions together:

  `aren't` ⇒ `aren't`

- Converting all characters to lowercase

  `HELLO Katy` ⇒ `hello katy`

- Removing more than two repetitions of the same character:

  `hellloooo` ⇒ `hello`

- Converting numbers to a special token:

  `won 9:3` ⇒ `won <number>`

  `it's 12.15` ⇒ `it's <number>`

- Converting email addresses to a special token:

  `bob@mail.com` ⇒ `<email>`

- Converting URLs to a special token:

  `http://tinyurl.com/abc` ⇒ `<url>`

Learning the embeddings

We constructed a corpus composed of all *SemEval* and a subset of 90M *Sheffield* tweets. Of the latter 30M contained positive emoticons, 30M contained negative ones and 30M contained none at all. This corpus, with a total of 1.3B tokes, was employed to create the word-embedding vectors of $d_w = 100$ using the *GloVe* software, developed at Stanford [25]. The algorithm applies a sliding window of size 15 to identify words which stand in a relation to each other and then factorizes the resulting shuffled co-occurrence matrix. Words which occur less than 5 times in the corpus are discarded, which results in a vocabulary $\mathbf{V}$ of 914354 unique words. The vectors for two special words are manually added:

- An *unknown-word* vector, assigned to all words not present in the vocabulary $\mathbf{V}$.
- A *zero-word* vector filled with zeros, used to pad the word matrix (see next section).

Stacking together the embedding vectors of all the words in the vocabulary $\mathbf{V}$ yields the *word-embeddings* matrix $\mathbf{E_w} \in \mathbb{R}^{|\mathbf{V}| \times d_w}$.

Furthermore, a *flag-embeddings* matrix $\mathbf{E_f} \in \mathbb{R}^{2^{d_f} \times d_f}$ with $d_f = 4$ is created, each row being a vector $\in \{0,1\}^{d_f}$. These binary vectors are appended to the embedding vectors when fed to the neural network (see Section 2.3.1 on page 22). The respective flags mark the following kinds of tokens :

- Words that belong to a hashtag
- Words that have been elongated (e.g. `hellllooo`, which is mapped to the same embeddings vector as the word `hello`)
- Words in which all characters are capitalized
- Punctuations which are repeated more than three times (e.g. `!!!!!`, which is mapped to the same embeddings vector as `!!!`)

Converting words to indexes

To speed up learning, the tweets which are to be fed to the neural network are encoded as vectors of integers. Every vector entry is the index to a unique word in $\mathbf{V}$. Stacking these vectors yields a *word-index* matrix, where each row represents one tweet. By feeding the neural network a matrix or parts of it instead of single vectors, the training phase can be drastically sped up. As a consequence, however, all sentences have to be represented by the same number of words n, which is chosen to be 37, given that only $\approx 0.15\%$ of all tweets contain more words. Vectors of sentences which contain *more* than 37 words are truncated, while those which contained *less* are right-padded with the abovementioned *zero-word*.

Similarly to the word-index matrix a *flag-index* matrix is created, holding indexes and referencing to the *flag-embeddings* matrix.

2.1.3 Neural Network Configuration

We train a convolutional neural network to predict the sentiments of tweets. Our network consists of two convolutional layers, each followed by a nonlinearity and a max-pooling layer. A soft-max regression is fully connected to the ultimate pooling layer, classifying every tweet as either 'positive', 'negative' or 'neutral'. A detailed mathematical description of the aforementioned network components is given in Section 2.3 on page 22.

The architecture, shown in Figure 2.1 on the facing page, is very similar to the one used by Severyn et al. 2015 [31], with the major difference being that we suggest two convolutional layers.

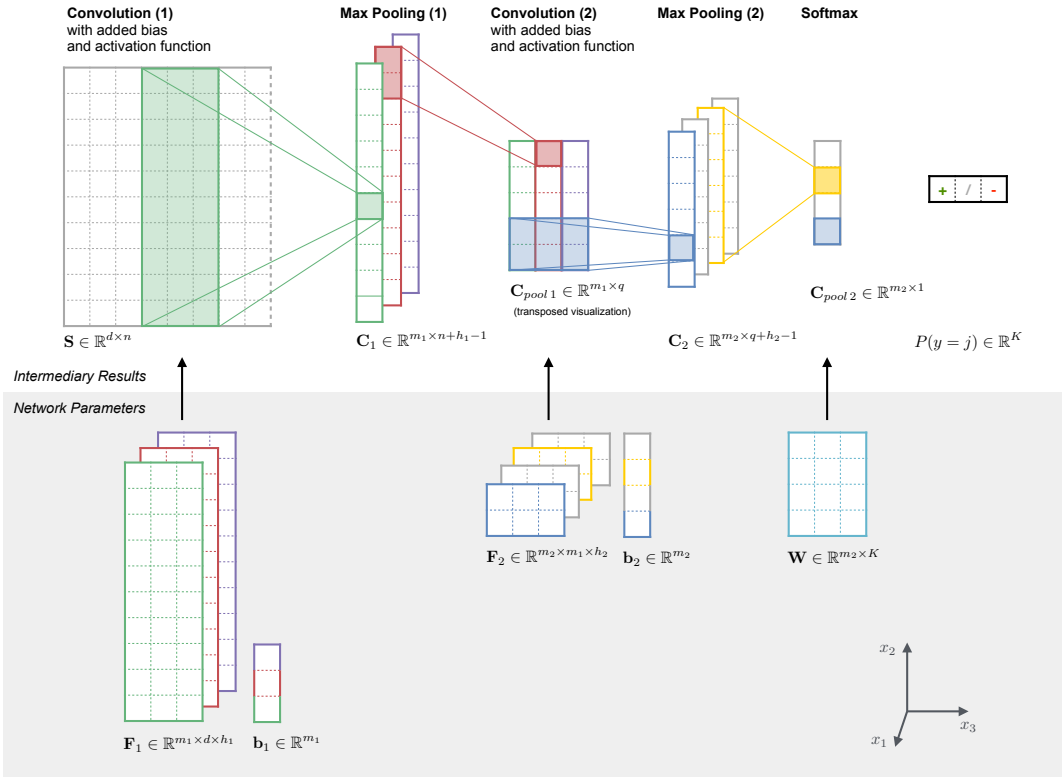

Figure 2.1: Architecture of neural network for tweet classification. The mathematical description of the components is detailed in Section 2.3 on page 22.

Training

During training the parameters $\Theta = \{\mathbf{E}_w, \mathbf{F}_1, \mathbf{b}_1, \mathbf{F}_2, \mathbf{b}_2, \mathbf{W}, \mathbf{a}\}$ are updated, where:

$\mathbf{E_w} \in \mathbb{R}^{|\mathbf{V}| \times d_w}$: The word-embeddings matrix, where each row contains the d_w-dimensional embedding vector for a specific word. Note that the flag embeddings matrix $\mathbf{E_f}$ is *not* trained.

$\mathbf{F}_i \in \mathbb{R}^{m_i \times \{d, m_1\} \times h_i}$: The weights for the m_i, $i \in \{1, 2\}$ convolutional filters of the first ($\mathbf{F}_1$) and second ($\mathbf{F}_2$) filter bank.

$\mathbf{b}_i \in \mathbb{R}^{m_i}$: The bias specific to every filter in the first ($\mathbf{b}_1$) and second ($\mathbf{b}_2$) filter bank.

$\mathbf{W} \in \mathbb{R}^{m_i \times K}$: The concatenation of the weights $\mathbf{w}_j$ for every output class in the soft-max layer.

$\mathbf{a} \in \mathbb{R}^K$: The biases of the soft-max layer.

As the number of hand-labelled tweets is relatively small (around 20k) compared to the number of trainable parameters (in the order of 90M, see Ta-

ble 3.3 on page 33), the neural network is initialized with a two-step approach, namely a distant supervised and a supervised step, as described in the following paragraphs.

Distant Supervised Training During this first phase, training sets are automatically generated form the *Sheffield* tweets. The raw corpus is filtered for tweets containing positive and negative emoticons, such as `:)` and `:(`, which are then used as weak labels, as already suggested by [15].

After random initialization of the parameters (except for the embedding matrices $E_{\{w,f\}}$), 30M tweets containing positive and 30M tweets containing negative emoticons are used to train the network for one epoch.

Supervised Training Using parameter values resulting from the distant supervised phase as starting point, the network is fine-tuned with the hand-labelled *SemEval* training sets. Parameters that were not present in the distant supervised phase, in particular the soft-max weights for the 'neutral' class, were initialized at random.

Score Metric

The main metric used for evaluation of the system is the averaged F1 score, whereby only the positive and negative class are considered: $\mathrm{F1}_{overall} = 0.5\,(\,\mathrm{F1}_{positive} + \mathrm{F1}_{negative}\,)$, where $\mathrm{F1} = 2\,\frac{\mathrm{precision}*\mathrm{recall}}{\mathrm{precision}+\mathrm{recall}}$. In addition also the accuracy is monitored.

Stopping Criteria

To prevent overfitting, the classification performance of the system is evaluated multiple times per epoch on a validation set, being independent from the training set. If a new best score is reached, the corresponding parameters are saved. Training is halted when no improvement was achieved for t number of epochs. For the distant supervised phase $0 \leq t \leq 2$, while for the supervised phase $5 \leq t \leq 15$.

2.2 Medical Health Record Analysis

This section discusses approaches of predicting ICD-9 (International Classification of Diseases) codes from medical health records. While the ICD-9 codes are highly standardized, hierarchical and well structured, the medical notes generally do not follow a strict form. Rather, they often consist of sentence fragments, contain many abbreviations and differ vastly in quality. Two prediction tasks are attempted on these health records:

Top-level categories: Instead of predicting the exact ICD-9 codes, for which there are over 14k options, their top-level parent categories are used. Of the latter only 20 distinct classes exist, further reduced to 8 in this work, which makes this task significantly more feasible for current machine learning techniques.

One-vs-all: This approach creates a data subset containing notes with a relatively common, specific ICD-9 code (*272.0*) on the one hand, and random notes with differing ICD-9 codes on the other. Classifiers trained on this binary dataset are thus able to decide if that condition can be inferred from a note or not.

2.2.1 Datasets

The raw datasets were obtained from the *RatschLab* at the *Memorial Sloan Kettering Cancer Center* (MSKCC), New York. The data is fully anonymized, meaning that names of patients and medical personnel and dates were previously removed, along with any other information which could disclose the identity of individuals mentioned in the documents. The dataset consists of three parts:

Clinical notes: The main corpus contains the patient notes, each identified by a unique *document_id*. The 2.14M documents consist of over 8.3B words but vary drastically in their individual length, ranging from less than 5 to thousands of words. The documents are already tokenized and split into paragraphs and sentences based on heuristics.

Notes metadata: This table links every document by its *document_id* to *patient_id* and states the time it was created.

ICD-9 metadata: The ICD-9 codes, mapping diseases to hierarchical codes, will later be used as labels for the documents. This table, however, only associates *codes* to *patient_ids* and the time this relationship was entered into the system. Over 7M relationships are defined in the table.

2.2.2 Preprocessing

Generating Labels

Taking the ICD-9 codes directly as categorical variables would not be practical, as over 9.4k unique codes are listed in the respective table, many of them with only one or very few occurrences (see Figure 2.2 on the following page for some statistical information on the codes' distribution). Therefore, the labels for the two cases introduced in the beginning of this chapter are generated as follows:

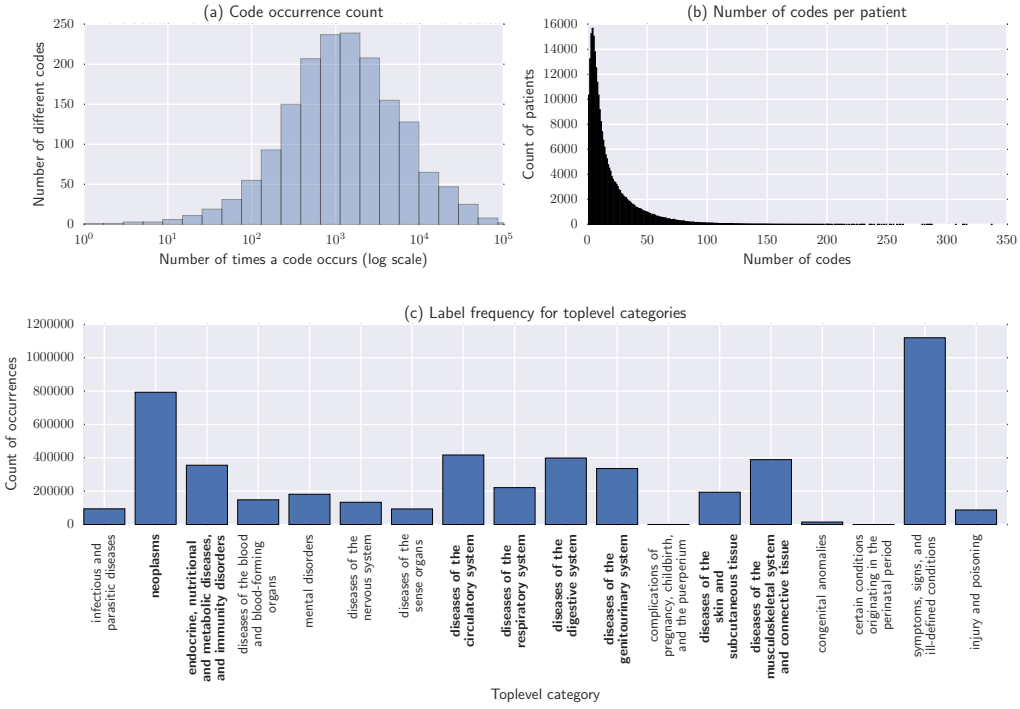

Figure 2.2: Statistical information of ICD-9 codes in the MSKCC dataset. Reading examples: (a) There a little over 50 codes that have around 100 occurrences in the dataset; (b) There are around 1000 patients that only have one ICD-9 code related to them; (c) A little over 80k ICD-9 codes map to the top-level category 'Neoplasms'. The categories printed in bold face are the ones kept for the prediction task.

Top-level categories: The specific codes are mapped to their top-level parent categories, leaving only 20 classes. Furthermore, categories containing miscellaneous codes and thus bearing little coherence (e.g. 'Supplementary Classification Of External Causes Of Injury And Poisoning' or 'Symptoms, signs, and ill-defined conditions') and such with only few samples are discarded, which results in the following 8 remaining categories:

1. Neoplasms
2. Endocrine, nutritional and metabolic diseases, and immunity disorders
3. Diseases of the circulatory system
4. Diseases of the respiratory system
5. Diseases of the digestive system
6. Diseases of the genitourinary system

7. Diseases of the skin and subcutaneous tissue
8. Diseases of the musculoskeletal system and connective tissue

One-vs-all: To find suitable target codes which can be predicted, the most common ICD-9 codes were analyzed by hand. It is probably no coincidence that the most common code (after discarding the 'supplementary classification' categories) is '401.9', translating to *Essential Hypertension, Unspecified*. This condition, however, was not considered an appropriate prediction candidate, due to it being relatively unspecific and a side effect of many more severe and concrete conditions. Applying these qualitative criteria regarding the specificity of codes the author's limited knowledge of the medical domain lead to the following selection: Code *272.0* or *Pure hypercholesterolemia*.

For this code a binary dataset is created, containing all notes that are linked to that code, called *target code*, and a similar number of notes that are related with other codes which are randomly selected.

Matching codes with notes

As both the clinical notes and the ICD-9 codes are identified with a unique patient, but not directly with each other, some heuristics are required to build such a relationship. For this purpose, the time codes are considered, which are given for both of these entities. An ICD-9 code is said to belong to a certain document, if its timestamp does dos not differ from the document's one by more than a certain amount of days. This results in a many-to-many relationship, where codes are identified with none, one or several documents. Analogously, notes can have none, one or multiple ICD-9 codes. The amount of matches depends on the allowed time shift, which is detailed in Figure 2.3 on the next page. For the subsequent steps the maximal valid time interval was set to 20 days, which leaves 655k notes attached to at least one code (see Table 2.2 on the following page).

Parsing Notes

As the notes vary greatly in length, structure and detail, preprocessing is required to generate suitable samples. Some steps, such as tokenziation, rough sentence splitting and replacing special tokens like dates with an according tag have already been done by *Stefan Stark*, from whom we received the data. After some analysis and manual inspection, the notes were scanned for paragraphs resembling the following content:

History: These sections contain information about the patients' general and medical history, medical history of their family and past information of the specific condition addressed in that note. Of all the notes, 90% matched at least one *history* search pattern.

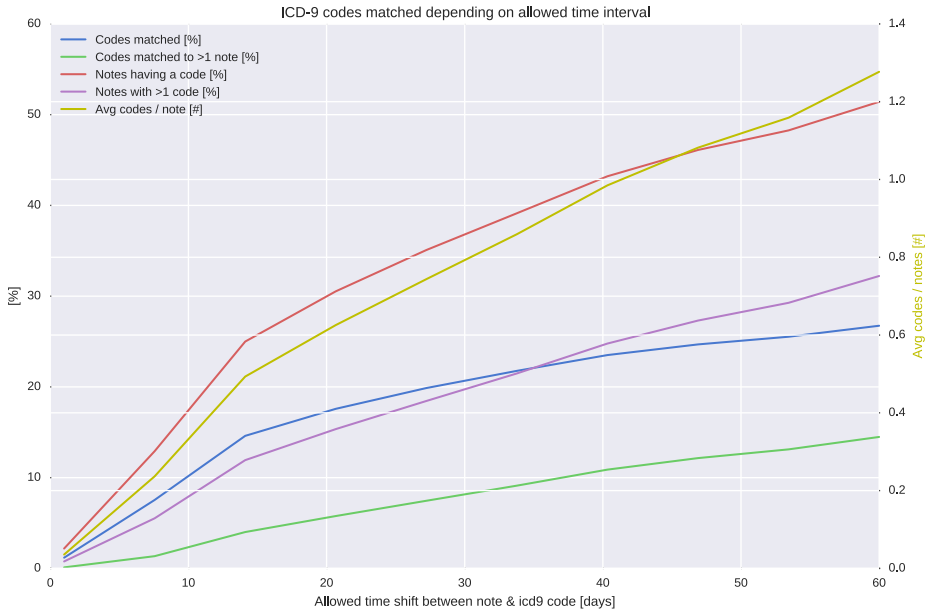

Figure 2.3: Matching ICD-9 codes to notes. Reading example: If the allowed difference between a note's and a code's timestamp is set to 20 days, a little less than 40% of all notes are assigned at least one ICD-9 code, while the average number of codes per note is around 1.0.

Plan: These sections describe the impression of the author regarding the patients' condition, examination results and their treatment plan. 86% contain some of the described information.

Only notes containing matches for *both* of these categories are considered in our study, which leaves 564k notes (see Table 2.2).

Table 2.2: Data samples after the preprocessing steps.

	Absolute	**Percentage of Original**
ICD-9 entries	7.1M	100%
ICD-9 in valid 8 top-level categories	3.1M	44%
Notes	2.1M	100%
Notes with at least one code	0.65M	31%
Notes with code & matching sections	0.56M	26%

Generating Datasets

Top-level categories: For these datasets the notes with more than one code attached are duplicated. In that way a note with four ICD-9 codes results in the same number of training samples with identical input but differing labels. One set is created where the *entire sections* are taken as one sample and another where each section is split into its sentences, creating a sample for *every sentence* (where all sentences from a section get the same label). For details on the dataset sizes see Table 2.3.

Table 2.3: Sample sizes of the medical datasets created for the top-level categories.

	Training set	Validation set	Test set
Entire sections	1.1M	5k	5k
Sentence-wise - section history	20.0M	83k	82k
Sentence-wise - section plan	13.2M	56k	56k
Sentence-wise - both sections	33.3M	139k	138k

One-vs-all: The creation of these datasets follows the same logic as for the top-level categories. As the target and non-target notes should have a similar number of occurrences, the resulting number of samples is much smaller than for the top-level categories. Details are listed in Table 2.4.

Table 2.4: Sample sizes of the medical datasets created for the one-vs-all scheme.

	Training set	Validation set	Test set
272.0			
Entire sections	52k	5k	5k
Sentence-wise - section history	1.0M	97k	98k
Sentence-wise - section plan	0.6M	62k	62k
Sentence-wise - both sections	1.6M	159k	160k

Learning the embeddings

To learn the word-embeddings the entire note corpus, containing 2.1M documents with 8.3B tokens, is employed. Using the same procedure as described in Section 2.1.2 on page 9, the embeddings are trained with *GloVe*, where words with less than 5 occurrences are discarded. This results in 50-dimensional embeddings for 440k unique words.

Converting words to indexes

Analogously to Section 2.1.2 on page 10, the words are mapped to integers for faster processing and concatenated into two matrixes (for each of the 'history' and 'plan' sections). This is done for all three datasets, where each row represents one training sample. As can be seen in Figure 2.4, the number of words per sample vary drastically. To avoid truncation on the vast majority of datasets but still keep the size of the matrix computationally feasible, the amount of words per sample n was set to 500 for the architecture taking entire sections as input, and to 50 for the one working on single sentences.

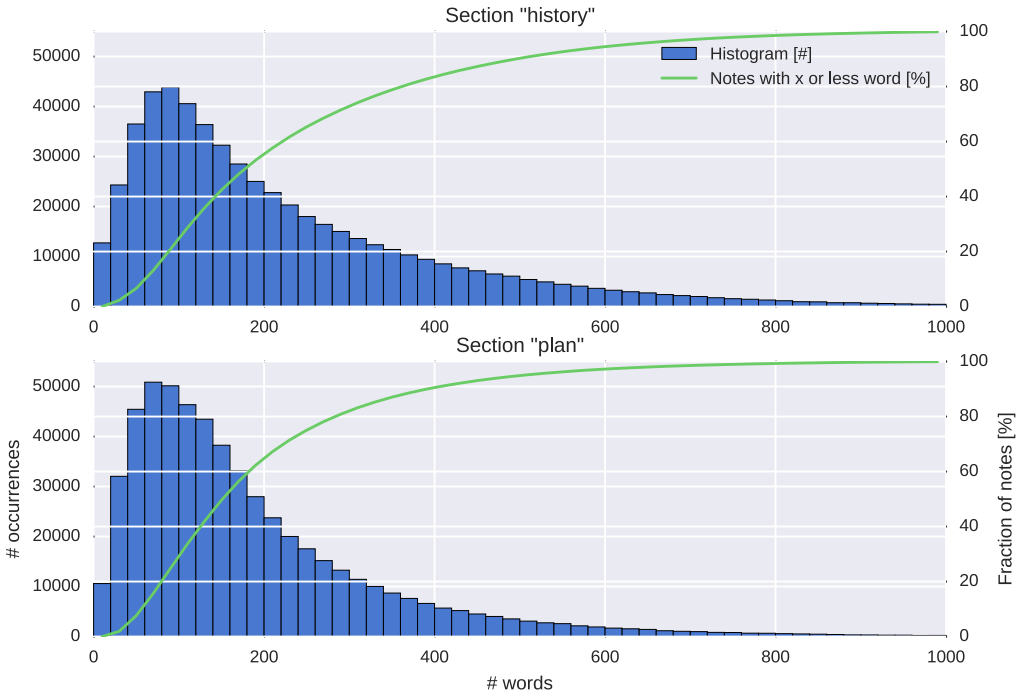

Figure 2.4: Histogram of words in medical notes [left y axis] and percentage of notes that have x or less words (i.e. notes that would not be truncated if n was set to that value) [right y axis].

2.2.3 Neural Network Classification

As described in Section 2.2.2 on page 15, two different parts are extracted from every note: The 'history' and 'plan' sections. Three slightly different setups were developed to suit this input, for each of which the network architecture was slightly adapted:

Section-wise classifier: The most straight-forward option is to have *one* input for the network, corresponding to *one* medical note. The input can either be the content from the 'history' or 'plan' section, or the concatenation of the two. By feeding the network only one section, we imply that the prediction is based on partial information and therefore, potentially reducing the accuracy. This can be avoided by using the concatenation of the two sections as input. However in this case, the samples contain up to thousands of words, raising the challenge for the network to distinguish between the few meaningful phrases and all the others.

The basic network architecture is very similar to the one used for tweet classification (see Figure 2.1 on page 11). However, the network suggested here only boasts one convolutional layer, followed by a nonlinearity and a softmax layer. This simplified design was chosen for performance reasons, as tests on the Twitter data showed that a second layer does improve accuracy, but not drastically. Furthermore, the flag-embeddings are omitted. During training the parameters $\Theta = \{\mathbf{E}_w, \mathbf{F}_1, \mathbf{b}_1, \mathbf{W}\}$ are updated. The layout is described in Figure 2.5.

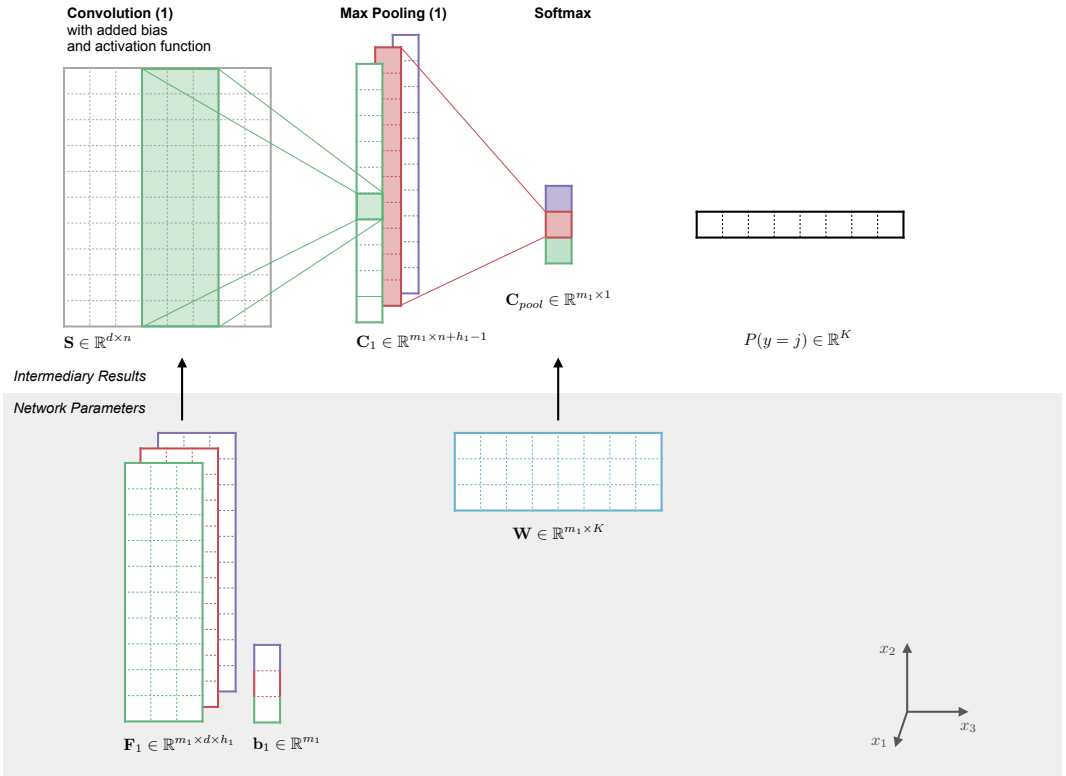

Figure 2.5: Architecture of a neural network for medical notes: The input can either be the 'plan' *or* 'history' section *or* the concatenation of the two. The mathematical description of the components is detailed in Section 2.3 on page 22.

Parallel section-wise classifier: To make better use of the varying language features and medical terms found in the two sections, the 'history' and 'plan' sections are inputted simultaneously but evaluated differently. This is achieved by feeding the two matrices to *separate* convolutional filter banks and then concatenating their results again: We start out with two sentence matrices $\mathbf{S}_{hist}$ and $\mathbf{S}_{plan}$, that are generated from their respective index matrices but with identical embeddings. These are then convoluted with separate filter matrices $\mathbf{F}_{hist}$ and $\mathbf{F}_{plan}$. After the max pooling step, the results are concatenated again and fed to one soft-max classifier. For details see Figure 2.6.

During training the parameters $\Theta = \{\mathbf{E}_w, \mathbf{F}_{hist}, \mathbf{b}_{hist}, \mathbf{F}_{plan}, \mathbf{b}_{plan}, \mathbf{W}\}$ are updated.

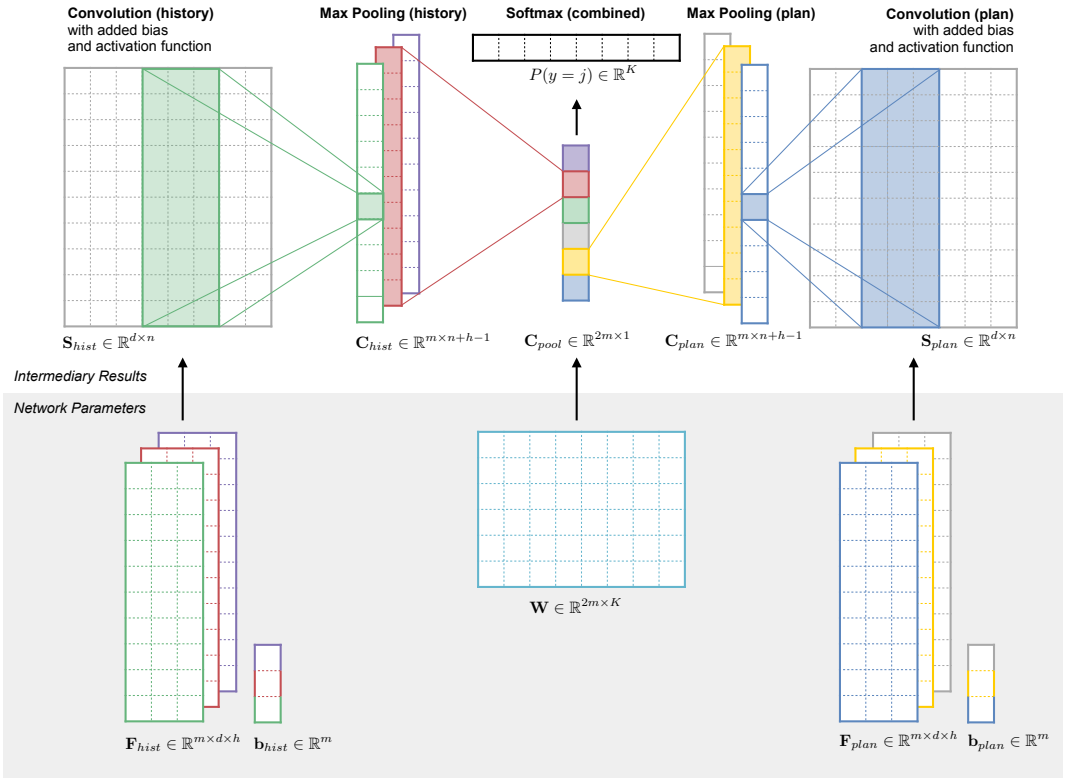

Figure 2.6: Architecture of a neural network for medical notes: Parallel input to simultaneously process 'history' and 'plan' sections using separate filter banks. The mathematical description of the components is detailed in Section 2.3 on page 22.

Sentence-wise classifier: The setups shown in the previous paragraphs treat every note as one sample, hence the amount of input words to the neural network can be between one and a few dozen sentences. Another

approach is to treat every sentence as individual sample, leading to tremendously shorter inputs (in the range of tweets). The neural network architecture is identical to the one described in Figure 2.5 on page 19. Here, after predicting the class probabilities, all results belonging to the same note are combined. The final answer is yielded by returning the class of the element with maximal probability.

Score Metric

The main metric used to evaluate the system is the averaged F1 score of all classes: $\mathrm{F1}_{overall} = \frac{1}{K} \sum_{i=1}^{K} \mathrm{F1}_i$. Samples annotated with multiple classes are counted as being correct if the prediction yields any one of these.

Stopping Criteria

As described in Section 2.1.3 on page 12, training is halted when no improvement was achieved on the validation set for t number of epochs. Depending on the classifier t is in the range $1 \leq t \leq 3$.

2.2.4 Support Vector Classification

To the best of our knowledge, no previous published work tackled the problem of classifying EHRs based on ICD-9 codes. Yet baseline comparison is necessary to assess the performance of the proposed neural network method. For that purpose a linear support vector machine is trained with a bag-of-words model.

Bag-of-words: A feature vector $x \in \mathbb{N}^{|\mathbf{V}|}$, with $\mathbf{V}$ being the vocabulary, represents the number of occurrences for each unique word. Specifically, x_i is the number of occurrences of the word i in the sample, where a sample is the concatenation of the 'history' and 'plan' sections. The order in which the words appear are not reflected in this feature vector, neither are words that are not present in the vocabulary. The vocabulary is identical to the one used for the neural network classification task, containing approximately 440k unique words (see Section 2.2.2 on page 17). As the vast majority of samples is composed by only a small fraction of the vocabulary $\mathbf{V}$, the feature vector is highly sparse.

Support vector classification: The bag-of-word vectors are fed to a linear optimizer, training K one-vs-rest support vector machines (SVM), with $K = 8$ being the number of classes. Given n training vectors x_i and their labels

$y_i \in \{-1, 1\}$ the SVMs minimize the following problem [1] [36]:

$$\min_{\mathbf{w},b,\zeta} \left[\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n} \zeta_i \right], \tag{2.1}$$

which is subject to the constraints $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \zeta_i$ and $\zeta_i \geq 0$ for all $i \in \{1, ..., n\}$. $C > 0$ is a constant penalty factor and ζ_i is a slack variable.

The decision function for the sample $\mathbf{x}_i$ is given by: $\text{sgn}(\mathbf{w}^T\mathbf{x}_i + b)$.

2.3 Neural Network Components

In the following sections, we will provide the details of the various components forming the neural network architectures shown in Figure 2.1, Figure 2.5 and Figure 2.6.

2.3.1 Embeddings Lookup

Inputs to the neural network are the word- and flag-*index* matrices (see Section 2.1.2 on page 10) and – if given – the word- and flag-*embeddings* matrices $\mathbf{E}_{\{\mathbf{w},\mathbf{f}\}}$ (see Section 2.1.2 on page 9). In a first step the elements of the index matrices are used as keys to look up the corresponding embedding vectors of size $d_{\{w,f\}}$ in the embeddings matrices. Concatenating the embeddings for all indexes in one row (representing one tweet) yields the sentence matrix $\mathbf{S} \in \mathbb{R}^{d \times n}$, with $d = d_w + d_f$.

2.3.2 Convolution Layer

The discrete convolution with a number number of filter matrices $\mathbf{F}_p \in \mathbb{R}^{d \times h}$ of width h is computed on the input sentence matrix $\mathbf{S}$, which corresponds to the intermediary matrix $\hat{\mathbf{C}}$ for higher-level layers. Let $\mathbf{S}_{[i:i+h]}$ denote the slice of matrix $\mathbf{S}$ from columns i to $i + h$. The zero-padded convolution operation then generates the feature $c_{p,i}$ by:

$$c_{p,i} := \sum_{k,j} \left(\mathbf{S}_{[i:i+h]}\right)_{k,j} \cdot \mathbf{F}_{p,k,j} \tag{2.2}$$

Concatenating all features $c_{p,i}$ for a sentence produces the feature vector $\mathbf{c}_p \in \mathbb{R}^{n+h-1}$. The filter can be seen as sliding over the columns of the input, performing an element-wise multiplication and summation on the current overlap, before moving one to the right. As the filter's dimension on the first axis is equal to the one of the sentence matrix, the resulting vector is

[1] The variable names employed in this paragraph are *not consistent* with the rest of the thesis

only one-dimensional. The operation can also be seen as a multiplication of the Fourier-transformed matrices **S** and **F**.

In order to recognize and react to different properties of the input, a whole set of filters is applied in parallel to the same data. Using m different filters results in a matrix $\mathbf{C} \in \mathbb{R}^{m \times n+h-1}$.

The trainable parameters of the convolution layer is a three-dimensional filter tensor $\mathbf{F} \in \mathbb{R}^{m \times d \times h}$.

2.3.3 Nonlinearity Layer

In order to make the network capable of recognizing nonlinear functions, a operation $\alpha(x)$ is applied element-wise to the convolution results $\mathbf{c}_p$. Popular choices for such a function, which were also tested in this work, are the hyperbolic tangent $tanh(x)$ and rectified linear '*ReLu*' function $max(x, 0)$. As both computational performance and classification accuracy were in favor of the *ReLu* activation, the majority of the tests were performed using said method. For enabling the layers to learn a suitable threshold, a bias b_p is added to the convolution vectors $\mathbf{c}_p$, summarized in the bias vector $\mathbf{b} \in \mathbb{R}^m$.

The nonlinearity layer thus performs the operation:

$$\hat{\mathbf{C}} = \begin{bmatrix} \hat{\mathbf{c}}_1 \\ ... \\ \hat{\mathbf{c}}_m \end{bmatrix} = \begin{bmatrix} \alpha(\mathbf{c}_1 + b_1 * \mathbf{e}) \\ ... \\ \alpha(\mathbf{c}_m + b_m * \mathbf{e}) \end{bmatrix}, \tag{2.3}$$

with $\hat{\mathbf{C}} \in \mathbb{R}^{m \times n+h-1}$ being the result matrix and $\mathbf{e} \in \mathbb{R}^{|\mathbf{c}_p|}$ the unit vector.

A dropout rate $0 \leq r \leq 1$ is applied to the output of the nonlinearity layer. A random set of result vectors $\hat{\mathbf{c}}_p$, reflecting a fraction r of the total number of result vectors m, is nullified. Thus if $r = 0$, there is no dropout effect. On the contrary, if $r = 1$ the output is all-zeros.

2.3.4 Pooling Layer

Pooling layers serve the purpose of reducing the dimensionality of the input and on the second level to obtain a fixed-length representation of the data. The row-wise (1D) pooling method used in this work is the maximum function on a fixed set of non-overlapping intervals with size s. It can be described using the following notation:

$$\hat{\mathbf{C}}_{pool} = \begin{bmatrix} max(\hat{\mathbf{c}}_{1[1:s]}) & ... & max(\hat{\mathbf{c}}_{1[s(q-1)+1:sq]}) \\ ... & & ... \\ max(\hat{\mathbf{c}}_{m[1:s]}) & ... & max(\hat{\mathbf{c}}_{m[s(q-1)+1:sq]}) \end{bmatrix}, \tag{2.4}$$

where $q = \left\lceil \frac{n+h-1}{s} \right\rceil$ is the number of resulting intervals given by the original vector length and the interval width. $\hat{\mathbf{c}}_{[a:b]}$ denotes the slice operator as introduced in Section 2.3.2 on page 22, returning the elements in the interval $[a, b]$ of the vector $\hat{\mathbf{c}}$.

In the special case of $s = n + h - 1$, with one interval spanning the entire results vector, the pooling layer acts as row-wise maximum function:

$$\hat{C}_{pool} = \begin{bmatrix} max(\hat{\mathbf{c}}_1) \\ ... \\ max(\hat{\mathbf{c}}_m) \end{bmatrix} \tag{2.5}$$

2.3.5 Soft-Max Regression

The outputs of the ultimate pooling layer $\mathbf{x} \in \mathbb{R}^m$ are fully connected to a soft-max regression layer which returns the class $\hat{y} \in [1, K]$ with the largest probability, i.e.:

$$\begin{aligned} \hat{y} &= \arg\max_j \; P(y = j | \mathbf{x}, \mathbf{w}, \mathbf{a}) \\ &= \arg\max_j \; \frac{e^{\mathbf{x}^\intercal \mathbf{w}_j + a_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\intercal \mathbf{w}_k + a_j}} , \end{aligned} \tag{2.6}$$

where $\mathbf{w}_j$ denotes the weights vector and a_j the bias of class j. The concatenation of the weights vectors $\mathbf{w}_j$ yields the weights matrix $\mathbf{W} \in \mathbb{R}^{m \times K}$.

2.3.6 Parameter Updates

In every iteration of the training process, a *batch* of samples is fed to the network in parallel. This significantly improves performance and also acts as a sort of regularization, thus reducing variability of the updates. The mean of the negative log likelihood was chosen as objective function f. For a batch of N datapoints $\mathbf{x}_i$, $i \in \{1...N\}$ with known labels y_i, f is defined as:

$$f(\theta) = NLL(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log P(Y = y_i | \theta, \mathbf{x}_i) , \tag{2.7}$$

with $\theta \in \Theta$ being the current value of one of the trainable parameters. An L_2 regularization can be added to the objective function to reduce overfitting. For a parameter θ that penalty term has the form $\lambda \|\theta\|_2^2$, with a parameter specific regularization strength λ.

Updates to the parameters Θ are computed through the *AdaDelta* update rule [37], which is a version of gradient descent automatically adapting the learning rate per-dimension using only first order information. *AdaDelta* is derived from *AdaGrad*, but differs from the latter in that it estimates the

step size in a local window, by accumulating an exponentially decaying average of the the squared gradients. It depends on two hyper-parameters: (i) the decay rate ρ and (ii) a constant ε added to the accumulated gradients, to "start off the first iteration and ensure progress [...] when the previous updates become very small."

The working principle of the update step is described in Algorithm 1, which is performed for every parameter $\theta \in \Theta$.

Algorithm 1 *AdaDelta* update step for $\theta \in \Theta$; from Zeiler et al. (2012) [37]

Require: decay rate ρ, constant ε
Require: initial parameter θ_1

Initialize accumulation variables $E[g^2]_0 = 0$, $E[\Delta\theta^2]_0 = 0$
for $t = 1 : T$ **do** # loop over all timesteps
  Compute gradient: $g_t = \frac{\partial f(\theta_t)}{\partial \theta_t}$
  Accumulate gradient: $E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2$
  Compute update: $\Delta\theta_t = -\frac{\text{RMS}[\Delta\theta]_{t-1}}{\text{RMS}[g]_t} g_t$, with $\text{RMS}[x] = \sqrt{E[x^2] + \varepsilon}$
  Accumulate updates: $E[\Delta\theta^2]_t = \rho E[\Delta\theta^2]_{t-1} + (1-\rho)\Delta\theta_t^2$
  Apply update: $\theta_{t+1} = \theta_t + \Delta\theta_t$
end for

2.4 Implementation

2.4.1 Computational Resources

The core routines were all written in `Python`, making heavy use of the mathematical expression compiler `Theano` [3], which compiles computationally intensive routines into `C` code when calling their `Python` wrappers for the first time. Furthermore it makes GPU utilization very straightforward while still ensuring portability and backward compatibility of the program. For further performance improvement the *CuDNN* library [5] was utilized, which contains optimized `CUDA` primitives for neural networks, developed by *Nvidia*.

Experiments were conducted on 'g2.2xlarge' instances of *Amazon Web Services* (AWS), which offer a *GRID K520* GPU with 3072 CUDA cores and 8 GB of GDDR5 RAM. To reduce costs, the prices of spot instances around the world were compared and the region selected accordingly. This resulted in about $1/8$ of the cost which would have been incurred using on demand instances, while still guaranteeing a very reliable service. At times over 30 instances were used in parallel for hyper-parameter tuning. The *StarClus-*

ter software developed at MIT was used to create a scalable cluster of AWS instances by making use of the *Sun Grid Engine* [13] batch-queueing system and a load-balancer that added or removed nodes from the cluster according to the number of waiting jobs.

Grid Searches on the hyper-parameters were initiated by a custom script, walking over the search-space of a predefined set of boolean, categorical and continuous variables exhaustively. Besides submitting the respective job, the script also writes every experiment to a *MongoDB* database on the master node. When being launched, during execution and after completion the worker nodes report progress and results to the database. This enables an efficient workflow for running a large number (up to several thousand) of experiments and for evaluating the results. Furthermore, a routine is included for handling exceptions, by automatically resubmitting failed experiments to the job queue.

Chapter 3

Experiments and Results

3.1 Twitter Sentiment Classification

3.1.1 Hyper-parameter Optimization

The classification and computational performance heavily depend on a number of hyper-parameters, some of which are strongly interdependent. Tuning these is very time-consuming due to the large number of experiments required and results can ultimately still be inconclusive, because of random correlations and not strictly deterministic implementation of some routines. In the following paragraphs some of the important hyper-parameters are discussed:

Number of convolutional filters: The number of filters $m_{\{1,2\}}$ used in the filter banks $F_{\{1,2\}}$ has a considerable impact on speed as well as accuracy. The reasoning for using more filters is to enable the system to distinguish between more kinds of patterns that convey information about the polarity of the input. However, more filters are only advantageous if they manage to capture *different* kinds of patterns, thus making a good (random) initialization and techniques like drop-out key enablers for larger m values. Results from experiments suggest a lower boundary of $m_1 \geq 100$, with a smaller number considerably hurting the classification performance, and an upper boundary of $m_1 \leq 300$, with a further increase not significantly improving classification results but weighing heavy in terms of computational costs. Similarly, a range of $50 \leq m_2 \leq 150$ was chosen.

Convolutional filter width: While the height of the convolutional filters was set to the embedding dimension d – as it would not make sense to convolute along the embedding vectors – their widths can be varied, according to the number of words (for $\mathbf{F_1}$) or higher-level features ($\mathbf{F_2}$) to be grouped together. As average sentences usually have 5-37 words and we

want each level of the neural network to be responsible for only a part of the information-aggregation, reasonable values of $h_1 = 6$ and $2 \leq h_2 \leq 6$.

Embeddings dimension: The dimension of the word-embeddings d_w determines how much different aspects of semantic and syntactic information each word vector can represent. In this work no gain was experienced when using more than 50 dimensions. Hence we finally set $d_w = 50$.

Max pooling intervals and overlap: The interval sizes $s_{\{1,2\}}$ of the max pooling layers determine the amount of downscaling and summarization incurred after the convolution layers. For the first of these layers a value of $s_1 = 3$ has shown to deliver good results. In order to yield a one-dimensional representation for the soft-max layer, s_2 was set to be equal to $|\mathbf{c_p}|$, i.e. the number of outputs per convolutional filter in the first layer. Experiments were also conducted using an overlay between the intervals. Yet, a potential minor gain in accuracy did not justify the drastic decrease in performance, due to implementation details of *CuDNN*.

Batch size: As the objective function f is an average over an entire batch of inputs, larger batch sizes tend to reduce variability of parameter values over time and the influence of random correlations. The discrepancy in oscillation can be observed by comparing Figure 3.2 on page 30, where $N = 2000$ and Figure 3.3 on page 31, where $N = 200$. On the other hand, it can also diminish the impact of samples with valuable information. Large batches are also in favor of computational performance, as the system can make better use of its highly parallel architecture. For noisy labels (in the distant supervised phase) a batch size of $1000 \leq N \leq 3000$ was chosen, while on the smaller hand-annotated sets $50 \leq N \leq 200$.

Dropout-rate: Applying a dropout-rate of $r > 0$ increases the number of training epochs needed for the system to reach a (local) maximum, and also the probability for convolutional filters to learn *different* kinds of patterns. In this work a dropout rate of $r = 0.5$ is used.

L2 regularization: To reduce overfitting and control the 'amount of learning' for each parameter $\theta \in \Theta$, a L_2 regularization was applied to the word-embeddings matrix $\mathbf{E}_w$, the convolutional filters $\mathbf{F}_i$ and the softmax weights $\mathbf{W}$. In Figure 3.1 the results for a set of experiments to determine the regularization strength are visualized. Trials have shown that the regularization strength should be $\lambda \leq 1e^{-3}$ for all mentioned parameters. Determining the exact value required a lot of fine-tuning, but proved to be crucial for achieving cutting-edge results. The λ values also differ depending on the

dataset: The distant phase with weak labels requires other settings than the supervised phase.

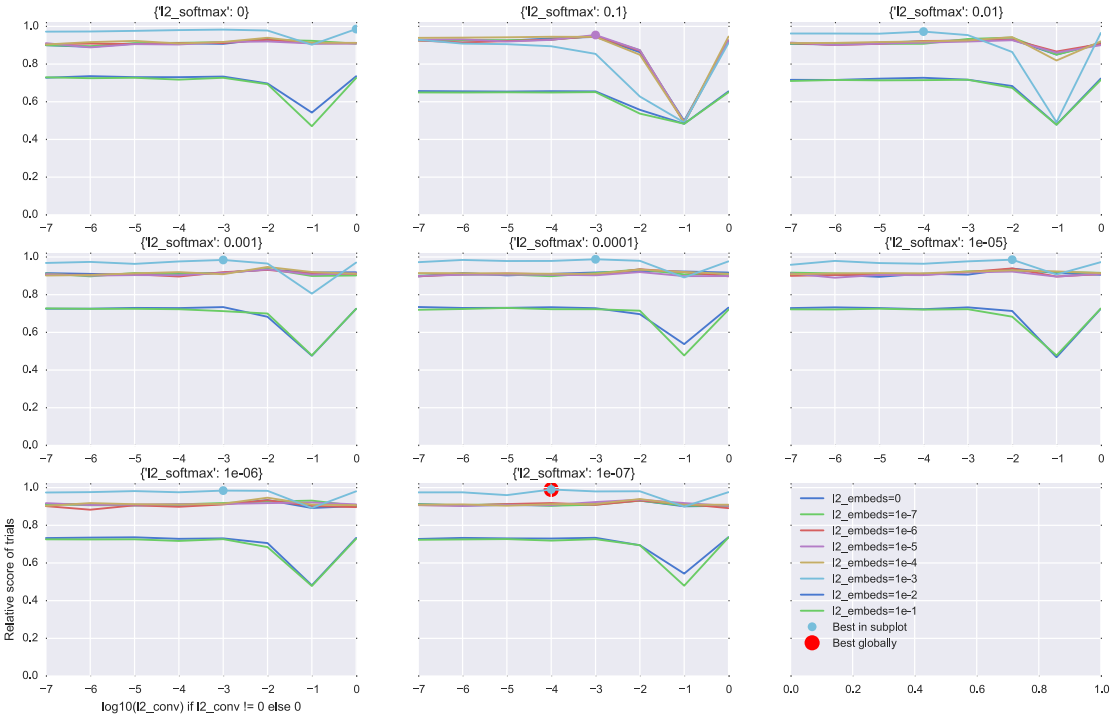

Figure 3.1: Relative classification performance for different L_2 regularization strength values.

Adadelta parameters: Adjusting the parameters ρ and ε for the *Adadelta* update steps alters the behaviour during training. In particular, higher values of ε increase the variability of the score over time. However, in terms of final classification performance the values of $\rho = 0.95$ and $\varepsilon = 1e^{-6}$ proposed by *Zeiler et al. 2012* [37] proved to be the most robust choice.

Nulling of zero-word: As described in Section 2.1.2 on page 9, sentences are padded with a *zero-word*, to enable representation of the input as a fixed-width matrix. However, during training, the value of the zero-word-embedding (originally set to be all zeroes) can change and thus a 'meaning' could be given to this word. To avoid this unwanted behaviour an option was tested to reset that embedding after every timestep.

Yet results indicate that this re-nulling is not advantageous to the classification performance, suggesting that the convolutional filters learn another embedding vector to represent 'meaninglessness'.

Balancing: As the datasets exhibit unbalanced distributions among the classes (see Table 2.1 on page 8), two methods were evaluated for counteracting this situation. One is to adjust the impact of misclassifications for every class in the objective function. This is done by multiplying the summands of the negative log likelihood (NLL) measure with a class-specific weight. The weight for class i with occurrence count N_i was calculated as $\frac{N}{N_i}$, with $N = \sum_{i=1}^{K} N_i$. Normalization is not required, as the NLL is averaged over the samples.
The other is to extend the training set by resampling from underrepresented classes, such that all classes reach the same occurrence count as the one with the largest N_i. Applied e.g. to the *Train 2013* set, this means that on average every sample labelled as *negative* is represented over three times.
Empirical results showed a slightly reduced classification performance when adjusting the class weights and no significant impact when using resampling techniques. Therefore, none of these methods was finally applied.

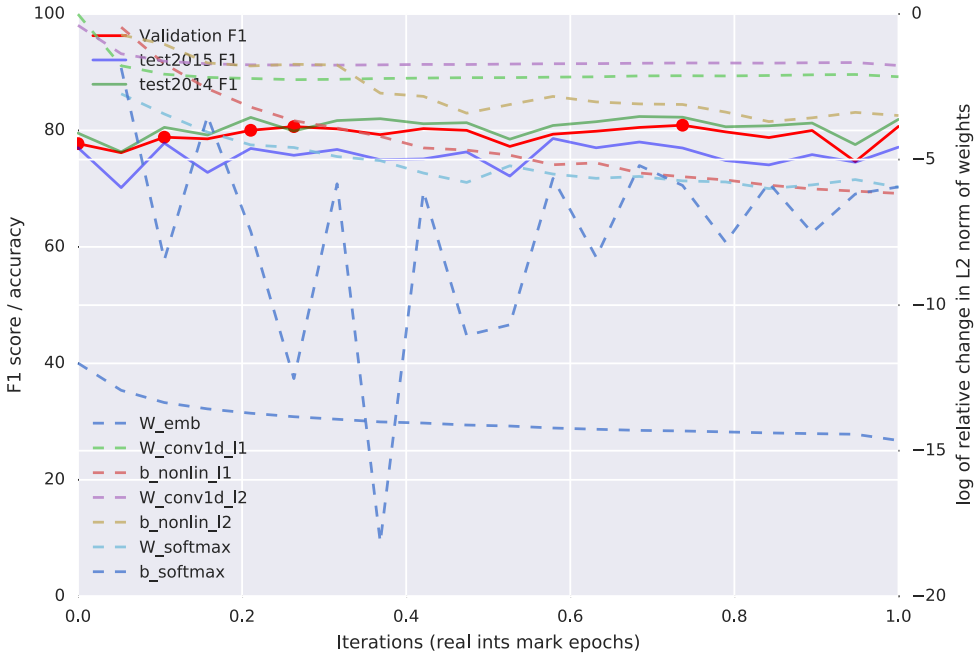

Figure 3.2: Evolution of scores [left y axis, solid lines] and relative change of parameters [right y axis, dashed lines] during training on Twitter data for distant supervised phase.

Shuffling: In the supervised phase multiple epochs of training are conducted. To avoid following similar paths in the parameter-space during these epochs, shuffling of the sequence of samples is necessary. In the adopted shuffling we permuted the order and composition of the mini-batches.

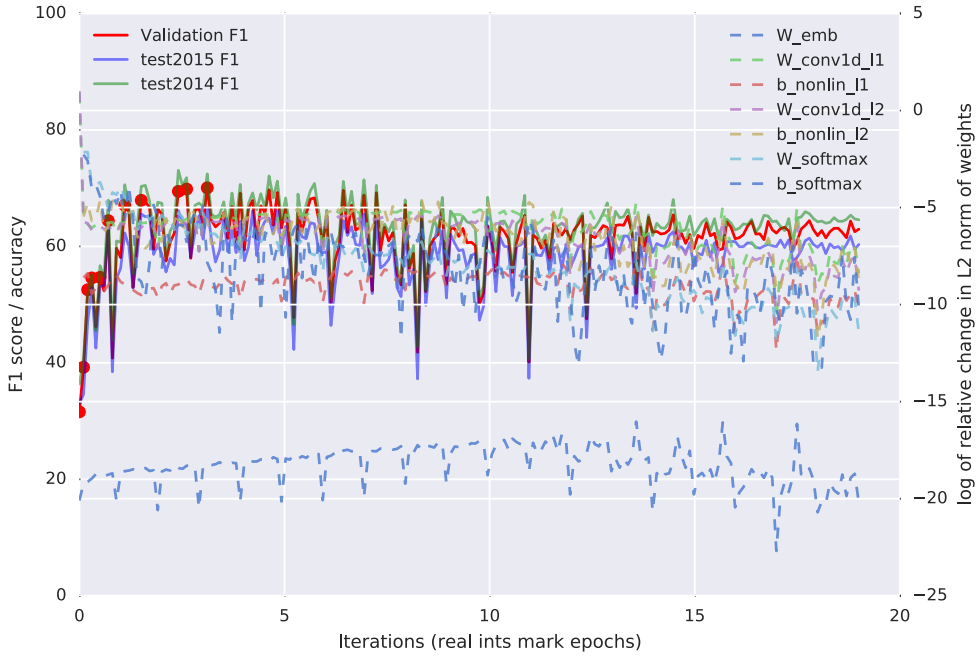

Figure 3.3: Evolution of scores [left y axis, solid lines] and relative change of parameters [right y axis, dashed lines] during training on Twitter data for supervised phase.

3.1.2 Classification Performance

Classification performance was optimized for task 4a of the *SemEval* contest 2016. Over 7k experiments were conducted for the supervised phase and over 100 for the distant supervised phase. The final chosen parameters are listed in Table 3.1 on the following page. On the distant supervised dataset the system was trained for one epoch (see Figure 3.2 on the preceding page) while on the supervised one it reached peak performance during epoch four (see Figure 3.3). As described in Section 2.1.3 on page 12, training was halted after $t = 15$ epochs without improvement on the validation set.

Table 3.1: Hyper-parameters used for *SemEval* competition. The network architecture is shown in Figure 2.1 on page 11.

	Supervised	**Distant supervised**
N (Batch size)	200	1000
m_1 (# Conv. filters L1)	300	300
m_2 (# Conv. filters L2)	100	100
h_1 (Conv. Filter width L1)	6	6
h_2 (Conv. Filter width L2)	4	4
s_1 (Max pooling window L1)	3	3
r (Dropout Rate)	0	0.3
λ_{conv} (L_2 strength conv.)	$1e^{-5}$	0
λ_{emb} (L_2 strength embed.)	$1e^{-4}$	$1e^{-4}$
λ_{soft} (L_2 strength softmax)	$1e^{-4}$	$1e^{-7}$
K (# softmax classes)	3 (pos., neg., neu.)	2 (pos., neg.)

With a final F1 score of 62.36 on the 2016 test set, the system proves to be on par with the most performant systems currently available. Had the results been submitted to the contest as such, they would have taken second place out of 34 industry and university participants, only surpassed by *SENSEI-LIF* (University of Marseille). Their approach is also based on CNNs but combines multiple embedding techniques. However, it was decided to create an ensemble of this system and one developed by Jan Deriu by employing a random forest on the class probabilities. Our team, *SwissCheese*, reached first place in the aforementioned competition. See Table 3.2 for detailed F1 scores on the various test sets [9].

Table 3.2: F1 score on *SemEval* task 4a test sets. SwissCheese: Ensemble to which this system contributed (winning team); SENSEI-LIF: Runner-up team. Best results are highlighted in bold face.

	This work	**SwissCheese**	**SENSEI-LIF**
Twitter2016	62.36	**63.30**	62.96
Twitter2015	66.63	**67.05**	66.16
Twitter2014	72.45	71.55	**74.43**
Twitter2013	70.05	70.01	**70.54**
LiveJournal2014	70.86	69.51	**74.07**
Tw2014Sarcasm	**62.74**	56.63	46.66

3.1.3 Computational Performance

The total number of trained parameters trained is 91.7M, of which over 99% are from the the word-embeddings matrix $\mathbf{E_w}$. However, in terms of computational intensity the convolutional layers are much more time-consuming, as the word-embeddings matrix merely serves as lookup-table. The detailed number of parameters is listed in Table 3.3.

Depending on the number of convolutional filters and their width, batch size, regularization, etc., the system is capable of processing between 500 and 5000 sentences per second during training. With the parameters listed in Table 3.1 on the facing page training one epoch in the distant supervised phase (60M sentences) takes a little over 4.5 hours, while one supervised epoch (18k sentences) is completed within 20 seconds.

Table 3.3: Sizes of parameters which are updated during training.

	Description	Tensor dim.	Trainable params.
$\mathbf{E_w}$	Word-embeddings	914354×100	91.4M
$\mathbf{F_1}$	Convolution 1	$300 \times 6 \times 104$	187k
$\mathbf{b_1}$	Bias 1	300×1	300
$\mathbf{F_2}$	Convolution 2	$100 \times 4 \times 300$	120k
$\mathbf{b_2}$	Bias 2	100×1	100
$\mathbf{W}$	Soft-max weights	100×3	300
$\mathbf{a}$	Soft-max bias	3×1	3
Total			91.7M

3.2 Medical Health Record Analysis

3.2.1 Top-level categories

This section describes the hyper-parameters and the resulting $F1$ scores for the system predicting the 8 top-level ICD-9 codes, as introduced in Section 2.2.2 on page 13. Experiments were conducted for the three kind of neural networks described in Section 2.2.3 on page 18, as well as for the support vector machine detailed in Section 2.2.4 on page 21. An overview of the results can be found in Figure 3.4 on page 35. The goal of this analysis was not to merely maximize the F1 scores, but to establish a fair comparison of different systems. Therefore, the hyper-parameters were not tuned independently for each system, but set equal for all systems. The values used in this work are listed in Table 3.4 on the next page.

Table 3.4: Hyper-parameters used for analysis of medical health records.

Parameter	Value
N (Batch size)	200
m_1 (# Conv. filters L1)	100
h_1 (Conv. Filter width L1)	4
r (Dropout Rate)	0.5
λ_{conv} (L_2 strength conv.)	0 or $1e^{-4}$
λ_{emb} (L_2 strength embed.)	0 or $1e^{-5}$
λ_{soft} (L_2 strength softmax)	0 or $1e^{-6}$
K (# softmax classes)	8

The **section-wise classifier** is remarkably robust, ranking very close to – if not better than – the more sophisticated and computationally expensive parallel network. It yields results in F1 scores between 51.3, when the 'plan' section is fed without applying regularization, and 55.8 (which is the highest score reached by any system), when using both sections and enforcing L_2 regularization. In the latter experiment (both sections & L_2 regularization), the by-category results are within the range of 45 and 60, thus exhibiting less variability between them than the other classifiers (see Figure 3.5 on page 36). In particular the worst score is still quite good and exceeds exceeds the equivalent result of SVM by 15.7 points.

The **parallel section-wise** network shows similar F1 scores as the section-wise approach: While outperforming the latter without L_2 regularization by 0.9 points (reaching 54.3) it falls behind by 0.9 points when a regularization is applied by (54.9). While quantitative results show no clear advantage in using either the parallel or the section-wise networks, the latter has the advantage of being computationally less expensive.

The **sentence-wise classifier** has the lowest scores compared to the other neural network designs and performs only marginally better than the support vector classifier: Its results range 45.5 and 48.3, with per-class scores in [28.3, 60.3].

The **support vector classifier** generally delivers the weakest results, with an averaged F1 score between 45.5 and 49.6. Examples of failure categories are 'Diseases of the respiratory system' and 'Endocrine, nutritional and metabolic diseases, and immunity disorders', where scores are below 30 and 33, respectively.

For all tested systems, it can be observed that the experiments using only the 'history' section consistently score higher than the ones relying only on

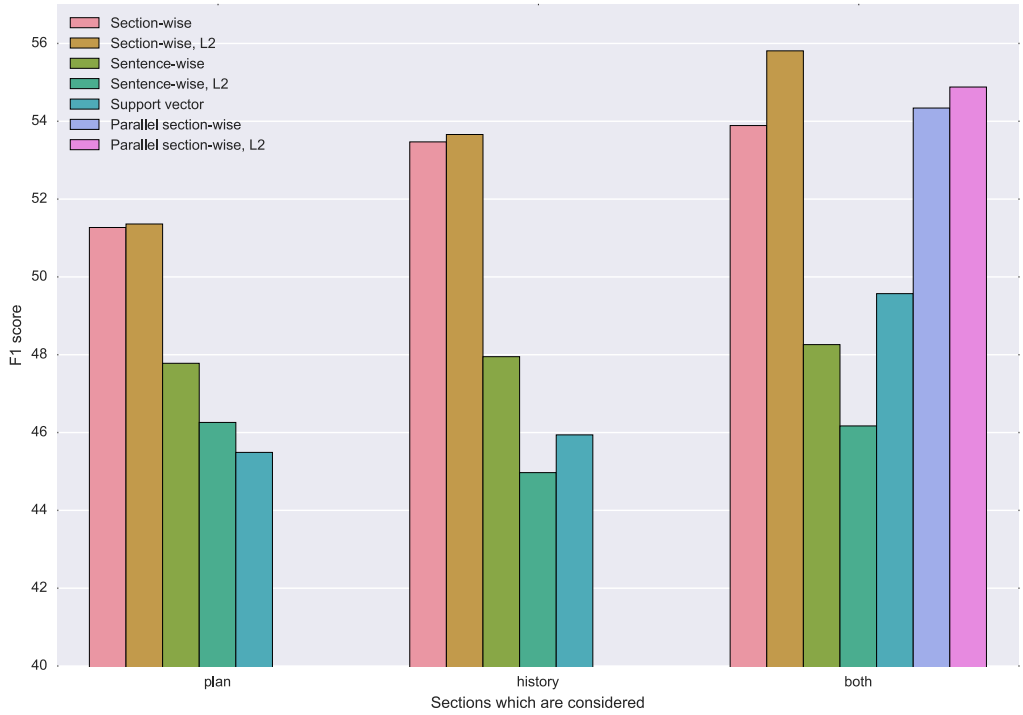

Figure 3.4: F1 scores for the top-level categories prediction for different classifier systems [bar colors] and sections they were fed [x axis].

the 'plan' part. When employing both sections the score is further increased, albeit only slightly. This suggests, that the systems perceive the content of these sections as being redundant to a large extent.
Furthermore it can be seen that a weak L_2 regularization enhances classification performance in all cases, although to a varying extent (see figure 3.4). As the focus of this thesis is the comparison between systems, further significant improvements could be easily achieved by tuning the hyper-parameters for each classifier, especially the regularization strengths of different layers, number of convolutional filters and filter widths.
When looking at the category-specific results (see Figure 3.6 on page 37), it can be seen that the categories 'Endocrine, nutritional and metabolic diseases, and immunity disorders' and 'Diseases of the respiratory system' perform distinctively worse than the average, while 'Neoplasms' are recognized by all classifiers above average. This may be partly due to the fact that the number of training samples is not consistent over all categories, with 'Neoplasms' exhibiting nearly 240k samples (for the section-wise approach) while 'Endocrine and [...] disorders' only demonstrate a count of

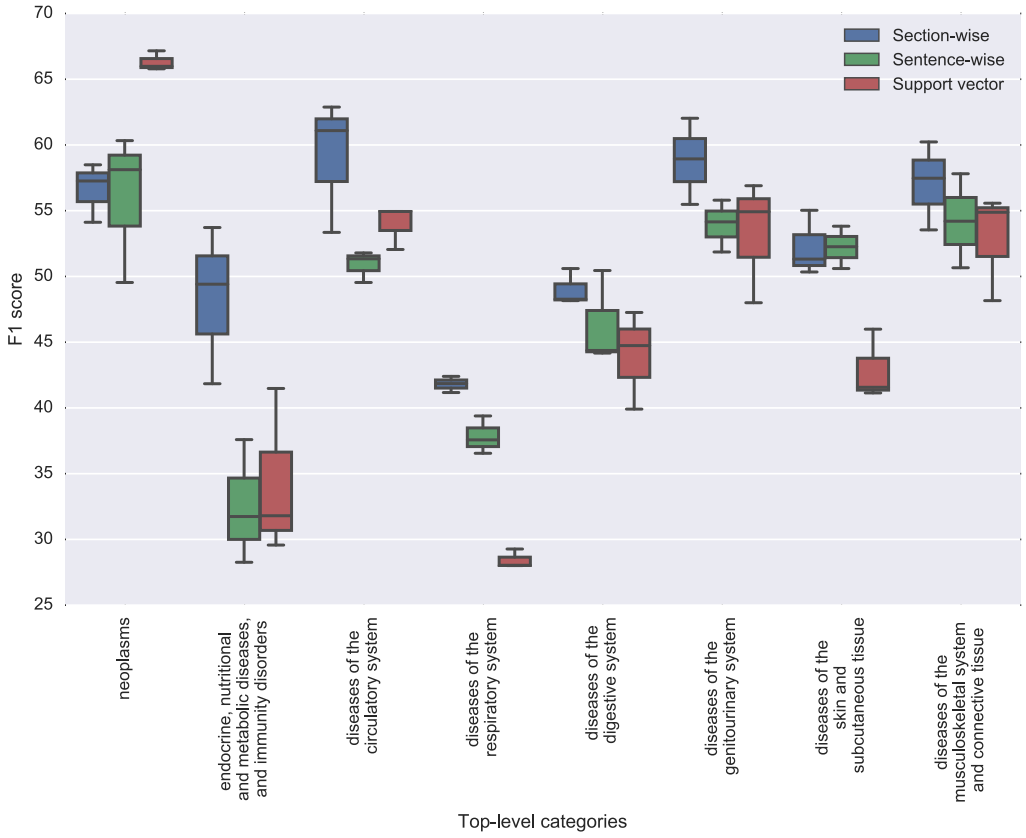

Figure 3.5: Boxplot of medical records classifiers for each label [x-axis] and classifier [color]. The plots depict the variability of using different input sections $\in$ {'history', 'plan', both} without the application of L_2 regularization.

90k. However, other categories with even fewer samples, such as 'Diseases of the musculoskeletal system and connective tissue' (72k), still achieve good results. Another explanation might that some categories display less consistency within themselves than others, in that they contain diseases with very different phenotypes and wording. Finally, the diseases in some categories might be more elaborately described than those in others, either due to their perceived severity or due to the specialization of the hospital (in this case a cancer center) or the physicians.

Besides the quantitative evaluation of correctness, results from the sentence-wise classifier can also be examined qualitatively: In Table 3.5 on page 38 and Table 3.6 on page 39 some examples of phrases are listed which the

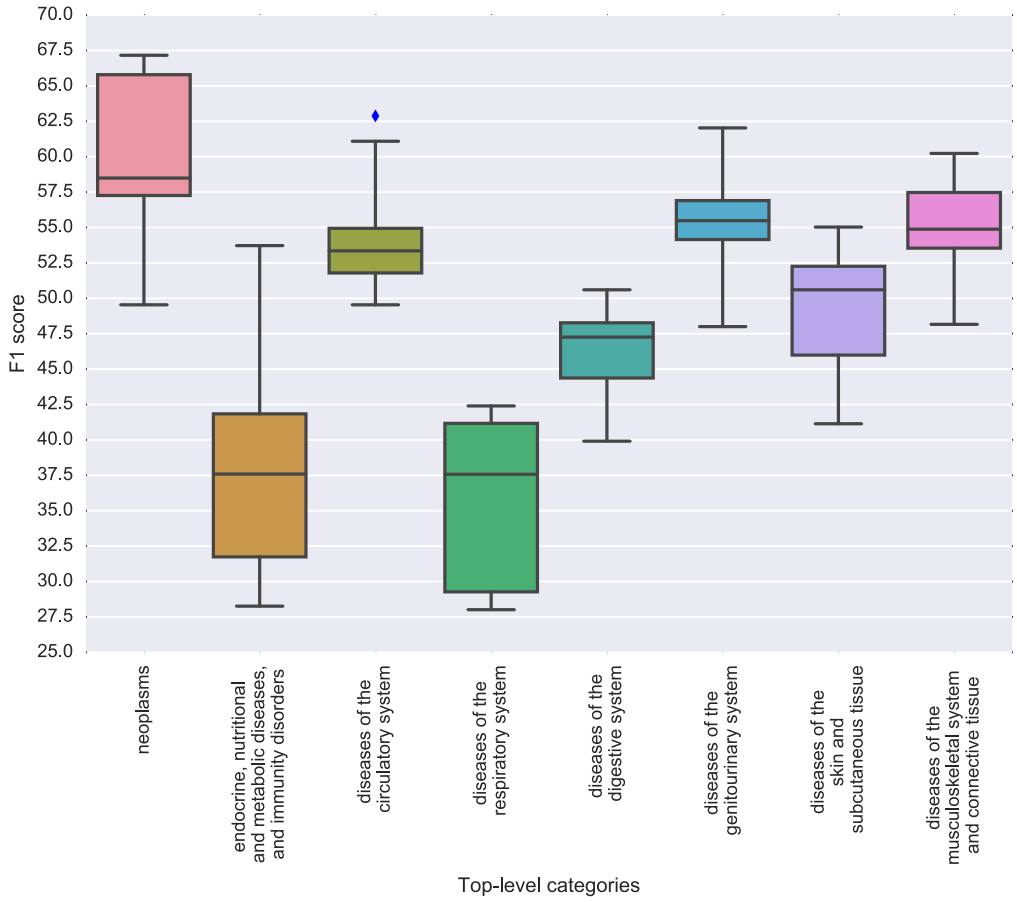

Figure 3.6: Boxplot of top-level classifiers for each label by category.

system classified with high certainty, all being in the top 10 for their category. The phrases were shown to a physician, who rated the plausibility of the prediction according to the following scheme:

Positive: Clearly correct or strong evidence in favour.

Potential: Plausible with some indicators pointing towards correctness, might however also not be the case.

Unsubstantial: There is no information in the phrase pointing towards that prediction.

In total 48 sentences were screened by a physician, 6 coming from each top-level category, all of which were among the top 10 certainty predictions. Out

of the 48 sentences 42 were rated as *positive* (88%), 5 as *potential* (10%) and 1 as *unsubstantial* (2%).

Table 3.5: Sample phrases out of the top 10 certainty predictions for top-level categories 1-4, annotated by a physician regarding plausibility.

Phrase	Plausibility
Neoplasms	
The lesion pathologically was found to be at the peritoneal reflection.	*Potential*
Concepts surrounding systemic therapy for risk reduction of distant recurrence of micrometastasis were reviewed with the patient at length.	*Positive*
Hence, a current standard approach would involve an infusional 5 FU based FOLFOX or FOLFIRI with the addition of Avastin to either of these regimens [...].	*Positive*
Endocrine, nutritional and metabolic diseases, and immunity disorders	
On [...], her TSH was 12.73, thyroglobulin was less than 0.2, her TG antibodies were elevated at 421.	*Positive*
She is a [...] year old woman who has a history of type 2 diabetes, non insulin requiring, as well as hypertension and hyperlipidemia [...].	*Positive*
I did explain to her that if she does choose radioactive iodine that would be the more common treatment, there is a slight risk that she might become hypothyroid afterwards and would have to be on Synthroid to bring her back up to a euthyroid state.	*Positive*
Diseases of the circulatory system	
He was then placed on a beta blocker, repeat Holter monitor showed recurrent paroxysms or non sustained VT. His Toprol at that time was increased from 50mg a day to 50mg twice a day.	*Positive*
He needs to remain on the sotalol at this point in time for control of his heart rate and rhythm.	*Positive*
A Holter monitor showed atrial tachycardia at 2:1 and 3:1 conduction, PVCs, couplets and non sustained VT.	*Positive*
Diseases of the respiratory system	
The patient will follow up with us in the clinic in three months' time for his continued COPD care.	*Positive*
His chronic dyspnea may be related to his moderate obstructive airways disease; for that he we started him on Spiriva 18 micrograms inhaled once daily, and albuterol inhaler prn.	*Positive*
She also has seasonal allergies, which she considers may be contributing to the cough with rhinitis, sneezing, and postnasal drip.	*Positive*

Table 3.6: Sample phrases out of the top 10 certainty predictions for top-level categories 5-8, annotated by a physician regarding plausibility.

Phrase	Plausibility
Diseases of the digestive system	
He denies any hematemesis, any dark tarry stools or bright red blood per rectum.	*Positive*
Regarding his esophagitis, it is noted that he has attempted to use multiple interventions to date; however, we will give a trial of oxycodone elixir today.	*Positive*
I have also encouraged him to start doing Magic Mouthwash again just before he starts to eat.	*Positive*
Diseases of the genitourinary system	
The patient had an endometrial biopsy [...] showing fragments of benign endocervical polyps and atrophic endometrium.	*Positive*
STD screening.	*Positive*
I am performing a Pap smear and a colposcopy today.	*Positive*
Diseases of the skin and subcutaneous tissue	
Lentigo versus seborrheic keratosis on his right lower eyelid.	*Positive*
Followup: The patient is scheduled to return to clinic in early [...] for reevaluation of the tan macule on her left upper arm and for reevaluation of current pruritic eruption.	*Positive*
She will continue to use topical metronidazole gel to her face, and will also use Renova to the lateral aspects of her cheeks and to a solar lentigo on her left neck.	*Positive*
Diseases of the musculoskeletal system and connective tissue	
She has a chronic nociceptive lower back pain due to her bone metastases with a component of arthritic pain.	*Potential*
He still has occasional tingling of the left hip area but overall the T12 radiculopathy symptoms are much improved since undergoing the T12 laminectomy.	*Positive*
She has longstanding cervical radiculopathy as well as arthritic problems in the shoulders and wrists and these pains and related hand weakness are worse.	*Positive*

3.2.2 One-vs-all

The one-vs-all classifier's goal is to estimate if there is evidence in the text for *hypercholesterolemia*, or elevated cholesterol levels. For this setup the same hyper-parameters were employed as for the top-level categories task (see Table 3.4 on page 34), except for the number of classes. Being a binary decision, there are only two of them. For this task we tested the the section-wise network, as it was the best performing classifier for the top-level classification.

For baseline comparison, we also conducted experiment using SVM. An overview of the results is shown in Figure 3.7. The best score is achieved by the regularized section-wise classifier, with an averaged F1 score of 80.6. As already outlined in the previous task, the 'history' section seems to offer more information than the 'plan' part. The per-class scores for the best experiment are 82.2 and 79.0 for the target and non-target notes, respectively.

As distant-supervision drastically improved scores in the sentiment prediction task, this technique was also applied to the one-vs-all classifier. For that purpose the network was initialized with the weights resulting from the top-level category classifier. Here no significant increase of the score could be observed, however, with results being in the range of [-0.5, 0.5] points to the unsupervised experiments. This allows two conclusions: Either the labels are too different to allow transfer-learning or the amount of supervised data is sufficient and not a bottleneck for the classifier's performance.

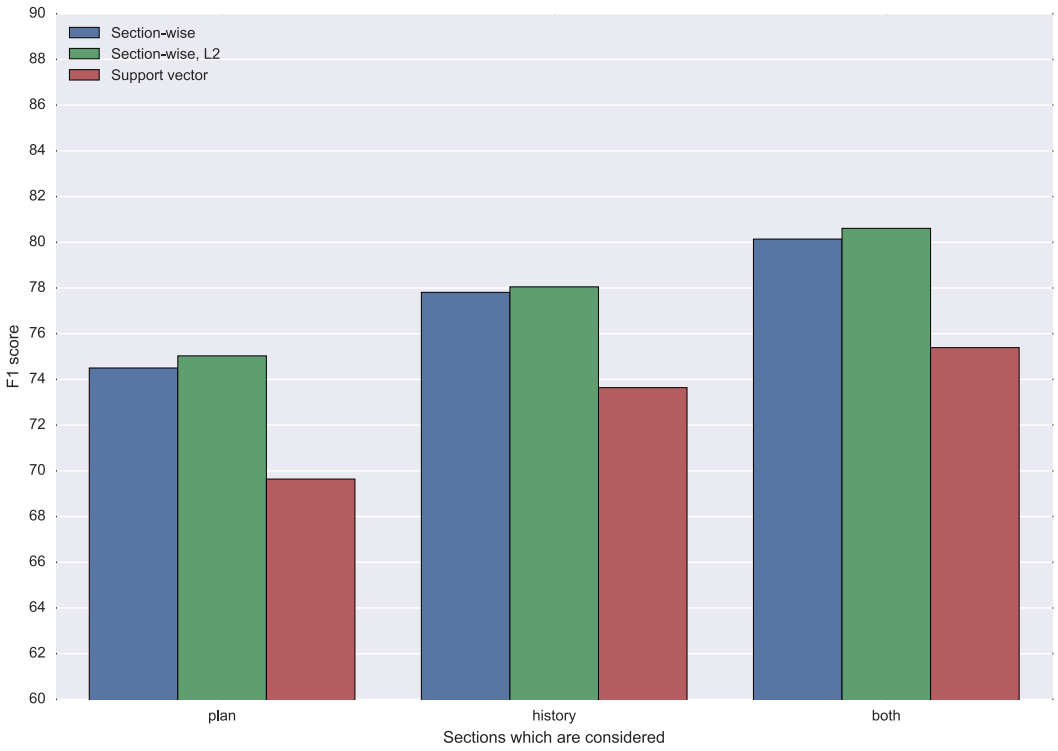

Figure 3.7: F1 scores for the one-vs-all prediction for different classifier systems [bar colors] and sections they were fed [x axis].

Testruns were also carried out for the sentence-wise classifier (not depicted in the figure), delivering a result of $F1 = 77.0$. Qualitative insights into the results can be revealed when looking at the test samples with the highest prediction certainty. For the top 500 predictions (out of 160k), all exhibiting a certainty ≥ 0.99, 495 contain the exact words (hyper)-cholesterolemia, cholesterol or some spelling variants thereof. A few other sentences of the top 1000 predictions *not* containing these words are listed in Table 3.7. A physician scored their plausibility according to the same scheme described in the previous section.
A reliable medical predictor often present in the mentioned sentences is the prescription of specific medications like. It also seemed to link high levels of cholesterol with the occurrence of cardiovascular diseases and therapy methods for the latter. Of the false positives identified during screening a majority is related to some kind of urological condition, for which the systems seems to have picked up a spurious correlation with *hypercholesterolemia*.

Table 3.7: Sample phrases out of top 1000 certainty predictions for *hypercholesterolemia*, annotated by physician regarding plausibility

Phrase	Plausibility
The local urologist confirmed the diagnosis with an office cystoscopy and subsequently performed a transurethral resection of the bladder tumor.	*Unsubstantial*
Aspirin and Plavix should be restarted as soon as feasible postoperatively.	*Positive*
I am asked for my advice and opinion on risk stratification on this [...] year old man with a history of hypertension, hyperlipidemia and myocardial infarction, diabetes mellitus, referred for risk stratification of upcoming TURBT.	*Positive*
Dr. [...] informed 2) Hypertension, cardioprotective beta blocker indicated.	*Positive*
Her ASA should be resumed as soon as possible post operatively because of his drug eluting stent.	*Positive*
Diabetes mellitus: Metformin should be held for 3 days prior to surgery.	*Potential*
His outside urologist recommended a transurethral resection of the prostate followed by BCG treatment.	*Unsubstantial*
Schedule a Cysto EUA TURBT in the SDH [...] under general anesthesia.	*Unsubstantial*

Chapter 4

Conclusion & Future Work

4.1 Conclusion

In this thesis we demonstrated the capabilities of CNNs as natural language analysis methods in two applied problems: *(i)* Twitter sentiment analysis and *(ii)* prediction of ICD-9 codes from medical health records.
For the first problem we empirically demonstrated the influence of various designs, architectures and hyper-parameter decisions on the classification performance. Own word-embeddings were created in an unsupervised manner on a large text corpus, including the datasets later used for training purposes. Through distant supervised training on weak labels, the score could be substantially increased. Being part of the winning team of the *SemEval competition 2016, task 4a*, the approach suggested approach proved to perform similarly to other currently cutting-edge systems. We achieved F1 scores of 62.36 with our system and 63.30 with the ensemble our system was part of on the *SemEval Twitter2016* test set. This also highlights the promising trend towards machine learning techniques relying on automatically generated features, convolutional filters.
The focus of the analysis of medical health records was to establish a solid baseline for systems predicting health conditions – specifically ICD-9 codes – from text data. For baseline comparison experiments were also conducted on support vector classifiers, although these performed poorly in comparison to the implemented CNNs. Two concrete *single label multiple class* tasks were designed for the systems: *(i)* predicting the top-level category (8 classes) for a health record and *(ii)* determining if a note is assigned a specific code or note (binary decision). Experiments yielded an F1 score of 55.81 for task *(i)* and 80.61 for task *(ii)*. Additionally we investigated which parts of the notes encode the most useful information for the classifiers. Significant differences in performance were also exhibited among the classes, for which possible reasons are discussed. Network architectures were developed to either take entire sections or single sentences as inputs, with the

former exceeding the performance of the latter. However, it was shown that distant training did not prove suitable for this task.

4.2 Future Work

The techniques presented in this work are a very active field of development and expected to make their way into useful applications over the next few years. In the following paragraphs a few technical and practical aspects are discussed as potential ways to improve performance and make use of the results:

Inner Workings of Neural Networks: Understanding the patterns learnt by neural networks is very challenging. It requires the translation of a large number and often multi-dimensional (3D or 4D) layer weights into a more abstracted logic that humans can make sense of. In computer vision tasks, these weights can be interpreted as images again, revealing the kind of features searched for in each layer [33, 38]. Yet in NLP tasks, their direct interpretation is tougher, as the word-embeddings represent already a level of abstraction.
One way of finding patterns to which the network reacts, is to feed it (randomly) mutated samples and measure the influence of these modifications on the output. Although this approach will probably present some clues, it offers no genuine explanations but merely sample-based evidence on the conditional probability distribution.
Another method would be to attach a *deconvolutional network* to the CNN, which allows the feature activity of intermediate layers to be mapped back to the input. Although this approach is sample based as well – the deconvolution is dependent on inputs to the corresponding CNN – it reveals *what patterns* caused the given activation. Due to the non-invertible nature of operations such as max-pooling, some approximations have to be made. In computer vision tasks this technique demonstrated promising results [38], illustrating the hierarchical structure of the filters, with those on lower levels reacting to abstract patterns and high-level ones matching very specific features, like noses or eyes. However, for NLP tasks this approach requires the additional and ambigious step of converting the reconstructed inputs back to real words.
A direct inspection of the filter weights would be most insightful, although there are currently no known methods for interpreting these in NLP tasks.

Applying Distant learning to Medical Data: As could be seen for the tweet sentiment prediction, distant supervised training on weak or unlabelled data can in some cases substantially improve results. Applied to the medical domain, this could mean making use of large publicly available

datasets such as *PubMed*, which contains over 25M articles. These articles could be used in an unsupervised manner to improve the word-embeddings. In addition, automatically generated labels created with metadata or heuristics could be employed to pre-train a neural network. Through sensible distant supervision the problem of small training samples can possible be alleviated. However, more research is required as to which cases are suited for transfer-learning methods and what kind of relationship the 'weak' and 'strong' labels should exhibit.

Re-utilization of hidden layer output as embeddings: While word representation as fixed-length vectors have experienced wide adoption and demonstrated promising results, the equivalent for entire sentences is still at an early stage. The difficulty here lies in encoding all semantic and syntactic information conveyed by a phrase in some standardized, numerical manner. Previous works suggested the use of techniques similar to the ones used for word-embeddings, like skip-gram models, on very short phrases [22]. Other approaches made use of context-aware recursive neural networks, working with sentences of a length up to 15 [34].
The adequacy and quality of sentence-embeddings heavily depends on their designated purpose and the domain they originate from. The characterization of writing styles for novel reviews requires radically different information than the categorization of medical text documents. For applications in areas where some kind of labels are available, the output of a hidden-layer from a convolutional neural network similar to the ones described in this work, could be used for that purpose: Once the network is trained, it inherently generates some fixed-length representation for given inputs. The network trained on the ICD-9 codes could thus be employed to get embeddings for medical health record sentences or even entire notes. By unsupervised clustering algorithms the notes could then be aggregated and matched with other data, such as gene mutations [4], to display and potentially discover correlations between text and genes.

Bibliography

[1] *Learning Semantic Representations Using Convolutional Neural Networks for Web Search*. WWW 2014, apr 2014.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[3] James Bergstra, Olivier Breuleux, Frederic Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math compiler in Python. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, (Scipy):1–7, 2010.

[4] K R Chan, X Lou, T Karaletsos, C Crosbie, S Gardos, D Artz, and G Rätsch. An Empirical Analysis of Topic Modeling for Mining Cancer Clinical Notes. In *2013 IEEE 13th International Conference on Data Mining Workshops*, pages 56–63, dec 2013.

[5] Sharan Chetlur and Cliff Woolley. cuDNN: Efficient Primitives for Deep Learning. *arXiv preprint arXiv: ...*, pages 1–9, 2014.

[6] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 539–546, 2005.

[7] Dan Cires and Ueli Meier. Multi-column Deep Neural Networks for Image Classification. *Applied Sciences*, (February):20, 2012.

[8] Dan Claudiu Ciresan, Ueli Meier, Luca Maria Gambardella, and Juergen Schmidhuber. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition. *Neural Computation*, 22(12):1–14, 2010.

[9] J. Deriu, M. Gonzenbach, F. Uzdilli, A. Lucchi, V. De Luca, and Jaggi. M. SwissCheese at SemEval-2016 Task 4: Sentiment Classification Using an Ensemble of Convolutional Neural Networks with Distant Supervision. *Int Workshop on Sem-Eval, 2016. Accepted*, 2016.

[10] Li Dong, Furu Wei, Chuanqi Tan, Duyu Tang, Ming Zhou, and Ke Xu. Adaptive Recursive Neural Network for Target-dependent Twitter Sentiment Classification. *Acl-2014*, pages 49–54, 2014.

[11] Dumitru Erhan, Aaron Courville, and Pascal Vincent. Why Does Unsupervised Pre-training Help Deep Learning ? *Journal of Machine Learning Research*, 11:625–660, 2010.

[12] Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. Learning Semantic Representations for the Phrase Translation Model. *arXiv preprint arXiv:1312.0482*, 2013.

[13] Wolfgang Gentzsch. Sun Grid Engine: Towards creating a compute power grid. In *Proceedings - 1st IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid 2001*, pages 35–36, 2001.

[14] Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Shortpapers*, number 2, pages 42–47, 2011.

[15] Alec Go, Richa Bhayani, and Lei Huang. Twitter Sentiment Classification using Distant Supervision. *Processing*, 150(12):1–6, 2009.

[16] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition With Deep Recurrent Neural Networks. *Icassp*, (3):6645–6649, 2013.

[17] Rie Johnson and Tong Zhang. Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. *NIPS*, pages 1–9, 2015.

[18] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A Convolutional Neural Network for Modelling Sentences. *Acl*, pages 655–665, 2014.

[19] Yoon Kim. Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1746–1751, 2014.

[20] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998.

[21] Yann. LeCun, Fu Jie Huang, and Leon. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, 2:97–104, 2004.

[22] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, pages 1–12, 2013.

[23] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černocký. Strategies for training large scale neural network language models. In *2011 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2011, Proceedings*, pages 196–201, 2011.

[24] Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. SemEval-2016 Task 4: Sentiment Analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, jun 2016. Association for Computational Linguistics.

[25] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[26] Adam Perer, Fei Wang, and Jianying Hu. Mining and exploring care pathways from electronic medical records with visual analytics. *Journal of Biomedical Informatics*, 56:369–378, 2015.

[27] Jonathon Read. Using Emoticons to reduce Dependency in Machine Learning Techniques for Sentiment Classification. *ACL Student Research workshop*, (June):43–48, 2005.

[28] Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M. Mohammad, Alan Ritter, and Veselin Stoyanov. SemEval-2015 Task 10: Sentiment Analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 451–463, 2015.

[29] Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. Ultradense Word Embeddings by Orthogonal Transformation. *CoRR*, abs/1602.0, 2016.

[30] David E Rumelhart, James L McClelland, and R J Williams. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. 1986.

[31] Aliaksei Severyn and Alessandro Moschitti. Twitter Sentiment Analysis with Deep Convolutional Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 959–962, 2015.

[32] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management - CIKM '14*, pages 101–110, 2014.

[33] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. *Iclr*, 2014.

[34] Richard Socher, Christopher D Cd Manning, and Andrew Y Ay Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks. *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9, 2010.

[35] Richard Socher, Alex Perelygin, and Jy Wu. Recursive deep models for semantic compositionality over a sentiment treebank. *Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[36] Joachims Thorsten. Transductive Inference for Text Classification Using Support Vector Machines. *Icml*, 99:200–209, 1999.

[37] Matthew D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv*, page 6, 2012.

[38] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8689 LNCS, pages 818–833, 2014.

[39] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. *NIPS*, pages 1–9, 2015.