Institute of Cognitive Science

Bachelor's Thesis

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

Philipp Hummel

June 20, 2017

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

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

When faced with exceptional clinical cases, doctors would often like to review information about similar patients to guide their decision-making for the current one. Retrieving 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 specified reference letter with information retrieval methods. 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 its performance through a psychological experiment assessing physicians' similarity judgments of those letters. We find that the similarity measure of our recommender system correlates strongly with doctors' similarity judgments. Additionally, we explore other text mining applications like searching for letters more intelligently, based on this dataset.

Acknowledgements

I would like to thank several people and acknowledge their contributions to this thesis. First and foremost I want to thank Dr. Frank Jäkel for his guidance, his expertise in cognitive science, psychology and machine learning, and his well thought-out remarks in general. Thanks also to Dr. Sascha Lange and the company PSIORI for making this thesis possible by selecting me for this project, providing the objective for the thesis, and sharing their contacts and resources.

Special thanks go to Prof. Dr. Roland Mertelsmann who envisioned this project for years, gave many fruitful comments, established the connection to the University hospital Freiburg and motivated several other physicians and students to contribute to this project among other things by participating in the experiment.

Lastly, I want to thank Astrid Feiten, Judith Schepers and Lisa Dieminger for their proof-reading, which made this thesis much more readable.

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 undergone 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 exclusively to case reports have emerged (Kidd and Hubbard, 2007)[1] and many people have argued for the value of case reports for research itself but also beyond (Williams, 2004; Dib et al., 2008; Sandu et al., 2016). Specifically, 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 physicians think that reviewing similar cases helps them in their work, we will argue that this can indeed improve their medical problem-solving.

Cognitive scientists have discussed the usefulness of examples for reasoning processes for a long time 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

[1] Additional Case Journals can be found at:
Journal of Medical Cases: http://www.journalmc.org/index.php/JMC/index
British Medical Journal Case Reports: http://casereports.bmj.com/site/about/index.xhtml
American Journal of Medical Case Reports: http://www.sciepub.com/journal/AJMCR

memory, experts apply the solution from the recalled situation to the current problem 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 doctors' decision-making in an actual clinical setting.

Research in cognitive science has also directly addressed the medical domain. 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 with 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 humans and machines can learn from specific examples and use them to reason about new problems.

Around the idea that artificial systems might learn from examples a whole branch of Artificial Intelligence (AI) has evolved, which is called "Case-based reasoning". This domain has been strongly influenced by psychological findings, some of them mentioned above. Researchers have 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 the medical area does not only offer straightforward usage of examples, but also has a need for automated 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 especially well-suited source for the retrieval of patient cases are databases of physician letters, as these letters provide concise summaries of the specifics of a patient 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 prototypical recommender system on those physician letters to do automatic retrieval of only the relevant documents from a database.

1.2 Dataset

To obtain a dataset for exploratory work on the recommender system we collaborated with the University Hospital Freiburg. The hematology and oncology department

has a database of approximately 190,000 German physician letters in PDF form (Spadaro, 2012). 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. An example of such a letter is shown in the appendix in Figures A.1 and A.2. The letters generally follow a rough structure. Almost all of them include a letterhead (a greeting and introduction), a diagnosis (summarizing diagnosed diseases bullet point like), a therapy history (listing the prior 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, which is added to the database. We refer to these letters as "follow-up" letters.

To get permission to use a subset of those letters for our experiment, it was necessary to ensure that all personal information was removed from them. A medical student of the university hospital was therefore paid to manually anonymize 307 of the letters and forward them to us. The letters were given to us in Microsoft Word XML format.

1.3 Use Case Scenario

The recommender we envision is built into the clinical everyday life of doctors. During the visit of a patient, physicians write a new or modify an existing letter for this patient. Already in this writing phase, we wish to retrieve letters of similar patients and present them on demand. Thereby, doctors' decision-making processes can be guided by these similar cases if the doctor deems this necessary. The perceived suitability of the retrieved letters has to be exceptionally high. Physicians' additional time resources are usually very limited. Therefore the recommender system will only be used in practice if almost all recommendations are considered useful.

With the dataset mentioned above, we set out to build this kind of recommender system. We first clear our dataset from duplicates with the help of basic information retrieval methods. This process is described in the subsequent chapter. We elaborate on methods for hiding and showing specific information from the letters in chapter 3. For this goal, we make extensive use of more sophisticated information retrieval methods and introduce them alongside there. Chapter 4 describes the recommender system itself, the experiment we conducted to assess its quality, and the evaluation of the obtained data. In chapter 5 we describe a related use case in a research setting, develop an automated aid for it and shortly evaluate how useful the developed system can be in practice. Finally, we review shortcomings, conceivable problems, and possibilities for further work in the discussion.

Chapter 2

Dataset Cleansing

The dataset we acquired from the university hospital contains several duplicate letters. To ensure undistorted test results, we have to identify as many of them as possible. It is clear that finding all of these duplicates manually is not only error-prone, but also very time consuming as in principle $\frac{(n-1)^2+(n-1)}{2}$ letter comparison have to be made. With $n = 307$ this amounts to almost 50,000 comparisons. Personal information, like names and dates of birth, would be helpful for this task, but has been removed during anonymization. Therefore we make use of two semi-automatic procedures for duplicate identification. First, we use a longest common subsequence matching method. This method would be sufficient for our problem if we only had to deal with exact duplicates. However, we face the issue of follow-up letters with modified information in our dataset. To overcome this problem we use a procedure well known in the information retrieval community: The bag of words model for representing texts as vectors, with which it is possible to compute distances between documents. Duplicates are then found by their vector proximity.

2.1 The Bag of Words Model

The bag of words model represents a text as the multiset (bag) of its words. This means that all word order information is disregarded and only the information how often a word is present in a text is encoded (Manning et al., 2008*b*). More concretely, in the bag of words model documents are represented as fixed length feature vectors, in which every feature is a word occurrence count. In the simplest approach, every feature is the word or term frequency $\mathrm{tf}(t, d)$ of term t in document d. Where $\mathrm{tf}(t, d)$ is the occurrence count of word or term 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 $\mathrm{tf}(t, d)$ for all $t \in V$. To represent text in the bag of words model, however, the text needs to be preprocessed first.

Preprocessing

In computers, text is most often represented as sequences of characters. The process of extracting words from such a sequence is known as tokenization (Manning et al., 2008*c*). 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 "The doctor treats the patient.". The naive approach yields the words "The", "doctor", "treats", "the" and "patient.". 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.

Some words are more important or representative of a text than others, and so it is a useful preprocessing step for the bag of words model to remove some frequently appearing but uninformative words, so-called stop words. A list of English stop words comprises 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 (Manning et al., 2008*c*).

Additionally, for the bag of words model, we are not interested in the exact grammatical form in which a word is present in a text. Removing this unnecessary information is achieved with a procedure called stemming, that maps word appearances to their stem (Manning et al., 2008*c*). It maps words such as "doctors" and "doctor" both to their common stem "doctor". Both removal of stop words and stemming seem at first glance to disregard information. This is true, but they allow to represent a text more compactly and thereby reduce the noise of the representation. We explain those preprocessing steps and the bag of words model with a more illustrative example.

Bag of Words Example

Assume the corpus consists of two documents d_1 = "The patients with disease A." and d_2 = "The patients with disease B.". After tokenization, removal of stop words, and stemming, the bag of words vectors representing the two texts are:

$$\mathbf{v}_{d_1} = \begin{matrix} 1 \\ 1 \\ 1 \\ 0 \end{matrix} \quad \mathbf{v}_{d_2} = \begin{matrix} 1 \\ 1 \\ 0 \\ 1 \end{matrix} \quad \begin{matrix} \text{patient} \\ \text{disease} \\ \text{A} \\ \text{B} \end{matrix}$$

Note how the punctuation character does not appear, as the tokenization has removed it from the list of words. The words "the" and "with" have been removed as stop words and the word "patients" has been stemmed to produce "patient". Then the list of words has been converted to a bag of words vector.

Application and shortcomings

The vectors $\mathbf{v}_{d_1}$ and $\mathbf{v}_{d_2}$ live in a four-dimensional vector space and we can either use them as feature vectors for a machine learning task or compute a measure of similarity between them. The standard similarity measure used for bag of words vectors is the cosine similarity $s_{\cos}(\mathbf{v}_1, \mathbf{v}_2) = \frac{\langle \mathbf{v}_1, \mathbf{v}_2 \rangle}{||\mathbf{v}_1|| \cdot ||\mathbf{v}_2||}$, where $\langle \mathbf{v}_1, \mathbf{v}_2 \rangle$ is the scalar product of $\mathbf{v}_1$ and $\mathbf{v}_2$ and $||\mathbf{v}||$ is the norm or length of vector $\mathbf{v}$. The cosine similarity has the appealing property that it normalizes the scalar product by the norm of the vectors used. Thereby it ensures, that the resulting measure lies in the interval of $[-1, 1]$. For positive spaces (i.e. spaces, where all vectors have only non-negative elements, like the bag of words vector space) this measure lies in the interval $[0, 1]$. Two documents d_1 and d_2 are then considered to be more similar than d_3 and d_4, if $s_{\cos}(\mathbf{v}_{d_1}, \mathbf{v}_{d_2}) > s_{\cos}(\mathbf{v}_{d_3}, \mathbf{v}_{d_4})$.

Generally, however, the bag of words model has severe limitations. Consider for example the two sentences d_3 = "The patients with disease A, but not B" and d_4 = "The patients with disease B, but not A". Their representation in the bag of words model is equivalent, although they express quite different meanings. Additionally, some phrases that express similar or identical meaning are treated very differently in the bag of words model. Consider the example disease word "Chronic-lymphocytic-leukemia". Many doctors will abbreviate this word to "CLL" or "B-CLL". The bag of words model regards these three forms of the same word as completely dissimilar. Even worse, when spelled "Chronic lymphocytic leukemia", the phrase is regarded as three distinct words, having nothing in common with each other or the words above. Still the bag of words model is a standard approach for information retrieval problems, as it has many desirable properties. Texts can be straight-forwardly embedded in a vector space (i.e. represented as vectors) and these embeddings can be used as fixed length feature representations for machine learning tasks.

2.2 Duplicate Detection

With the bag of words model and the longest common subsequence matching method, we semi-automatically identify duplicates in our data. We had a senior oncologist identify all duplicates present in a subset of 150 of these letters manually to have a baseline to which we can compare the methods' performances.

The longest common subsequence matching procedure scans two documents and finds the longest sequence of characters they have in common. If the length of this sequence exceeds a threshold the pair is marked as possibly duplicate and manually inspected. To get more hits we lower the threshold until the false positive rate becomes too high. Secondly, we preprocess all texts as described above and map them into the bag of words representation. We use the cosine similarity as a measure of relatedness between pairs and mark all pairs with distance lower than a threshold as possibly duplicate and again iteratively decrease the threshold.

The bag of words approach identifies all manually detected duplicates and follow-ups while having a very low false positive rate. It detects one additional manually

undetected follow-up pair. The string matching procedure is not as useful due to a high false positive rate and does not detect all manually found duplicates. Its performance is worse than the performance of the bag of words procedure, especially for follow-up letters. However, it finds a second manually undetected follow-up pair, which we have not identified with the bag of words procedure either. We use both approaches with thresholds from the "training" set on the remaining 157 letters as well. Overall we detect 18 exact copy pairs and 17 follow-up pairs. We additionally remove three letters from the dataset as they include almost no information (an artifact of the specific documentation procedure at the university hospital). This results in our final dataset consisting of 269 unique physician letters (follow-ups are only included, where mentioned explicitly).

Chapter 3

Fine-grained information presentation

Many physician letters are rather long documents, spanning several pages. To increase the usability of a recommender system used in practice, it is desirable to be able to show specific information like the diagnosis or the therapy history on demand or hide unnecessary parts of the letters like the introduction. It might also be desirable to compare letters only based on a specific section like therapy history. A first step towards this goal is the automatic extraction of the relevant paragraphs.

3.1 Paragraph Extraction

The XML data format of the letters allows automatized inspection of rather fine-grained structures present in the letters. One can automatically determine boldfaced characters for example. Because of this and because the documents are similar in structure, 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
```

A regular expression is defined that matches the word "Diagnose", which is the German word for diagnosis. The text of every node in the XML tree is checked for a regular expression match and whether it is boldfaced. If an XML node matches those criteria and is preceded by a newline, it is marked as the beginning of the diagnosis paragraph. With a set of rules like the one above we automatically extract

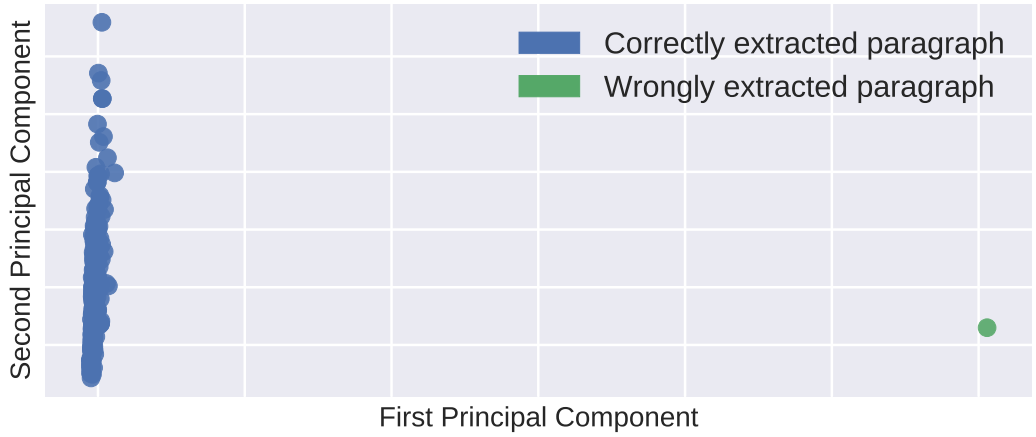

Figure 3.1: 2D PCA projection of bag of words representation of one incorrectly extracted diagnosis paragraph and several correctly extracted ones.

the paragraphs of interest from the documents. This approach, however, is not completely reliable, as the doctors are free to write the documents in the way they please. Indeed we find several wrongly extracted paragraphs, which 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 work and is not scalable to bigger datasets. We, therefore, explore whether we can in principle make use of other automated methods to find paragraphs for which the rule-based extraction process does not produce desired results. We, therefore, take several correctly extracted and one incorrectly 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 (PCA) to get a lower dimensional approximation of the vectors. PCA finds the linear subspace with desired dimensionality of the original space that preserves as much of the variance of the vectors in the original space as possible. Thereby one can gain a low dimensional approximation of the high dimensional data and use this approximation for visual inspection. See Figure 3.1 for a 2D PCA plot of the bag of words representation of the correctly and incorrectly extracted diagnosis paragraph. As is apparent from the figure, it would not be a hard task to detect the outlier automatically. In this case the incorrectly extracted paragraph includes 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, due to the way our rules are made. Indeed, we did not find a case of this problem in our dataset.

3.2 Paragraph Classification

In our sample dataset, we can automate paragraph extraction as shown above. We can also detect for which paragraphs the procedure yields 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 classifying the extracted paragraphs into three 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 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 viewpoint. We compute the vector representation for every paragraph and use a classifier trained on these representations to predict the correct paragraph label. To get the best results, we compare different text embedding methods. We test the standard bag of words, term frequency—inverse document frequency (tf-idf) and paragraph vector models. We also use Latent Semantic Analysis (LSA) and Latent Dirichlet Allocation (LDA) to get more condensed feature vector representations based on the tf-idf vector space. Before presenting the results of the different approaches, we subsequently introduce these embedding methods.

Additional vector space models:

Term Frequency—Inverse Document Frequency

An extension of the bag of words model that creates feature vectors based on term frequencies $\mathrm{tf}(t, d)$ of term t in document d , is the term frequency—inverse document frequency model (tf-idf). A common problem with the bag of words model is that some terms (even after filtering stop words) often appear 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 often appearing in a few texts but rarely in others are good discriminative features for those texts. tf-idf refers to a specific scaling scheme, that downscales the importance of frequent words, while upscaling the importance of rare words.

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

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

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 $\text{tfidf}(t, d, C)$ is then calculated as

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

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*b*).

Latent Semantic Analysis

An often occurring machine learning (ML) problem is that very high dimensional feature vectors, as occurring in bag of words and tf-idf models, tend to generalize poorly in subsequent tasks like classification. In those cases, 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 $\mathbf{M}[i, j] = \text{tf}(\text{t}_\text{i}, \text{d}_\text{j})$, i.e. row i represents the relation of term i to all documents, while column j represents one document and the relation to all its terms. So column j represents document d_j's vector representation $\mathbf{v}_{d_j}$.

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

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. (1990) 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 be used not only with the bag of words space as a basis, but also with the tf-idf space. Throughout this thesis, we use LSA in the latter way.

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. More specifically that means that documents in LDA space are represented by vectors of dimensionality k. The elements of one vector then represent the strength of association of the document with the respective topic. 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).

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 $2c$ predictions for every center word considered. The model parameters are updated with gradient descent. After training, either only the input word vectors or the input and output word vectors are used. In the latter case, they 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 can train on a vast amount of text data (billions of words) in a matter of hours. Mainly through those speed ups (and the concomitant larger amounts of processable data) Word2Vec can produce better word vectors, than previously possible.

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, which 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).

Classification Results

After having introduced the additional embedding methods tf-idf, LSA, LDA and paragraph vector, we now use their embeddings for classification of the paragraphs into the respective categories. Therefore we take all the extracted greeting, diagnosis, and anamnesis paragraphs and map them to one of the embedding representations. Then logistic regression is trained on a training portion of the dataset and the performance evaluated on a testing portion. As our dataset is relatively small, we use leave-one-out cross-validation to be able to use every data point as test data. Thereby we obtain as much information about the performance as is possible with such limited data.

As vector embedding methods we test all methods described above. For LSA and LDA we tune the hyperparameter of dimensionality k to obtain best results. The paragraph vector method has several rather unintuitive hyperparameters, which need tuning. We start with hyperparameters reported as working well in the literature (Lau and Baldwin, 2016). From there we use a randomized search strategy to find better fitting parameters for our specific problem. Results of our evaluation can be found in Table 3.1. Several aspects of the results are noteworthy. 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 of this subspace is even substantially smaller, as LSA vectors produce the same results in classification accuracy when reducing the number of dimensions even much further.

Embedding Method	BOW	TF-IDF	LSA	LDA	Para2Vec
Classification Accuracy	0.995	0.997	0.997	0.992	<u>1.0</u>

Table 3.1: Mean paragraph classification accuracy of logistic regression measured with leave-one-out cross-validation for different vector embedding methods.

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 3.2 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

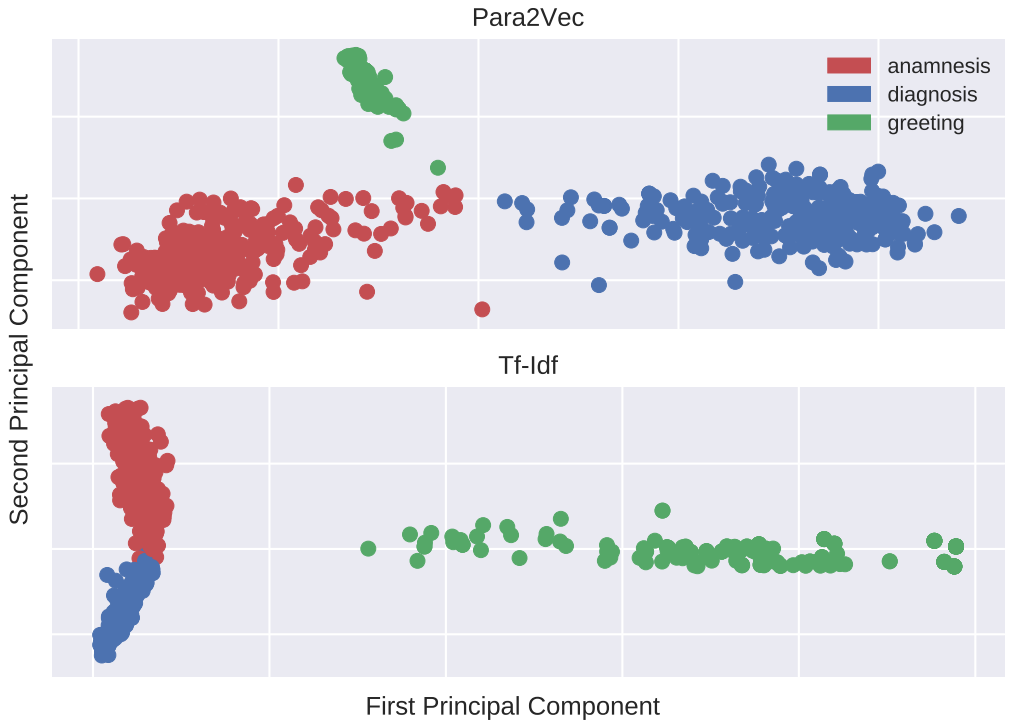

Figure 3.2: 2D PCA projections of a vector space embedding of the physician letter paragraphs. Colors encode the respective paragraph category for each vector. **Top:** Paragraph vector space. **Bottom:** Tf-idf vector space.

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 conclude that paragraph vector is the best-suited method for this classification task. Additionally, we note that it works surprisingly well with very limited training data.

Chapter 4

Recommender System

The main objective of this thesis is to build a prototypical recommender system for the physician letters and assess its quality. We do so based on the document embedding methods introduced in chapters 2 and 3. Once documents are embedded in a vector space, similarities between two documents d_1 and d_2 can be computed as the cosine similarity of the corresponding vectors $\text{sim}_{\cos}(\mathbf{v}_{d_1}, \mathbf{v}_{d_2})$. To find the most relevant letters, given a reference letter, we compute the similarity of the reference letter to all other letters in the database. Thereby we get a ranking of all other letters, given the reference letter. The "best" fitting letter, i.e. the letter with rank one, is considered the most relevant according to the algorithm. The n most relevant letters are the ones on the n highest ranks or the n letters with highest cosine similarity to the current reference letter. These letters can then be retrieved from the database and presented to a user.

4.1 Fine-tuning

Based on the cosine similarity between vectors of the corresponding texts, all document embedding methods can in principle be used as a base for the recommender system. To identify which hyperparameters and which embedding method is most suitable for our task, we use supervised similarity information. For this, the most desirable information would be an expert rating of similarity between letter pairs. However, this information is hard to come by because its generation is time-consuming and requires a medical expert. We therefore draw on a different, more easily acquirable dataset for the task of fine-tuning. Instead of a pairwise similarity rating, 135 of the letters were categorized into 50 non-overlapping groups of similar patients by a medical expert. Some groups only contain a single patient (if he or she was dissimilar to all others), some contain several and the average group consists of 2.7 patients. This grouping is not equivalent to a correct measure of similarity. Nevertheless, it can be used as an approximate measure of similarity to tune our algorithm by considering letters from the same group as similar and letters from different groups as dissimilar. One measure of goodness for a real recommender sys-

Embedding Method	BOW	TF-IDF	LSA	LDA	Para2Vec
Continuous Measure Score	0.794	0.870	0.868	0.634	0.830

Table 4.1: Performance of different embedding methods (with tuned hyperparameters) on the grouping dataset evaluated with the continuous measure.

tem is how often the system ranks the "truly" similar letters into the top n, where truly refers to the patient groups defined by the expert. For a good performance, the recommender system should, given a reference letter, rank these truly similar letters among the top n most relevant letters. We call this the top-n measure. This measure, however, does not take all available information into account. For instance, if $n = 5$, the top-n measure returns the same score irrespective of whether a truly similar letter is ranked as the 6th most similar or the least similar to a reference letter. A measure that assigns a higher score in the first situation than in the second would be preferable for the fine-tuning of the algorithm. We therefore develop a continuous measure that assigns a score from the interval $[0, 1]$ for all possible ranking situations. A score of 1 is given if the truly similar letters occupy the foremost positions, a score of 0 if the truly similar letters occupy the last positions, a score of 0.5 if the truly similars occupy the positions in the center. A score of 0.5 is expected if the algorithm sorted the letters by chance.

Based on the continuous measure, we start with doing hyperparameter tuning for the embedding methods as applicable. Afterwards, we select the best performing embedding method in the same manner. Table 4.1 shows the scores obtained by each embedding method when testing them with the continuous measure. The performance of all methods is high above chance level. It is noteworthy that the simple and hyperparameterless tf-idf method outperforms all other methods including LDA and the recent and hard-to-tune paragraph vector. We, therefore, choose tf-idf as the embedding method for the recommender system.

There is a relatively intuitive reason why tf-idf works well for our problem. It is plausible that the main features for human similarity judgments of letter pairs are based on the diagnosis of the patients. The diagnosis and diseases words are features that the tf-idf method will judge as very important, as they appear only in few documents overall.

4.2 Recommendation Quality—Experimental Setup

Having fine-tuned the recommendation procedure as described above, the next step consists of assessing the quality of the recommendation. We test this quality in a psychological experiment. To this end, we probe the similarity ratings that medically trained subjects give to pairs of letters and compare them to the similarity measure of our algorithm.

We construct an experiment with 32 trials, in which subjects have to compare letter pairs for similarity. More precisely, we select 32 letters as "reference letters"

out of our database and asked subjects to rate the similarity between the reference letters and five other letters each. Thus, we collect ratings for 160 letter pairs. Half of the reference letters have a follow-up letter in our database. Trials with this kind of reference letter are called "follow-up trials" and letter pairs of follow-ups are called "follow-up pairs". The remaining 16 reference letters are selected randomly among the letters without follow-ups present in our dataset. The five letters that are compared to one reference letter are called the comparison letters. Four of them are selected based on the cosine similarity between the reference letter and all other letters. These four are the ones with highest cosine similarity to the reference letter according to our algorithm. That means, they are the best matches to the reference letter according to our algorithm—or equivalently, they are ranked on places one to four given a reference letter. The fifth comparison letter is randomly selected from all other letters and then fixed. Subjects are presented with one reference and one comparison letter at a time. After rating the similarity of the pair they are presented with the next comparison letter. One trial consists of the comparison of one reference letter to all of its five comparison letters. Once a trial is done, the next reference letter and the accompanying comparison letters are presented. The order of the reference and comparison letters is random, but fixed. Subjects are forced to give a rating in the range of 1 (very dissimilar) to 7 (very similar) for each letter pair. See Figures A.3, A.4, and A.5 (in the Appendix) for screenshots of the website on which the subjects perform the experiment.

We design the experiment such that the first trial is a follow-up trial and the second one is a non-follow-up trial. Thereby, we ensure that subjects can adjust for the upper similarity bound of having to compare letters of the same patient. These two trials are excluded from the later analysis. We have six subjects performing the experiment, four experts (oncologists with at least five years of practical experience) and two novices (medical students more than halfway through their study course).

4.3 Recommendation Quality—Results

Inter-Rater Agreement

We first analyze the inter-rater agreement among subjects. We believe it is not trivial for them to rate physician letter pairs for similarity as it is not directly clear along which dimension similarity is to be judged, a problem well known from the cognitive science literature (Medin et al., 1993). One might for example judge similarity based on diagnosis or based on therapy. We also believe that experts and novices (or students) base their judgments on different features. Experts generally have higher agreement when categorizing stimuli, than novices do, and usually use more abstract, less accessible features (Chi et al., 1981; Linhares and Brum, 2007; León-Villagrá and Jäkel, 2013). In line with these results, we find that the inter-rater agreement is higher among the experts than among the students (expert agreement: 0.76; student agreement: 0.59). We measure this agreement with the Spearman rank correlation coefficient, which produces a number between -1 (perfect anti-correlation)

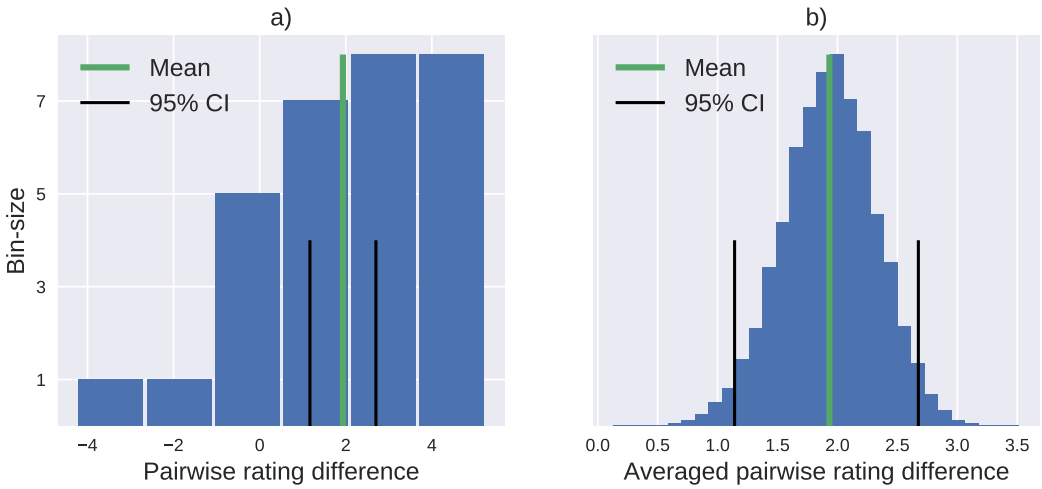

Figure 4.1: (a): Histogram of differences in rating of the "best" and the random comparison letter for each trial. **(b):** Histogram of the average differences in averaged subject ratings of the two groups for 100.000 bootstrap resampled datasets.

and 1 (perfect correlation). Spearman's coefficient is used instead of the often used Cohen's kappa coefficient, because the former can deal with ordinal data, whereas the latter can only deal with nominal data (Spearman, 1904; Cohen, 1960). Our number of subjects is quite low for concluding that students rate letter pairs worse than experts. Still, as our findings are in line with well-established research, we discard the student rating data for further analysis.

Note that for the following analyses we furthermore exclude data of follow-up pairs, except where explicitly stated otherwise. Subjects rate the similarity of these pairs as very high and almost any information retrieval system will find them to be similar. Including them would improve positive correlation statistics in our analysis, although retrieving them is useless in practice. We will show later that our recommender can easily distinguish them from non-follow-up pairs.

Ranking and Subject Ratings

The first step of assessing the quality of our recommender system is to evaluate whether the recommendations are better than chance. For this we compare the average subject rating of the "best fitting" letter (as computed with our algorithm) and the random comparison letter for each trial. The best fitting comparison letter is the one with the highest cosine similarity to the reference letter. Figure 4.1a shows a histogram of the differences in subject rating of the best fitting and the randomly selected letter pairs. The figure also shows the mean difference and a 95% interval for this mean. The confidence interval was calculated using the Standard Error of the Mean (SEM). The lower and upper bound is then calculated as $bound_{1/2} = mean \pm 1.96 \cdot SEM$ respectively. This estimation of the confidence interval, however, relies

on the assumption that sample means are normally distributed. As those means are calculated over 30 values in each sample, this assumption is hardly questionable in most scenarios due to the central limit theorem. Still, as the differences in subject ratings are discrete values bounded between -6 and +6, we might face the issue of a skewed distribution because of bounding problems. Therefore we verify the assumption of normality with bootstrapping (Efron, 1979).

Bootstrapping is a procedure in which many random resamples of the dataset are computed. If our collected dataset represents the true distribution of differences in ratings of best fitting and random letter pairs well enough, then the resamples are to our sample as our sample is to the true distribution. Thus, the mean of the average difference of the resampled datasets gives an estimate of the average differences, when collecting many samples from the real distribution. We take 100.000 resamples with replacement from our data and compute the average difference of the resamples for every one of them. The distribution of these average differences of the resamples can be seen in Figure 4.1b. From this figure it is apparent that the assumption of normality for the distribution of average differences indeed holds. The figure also shows bootstrap estimates for the mean and the 95% confidence interval. As expected these estimations match closely with the ones estimated with the standard error of the mean as can be seen from the following table.

Estimated Statistic	Mean	Confidence Interval
Standard Estimation	1.93	[1.17, 2.70]
Bootstrap Estimation	1.93	[1.14, 2.68]

Cosine Similarity and Subject Ratings

After assessing whether ranking letter pairs with our algorithm works better than chance, we analyze the relationship of the cosine similarity and the average subject ratings of letter pairs more directly. We visualize the mean cosine similarity, which our algorithm assigns to pairs, as a function of the subject rating in Figure 4.2. The data shows a positive, close to linear correlation (Pearson correlation: 0.96) of those two variables, suggesting that our recommendation method captures at least some aspects of perceived similarity.

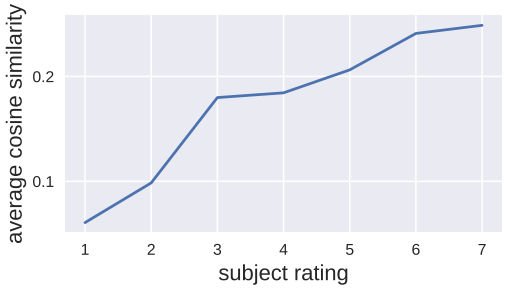

Figure 4.2: Averaged cosine similarity of all letter pairs that were rated into one of the seven possible rating categories.

Next, we analyze the relationship of the cosine similarity and the subject ratings more thoroughly. In Figure 4.3 all letter pairs are visualized as points in a plot of average rating versus cosine similarity of a pair. Follow-up pairs are marked green

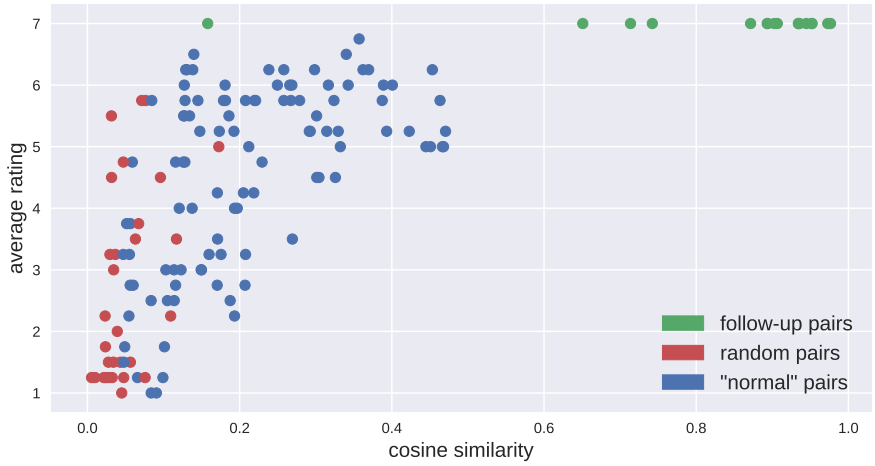

Figure 4.3: Average subject rating versus cosine similarity of all letter pairs.

and the random comparison letter pairs red. The figure shows that follow-up letters can be easily distinguished with the cosine similarity from other letter pairs. The randomly selected pairs mostly have a low cosine similarity as expected, however several of them are still rated rather highly by subjects. Some well fitting random letter pairs are expected by chance and we adopt this conclusion after manual inspection of these cases.

Considerations of Precision and Recall

A standard approach to assess the quality of an information retrieval system uses two measures called precision and recall. Precision and recall are employed as indicators of the performance of an information retrieval system, when all documents are binarily classified as either relevant or irrelevant given a current information need or query (Manning et al., 2008*a*). In our case, a query is a reference letter and we analyze whether a comparison letter is relevant or irrelevant given this reference letter. Precision is then a measure of the correctness of the retrieved results. In probabilistic terms, it is the probability that a document is relevant given that it is retrieved $\mathrm{P}(relevant|retrieved)$. Recall is a measure of completeness of the retrieved results and can be expressed as the probability of being retrieved given that the document is relevant $\mathrm{P}(retrieved|relevant)$. Both quantities usually exhibit an inverse relationship. One can trade higher precision for lower recall or vice versa. Both precision and recall always lie in the interval $[0, 1]$. Note that we classify letter pairs with an average rating of five or higher as relevant and others as irrelevant. This threshold is somewhat arbitrary but was set after several personal discussions with the contributing physicians.

We can assess the precision of our system in different scenarios. Recall, however,

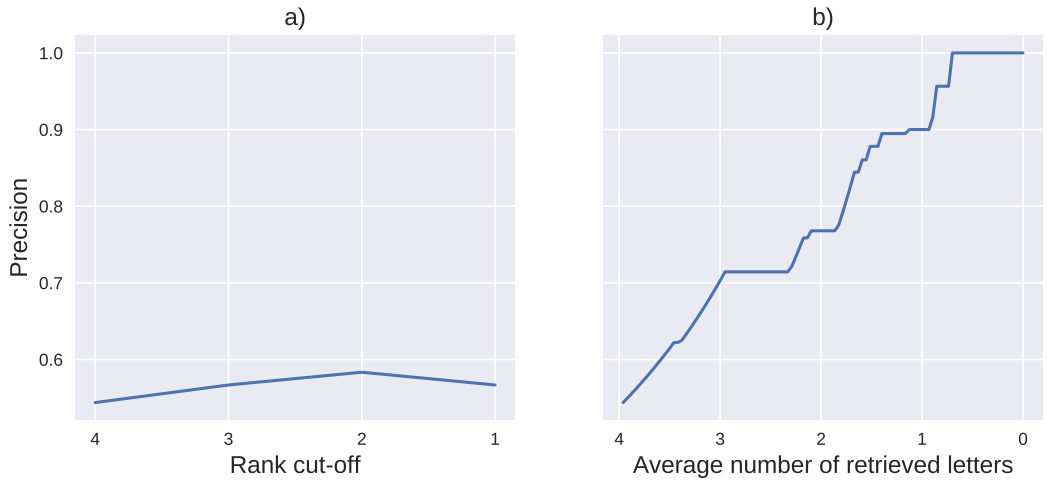

Figure 4.4: (a): Precision as a function of rank cut-off. **(b):** Precision as function of a varying cosine similarity threshold. The x-axis shows the average number of retrieved letters per reference letter at the threshold instead of the threshold itself.
Note that both plots start at the same point, where all letters with rank four or lower are retrieved, and share a common y-axis.

is impossible for us to measure. This is because we do not know the set of relevant letters for a given information need (i.e. reference letter) and therefore cannot compute P($retrieved|relevant$) (i.e. recall). Precision is a quantity of high interest for the recommender system in the use case we envision. For applicability and acceptance among possible users in a clinical everyday life, it is crucial that a large fraction of recommended letters is deemed useful by doctors. If this was not the case, physicians would quickly stop using the system. Recall on the other hand is not an interesting quantity in this situation. Doctors will only look at very few recommended letters such that it is of no concern whether or not a large fraction of relevant documents is retrieved. However, an interesting question, somewhat related to recall, is whether or not the most relevant letters are retrieved first. We will address this question after examining the precision of our system. It is important to remark that for another use case recall might be more important than precision.

In our usage scenario, a possible implementation of the system could always retrieve the four highest ranked letters to a given reference letter. Thus, it is worth investigating what precision we can achieve, when recommending the "best" four or even fewer letters. Figure 4.4a shows the level of precision that the system achieves when stopping retrieval of letters at a given rank. As shown, the retrieval of all four "best" matches results in a rather low level of precision (precision = 0.55). Unfortunately, reducing the number of letters to be retrieved does not significantly increase precision, with a maximal precision of 0.59 when presenting the two letters with highest cosine similarity. This means that little more than every second retrieved letter is relevant. From Figure 4.3 it is apparent, however, that we can do better,

when incorporating information about the absolute cosine similarity and not only the ranking of the letters. One can see that we can get up to a precision of 1, if we present only letters with a cosine similarity higher than 0.4. This, in turn, means that only for few reference letters some comparison letters are retrieved. In Figure 4.4b we examine this relationship more closely. For this we restrict the system to letters of rank four or lower and plot the precision of retrieved letters as a function of a varying cosine similarity threshold. Instead of labeling the x-axis with this threshold we show the average number of retrieved letters per query or reference letter. Thereby we can visualize the trade-off between higher precision and fewer retrieved letters. Increasing precision in this way is easy to achieve. This though comes with the cost that for some reference letters no comparison letters might be retrieved. Unfortunately, it is likely that the recommender system is of highest value to doctors in cases of very uncommon patients. In those cases, however, it is probably least likely to find other letters with high cosine similarity to the current one in the database. Therefore, the very promising results of Figure 4.4b have to be taken with a grain of salt for real application scenarios.

Next, we turn to the question whether the most relevant letters are retrieved first. This is hard to answer, since we only know of five letters how relevant they are to a reference letter. It remains unclear which similarity rating subjects would have assigned to the other letters from our database. However, comparing the ranking computed by the algorithm with the ranking by the experts, can provide some insight into this problem. We compare these two rankings with the Spearman rank correlation coefficient. Again we estimate effect sizes with 100,000 bootstrap resamples. The correlation between the ranking given by the averaged expert rating and the ranking of the algorithm is 0.39 (95% CI: $[0.22, 0.56]$). While this is not particularly high, it is still far better than chance (Spearman coefficient of 0). The inter-rater agreement among the experts when calculated accordingly (i.e. without follow-up pairs) is 0.71 (95% CI: $[0.63, 0.79]$). We also directly estimate the difference between inter-expert agreement and algorithm-to-expert agreement. On average the algorithm-to-expert agreement lies 0.33 below the inter-expert agreement (95% CI: $[0.15, 0.52]$). Summarizing, one can state that our system's ranking is well above chance, but still quite far below human gold standard.

Considerations for further work

We have two reasons to assume that our system performs even better in reality than expected from the considerations above. First, when extending our recommender to work on a much larger database, we expect that the best fitting letters to a given reference letter will have much higher cosine similarity than is currently the case using our comparatively small database. This can be explained with the probabilistic argument that in a much larger database more well fitting documents are expected for each reference letter. Regarding Figure 4.3, this means that we would expect almost no blue points on the left and many more in the region around a cosine similarity of 0.4. These points most likely will be rated as highly similar by subjects,

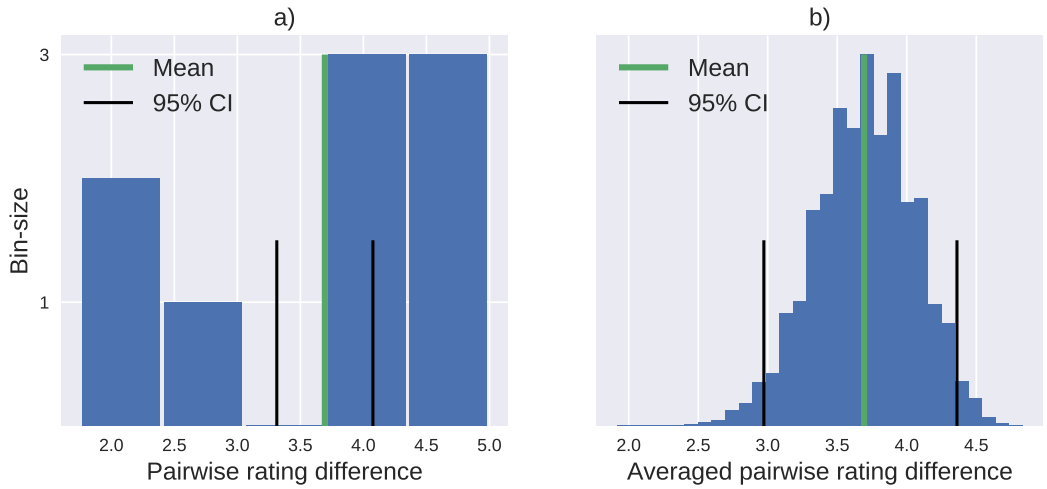

Figure 4.5: **(a):** Histogram of differences in rating of the "best" and the random comparison letter of selected trials. Selection occurred with a combination of paragraph vector and tf-idf embeddings. **(b):** Histogram of the differences in the means of averaged subject ratings of the two groups for 100.000 bootstrap resampled datasets.

as are all points with such a cosine similarity in our experiment. In Figure 4.4a and 4.4b this effect of a large database likely shifts both curves up higher. Second, there is reason to believe that some of the letter pairs with low cosine similarity, but high subject rating are due to psychological adjusting. If, during one trial, only rather badly fitting letters are presented, subjects might adjust their rating scale and rate partially fitting letters higher than under different circumstances. Effects like this are often observable in psychological experiments (Poulton, 1975). In consequence, it is plausible that points in the upper left corner of Figure 4.3 are not as problematic as it might seem. In the figure, it looks like some letters are highly relevant, although they have very low cosine similarity and will therefore probably not be retrieved by the recommender system. If the assumption that subjects adjust their rating scales is correct, then some of the letters in this region are probably less relevant than it seems from our data.

Finally one can ask whether there are possibilities to improve the system. The tf-idf embedding method does not have hyperparameters, so no further tuning of those is possible. We believe, however, that combining similarity measures from several embedding approaches is useful. Of the five embedding methods we tested, only paragraph vector is a promising option for improving the results of the retrieval. Bag of words is more or less the simple basis for tf-idf; LSA is highly correlated with tf-idf as it is a feature compressed version of the latter; LDA performed severely worse than all other methods; leaving only paragraph vector. Paragraph vector performed almost as well as tf-idf during our fine-tuning and computes feature vectors in a very different way, possibly complementing tf-idf well. Indeed we find cues that combining these two methods yields substantially better retrieval results. In a preliminary

version of such a combination we again compare the ratings given to letter pairs from two groups. One of the groups consists of the letters from our experimental dataset that paragraph vector ranks on the first place out of all letters from the entire database. Hence, letters in this group are ranked among the top four by tf-idf and top one by paragraph vector in our whole dataset. We only find nine of these letters out of the 30 possible trials. The other group consists of the nine randomly selected letters from the corresponding trials. Again we look at a histogram of differences in rating in Figure 4.5a and the distribution of average differences for 100.000 bootstrap resamples in Figure 4.5b. First, one has to note that both the bootstrap and the standard calculated means and confidence intervals are to be treated cautiously at a sample size of nine. Additionally, one can see from Figure 4.5b that the assumption of normality does not hold perfectly this time as we face an upward bounding issue and the distribution's left tail is longer, than the right tail. Due to the violation of the assumption of normality, the confidence interval estimates differ. Nevertheless, comparing the mean and confidence interval estimates with Figures 4.1a and 4.1b, where only tf-idf information was used, we see that additionally using information from the paragraph vector embedding improves results substantially. The mean of the average difference in ratings, for example, is improved from 1.93 to 3.69 units on the rating scale from 1 to 7. Therefore we conclude that using information from both embedding methods is highly useful, although we distrust the estimation of the mean to some degree due to the small sample size. In summary, work addressing how to combine information from different embedding methods seems to be a worthwhile task and should be undertaken by further research.

Chapter 5

Intelligent Search

5.1 Research Use Case

With the set of methods used throughout this work, we can address a related, but distinct problem. For some research purposes, it is necessary to find as many documents of patients with a certain feature as possible. Traditionally, this can be done with the help of string matching algorithms, which search the database. Even in easy scenarios, though, these algorithms might fail to retrieve all relevant documents. One reason for this is that, as mentioned earlier, diseases are spelled differently in different documents. Additionally, a string matching algorithm will for example also return documents in which a specific disease is mentioned in the family anamnesis, while one is only interested in patients having this disease themselves. In the specific use case we envision, a researcher starts with the physician letter of one patient and whishes to retrieve letters of similar patients from the database. The initial letter is given as input to our system and the most similar letter is retrieved from the database. The user then has to decide whether this letter indeed has the demanded feature. As a result of this decision, the document is either labeled as relevant or non-relevant. This procedure of retrieving one letter and deciding about its usefulness is repeated until the user has found sufficiently many letters with the required feature. We can address this use case with our embedding methods in the following way.

5.2 Search System

To achieve the goal of retrieving the letters with the required features, we make use of a hybrid approach of an exemplar classifier and logistic regression. In the beginning, we employ the exemplar classifier to retrieve more samples. Accordingly, the next document to be retrieved is the one with the highest average cosine similarity to the thus far obtained true positive (i.e. relevant) samples. Note that in the beginning this is equivalent to the standard recommending procedure, as the letter with the highest cosine similarity to one reference letter is retrieved. The user has then

to label the retrieved letter as either relevant or non-relevant. Once enough letters have been labeled, the exemplar classifier is replaced by a logistic regression classifier (lrc). This lrc is trained with all thus far labeled documents. The next letter to be retrieved is the one that the lrc assigns the highest probability. It is labeled by the subject and the lrc is retrained with the new set of labeled samples. In this way, one can retrieve as many relevant letters as desired. The important measure of performance for this search system is given by the ratio of irrelevant or false positive (FP) samples, which have to be looked through, to relevant or true positive (TP) samples obtained, i.e., FP/TP. We can measure it for different fractions of relevant letters retrieved. It is equivalent to measuring precision at different levels of recall, but easier interpretable for the usefulness of this particular system.

5.3 System Performance

To measure FP/TP empirically, we take a subset of 135 physician letters and label them manually for five prevalent diseases—"Chronic lymphocytic leukemia" (cll), "multiple myeloma" (mm), "breast cancer" (bc), "follicular lymphoma" (fl) and "diffuse large B-cell lymphoma" (dlbcl). In the selected subset of letters, these diseases appear between 9 and 14 times. To emulate the intelligent search, we start out with one letter from one of the groups, retrieve the best match according to our retrieval method, classify this retrieved letter as either relevant or non-relevant, update our retrieval method and present the next candidate. This is repeated until all relevant letters are retrieved. We apply this procedure to every possible starting letter. Averaging over these trials, we obtain a mean estimate for FP/TP when trying to find the first relevant, and 50%, 90%, and 100% of relevant documents for the case of each of the five diseases tested. The results of this experiment can be seen in Figure 5.1. We replace the exemplar classifier with the lrc after five false positive letters have been retrieved (a rather arbitrary value that was found through manual adjustments).

On the y-axis of this figure, we plot the means and standard errors of the means of FP/TP for each disease for a prespecified fraction of retrieved relevant documents. This allows to be able to directly read from the plot how many false positives one has to look at to obtain a new true positive sample. In the example of cll we have to look at roughly one false positive for every five true positives we retrieve, in the case of 100% retrieval. This makes it very convenient to search the database for more cll patients. However, as one can see from the figure, it works considerably less well for other diseases. Still, even in the case with poorest FP/TP ratio, one has to look at only 3.5 false positives to get one true positive sample. For research purposes, we deem this value good enough. It is also apparent from the figure that often it is relatively hard to retrieve the first match and becomes easier afterwards. One can overcome this initial phase by providing the system with several positive examples, to begin with.

With this evaluation, we showed that, in principle, it is possible to construct

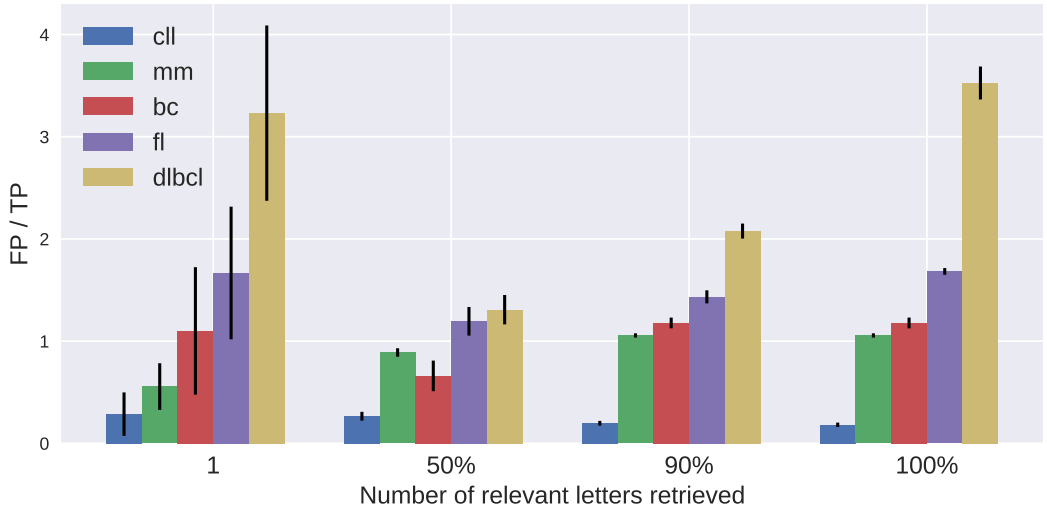

Figure 5.1: Historgram of the means (and standard errors of the means) of the false positives divided by the true positives versus how many relevant letters are retrieved for each of the five diseases—"Chronic lymphocytic leukemia" (cll), "multiple myeloma" (mm), "breast cancer" (bc), "follicular lymphoma" (fl) and "diffuse large B-cell lymphoma" (dlbcl).

a research retrieval system that is usable in practice. It remains unclear though, how well this system works for less evident features than diseases. One might be interested in relatively less defining features of the document's text, than disease. In those cases, it is probably more difficult for the system to retrieve relevant documents. A great advantage of this system, however, is that it learns from examples. One might be interested in an unusual combination of features, for example all male smoking adults with two specific types of cancer, who suffered from depression in their youth. Retrieving letters of this patient group is hardly feasible with classic string matching searches, but conceivable with the here presented method.

Chapter 6

Discussion

6.1 Theoretical Reflections

In the motivation section of our introduction, we discussed several cognitive science results about human reasoning. More specifically, we presented evidence for the hypothesis that humans often reason from examples stored in memory. The theory on this aspect of cognitive science is not as clear-cut as we presented it, though. There is, in fact, a great controversy between two different views on this subject. So-called exemplar theories indeed argue that humans reason based on specific examples stored in memory. The contrary viewpoint—prototype theory—suggests that humans form abstract prototypes and cognitive processes are based on these prototypes rather than specific examples (Rosch and Mervis, 1975; Bordage and Zacks, 1984).

A second, complicating issue arises, because there is, additionally, evidence that humans reason with conscious hypothesis testing rather than automatic pattern matching (based on examples or prototypes) (Elstein et al., 1978). The latter controversy is resolved by findings that suggest that humans use a flexible approach to problem-solving preferring either pattern matching or hypothesis testing based on the perceived difficulty of the problem. Difficult problems usually cannot be solved by pattern matching and subjects fall back to hypothesis testing (Elstein and Schwarz, 2002). Additionally, the pattern matching kind of problem-solving does not occur when presenting information as written feature lists rather than pictorially (Allen and Brooks, 1991). Hence, it is plausible that in our recommender system's use case of physician letters of uncommon cases, doctors apply conscious hypothesis testing. Consequently, the controversy between exemplar-based and prototype-based reasoning is not important for theoretical reflections on the usefulness of our system.

It is highly plausible that in our use case the model of Klein (2008) is applicable. As already described in the introduction, his view of real-world decision-making assumes that subjects generate hypotheses for the current problem based on exemplars stored in memory. The solutions from the exemplars are applied to the current problem and refined or discarded if necessary. In our application scenario, the exemplars

are provided by our algorithm rather than retrieved from memory. We can thereby extend the physician's exemplar retrieval to the whole database of the hospital.

6.2 Further Research

As explained above, there is good theoretical ground to assume the usefulness of presenting examples in aiding reasoning about difficult problems. Furthermore, we showed in chapter 4 that our system can indeed retrieve useful instances. Nevertheless, it remains unclear whether physicians' decisions will improve when using the presented system. Further research can address this issue in the following way: Medical students or unexperienced doctors can be asked to make therapy decisions with and without the recommender system. An expert can then review the benefits in therapy decisions when using our algorithm. While this experiment addresses the quality of therapy decisions as judged by human experts, it still does not address whether patients' health outcomes indeed improve. Only a clinical trial can answer this question.

6.3 Future Work

The next steps for this project consist in extending this prototype to work on the large database of the University Hospital Freiburg. We already foresee several challenges that will arise when working on this extended database. First, new data formats (other than Microsoft Word XML files) will have to be processed. Second, the current algorithms are not optimized for speed or low memory footprint. Surely, these issues must be addressed when extending the database by several orders of magnitude. Moreover a usable prototype will have to be made available for doctors in the hospital. Initially, we will have to encourage them to use the recommender. After a testing period their evaluation and criticism can be collected and assessed. If yielding positive results, the system can be incorporated into the clinic's documentation system. At that point, recommendations can be made automatically available during the writing of a new letter. In the long run, it may also be possible to extend the system to work on databases of several clinics. If these databases are used collectively for the recommendation procedure, issues of anonymization will arise. Doctors from one clinic are not allowed to view the patient data from other clinics without the patient's consent. In a realistic scenario, the system might only tell doctors that letters of similar patients are available at other clinics and the doctor has to obtain those letters manually. In this case, though, the usefulness will suffer greatly and it is likely that doctors only draw on this procedure for very uncommon cases, due to the increased effort of obtaining the letters manually.

Still, we believe that our system can improve the health outcomes of patients and should therefore be implemented in a concrete application. From personal feedback of staff of the hospital, it seems indeed likely that the system will be extended to work within the University Hospital Freiburg soon.

Appendix A

Screenshots

Sehr geehrte Frau Kollegin, sehr geehrter Herr Kollege,

wir berichten Ihnen nachfolgend über o.g. Patienten, der sich am 19.05.2016 in unserer Ambulanz vorstellte.

Diagnosen:

1. **Multiples Myelom IgG kappa** ED 01/10
   Stadium IA nach Durie & Salmon (initial Hb 14,2 g/dl, Ca 2,24mmol/l;
   im Pariser Schema keine Osteolyse; IgG 2180 mg/l)
   ISS Stadium I (ß2-Mikroglobulin 1,61 mg/l, Albumin 4,3 g/dl).
   Zytogenetik: Deletion 13q14, Zugewinn 1q21
2. Inaktive Antrumgastritis
3. Alopezie seit ca. 1977, vermutlich immunologisch bedingt
4. Arterielle Hypertonie ED 09/11
5. Hämangiom Leberlappen re 1,25cm ED 11/09
6. Benigne Prostatahyperplasie
7. Z.n. Pneumonie 08/13

Aktueller Remissionsstand: SD
Aktueller Karnofsky-Index: 100%
Vorsorgeuntersuchungen: ÖGD/Koloskopie 02/11 mit Polypabtragung, nächste ÖGD + Koloskopie für 2016 geplant.
Jetzt: Ambulante Verlaufskontrolle
Verlaufsparameter: Tumormarker, KMP, Immunfixation Serum/Urin

Verlauf und Therapie:

seit ca. 2005	wiederholt nachgewiesene, erhöhte Gammaglobuline in der Eiweißelektrophorese, graduell steigend
26.11.09	KMP: histologisch ca. 20% Infiltration durch **Multiples Myelom hohen Reifegrades vom Leichtkettentyp kappa**
27.11.09	gamma-Globuline 29,7% in EW-Elektrophorese
21.12.09	Röntgen Pariser Schema: Alte Fraktur 8. Rippe li lateral, degenerative WS-Veränderungen betont HWK 6/7, mittlere BWS, untere LWS. Leichtgradige Coxarthrose bds. Keine Osteolysen.
19.01.10	Erstvorstellung Ambulan: **watch and wait**
03-09/10	IgG, ß2-MG ansteigend, k fr LK sinkend
11/10	IgG, k fr LK leicht ansteigend.
03/11	IgG leicht fallend (von 2346 auf 2284 mg/dl), k fr LK leicht steigend (von 90 auf 101 mg/l).
09/11	IgG ansteigend (2567 mg/dl), k fr LK fallend (79 mg/l).
17.01.12	MR-Thorax bei persistenter Schmerzsymptomatik Rippenbogen re: keine Plasmozytommanifestationen, bekanntes Leberhämangiom. k fr LK 99 mg/l, IgG 2467 mg/dl
15.03.12	IgG 2508 mg/dl, ß2-MG 2,5 mg/l, k fr LK 86,1 mg/l
22.11.12	IgG 2778 mg/dl, ß2-MG 2,33 mg/l, k fr LK 92,8 mg/l Pariser Schema: kein Hinweis auf Osteolysen
13.06.13	IgG 2673 mg/dl, k fr LK 96,4 mg/l
12.12.13	IgG 2988 mg/dl, k fr LK 64,5 mg/l
27.02.14	IgG 2983 mg/dl, k fr LK 63,4 mg/l
28.08.14	IgG 3045 mg/dl, k fr LK 59,3 mg/l
05.03.15	IgG 2844 mg/dl, k fr. LK 106 mg/l (cave: Umstellung der Meßtechnik im Labor)
21.05.15	IgG 3006 mg/dl, k fr. LK 115 mg/l --> Start Therapie mit **20 mg Decortin 1x/Monat**
25.06.15	IgG 2985 mg/dl, k fr. LK 122 mg/l, **monatlich 40 mg Decortin**
23.07.15	IgG 2951 mg/dl, k fr. LK 136 mg/l, **monatlich 40 mg Decortin**
24.07.15	Vorstellung Uniklinik Heidelberg: Knochenmarkpunktion: zytologisch 9% Plasmazellen, histologisch 10 bis 15% Plasmazellanteil. Zytogenetik: Deletion 13q14 (74%), Zugewinn 1q21 (73%, 3 Kopien). Ganzkörper-MRT: keine myelomtypischen Läsionen
19.05.16	IgG 2792 mg/dl, k fr. LK 126 mg/l, SD

Figure A.1: First page of one exemplary physician letter. At the top is a short and general greeting. Next one can see a list of several diagnosis with additional information about medical parameters. The next section includes more medical parameters not related to one specific diagnosis. Then the therapy history is shown, including medical disease parameters over time and special results obtained e.g. during an MRI.

Bisherige Medikation: Ebrantil 60 mg 1-0-1, Candesartan 32 mg 1-0-0, Urotec 1-0-0

Diagnostik:
Labor vom 19.05.2016:
Hämatologie: Leukozyten 7,28 Tsd/µl; Thrombozyten 228 Tsd/µl; Erythrozyten 4,65 Mio/µl; Hämoglobin 14,0 g/dl; Hämatokrit 40,3 %; MCV 86,7 fl; MCH (HbE) 30,1 pg; MCHC 34,7 g/dl;
Differentialblutbild: Neutrophile masch. 60,7 %; Lymphozyten masch. 24,5 %; Monozyten masch. 8,1 %; Eosinophile masch. 5,2 %; Basophile masch. 1,4 %; Neutrophile masch.abs. 4,42 Tsd/µl; Lymphozyten masch.abs. 1,78 Tsd/µl; Monozyten masch.abs. 0,59 Tsd/µl; Eosinophile masch.abs. 0,38 Tsd/µl; Basophile masch.abs. 0,10 Tsd/µl; Hämolyse-Index (Serum) 12;
Klinische Chemie:
Natrium 142 mmol/l; Kalium 4,4 mmol/l; Calcium 2,26 mmol/l; Magnesium 0,83 mmol/L; Harnstoff 40 mg/dl; Serum-Kreatinin 1,09 mg/dl; GFR-Abschätzung(MDRD) 66 ml/min/1.73qm; CKD-EPI GFR geschätzt 66 ml/min/1.73qm; Glukose im Serum 91 mg/dl; LDH 179 U/l; GOT (AST) 30 U/l; GPT (ALT) 23 U/l; Alk. Phosphatase 93 U/l; Gamma-GT 49 U/l; Bilirubin gesamt 0,8 mg/dl; C-reaktives Protein < 3 mg/l; Gesamt-Eiweiß 8,3 g/dl; Albumin 4,2 g/dl; Albumin elektroph. %; alpha-1-Globulin %; alpha-2-Globulin %; beta - Globulin %; gamma - Globulin %; Quant. M-Gradient g/dl; Quant. M-Gradient % %; Immunglobulin A 126 mg/dl; Immunglobulin G 2792 mg/dl; Immunglobulin M 58 mg/dl; ß2-Microglobulin 2,16 mg/l; k fr.Leichtketten (Serum) 126,00 mg/l; l fr.Leichtketten (Serum) 13,20 mg/l; k/l-Quotient (Serum) 9,55; k/l-Differenz (Serum) 112,80 mg/l;

Anamnese und Epikrise:
Der Patient stellt sich bei bekanntem multiplem Myelom vom Typ IgG kappa zur klinischen Verlaufskontrolle in unserer Ambulanz vor.
Auf Nachfrage werden Infekte und eine B-Symptomatik verneint. Es besteht eine gute Belastbarkeit im Alltag. Es liegt lediglich eine leicht belegte Stimme vor ohne Hinweis auf eine therapiebedürftige Infektion. Eine kardiologische Kontrolle bei arterieller Hypertonie erbrachte zuletzt keinen pathologischen Befund.
Unter dem Absetzen der monatlichen Therapie mit Decortin 40 mg seit Oktober 2015 zeigt sich erfreulicherweise eine Konstanz des IgG und der kappa freien Leichtketten.
Wir besprachen mit dem Patienten, dass aus unserer Sicht aktuell anhand der vorliegenden Befunde die Therapie mit Decortin weiter pausiert werden könne und eine Wiedervorstellung in 6 Monaten vorgesehen ist. Sollte es im mittelfristigen Verlauf zu einem Anstieg der Myelomparameter kommen, kann über eine Wiederaufnahme der Decortintherapie nachgedacht werden. Bezüglich eines Lipoms am Hinterkopf rechtsseitig wägt der Patient momentan eine chirurgische Entfernung ab. Wir empfehlen hier eine konsiliarische Beratung.

Procedere:
Einen unkomplizierten Verlauf vorausgesetzt, Wiedervorstellung am 17.11.16 zur Blutentnahme der betreuenden Station und in unserer Ambulanz zur Besprechung der Befunde.

Mit freundlichen kollegialen Grüßen

Figure A.2: Second page of one exemplary physician letter. At the top one can see the so far used medication. The next section describes detailed laboratory examination results of blood counts. Then the current anamnesis is shown, where e.g. the patient's current complaints are listed. The subsequent paragraph discusses the procedure for the next six months. Finally "goodbye regards" are given.

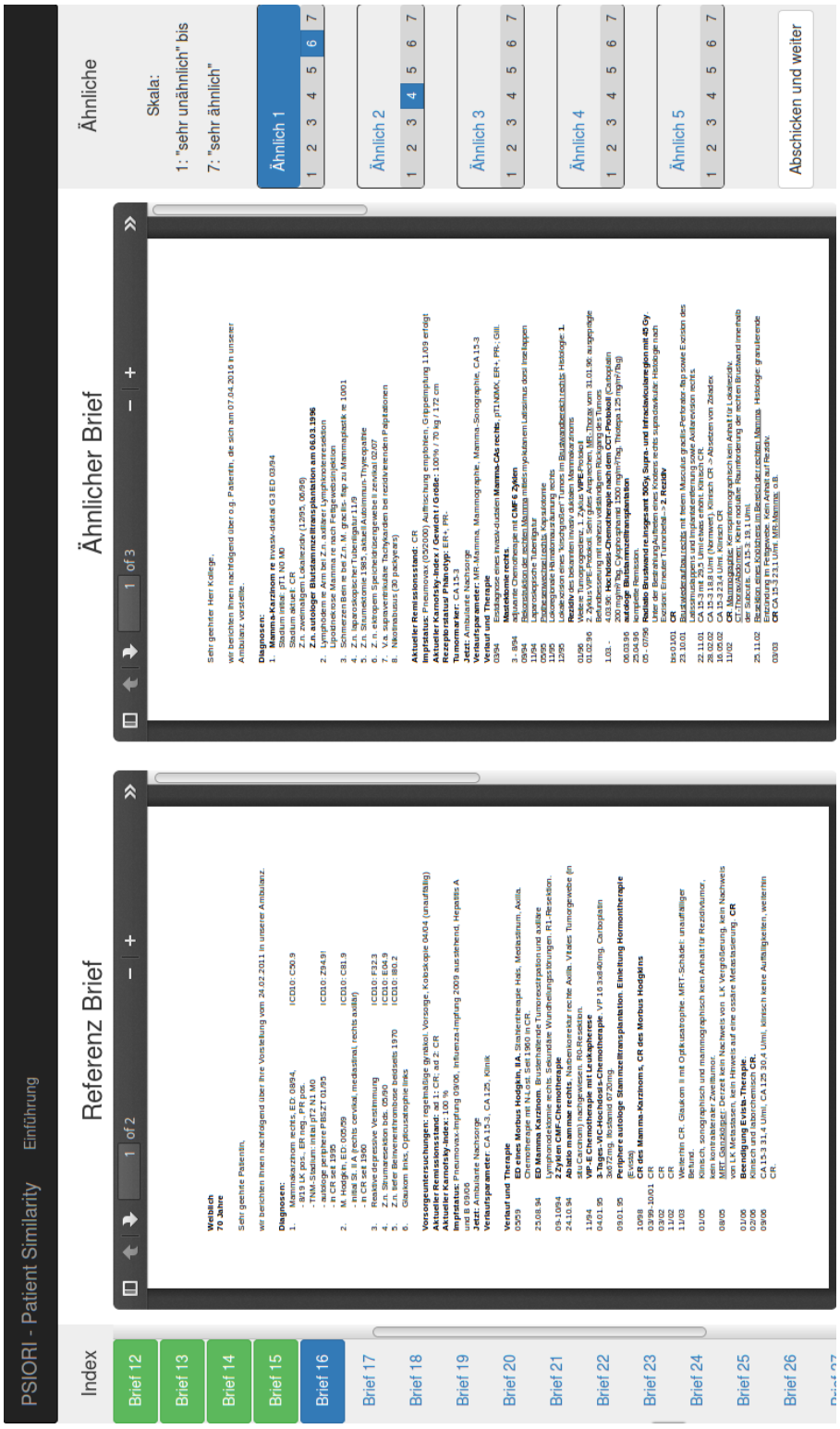

Figure A.3: Screenshot of the whole webpage, where subjects conducted the experiment.

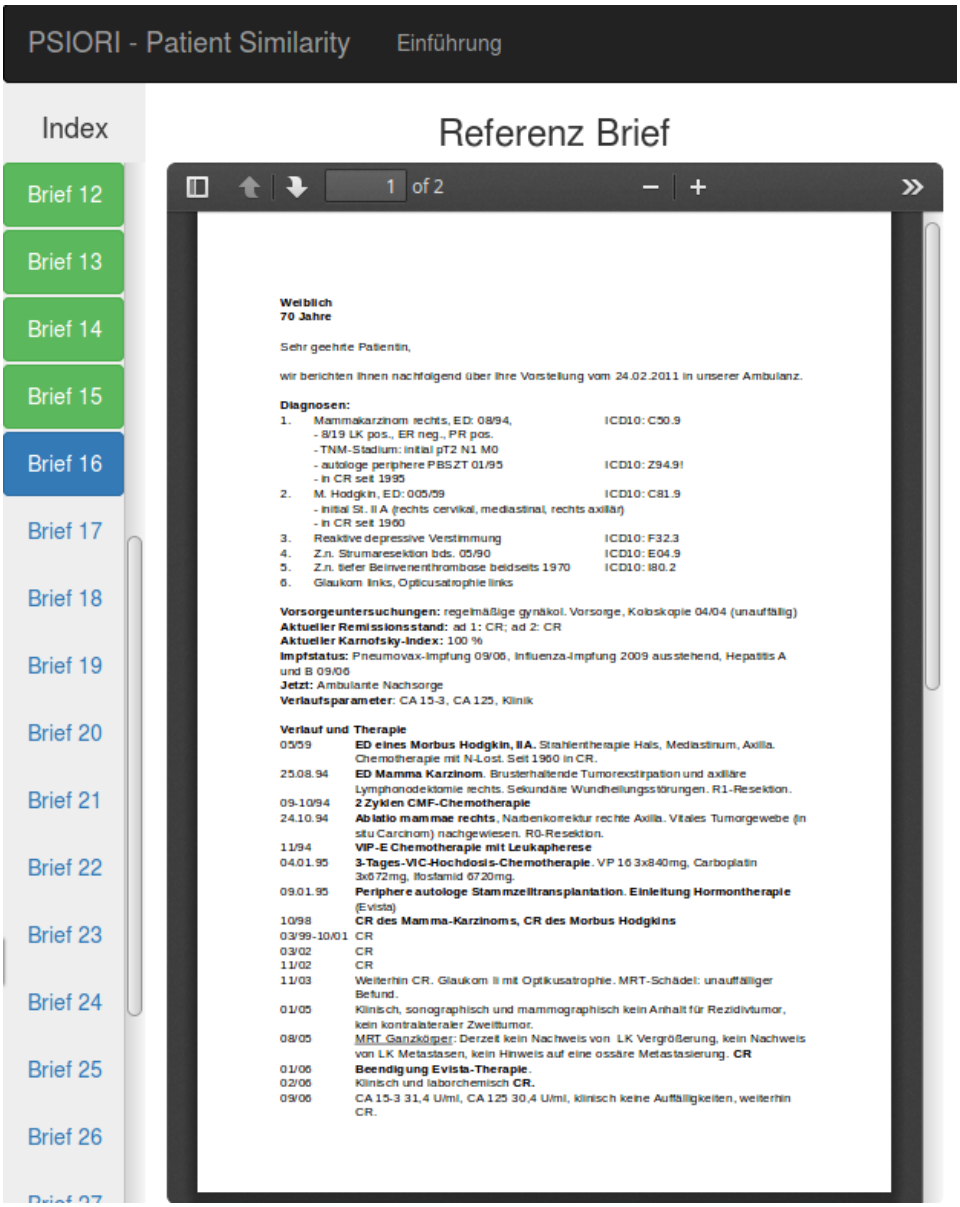

Figure A.4: Screenshot of the left part of the webpage, where subjects conducted the experiment. In the left column the different reference letters can be selected. The chosen reference letter is displayed right of this column.

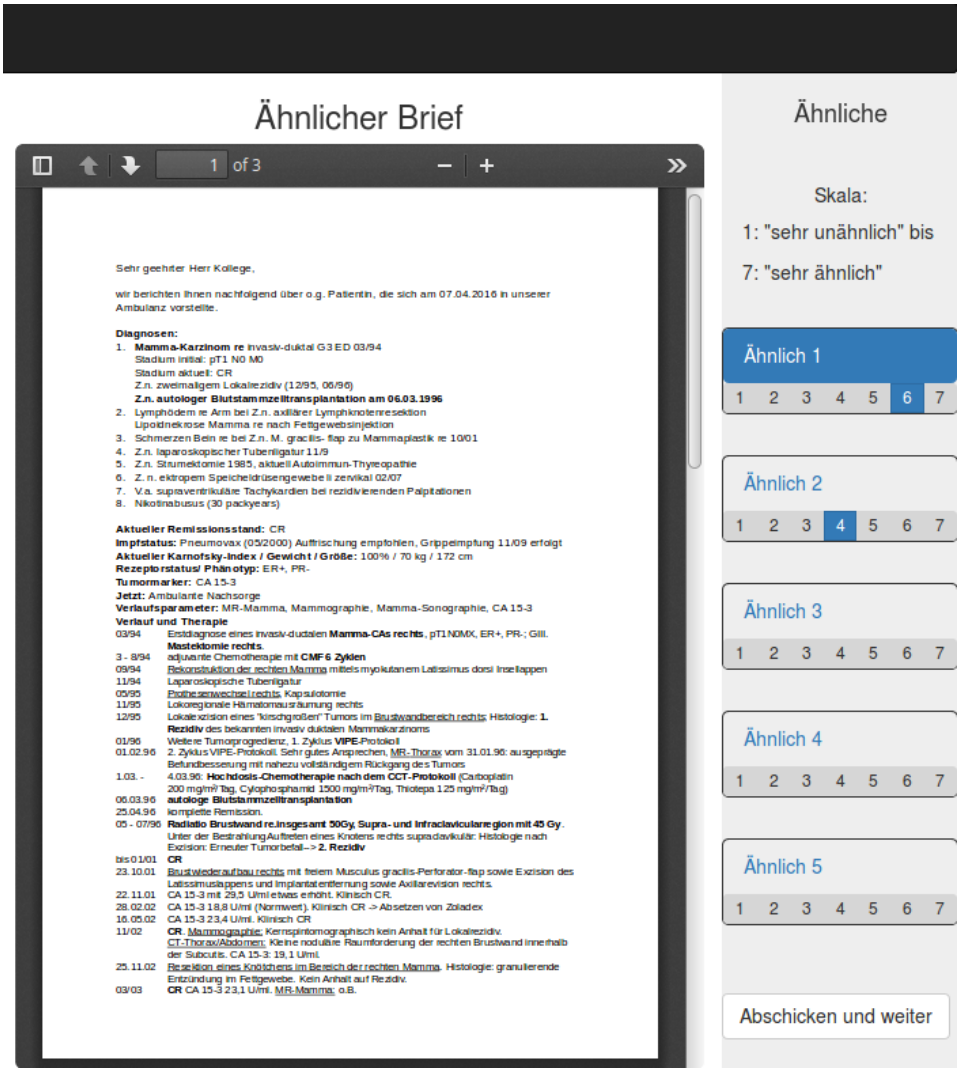

Figure A.5: Screenshot of the right part of the webpage, where subjects conducted the experiment. In the right column the different comparison letters can be selected and rated for their similarity to the current reference letter. The chosen comparison letter is displayed left of this column.

Appendix B

Code

The Python code comprising the recommender system itself and all experiments and evaluations we have done is attached to the printed version of this thesis as a hard copy on DVD. However, the dataset of physician letters is not included. Personal information of the patients might still be present despite our efforts to remove it. Consequently, we prefer not to make this dataset publicly available.

Bibliography

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

Allen, S. W. and Brooks, L. R. (1991), 'Specializing the operation of an explicit rule', *Journal of Experimental Psychology* **120**(1), 3–19.

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.

Bordage, G. and Zacks, R. (1984), 'The structure of medical knowledge in the memories of medical students and general practitioners: categories and prototypes', *Medical Education* **18**(6), 406–416.

Chi, M. T. H., Feltovich, P. J. and Glaser, R. (1981), 'Categorization and Representation of Physics Problems by Experts and Novices', *Cognitive Science* **5**(2), 121–152.

Cohen, J. (1960), 'A Coefficient of Agreement for Nominal Scales', *Educational and Psychological Measurement* **20**(1), 37–46.

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.

Efron, B. (1979), 'Bootstrap Methods: Another Look at the Jackknife', *The Annals of Statistics* **7**(1), 1–26.

Elstein, A. S. and Schwarz, A. (2002), 'Clinical problem solving and diagnostic decision making: selective review of the cognitive literature', *British Medical Journal* **324**(7339), 729–732.

Elstein, A. S., Shulman, L. S. and Sprafka, S. A. (1978), *Medical Problem Solving: An Analysis of Clinical Reasoning*, Harvard University Press, Cambridge, Massachusetts.

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), 1.

Klein, G. (2008), 'Naturalistic Decision Making', *Human Factors: The Journal of the Human Factors and Ergonomics Society* **50**(3), 456–460.

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), 420–431.

Lau, J. H. and Baldwin, T. (2016), An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation, *in* 'Proceedings of the 1st Workshop on Representation Learning for NLP', Association for Computational Linguistic, Berlin, pp. 78–86.

Le, Q. and Mikolov, T. (2014), Distributed Representations of Sentences and Documents, *in* 'Proceedings of the 31st International Conference on Machine Learning', JMLR Workshop and Conference Proceedings, Beijing, pp. 1188–1196.

León-Villagrá, P. and Jäkel, F. (2013), Categorization and Abstract Similarity in Chess, *in* 'Proceedings of the 35th Annual Meeting of the Cognitive Science Society', Cognitive Science Society, Berlin, pp. 2860–2865.

Linhares, A. and Brum, P. (2007), 'Understanding Our Understanding of Strategic Scenarios: What Role Do Chunks Play?', *Cognitive Science* **31**(6), 989–1007.

Manning, C. D., Raghavan, P. and Schütze, H. (2008*a*), Evaluation of unranked retrieval sets, *in* 'Introduction to Information Retrieval', Cambridge University Press, Cambridge, chapter 8, pp. 154–157.

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

Manning, C. D., Raghavan, P. and Schütze, H. (2008*c*), The term vocabulary and postings lists, *in* 'Introduction to Information Retrieval', Cambridge University Press, Cambridge, 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., Goldstone, R. L. and Gentner, D. (1993), 'Respects for Similarity', *Psychological Review* **100**, 254–278.

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', *arXiv preprint arXiv:1301.3781* .

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 26', Curran Associates, Inc., South Lake Tahoe, pp. 3111–3119.

Morin, F. and Bengio, Y. (2005), Hierarchical Probabilistic Neural Network Language Model, *in* 'Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics', The Society for Artificial Intelligence and Statistics, Barbados, 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.

Poulton, E. C. (1975), 'Range Effects in Experiments on People', *The American Journal of Psychology* **88**(1), 3–32.

Rosch, E. and Mervis, C. B. (1975), 'Family resemblances: Studies in the internal structure of categories', *Cognitive psychology* **7**(4), 573–605.

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.

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

Spearman, C. (1904), 'The Proof and Measurement of Association between Two Things', *The American Journal of Psychology* **15**(1), 72–101.

Williams, D. D. R. (2004), 'In defence of the case report', *The Royal College of Psychiatrists* **184**(1), 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, June 20, 2017