Institute of Cognitive Science

Bachelor's Thesis

A textual recommender system and other text mining applications for clinical data

Philipp Hummel

May 8, 2017

First supervisor: Dr. Frank Jäkel
Second supervisor: Prof. Dr. Gordon Pipa

A textual recommender system and other text mining applications for clinical data

In atypical clinical situations doctors would often like to review cases of similar patients to guide their decision making for the current one. To retrieve the relevant cases however is a hard and time consuming task. This thesis works on building a recommender system that automatically finds physician letters similar to a reference letter in an information retrieval like manner. We use a small dataset of free text physician letters of oncology patients to do exploratory work on this issue. We provide a prototypical system that gives recommendations on this dataset and verify it's performance through a psychological experiment assessing doctor's similarity judgments of those letters. We find that the similarity measure of our recommender system correlates strongly with doctor's similarity judgments. Additionally we explore other text mining applications like automatic diagnosis extraction based on this dataset.

Acknowledgements

acknowledgements

Contents

Chapter 1

Introduction

1.1 Motivation

Publishing the case of a patient with a particularly interesting medical phenomenon in the form of a clinical case report has seen a change in popularity in the medical community. The number of published case reports has been declining in standard journals. This has happened not only because case reports can hardly contribute to a good impact factor, but also because their scientific benefit has been questioned (Mason, 2001). However, several new journals dedicated only to case reports have emerged (Kidd and Hubbard, 2007)(how to cite a journal's existence, which does not have an introductory article?) and many people have argued for the value of case reports for research itself but also beyond (Williams, 2003; Dib et al., 2008; Sandu et al., 2016). Importantly case reports allow practitioners a more in-depth understanding of specific disease courses and provide educational material for students (Nissen and Wynn, 2014). Although case reports lack the scientific validity of large empirical studies, it is apparent that people have strong intuitions for the usefulness of them. During our collaboration with practicing doctors we also found that practitioners use the clinical records of similar patients to guide problem solving for the current patient. Especially when faced with hard or unusual cases doctors seek similar patient information from the hospital database. While this only shows that doctors think that the presentation of similar cases helps them in their work, we will argue that it can indeed improve medical problem solving.

Cognitive Scientists have discussed the usefulness of examples for reasoning processes for long and found that at least in some experimental settings reasoning processes are based on earlier presented examples (Medin and Schaffer, 1978). More recently these reasoning processes have also been studied in more realistic scenarios. Klein (2008) reviews models for decision making under real world circumstances. According to him experts interpret a situation based on its resemblance to remembered situations. Once a sufficiently similar situation has been retrieved from memory, experts apply the solution from the remembered situation to the current one in a thought experiment. They evaluate whether or not this solution strategy will

lead to success and adjust it, if necessary. In case no way to adequately adjust the solution can be found, another situation is retrieved from memory. This process is repeated until a sufficiently good solution is found. Presentation of similar cases should therefore aid doctor's decision making in an actual clinical setting. The medical domain has also been directly addressed by research in cognitive science. Elstein and Schwarz (2002) have concluded that for medical problem solving reasoning processes can be divided into two distinct categories. For cases perceived as easy doctors apply a kind of pattern recognition based on the examples they have encountered before and use solutions stored in memory. For harder cases, however, doctors need to rely on a more elaborate reasoning process. They have to consciously generate and eliminate hypotheses to be able to solve the problem. It is plausible that hypothesis generation as well as hypothesis falsification is also guided by the doctor's experience of earlier patients. From a more theoretical perspective Kolodner and Kolodner (1987) have specifically argued that "[i]ndividual experiences act as exemplars upon which to base later decisions" in medical problem solving. Their research was partially driven by the desire to understand the way in which clinicians perform problem solving but also by the goal of building artificial systems that can aid in this process. They argue that both learn from specific examples and use them to reason about new problems.

Around the idea that artificial systems might learn from examples has evolved a whole branch of Artificial Intelligence (AI), which is called "Case-based reasoning". This domain has been greatly influenced by psychological findings, some of them mentioned above. It has successfully built systems used in real world applications, that reason from the examples provided (Aamodt and Plaza, 1994). Within this domain of AI one of the greatest application areas is medicine. It seems that this area does not only offer straight forward usage of examples, but also has a need for automatic aids for problem solving (Begum et al., 2011).

Given the practical, psychological and theoretical reflections above we believe that it would be helpful for practitioners to be able to review cases of similar patients. One particularly well suited source for the retrieval of patient cases are databases of physician letters, as they provide concise summaries of the specifics that matter in practice. Search in these databases is, to our knowledge, usually limited to character matching procedures and therefore provides limited practical value for doctors. We therefore set out to build a recommender system on those physician letters to do automatic retrieval of only the relevant documents from a database.

1.2 Dataset

Chapter 2

Dataset

The university hospital Freiburg has a database of approximately 190,000 German physician letters in PDF form (Spadaro, 2012) (in der clinicon Beschreibung heißt es 190.000 medizinische Dokumente und Arztbriefe) (außerdem bin ich noch nicht zufrieden, wie das item in der bibliography erscheint. Ich weiß allerdings auch nicht wie es richtig wäre.). These physician letters are free text documents written by the doctors to keep record of the patient's visit. They usually include information about the patient's age, sex, diagnosed diseases, therapy history, current complaints, many more medical details like blood counts, but also personal information like names and birth dates. The letters usually follow a rough structure. Almost all of them include a letter head (a greeting and introduction), a diagnosis (summarizing diagnosed diseases bullet point like), a therapy history (listing the past therapies with dates) and an anamnesis (free text about current complaints etc.) section, separated into individual paragraphs. In principal though, doctors are free to document this information in the way they please. The database does not, however, contain the information of 190,000 unique patients. For many patients several letters are included, as a new visit will often result in an updated letter, that is added to the database.

From this database we acquired a subset of 307 anonymous letters in Microsoft Word XML format. Unfortunately 18 of the 307 letters are duplicates and thereby not usable for our analysis. We excluded another 3 letters, that contain almost no information about the patient (this kind of letter is produced due to the specific kind of documentation process at the clinic). Another 17 are "follow-up" letters i.e. letters of the same patient at a later point in time. This left us with the letters of 269 individual patients (the follow-up letters were only used where stated explicitly). Note that it is not a trivial task to ensure that the set of letters contains neither duplicates nor follow-ups. Automated aids for this problem will be discussed below.

For our goals of automatically extracting information from and finding similarities between the letters we additionally needed supervised information. For a subset of 135 letters we obtained supervised labels of two kinds. One, we manually labeled the letters for whether or not the patients suffered from specific diseases common

among those patients. These labels were checked by an expert for correctness. Two, we obtained a grouping of these 135 patients into 50 non-overlapping groups from an expert. This grouping is meant to reflect the expert's intuition about similarities between patients, that might not be capturable in simple rule based grouping approaches like grouping by disease. Additionally we performed a psychological experiment with [replace] medicine students and [replace] doctors, probing their intuition about letter similarity for a larger set. With these data we tried several approaches for automatic extraction of information from the letters and automatic similarity ratings between them. The methods used in our approaches are described in the next chapter.

Chapter 3

Methods

3.1 Notes on Notation and Naming Conventions

In this thesis vectors are represented by lowercase boldface characters such as $\mathbf{v}$ and matrices by uppercase boldface letters such as $\mathbf{M}$.

In the context of Natural Language Processing in this thesis the words "document" and "text" are used interchangeably and refer to texts of variable length such as sentences, paragraphs or whole documents. "term" and "word" are used interchangeably as well. Although "term" in general often refers to an entity within the text, that might be composed of several "words", for example "New York", here both will be used to refer to single words.

3.2 Preprocessing

One very common way of representing text in a computer is to view the text as a sequence of individual characters, a string. However, for many methods of information retrieval and machine learning, including the ones described subsequently, the units of text we would like to deal with are words or terms not sequences of characters. The process of extracting words or terms from a sequence of characters is known as tokenization (Manning et al., 2008*b*). A first naive approach to tokenization might be to split the sequence of characters on every whitespace and regard everything in between as a word. This approach, however, can easily lead to unexpected results. Consider the example string "john loves mary.". The naive approach yields the words "john", "loves" and "mary.". Note how the last word contains the punctuation character ".". So even for very easy examples the naive approach does not produce expected results. Luckily many more sophisticated algorithms are implemented in ready-to-use open-source software packages.

A second common step in preprocessing is the removal of so called stop words. Stop words are usually the most commonly used words of a language, that do not contain much meaning, e.g. words like "the", "a", "just" or "some". While they are necessary for the grammatical structure of a language, they do not convey much

information about the similarity of documents and can be disregarded in further analysis steps (Manning et al., 2008*b*).

Another often used preprocessing method is stemming. Stemming refers to the process of mapping (inflected) words to their stems, e.g. mapping words such as "am", "is" and "are" to the word "be". The rationale behind using stemming is that for finding e.g. the topic of a text it should not matter which form of the word is present in a text, but only that a word is present in a text (Manning et al., 2008*b*).

All three preprocessing steps described above are used in this thesis except where noted otherwise. The python package NLTK (natural language toolkit) provides many specific implementations of these algorithms and was used throughout this thesis. For tokenization the "TreebankWordTokenizer" and for stemming the "SnowballStemmer" are used. A list of German stopwords is also taken from the package.

3.3 Bag of Words

After preprocessing of a text one of the standard approaches for representing documents for information retrieval and machine learning algorithms is in the bag of words model. In the bag of words model a text is represented by the multiset (bag) of its words. That means all word order information is disregarded and only the information whether (or how often) a word is present in a text is encoded (Manning et al., 2008*a*). (Mit der Art wie das Buch hier zitiert wird bin ich noch nicht glücklich. Es sollte eigentlich chapter 2 und 6 sein und nicht 2008 a und b.)

In this model documents can easily be represented as fixed length feature vectors, where every feature is a kind of word occurrence count. In the simplest approach every feature is the term frequency $tf(t,d)$ of term t in a document d. Where $tf(t,d)$ is simply the occurrence count of word t in document d. To compute feature vectors for documents in a corpus the vocabulary V of the corpus needs to be established first. Documents are represented by a $1 \times |V|$ vector of the values $tf(t,d)$ for all $t \in V$. Assume the corpus consists of two documents d_1 ="john loves mary" and d_2 ="mary loves pizza". The vectors representing the two texts are:

$$\mathbf{v}_{d_1} = \begin{matrix} 1 \\ 1 \\ 1 \\ 0 \end{matrix} \quad \mathbf{v}_{d_2} = \begin{matrix} 0 \\ 1 \\ 1 \\ 1 \end{matrix} \quad \begin{matrix} john \\ loves \\ mary \\ pizza \end{matrix}$$

This representation obviously has severe limitation, consider for example the two sentences d_3 ="john loves mary" and d_4="mary loves jon". Their representation in the bag of words model is equivalent, although they express quite different meanings. Still the bag of words model is a standard approach for information retrieval problems, as it has many desirable properties. For example documents can be expressed as a fixed length vector, necessary for many machine learning tasks, and distances in the vector space can straight forwardly be interpreted as a measure for

document dissimilarities. In information retrieval often the focus lies on the topic of a text rather than it's specific content. Some of the weaknesses of the bag of words approach are then neglectable. Consider again the sentences d_3 and d_4 and additionally the sentence d_5 ="computers automate tasks". If we are interested in the topic of the sentences or in their relevance to each other, then the bag of words model will give the reasonable result that d_3 and d_4 are more relevant to each other than to d_5.

3.4 N-Grams

In order to extend the bag of words representation to capture word order to some extent n-grams have been proposed. The idea of n-grams is to treat a sequence of n words as a distinct token or term. For the sentence d_3 the list of 2-grams or bigrams is [(john, loves), (loves mary)] and for $d_4$it is [(mary, loves), (loves, john)]. So the problem of the equivalent representation of d_3 and d_4 has been solved by capturing word order information in small context windows. However, this comes at the cost of a higher dimensionality of the resulting feature vector as there are more distinct n-grams in a corpus than single words. In principle the cardinality of the n-gram vocabulary could be as large as $|V_n| = |V_1|^n$, where V_n is the n-gram vocabulary. However, in natural texts not all possible n-grams occur and the feature vector dimensionality grows more moderately than an exponential function with increasing n. (Vielleicht kommt der Absatz doch wieder raus. Habe n-grams jetzt doch noch gar nicht verwendet.)

3.5 Term Frequency - Inverse Document Frequency

A common problem with the bag of words model is that some terms (even after filtering stop words) appear often across texts in a corpus, i.e. have a high term frequency yet do not constitute a good feature for discrimination between texts. Therefore a scaling factor for the term frequencies is desired which captures the intuition that words appearing often in a few texts but rarely in others are good discriminative features for those texts. Term Frequency - Inverse Document Frequency (tf-idf) refers to a specific scaling scheme, that downscales the importance of frequent words, while upscaling the importance of rare words.

Term frequency $tf(t, d)$ usually refers to the standard word count. Inverse document frequency $idf(t, C)$ of term t and corpus C can be computed as

$$idf(t, C) = log_2(\frac{|C|}{|\{d \in C : t \in d\}|})$$

where $|C|$ is the total number of documents in the corpus and $|\{d \in C : t \in d\}|$ is the number of documents in which term t appears at least once.

Term frequency - Inverse document frequency $tfidf(t, d, C)$ is then calculated as

$$tfidf(t, d, C) = tf(t, d) \cdot idf(t, C)$$

tf-idf can be used as a feature for the representation of the document that is more robust to uninformative changes in the distribution of common words and more expressive for rare words (Manning et al., 2008*a*).

3.6 Latent Semantic Analysis

An often occurring machine learning problem is that very high dimensional feature vectors, as occurring in bag of words models described above, tend to generalize poorly in subsequent tasks like classification. Therefore it is desirable to have a more condensed lower dimensional feature representation of documents that still captures most of the variance in the bag of words representation. Latent semantic analysis (LSA) of Deerwester et al. (1990) in its simplest form takes the plain bag of words vectors of all documents in the corpus and constructs a term-document matrix $\mathbf{M}$, where $\mathbf{M}[i,j] = tf(t_i, d_j)$, i.e. row i represents the relation of term i to all documents, while column j represents one document and the relation to all it's terms. So column j represents document d_j's vector representation $\mathbf{v}_{d_j}$.

$$\begin{array}{cc} & \mathbf{v}_{d_j} \\ & \downarrow \\ \mathbf{v}_{t_i}^T \rightarrow & \begin{bmatrix} tf(1,1) & \dots & tf(1,|C|) \\ \vdots & \ddots & \vdots \\ tf(|V|,1) & \dots & tf(|V|,|C|) \end{bmatrix} \end{array}$$

Singular value decomposition is then used on the term document matrix $\mathbf{M}$, allowing to find a k-dimensional ($k < dim(\mathbf{M})$) linear subspace of $\mathbf{M}$, $\hat{\mathbf{M}}$, that still captures as much of the original information as possible, i.e. that captures as much of the variance in the original space as possible. Column j of $\hat{\mathbf{M}}$ contains a k-dimensional, approximate representation of the document j's feature vector $\mathbf{v}_{d_j}$, called $\hat{\mathbf{v}}_{d_j}$. The document representations $\hat{\mathbf{v}}_{d_j}$ in $\mathbf{M}$'s linear subspace spanned by $\hat{\mathbf{M}}$ do no longer have an intuitive interpretation as word counts, but can still be used as feature vectors for subsequent ML tasks or can be analyzed for similarity. Deerwester et al. argue that the resulting features have several appealing properties. The LSA features can simply be viewed as a noise-reduced version of the original features, but according to the original paper they can also better deal with linguistic issues such as synonymy and polysemy. This works because every original observation (i.e. document) is represented as a linear combination of k hidden (or latent) semantic concepts. Terms that often appear together (i.e. are semantically related) will then be mapped onto similar LSA representations and can thereby for example capture some aspects of synonymy.

The value of k is a parameter of the model and has to be specified by the researcher. LSA can also be used with tf-idf features.

3.7 Latent Dirichlet Allocation

A popular probabilistic method for finding latent semantic features of documents in a corpus is latent dirichlet allocation (LDA). As with LSA the rationale is that it would be useful to find a shorter or lower dimensional description for documents in the bag of words vector space format for subsequent tasks. In the LDA context the latent features are called topics and texts are represented as probabilistic mixtures of these topics. A topic is represented by a probability distribution over words and so a document is represented as a probabilistic sample from several topic distributions over words. As in the case of LSA the number of topics k is a parameter and needs to be specified by the researcher (Blei et al., 2003).

3.8 Word Vectors

In the standard bag of words model no relationship between words is encoded, i.e. every word is dissimilar from every other word. Thereby a lot of information is lost, as e.g. the words "car" and "automobile" are considered by the system to be as similar to each other as to the word "blue". One way to encode relationships between words is to embed them in a vector space. Just like the way document similarities can be computed from their distances in the vector space model, similarities of words can be computed, if the words are represented as vectors. A first successful approach to this problem used the LSA model. Here a single word can be represented as a pseudo document and thereby be mapped into the LSA space. The resulting vector in the LSA feature space can be considered a vector representation for the word and be used for comparisons with other words (Deerwester et al., 1990).

Bengio et al. (2003) introduced a neural language model that uses word vectors to predict subsequent words in a text and updates the model as well as the vectors with gradient descent. It seems that by being useful for prediction of subsequent words, the vectors represent statistical information about word contexts and capture meaning of the words to some extent.

Recently Mikolov, Chen, Corrado and Dean (2013) proposed their influential Word2Vec model. This model utilizes a neural network for the prediction of word vectors, given other nearby word vectors. However, the authors focused on the refinement of the word vectors only and were able to simplify the net substantially, while gaining word vector "performance" (i.e. vectors that perform better on a task designed to measure how well the vectors capture human intuitions about the words). Their model works simply with scalar products of input and output word vectors (it has two vectors per word) and a softmax output layer for normalization. This model comes in two architectural types: "Continuous Bag of Words" (CBOW) and "Continuous Skip-gram" (skip-gram). The former predicts the output vector of a center word, given the n previous and n subsequent input word vectors. The input vectors of the $2n$ surrounding words are averaged, the scalar product of this average with every output word vector is computed and a softmax normalization produces final output probabilities. The skip-gram model utilizes the input vector

of the center word to predict the output vectors of surrounding words in a left and right context window of maximal size w. The current window size c is sampled uniformly from $range(1, w)$ at every iteration. This results in $2 \cdot c$ predictions for every center word considered. The model parameters are updated with gradient descent. After training either only the input word vectors are used or the input and output word vectors are either averaged or concatenated to produce the final word vectors used for subsequent tasks.

A note on computational cost: The softmax output layer is very inefficient as it requires the computation of $|V|$ scalar products (every word in the vocabulary). As a first speed up a hierarchical softmax (Morin and Bengio, 2005) is used as an approximation to the real softmax, that reduces the number of scalar product computations to $log_2(|V|)$ (Mikolov, Chen, Corrado and Dean, 2013). In a subsequent publication the authors introduced a simplified variant of Noise Contrastive Estimation (Gutmann and Hyvärinen, 2012), called Negative Sampling, that further reduces the computational complexity, while still giving useful word vectors (Mikolov, Sutskever, Chen, Corrado and Dean, 2013).

With these architecture and approximation simplifications the Word2Vec model is able to train on a vast amount of text data (billions of words) in a matter of hours and thereby produce better word vectors, than previously possible.

3.9 Paragraph Vector - An extension of Word2Vec

A simple extension of the word2vec model allows to obtain vectors for sentences or longer passages of text. An additional vector is introduced for every piece of text, that we regard as a separate entity (a sentence, a paragraph or a document). This so called paragraph vector is then used in two different ways in the extensions of CBOW and skip-gram. In the Distributed Memory (DM) model, an extension of CBOW, the paragraph vector is used in combination (average or concatenation) with the vectors of surrounding words to predict a center word. In the Distributed bag of words model, an extension of skip-gram, the paragraph embedding is used to predict words randomly sampled from the paragraph.

After the initial training phase a second step, called inference phase, is necessary to gain paragraph embeddings. A paragraph vector is initialized randomly, the rest of the model is kept fixed and the normal training procedure of word prediction and gradient descent on the paragraph vector is done. Thereby the final document embeddings are produced, that can be used like the embeddings of the methods described above. Note, however, that the time until the model reaches convergence both in the training and in the inference phase is unknown and a number of running epochs for both phases must be specified beforehand (Le and Mikolov, 2014).

Chapter 4

Results

Vielleicht sollte das Kapitel eher challenges and solutions heißen. Es sind ja nicht nur results, sondern auch immer die Problemstellungen beschrieben.

4.1 Duplicate Detection

Exploring our dataset we realized that we had to deal with two duplication problems. The first being that several letters were contained more than once in our dataset. The second, more challenging problem was that for several patients letters of different time points with somewhat different content were present. To ensure we would not distort our results, because of duplicates present in our test set, we tried to identify all duplicates of the two kinds. On first glance at least the first problem seems easy. Finding exact duplicates is not a challenging task. However, as the letters were made anonymous separately, they are not exact copies of each other. Names and other personal information were eradicated from the word documents by hand, so extra whitespaces and similar subtle differences were introduced. For a subset of 150 of the letters we manually searched for the duplicates. However, this work is tedious, error-prone and does not scale to bigger datasets. To semi-automatically find the true duplicates and the follow-up letters we used two approaches.

The first method searches two letters for their longest common subsequence of characters. If this longest common subsequence exceeds a threshold relative to the longer document (e.g. 3 % of the length of the longer document), then the two letters are marked as possibly related. In the second approach we use the bag of words representations of the documents and their cosine distance in the vector space as an indicator. If the distance between two vectors is lower than a threshold (e.g. 0.2), they are also marked as possibly related.

We start in both methods with high thresholds, manually check the results and successively lower them until the false positive rate becomes too high. With this procedure we were able to quickly find all the duplicate pairs we already found manually before. Additionally we found two follow-up pairs among the 150 that we did not identify with the manual procedure. The bag of words approach is

somewhat more reliable and identified almost all the duplicates, before including false positives. However, we did not find two other duplicates pairs, that the string matching procedure quickly identified. So it seems worth the effort to use both methods.

Later we found one more follow-up pair by chance. However, these two letters are radically different in content and so this event does not undermine our confidence, that we were able to find a sufficient portion of the duplicate pairs to not distort our test results. (Will ich diesen letzten Absatz überhaupt drin haben?)

4.2 Paragraph Extraction

As already mentioned above, the letters almost always contain separate paragraphs like greeting, diagnosis, therapy history and anamnesis. To be able to hide unnecessary information or to only present requested information it would be useful to automatically extract individual paragraphs from the documents. As the documents are similar in structure and the word XML format allows to automatically examine the XML tree structure of a document easily, we use a rule based approach for extracting the individual paragraphs. A simplified rule to find the beginning of the diagnosis paragraph is shown in pseudocode:

```
diagnosisRegex = '[dD]iagnose(n)?'
text = thisXmlNode.text()
if regex.match(text, diagnosisRegex) and boldface(text) and
 precededByNewline(thisXmlNode) then
 | diagnosisStart = thisXmlNode
end
```

Algorithm 1: Simplified pseudocode algorithm to find the beginning of the diagnosis paragraph

```
diagnosis_regex = '[dD]iagnose(n)?'
text = this_xml_node.text()
if diagnosis_regex.match(text)
                and boldface(text)
                and preceded_by_newline(this_xml_node):
then:
        diagnosis_start = this_xml_node
```

(Weder der erste noch der zweite pseudocode gefällt mir bis jetzt. Wird noch geändert.)

With a set of rules like the one above we automatically extract the paragraphs of interest from the documents. This approach, however, is not very reliable, as the doctors are free to write the documents in the way they please. Indeed we find several

wrongly extracted paragraphs, that e.g. include the subsequent paragraph as well. For our dataset it is possible to check the extraction process by hand. However, this is tedious and error-prone work and is not doable for bigger datasets. We therefore explore whether we can in principle make use of other automated methods to find paragraphs for which the extraction process does not produce desired results. We therefore take the extracted diagnosis paragraphs and convert them to their bag of words representation. To get a feeling for how these vectors behave we use Principle Component Analysis to get a lower dimensional approximation, that is a linear subspace of the original space and keeps as much variance as possible. See figure 4.1 for a 2D PCA plot of the bag of words representation of one incorrectly extracted diagnose paragraph and some correctly extracted ones. As is apparent from the figure, it would not be a hard task to automatically detect the outlier. In this case the incorrectly extracted paragraph included not only the diagnosis, but also the therapy history. In cases like this with additional text present, it is an easy task to identify the incorrect ones. A harder problem arises, when only parts of the paragraph of interest have been extracted. However, we believe that this problem is of little concern. The way our rules are built it is very unlikely that we will face this problem. The paragraph would have to include an empty line, the subsequent one would have to contain only boldface characters and a few more conditions would have to be fulfilled for this problem to arise. Indeed, we did not find a case of this problem in our dataset. (Ich könnte auch mal tatsächliche outlier detection machen, wenn du das für sinnvoll hälst. Hab ein paar Ideen, die gut funktionieren könnten, wollte aber keine Zeit rein stecken, falls wir es nicht benutzen wollen.)

4.3 Paragraph Classification

In our sample dataset, we can automate paragraph extraction as shown above. We can also semi-automatically detect for which paragraphs the procedure produces incorrect results. This approach works well only because the documents in our dataset generally adhere to a rough structure. For datasets from other clinics constructing a rule based extraction procedure is not only time consuming, it might not be possible at all. We therefore test an approach to classification of extracted paragraphs into the respective categories - greeting, diagnosis and anamnesis. Our findings show that surprisingly this is not a hard problem. On unseen datasets it might therefore be possible to split text into unlabeled paragraphs with a basic rule based approach. One can possibly define a new paragraph to begin after a blank line and automatically label the resulting paragraphs with a predefined category. This way one would be able to hide or show specific information on demand even on datasets from other clinics.

We approach the problem again from a vector space based view-point. We first compute the vector representation for every paragraph with different text embedding methods. Then logistic regression is trained on a training portion of the dataset and the performance evaluated on a testing portion. We use leave-one-out cross-

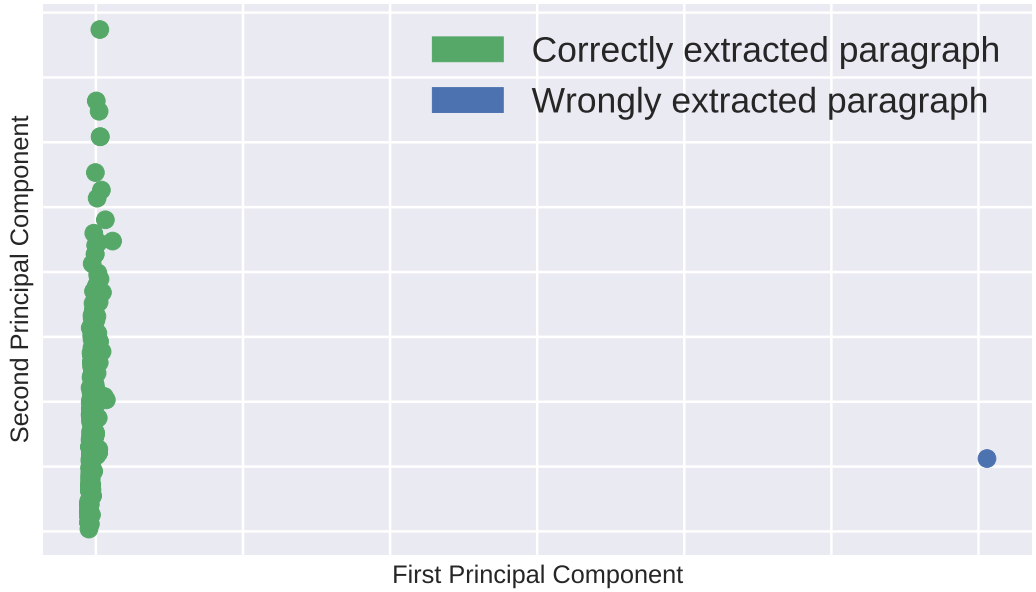

Figure 4.1: 2D PCA projection of bag of words representation of one incorrectly extracted diagnosis paragraph and several correctly extracted ones.

validation to obtain a good estimate of the performance even on our limited dataset.

As vector embedding methods we tested the standard bag of words, tf-idf and paragraph vector models. We also used LSA and LDA to get more condensed feature vector representations based on the tf-idf vector space. Results of our evaluation can be found in table 4.1 . Several things are noteworthy about the results. First it is surprisingly easy in general to use a small number of training paragraphs (less than 300 per category) to predict its label with very high accuracy. Second all methods are indeed outperformed by the more recent paragraph vector approach. However, the paragraph vector performance comes with the cost of needing to tune many hyperparameters, whose influence is not intuitively clear. Third LSA performance is always smaller or equal to tf-idf performance. As the tf-idf vector space has several thousand dimensions, but we only have several hundred texts, all these texts must fall into a linear subspace with dimension no greater than the number of texts. We assume the dimension is even substantially smaller, as LSA vectors produce the same results in classification accuracy when reducing the number of dimensions until 21. Reducing dimensionality further diminishes accuracy.

To gain a more intuitive understanding of the performance of these approaches we use PCA to get a 2D approximation of the vectors of the extracted paragraphs. In figure 4.2 one can compare the 2D PCA projections of the tf-idf and the paragraph vector models. While it is obvious that both methods can produce good results even just using a linear classifier, it is also easy to see that the paragraph vectors are easier separable (although not linearly separable in the 2D projection). We

Embedding Method	BOW	TF-IDF	LSA	LDA	Para2Vec
Classification Accuracy	0.995	0.997	0.997	0.992	1.0

Table 4.1: Mean classification accuracy of logistic regression with leave-one-out crossvalidation by vector embedding methods.

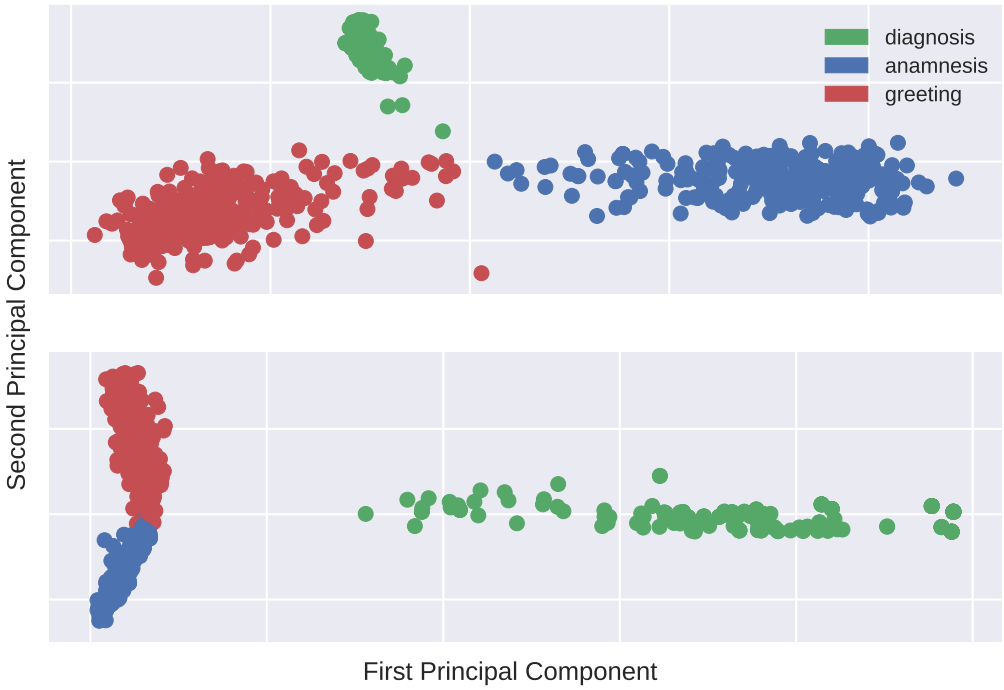

Figure 4.2: 2D PCA projections of a vector space embedding of the physician letter paragraphs. Colors encode the respective paragraph category for each vector. **Top:** Paragraph vector space. **Bottom:** Tf-idf vector space.

conclude that paragraph vector is the best suited method for this classification task and surprisingly performs well even with very limited training data, a finding not documented in the literature.

4.4 Disease Classification

We tried to automatically assign to each letter which disease the patient had. This is useful for two reasons. First, people might want to search for all patients with a particular disease (more useful than full text search as diseases can be written many ways.) Second, many interesting pieces of information are not coded into a structured database, but are "hidden" in the free text. With these methods we can find this information.

4.5 Letter Similarity

With the vector embedding methods we are able to get an estimate of the dissimilarity of texts simply through a vector distance. This way we are able to suggest letters that are similar to a reference letter. As it worked better than expected we conducted the experiment detailed below

Chapter 5

Experiment

Because doctors quickly gave us positive feedback to the algorithms usefulness, we conducted an experiment probing doctors intuitions about similarities between pairs of physician letters. We found that their ratings correlate strongly with the distance measures our algorithm provides for letter pairs.

Chapter 6

Prototype

Ich weiß ehrlich gesagt nicht genau, ob ich die Website als Prototypen hier überhaupt reinschreiben soll. Aber ich würde natürlich gerne dokumentieren, dass ich da was gemacht hab. Falls das also hier reingehört, soll ich dann einfach ein paar Screenshots machen und es kurz erklären oder wie kann so was aussehen?

Chapter 7

Conlusion / Discussion / Further Work

We showed that it is possible to automatically retrieve physician letters that are similar to a reference letter from a database. We evaluated that our automatic similarity ratings match well with expert intuitions of similarities.

Further work should address whether agreement between expert intuition and machine rating is high enough that a system like our prototype is useful in practice for doctors in a hospital (Vielleicht können wir das auch schon sagen). Such a system can then be deployed in the university hospital in Freiburg. If doctor's responses to the system are positive extensions of the system are possible. Deploying the system in another hospital or combining the databases of two hospitals are possible further paths from there. However, when letters are exchanged between clinics issues of anonymization will have to be addressed. In case letters have to be made anonymous before exchange the paragraph classification of this thesis can serve as a starting point for semi-automatic anonymization. Unnecessary parts of the letter (like the greeting) can be excluded. Thereby reducing the amount of work necessary for anonymization by quite a bit. Not only less text has to be read, but also the paragraphs containing most personal information are excluded this way.

Bibliography

Aamodt, A. and Plaza, E. (1994), 'Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches', *AI COMMUNICATIONS* **7**(1), 39–59.

Begum, S., Ahmed, M. U., Funk, P., Xiong, N. and Folke, M. (2011), 'Case-Based Reasoning Systems in the Health Sciences: A Survey of Recent Trends and Developments', *IEEE Transactions on Systems, Man, and Cybernetics* **41**(4), 421–434.

Bengio, Y., Vincent, P. and Jauvin, C. (2003), 'A Neural Probabilistic Language Model', *Machine Learning Research* **3**, 1137–1155.

Blei, D. M., Ng, A. Y. and Jordan, M. I. (2003), 'Latent Dirichlet Allocation', *Machine Learning Research* **3**, 993–1022.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K. and Harshman, R. (1990), 'Indexing by Latent Semantic Analysis', *American Society for Information Science* **41**(6), 391–407.

Dib, E. G., Kidd, M. R. and Saltman, D. C. (2008), 'Case reports and the fight against cancer', *Journal of Medical Case Reports* **2**(1), 39.

Elstein, A. S. and Schwarz, A. (2002), 'Clinical problem solving and diagnostic decision making: selective review of the cognitive literature', *British Medical Journal* **324**(March), 729–732.

Gutmann, M. U. and Hyvärinen, A. (2012), 'Noise-Contrastive Estimation of Unnormalized Statistical Models, with Applications to Natural Image Statistics', *Machine Learning Research* **13**, 307–361.

Kidd, M. and Hubbard, C. (2007), 'Introducing Journal of Medical Case Reports', *Journal of Medical Case Reports* **1**, 1.

Klein, G. (2008), 'Naturalistic Decision Making', *Human Factors: The Journal of the Human Factors and Ergonomics Society* .

Kolodner, J. L. and Kolodner, R. M. (1987), 'Using Experience in Clinical Problem Solving: Introduction and Framework', *IEEE Transactions on Systems, Man, and Cybernetics* **17**(3).

Le, Q. and Mikolov, T. (2014), Distributed Representations of Sentences and Documents, *in* 'Proceedings of the International Conference on Machine Learning', Vol. 32, pp. 1188–1196.

Manning, C. D., Raghavan, P. and Schütze, H. (2008*a*), Scoring, term weighting and the vector space model, *in* 'Introduction to Information Retrieval', Cambridge University Press, chapter 6, pp. 117–120.

Manning, C. D., Raghavan, P. and Schütze, H. (2008*b*), The term vocabulary and postings lists, *in* 'Introduction to Information Retrieval', Cambridge University Press, chapter 2, pp. 22–27, 32–34.

Mason, R. A. (2001), 'The case report - an endangered species?', *Anaesthesia* **56**(2), 99–102.

Medin, D. L. and Schaffer, M. M. (1978), 'Context Theory of Classification Learning', *Psychological Review* **85**(3), 207–238.

Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013), Efficient Estimation of Word Representations in Vector Space, *in* 'Proceedings of Workshop at the International Conference on Learning Representations'.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. and Dean, J. (2013), Distributed Representations of Words and Phrases and their Compositionality, *in* 'Advances in Neural Information Processing Systems', pp. 3111–3119.

Morin, F. and Bengio, Y. (2005), Hierarchical Probabilistic Neural Network Language Model, *in* 'Proceedings of the international workshop on artificial intelligence and statistics', pp. 246–252.

Nissen, T. and Wynn, R. (2014), 'The clinical case report: a review of its merits and limitations', *BioMedCentral Research Notes* **7**(1), 264.

Sandu, N., Chowdhury, T. and Schaller, B. J. (2016), 'How to apply case reports in clinical practice using surrogate models via example of the trigeminocardiac reflex', *Journal of Medical Case Reports* **10**(1), 84.
URL: *http://dx.doi.org/10.1186/s13256-016-0849-z*

Spadaro, S. (2012), ClinicOn Kurzbeschreibung für neue und interessierte Anwender, Technical report.

Williams, D. D. R. (2003), 'In defence of the case report', *The Royal College of Psychiatrists* **184**, 84–88.

Declaration of Authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

Osnabrück, May 8, 2017