

# Keyword extraction of reviews of ZorgkaartNederland

A report of an internship at ██████

Aucke Bos

s4591496

July 5, 2021

## 1 Structure of the internship

This report describes my external internship for the master Data Science at the Radboud University. For the internship, I joined a team of data analysts at ██████. The team mainly consists of data- and intelligence analysts; they work on a project basis. Their goal is to support other teams within ██████, by providing new insights using data analyses and models. We have a weekly meeting where we discuss the progress of the projects people are working on, their challenges, and possible new projects. This meeting is also the place where people present their results when found interesting for the rest of the team. I have also kept the team updated about our project by means of presentations, such that the experience and insights are shared within the team and the company.

During the internship, I mainly focused on my own project, which is described in this report. However, I also had the opportunity to have a look at the projects of other team members and other teams, to get an idea of how the company operates.

My supervisor within the company is ██████; he is working as a Senior Intelligence Analyst within ██████. We had meetings at least once a week. In these meetings, we discussed progress, any difficulties we encountered, and the direction and goal for the upcoming week. During the week we had contact whenever we felt we needed to discuss something, about any problems we encountered, or design choices we had to make. ██████ has experience with natural language processing, and we regularly had useful discussions about the problems at hand. Many challenges we encountered were new for both of us. In these cases, we discussed what we thought would work best, and made design choices based on these discussions.

## 2 Domain

The project we have been working on is about ZorgkaartNederland (ZKN). This is a platform where people write reviews about their experiences with healthcare providers. Its purpose is a) to provide patients with information about healthcare providers, such that they can base their decision on experiences of other patients; b) to provide a platform for patients to share their experiences of healthcare providers with other patients. Someone who is looking for healthcare can select a type of organization they are looking for, for example, a hospital, dentist, or general practitioner, or a specific disease they'd like care for. ZKN then provides a list of healthcare providers that should be able to help them, along with all the reviews that other people wrote about that provider. ZKN hereby contributes to the transparency of Dutch healthcare [1]. This transparency is important because in the Netherlands we have a regulated competition between healthcare providers, thus a patient must be able to get reliable information.

ZKN is a growing platform, with at the moment almost a million reviews about more than 130.000 healthcare providers. The rate of growth is a strong indicator that patient reviews play an important role in making the decision for healthcare. Literature also increasingly adopts the viewpoint that the reviews of healthcare users should be taken into account when measuring the quality of healthcare providers [6][3].

## 3 Motivation

For health insurers such as ██████, insight into patient reviews is very valuable. They might for example be used as a guideline when deciding which provider will be contracted, or as a quality

indicator. ZKN provides a package for healthcare providers and insurers, this package includes services like a dashboard with trends about ratings, and notifications of new reviews. A study compared the distribution of grades of ZKN with the Consumer Quality Index and concluded that the reviews of ZKN are representative of the population. [5] Although these insights are useful, they solely focus on the grades that are given in the review. █████ already refers its customers to ZKN and shows grades of providers in their services.

However, up until now the texts belonging to these reviews are only evaluated manually. Individual reviews provide information about single experiences, but general conclusions cannot be drawn. This project is therefore concerned with analyzing the textual part of the reviews. We'd like to draw conclusions about what patients *generally* think about certain aspects, in certain regions, for certain providers, about certain diseases, etc.

Secondly, this project will help to bring more focus within █████ to NLP as a tool to solve problems. NLP is not yet widely used within the company right now, although people are very interested in what they can achieve with NLP. If we get good results, the team will surely want to use the approach for other applications. In any case, the methodology that is used will be useful for any NLP-related research or projects within the company. Therefore an important part of the project is to discuss with the team our findings and encountered challenges, and discuss and document how we got to these results.

## 4 Task

As mentioned, our main goal is to retrieve new and structured information about the textual reviews of ZorgkaartNederland. This goal is of course very broad, and 'new information' by itself is not very valuable. However, if we are able to represent and analyze it in a structured way, we should be able to answer all kinds of questions. At the start of the project, we have defined several questions we would like to get answers to, by analyzing the reviews. However, As will be discussed later on, the focus of the project will be on keyword extraction. If we succeed in extracting the keywords with high accuracy, answering these questions should be much easier. Actually investigating these topics will be mostly outside the scope of the project.

A requirement of the project is that we program it in R. This way we are sure it integrates well with other projects, as this is the main programming language used within the company.

### 4.1 Sentiment analysis

The structure of the dataset will be discussed in Section 5. But each review has a list of grades, with a corresponding average grade. We can use these grades in the context of sentiment analysis. We can for example divide reviews into the categories positive and negative, by classifying grades  $\leq 7$  as negative reviews, and grades  $> 7$  as positive. Can we find out which subjects cause people to write negative reviews? To what extent do review texts correlate at all with the average grades? And can we use this information to improve services to increase the satisfaction of patients? These questions also play a role when █████ purchases healthcare at the beginning of a year. If certain providers consistently score badly on specific areas of their service, we might want to stimulate them to improve their service in these areas.

### 4.2 Digitization of healthcare

One specific topic that is interesting is the digitization of healthcare. In the context of sustainability, cutting costs, and a global pandemic, increasingly many hospital appointments are online. Can we see this shift in the grades and texts of reviews? If so, what do people think of this shift? Do they feel it is just as effective as a physical meeting? Do they feel more (un)comfortable in online appointments? Do they have the feeling they are listened to worse than before? We might find answers to these kinds of questions when we compare reviews of the past year with earlier years.

### 4.3 Satisfaction across geographical regions

For most providers on ZKN, their location is also provided on the website. We could use this information to map patient satisfaction across the country. Do certain regions score lower than average? If so, can we find out which aspects of healthcare are negatively experienced, and can we interpret these results? █████ already uses policies based on geographical regions, since population differs across the country. Are these policies in line with what we see in the reviews, or should they be changed?

## 4.4 Topic analysis

We can use topic analysis to map which topics are regularly discussed for specific healthcare providers. We might be able to use these topics to assign labels to providers, indicating which aspects of healthcare are in their expertise. If someone is looking for a healthcare provider regarding a specific issue, we might pinpoint suitable providers that have much experience in the issue, using these labels. If we can accurately assign topics to reviews, another application would be to apply this method to the reviews of the mobile application [REDACTED]. Some of those reviews are read manually, but no processing is done to draw conclusions out of them.

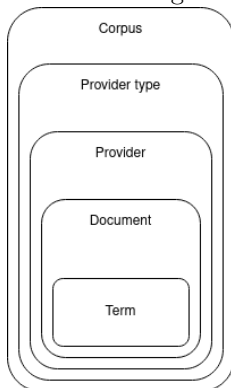
## 4.5 Corona and healthcare

In the past year, we often see news articles about capacity problems in hospitals. How often and to what extent do patients experience these problems, and how frustrating are they? Can we also see the influence of this in the grades people give about their experiences?

# 5 Data retrieval

Our first step is to create the dataset on which we will be working. ZKN does not have an API, thus we have to scrape the reviews of the website. The way we retrieve our reviews is related to the structure of the data on ZKN; it is shown in Figure 1. Each review is a document, which consists of terms (words). A review is always about a specific healthcare provider, for example, '[REDACTED]'. A provider has a provider type, for example, 'Health insurer'. We denote all documents of all providers of all provider types as our corpus. The algorithm we discuss in Section 6 is also based on this structure. A few examples of some reviews on ZorgkaartNederland are shown in Appendix B. We scrape the website as follows:

Figure 1: The structure of the dataset of ZorgkaartNederland.



1. We retrieve all provider types using the overview page<sup>1</sup>. For each type, we save the name and the overview URL.
2. For each provider type, we load all registered providers of that type, using the overview page of that provider type<sup>2</sup>. For each provider, we save the name, its type, and the overview URL.
3. For each provider, we load all basic data of all reviews, using the webpage of that provider<sup>3</sup>. The reviews are listed 20-per-page. Without following the link to the review, we save for each review the basic data: The provider type, the provider, the URL to the review, the date the review was written, and the average grade.
4. For each review, we follow the link<sup>4</sup> and save the grades on different levels, the text, and optionally the department and condition of the treatment.

These steps provide us with a dataset of about 650.000 reviews.

## 6 Approach: Keyword extraction

As discussed in Section 4, the main part of this project is to represent our dataset in a structured way. After scraping, we have a very large table with information about the reviews, including a column with the text data. We want to represent the texts of these reviews in a more compact and structured way, such that we can analyze and compare them. We have chosen to try to do this using keyword extraction. The idea is that the text of a review can be summarized by a few keywords extracted from

<sup>1</sup><https://www.zorgkaartnederland.nl/overzicht/organisatietypes>

<sup>2</sup><https://www.zorgkaartnederland.nl/{type}>

<sup>3</sup><https://www.zorgkaartnederland.nl/zorginstelling/{provider}-{id}/waardering>

<sup>4</sup><https://www.zorgkaartnederland.nl/zorginstelling/{provider}-{provider-id}/waardering/{title}-{review-id}>

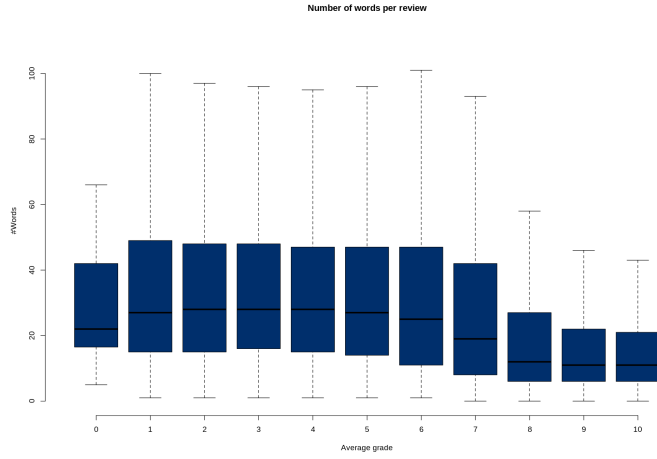


Figure 2: The average word count per grade. Stopwords and punctuation tokens have been removed from the data. The number of words per review is very small.

that text. This way each text is represented in only a few words that hold most of the information about what is being said. A lot of redundant and non-informative words are now removed, and only the core information remains. We can then analyze these keywords, and investigate the questions of Section 4.

To extract the keywords of a document, we want to assign a score to each term. The score for each term indicates the ‘informativeness’ of the term: The higher the score, the more informative that term is within the document. With some threshold for a minimum value, we can then extract the top  $n$  terms of a document as its keywords.

### 6.1 A failing idea: TF-IDF

The first thought is to compute the TF-IDF value for each term in each document. The higher this value, the more informative the word should be. However, due to the structure of our data, this approach does not work. As can be seen in Figure 2, our documents are very short: most of the reviews have fewer than 30 words. As a result, we encounter the TF=1 challenge [9]. The Term Frequency is calculated via Equation 1, where  $f_{t,d}$  is the frequency of term  $t$  in document  $d$ .

$$TF = \frac{f_{t,d}}{\sum_{t' \in D} f_{t',d}} \quad (1)$$

For short documents, each term will only occur once within the document. If you compute TF as the absolute frequency, instead of relative to the document length, this means that the TF for all terms is 1, hence the TF=1 challenge. This means that however informative a word is within a document, its term frequency will always be equal for all terms in the document. Thus the TF-IDF score is actually only based on IDF, which means the ‘informativeness’ is only based on the number of documents a term occurs in: The fewer documents a word occurs in, the higher its TF-IDF, the higher its informativeness. In Appendix A we provide the TF-IDF scores of a few sample reviews, which clarify the problem. In the first review, all terms have a TF of 1, thus the final scores are only based on the IDF value. In the second review, some parsing/spelling errors occur. Such a term is thus very unique in the corpus, resulting in a very high IDF score. The last review contains a term with a  $TF > 1$ ; we see this results in a very high TF value with respect to the other terms.

We concluded that the TF-IDF approach is too limited when computing scores for our corpus. We need a more sophisticated way to compute informativeness, taking into account more than only the term frequency and document frequency.

We found a paper by Toivanen et al [8] that describes an algorithm for keyword extraction on a very similar dataset as ours. We have used this paper as a basis for our implementation of keyword extraction, which we discuss extensively in the next section.

## 6.2 Solution: a more sophisticated approach

This section describes the algorithm called "Informativeness-based Keyword Extraction (IKE)" and the idea behind it, as created by Toivanen et al. In later sections we elaborate on the evaluation and tuning to our own dataset.

The idea of the approach is to define the informativeness of a term based on more than simply the term and document frequency. To do this, we base the final informativeness scores on three levels. We list them here and elaborate upon them below.

- **The corpus level.** The informativeness of a word on the corpus level is based on its IDF value.
- **The category level.** This score indicates informativeness of a term within the category it occurs in, with respect to the other terms in that category.
- **The document level.** The score on the document level is computed by combining the above two scores and adding some other characteristics of the terms.

### 6.2.1 Corpus level scores

The idea of the corpus level score is to extract important words on an abstract level. The result is a score for each unique word in the corpus. The informativeness of a word within the corpus is related to its IDF. If a word is too common, it is probably not very informative. The same holds for words that are very rare: these are also not very informative in most cases. Therefore we define an optimal IDF for each term. This optimal IDF, called *FW* for *Frequency Weight*, is used to penalize words that do not have an IDF equal to that value. *FW* is calculated via Equation 2, where  $tf_c$  is the word frequency of the term within the corpus.  $\alpha$  and  $n_0$  are hyperparameters to be set based on characteristics of the corpus, which we discuss in Section 9.  $n_0$  is the optimal frequency for a term; a word with this frequency should have the highest informativeness value. We take the absolute value of the logarithm such that frequencies both higher and lower than the optimal frequency are penalized.

$$FW(t) = \alpha \cdot \left| \log_2 \frac{tf_c}{n_0} \right| \quad (2)$$

$\alpha$  is used as a penalty for infrequent words. For all terms  $t$  with  $tf_c < n_0$  that only occur at most once in a document ( $tf_c = df$ ), we need this  $\alpha$  value. If we would omit it, an increase in  $tf_c$  results in a decrease in the IDF that is equal to the decrease in *FW*. Such that all those terms  $t$  would have the same corpus level score, regardless of the value of  $tf_c$ , as long as its  $< n_0$ .

Our corpus level score  $IDF_{FW}(t)$  is computed by extracting the penalty from the actual IDF, as shown in Equation 3.

$$IDF_{FW}(t) = IDF(t) - FW(t) \quad (3)$$

### 6.2.2 Category level score

This score defines the informativeness of a term within the category it occurs in. If a word occurs often within category  $c$ , and not often in other categories, it is informative within its category. If a term occurs often in all categories, the category level informativeness should be low.

For clustering the documents, Toivanen et al apply Agglomerative clustering. In our dataset, the clusters are more naturally present. As can be seen in Figure 1, our data is clustered by design. Two intuitive cluster levels are found: Provider level and Type level. Clustering on a Provider level means that each provider gets its own cluster; all reviews belonging to that provider are in that cluster. Type level clustering results in much larger clusters: one cluster per provider type. Using these intuitive clusters also benefits the explainability of the algorithm. Other clustering methods might be useful when using the keywords for different goals, an example is given in Section 10. In the first version, we cluster on a Provider level.

The category level score consists of two terms: The Term - Corpus Relevance (*TCoR*), and the Term - Category Relevance (*TCaR*). To compute the *TCoR*, we divide the documents into shorter texts, called fragments. The idea of these fragments is that they are stand-alone entities that provide information by themselves. Now the intuition is that terms are more informative if they occur more 'alone'. This would mean that the term needs few other terms to provide meaning. The fragment breaks we use differ a little from those that Toivanen et al use. We base them on two POS tags as computed by Spacy<sup>5</sup>: all tokens of type PUNCT (punctuation) and CONJ (coordinate

---

<sup>5</sup><https://spacy.io/>

conjunction) define the fragments breaks of a document. Splitting documents on these tags should result in fragments that contain meaning by themselves. This intuition seems valid when manually investigating the resulting fragments in the corpus. The list of breaks Toivanen et al use is less inclusive, as they only use some specific punctuation marks (, ! ? . : ;), and the words "and", "or", and "both". We saw that many coordinating conjunctions split the documents in meaningful fragments, hence we chose to use these instead.

Using a Fragment-Term-Matrix <sup>6</sup>, we compute for each unique term  $t$  in the corpus the Inverse average fragment length  $ifl(t)$  by Equation 4, where  $avg(l_f(t))$  is the average fragment length of term  $t$ . Using a Document-Term-Matrix we compute the Inverse category count  $ic(t)$  by Equation 5, where  $c_t$  is the count of unique categories in which term  $t$  occurs. The  $TCoR$  is now given by simply adding both values, as in Equation 6.

$$ifl(t) = \frac{1}{avg(l_f(t))} \quad (4)$$

$$ic(t) = \frac{1}{c_t} \quad (5)$$

$$TCoR(t) = ifl(t) + ic(t) \quad (6)$$

$ifl(t)$  is higher if  $t$  occurs in shorter fragments, in line with the intuition that it should hold more information.  $ic(t)$  is higher if  $t$  occurs in fewer categories, scoring terms higher if they occur in a single or a few categories.

The idea behind the  $TCaR$  is to assess the informativeness of a term among categories. If a term occurs often in one category, and rarely in others, it is informative within that category. The score is computed using two probabilities:  $P(c|t)$  and  $P(d_t|c)$ . Equations 7 and 8 show how we compute them, where  $d_{t,c} \in D_c$  are the documents of category  $c$  that contain  $t$ ,  $D_t$  is the set of documents in which term  $t$  occurs, and  $D_c$  is the set of documents in category  $c$ . We compute  $TCaR$  by multiplying these values, as in Equation 9.

$$P(c|t) = \frac{|d_{t,c} \in D_c|}{|D_t|} \quad (7)$$

$$P(d_t|c) = \frac{|d_{t,c} \in D_c|}{|D_c|} \quad (8)$$

$$TCaR(t, c) = P(c|t) \cdot P(d_t|c) \quad (9)$$

The category level term score is now computed via Equation 10. Terms that occur only in a single category will have much higher values than terms occurring in many categories, likewise for terms that occur in short fragments as opposed to terms that occur in longer fragments. The term score for a category is high if the term occurs in many documents of the category, but rarely in other documents.

$$s_{category}(t, c) = TCoR(t) \cdot TCaR(t, c) \quad (10)$$

### 6.2.3 Document level score

The document level score is computed by combining  $s_{category}(t, c)$  and  $IDF_{FW}(t)$  as a weighted average. Toivanen at all first convert  $IDF_{FW}(t)$  to  $s_{n,corpus}(t)$  by dividing  $IDF_{FW}(t)$  with the maximum corpus level score over the whole corpus, such that  $s_{n,corpus}(t) \in [0, 1]$ . They do this such that the category and corpus level scores are comparable. However, when looking at Equation 10, we see that  $s_{category}(t, c) \in [0, 4]$ . Thus if we only normalize  $IDF_{FW}$ , the scores are not very comparable. Another problem is the distribution of the category level score when not normalized. The distributions for both un-normalized scores are shown in Figure 3. There are a few high-scoring outliers, but the average category level score is  $< 0.01$ , while the average of the normalized corpus scores is  $\sim 0.7$ . The category score has almost no influence on the average of these two values, hence only the corpus score is of any importance. This way the final score is ultimately still only based on the document frequency of the terms (as in Section 6.1). Therefore we decided to apply log transformation and

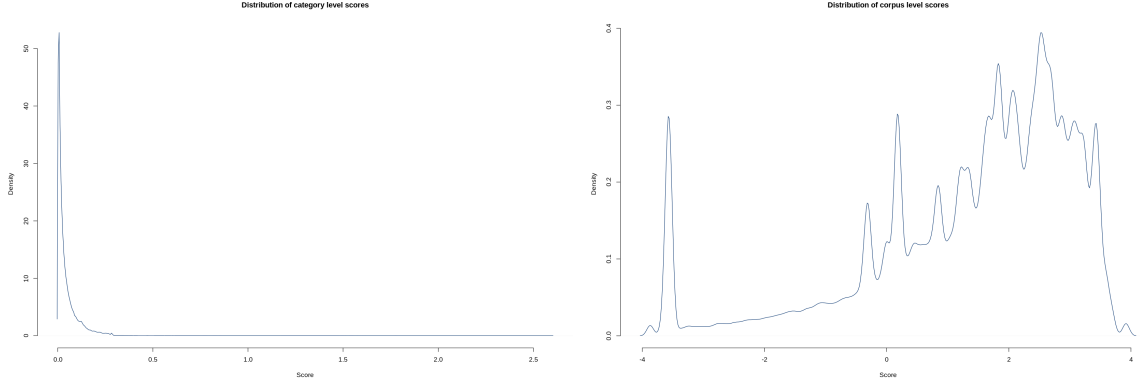


Figure 3: Distributions of the category level scores (left) and corpus level (right) scores, before normalization. The scores are not comparable, and the category score is very skewed.

apply the same normalization technique as for the corpus score. The distributions for the normalized scores are shown in Figure 4. We compute our document level score by Equation 11.  $c$  is the category of document  $d$ ,  $\beta$  defines the weighting; it is another hyperparameter we tune in Section 9.

$$s_{doc}(t, d) = \frac{\beta \cdot s_{category}(t, c) + (1 - \beta) \cdot s_{n,corpus}(t)}{2} \quad (11)$$

#### 6.2.4 Final score

For the final score, Toivanen et al add two other terms. The first term is the distance factor  $di(t, d)$ , as computed in Equation 12, where  $i(t, d)$  is the position of term  $t$  in  $d$ , and  $|d|$  is the document length.

$$di(t, d) = 1 - \frac{i(t, d)}{|d|} \quad (12)$$

The intuition behind this term is that important words often occur at the beginning of a document. We investigate the influence of this term on the final score in Section 9.

The second term is the POS-weight of the term:  $w_{POS}$ . For this term, we define a POS-weight  $\in [0, 1]$  for each POS tag in the corpus. Words with POS tags with a higher weight will hereby have a higher final score. The idea is that some POS tags have a high chance of carrying meaning, for example, nouns and adjectives. Other words, such as adpositions, have a low chance of high informativeness.

The final score is computed by multiplying these three terms, as shown in Equation 13.

$$s(t, d) = s_{doc}(t, d) \cdot di(t, d) \cdot w_{POS}(t) \quad (13)$$

---

<sup>6</sup>[https://en.wikipedia.org/wiki/Document-term\\_matrix](https://en.wikipedia.org/wiki/Document-term_matrix)

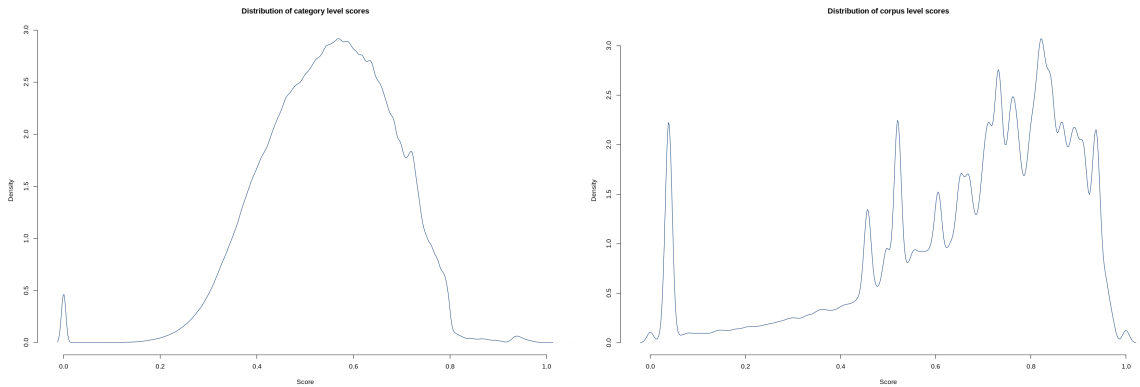


Figure 4: Distributions of the log-transformed and normalized category level scores (left) and the normalized corpus level scores (right). They are much better comparable.

We now have an informativeness score for each word in each document. Converting these scores into a list of keywords is done via a threshold. The value of this threshold is dependent on the words of each document. We set the threshold to  $0.5 \cdot \max(s(t, d))$  for each document, and limit the maximum number of keywords to 10. Thus a word is selected as a keyword if it has at least half the value of the highest score in the document, with a maximum of 10 keywords per document.

## 7 Implementation

We have implemented IKE as described in the previous section, in the programming language R. We use the package SpacyR<sup>7</sup> based on Spacy to parse our corpus into tokens. The tokenization of Spacyr provides us with a very large table, with a row for each term in the corpus. But the library is much more comprehensive than simple tokenization. It also provides us with characteristics of the term, like the position in the document, whether the word is a stopword, and the sentiment of the word (based on a list of sentiment words).

To transform the tokens into a Document Term Matrix, we use Quanteda<sup>8</sup>, which integrates nicely with SpacyR. We create a Document Frequency Matrix (DFM): a matrix with one row for each document, one column for each unique term in the corpus, the cells hold the frequencies. We also create a Fragment Frequency Matrix, with one row for each fragment. With these two matrices, we are able to compute all scores needed for the algorithm.

## 8 Evaluation

Now that we have a system to compute keywords for a set of documents, we need to evaluate it. This is not a trivial task. The only 'labels' we have about the data are the grades of the reviews. As a first attempt, we tried to use these grades as a label for classification.

The idea is to divide the corpus into two groups: positive and negative reviews. Reviews with a grade  $< 7$  get label 0 for negative, all other reviews get label 1 for positive. We then use solely the keywords to represent a document. We create a Document Frequency Matrix with one column for each unique keyword in the corpus. The values are the frequencies of those words in the documents. We then try different classification models to classify the documents into the positive or negative class. We use the function `train_models` of the library RTextTools<sup>9</sup>. This library provides a clean API to run different models on a DFM.

There are two problems with this approach. Firstly, the DFM is very large and very sparse. This matrix can only be saved in memory using a special data structure in R, called SparseMatrix. RtextTools supports inputs of this format, but it converts it to a normal matrix before applying the models. Such a matrix would need a few hundred gigabytes of memory, so this won't work. We thus only sampled a small fraction of the reviews (1000 documents) to train the models on. Random Forest seems to work best, achieving an F1 score of 0.71. However, given the simplicity of the task (binary classification), this was not very satisfying.

The second problem with this approach is that we are running a different model on our algorithm than what it was designed for. IKE is built to extract keywords from documents, which are those words that carry the most meaning. But we train models to verify whether we are able to classify the sentiment of documents using these keywords. So however high or low our score is, we still do not know whether the keywords are the most informative words in the documents.

### 8.1 Building our own testset

Because our previous attempt did not work, and we do not have any other labels, we decided to build our own testset.

As mentioned in Section 3, part of the project is to get NLP on the radar of other (team)members at [REDACTED]. Halfway through the internship, we, therefore, planned a presentation for our team, where we presented our current results. In this presentation, we explained a little bit about how NLP in general works, what it can be used for, how we used it, and we explained the algorithm we used. At that point, we had just finished the first version of the algorithm. We concluded the presentation with the statement that we were unsure how well the algorithm performed, and we were unable to test its performance, as we had no testset. We asked the team to help us create such a testset.

---

<sup>7</sup><https://github.com/quanteda/spacyr>

<sup>8</sup><https://quanteda.io/>

<sup>9</sup><http://www.rtexttools.com/>



We set up an online annotation tool called Doccano<sup>10</sup>, it is an open-source self-hosted annotation tool. We set up the project, imported the data, and made it available at <https://zkn-annotations.auckebos.nl><sup>11</sup>. We created a user for each of our team members and uploaded 300 random reviews of our dataset, which is similar to the size of the testsets used by Toivanen et al [7].

After the presentation, the team knew about the context of the project, and we had explained what a keyword should be about. The goal of the annotation tool was to create a dataset of reviews, annotated with their corresponding keywords. We wanted each review to be annotated at least twice, and we would subset those selections and calculate the Inter Annotator Agreement. If some reviews would only be reviewed once, either [REDACTED] or I would annotate it a second time, such that our dataset would be large enough.

Almost everyone annotated at least some reviews in the following week, and some team members even annotated a lot of them. Halfway through the week, we uploaded 50 more reviews, because we were almost through with the first 300. At the end of the next week, [REDACTED] and I finished the reviews that were not or only once annotated, and we ended up with a dataset of ~350 reviews.

When manually investigating the results, we found that it would have been better to select the testset manually. Randomly selecting 350 reviews resulted in many different categories (providers), with only a few documents per category. It would probably have been better if we selected only a few providers, and then selected 350 reviews of those providers for the testset. With only a few reviews in a category, chances are that these are not representative of the whole category. By now this testset was already annotated, so we used it for evaluation anyway.

## 8.2 Inter Annotator Agreement

The default measure to compute IAA is Cohen’s Kappa. However, this approach does not work very well for keyword selection[4][2]. The main reason is that Cohen’s Kappa requires the number of negative cases, eg non-keywords, to be known, but we do not know the number of keywords of a review beforehand. An alternative would be to compute Cohen’s Kappa for each word separately. But in that case, we would have many negative examples, as most words are not keywords. This would give a very distorted result.

Therefore, we compute IAA using F1 score instead. To compute the actual labels for each word, we created a "Golden standard" for each review. A term will be in this golden standard if it was selected by at least `min_annotations_for_gold_standard = max( $\lceil \frac{\text{num\_annotators}}{2} \rceil, 2$ )`, where `num_annotators` is the number of annotators that annotated the review. We can now compute the F1 for each annotation for each review, where an annotation is the list of all selected keywords. The resulting IAA is quite high: The average recall is 0.91, the average precision is 0.72, the average F1 is 0.76.

## 9 Hyperparameter tuning

Now that we have a testset, we can measure the performance of the algorithm. We again use the F1 measure for this and create the golden standard in the same way as in the previous section. The results are not very promising: An F1 of about 0.32. We see that the recall is a lot higher than precision: 0.46 vs 0.30. Manually investigating the keywords shows that the algorithm generally selects much more words as keywords. We also see that in many cases, there is something to say both for and against the annotations and the output of the algorithm. Two examples are given in Appendix B. In the first table, we see that the algorithm tends to select words that describe the entities (nouns), while the annotators select the 'feeling' the reviewer has about their treatment. In this case, the output of the algorithm seems to select better keywords, but it achieves a low F1 score. The second table shows another sample. Here we see that the output of the algorithm is too concise, and the keywords it selects are not descriptive for the review. The manual annotations are better, although they mostly capture the sentiment rather than the meaning.

This manual investigation shows us that the task might be too subjective to achieve a high score. The definition of a 'keyword' is different in different contexts and for different people. For some tasks, we mostly want to capture what type of entities the reviews talk about, for example the type of treatment or personnel. In another task, we might be more interested in the sentiment of the review, for example what someone thinks about the length of the waiting time or friendliness of the doctors.

<sup>10</sup><https://github.com/doccano/doccano>

<sup>11</sup>A dummy user has been created for a playground: `report_user`, `report_password`.

Table 1: Evaluation results for `testset_strict`, ordered by decreasing F1.

Precision	Recall	F1	$\alpha$	$n_0$	$\beta$	Corpus norm.	Category norm.	Include $di(t, d)$
0.31560	0.70146	0.40532	1.5	0.1	0.75	False	False	False
0.31584	0.70082	0.40519	2	0.1	0.75	False	False	False
0.31321	0.70058	0.40228	1.1	0.2	0.75	False	False	False
0.31163	0.70180	0.40177	1.1	0.1	0.75	False	False	False
0.31353	0.69597	0.40146	1.5	0.2	0.75	False	False	False
0.31289	0.69303	0.40055	2	0.2	0.75	False	False	False
0.31561	0.68782	0.40006	1.5	0.1	0.75	True	False	False
0.31669	0.68447	0.39993	2	0.1	0.75	True	False	False
0.31209	0.69090	0.39977	1.1	0.3	0.75	False	False	False
0.31158	0.68939	0.39893	1.5	0.3	0.75	False	False	False

Nevertheless, we decided to use this testset to tune our hyperparameters. For the initial algorithm, we used all the default values of all parameters, but they must be tuned on our dataset for optimal performance. We have defined seven different hyperparameters. Six of them are evaluated at once, we will discuss them here. The last hyperparameter is a little different, and we'll discuss it in the next section.

- $\alpha$  is a hyperparameter in Equation 2. As explained in Section 6.2.1, it defines the level of penalty for infrequent words where  $tf_c < n_0$ . Its value should be  $> 1$ , as a value  $< 1$  would mean an award those infrequent words. Toivanen et al set this value to 1.1. We evaluate performance with  $\alpha \in \{1.1, 1.5, 2\}$ .
- $n_0$  is also a hyperparameter in Equation 2. It defines the number of documents a term should occur in, for an optimal corpus level score. Toivanen et al set  $n_0$  to  $0.03 \cdot |D|$ , eg to 3% of the total number of documents. We evaluate with  $n_0 \in \{0.1 \cdot |D|, 0.2 \cdot |D|, 0.3 \cdot |D|\}$ .
- $\beta$  is a hyperparameter of Equation 11. It defines the weight of the category level score with respect to the corpus level score when combining them into the document level score. Toivanen et al set this value to 0.5, which results in a harmonic mean between the two scores. We evaluate with values  $\in \{0.25, 0.5, 0.75\}$ .
- **Normalizing the corpus level score per document.** Toivanen et al normalize the corpus score within each document. This means that for each document, the highest corpus level score has a value of 1. An alternative is to normalize the scores over the whole corpus, such that only the highest corpus level score over the whole corpus has a score of 1. Manual investigation shows that this probably makes the category level and corpus level scores more comparable. This parameter takes values  $\in \{\text{True}, \text{False}\}$ .
- **Normalizing the category level score per category.** Similar to the corpus level score, we define two options for category level score normalization. Normalizing within each category means that the highest score within each category has a value of 1. Normalizing over the whole corpus means the highest overall score has a value of one. This parameter takes values  $\in \{\text{True}, \text{False}\}$ .
- **Using the distance factor.** The distance factor is discussed in Section 6.2.4. This factor has a high influence on the final score for each term, and it is unclear whether the index of a term in the document should be so important when computing our keywords. This parameter indicates whether we include or exclude the distance factor in the final score. It takes values  $\in \{\text{True}, \text{False}\}$ .

As discussed above, we see that in many cases the output consists of more keywords than the golden standard. We have therefore created two versions of the testset. The first testset, `testset_strict`, is created as discussed in Section 8.2. In the second version, `testset_loose`, we set `min_annotations_for_gold_standard = 1`. This means that any keyword that was selected by any annotator, will be in the testset. This increases the number of keywords; we measure our performance on this test set as well.

For all combinations of the above parameters, we compute our performance on both testsets. This provides us with two tables of 216 rows. We provide the 10 rows with the highest F1 of both tables in Tables 1 and 2. Although there is not one clear best value for each parameter, we select a value for each of them:

Table 2: Evaluation results for `testset_loose`, ordered by decreasing F1.

Precision	Recall	F1	$\alpha$	$n_0$	$\beta$	Corpus norm.	Category norm.	Include $di(t, d)$
0.52072	0.68654	0.56619	1.5	0.1	0.75	False	False	False
0.52087	0.68714	0.56607	2	0.1	0.75	False	False	False
0.51692	0.68946	0.56438	1.1	0.1	0.75	False	False	False
0.51817	0.68428	0.56282	2	0.2	0.75	False	False	False
0.52395	0.67451	0.56243	1.5	0.1	0.75	True	False	False
0.52553	0.67067	0.56177	2	0.1	0.75	True	False	False
0.51596	0.68123	0.56091	1.1	0.2	0.75	False	False	False
0.51623	0.68232	0.56088	1.5	0.3	0.75	False	False	False
0.51625	0.68188	0.56080	1.5	0.2	0.75	False	False	False
0.51687	0.67621	0.56034	2	0.1	0.5	False	False	False

- $\alpha = 1.5$ . For both sets, there is no clear best value. It seems this value is not to be of much importance. We choose the middle value.
- $n_0 = 0.1 \cdot |D|$ . There is no obvious best value for this parameter, but 0.1 seems like a good value. This is a lot higher than the 0.03 Toivanen et al used, but early manual investigation showed that a higher value works better on our data. If the value is too low, the algorithm will select too infrequent keywords, which are not of any interest to us.
- $\beta = 0.75$ . This value clearly is the best option on both sets. We investigated even higher values, of 0.8 and 0.9. These do not seem to increase performance any further, so we choose 0.75. This means the category level score is weighted 3 times as much as the corpus level score.
- Normalizing the corpus level score per document = False. Almost all configurations in the top 10 have this value at False, so we will normalize over the whole corpus.
- Normalizing the category level score per category = False. Again, all configurations in the top 10 have this value at False, so we will normalize over the whole corpus.
- Using the distance factor = False. Again, all configurations in the top 10 have this value at False, so we exclude the distance factor in the final score.

## 9.1 Tuning POS-weights

The last 'parameter' is the configuration of POS-weights. Toivanen et al do not provide much information about the weighting scheme they used. For the evaluation discussed above, we used the following weights:

NOUN = 1  
 VERB = .5  
 ADJ = .8  
 ADV = .7

The available POS tags can be found at [universaldependencies.org](https://universaldependencies.org)<sup>12</sup>. This list of POS weights was set manually. The idea is that nouns contain the most information in a review, and adjectives provide more context. Verbs and their adverbs often contain important information as well, although less than nouns. Any term with a POS tag not in the above list gets a weight of 0.1. We do not set it

<sup>12</sup><https://universaldependencies.org/docs/u/pos/>

Table 3: POS-weight evaluation results for `testset_loose`, ordered by decreasing F1. The POS weight values are for the tags NOUN, VERB, ADJ, ADV.

Precision	Recall	F1	POS weights
0.53787	0.70287	0.58204	1, 0.5, 1, 0.5
0.52328	0.68424	0.56755	1, 0.5, 0.8, 0.25
0.52328	0.68424	0.56755	1, 0.5, 0.8, 0.25
0.52072	0.68654	0.56619	1, 0.5, 0.8, 0.7
0.50089	0.70342	0.55704	1, 0.8, 0.8, 0.8
0.49100	0.70270	0.54997	1, 1, 1, 1

at 0 for two reasons. Firstly if the document score is extremely high for a certain term, it should be able to become a keyword, no matter its POS tag. Secondly, we know that POS-tagging is not 100% accurate. If a term gets assigned the wrong POS tag, we still want it to be possible for it to become a keyword, if its document score is very high.

We evaluated different POS weight configurations on the test set. We have set the other six hyperparameters as defined above and tested seven different POS weight configurations. The resulting scores for both testsets are shown in Tables 3 and 4. We choose our POS weight configuration based on these results as

NOUN = 1  
 VERB = .5  
 ADJ = 1  
 ADV = .5

Evaluating our hyperparameters increased our F1 score on the strict testset by 0.1: from 0.32 to 0.42. On the loose testset, our highest F1 score is 0.58. These scores are not very high. But if we compare them to the scores Toivanen et al achieved on their test sets [8], they are quite alike. As with their testsets, although IKE often does not select the keywords annotated by humans, in many cases they are still feasible.

## 10 Proof of concept: Sentiment & Topic analysis

Now that we have implemented and configured our algorithm, we would like to use the keywords for sentiment analysis.

As a proof of concept, we want to be able to show the top-scoring keywords for a given provider, in the class 'positive' and 'negative'. Therefore we assign a binary class to each review as we did in Section 8: Reviews with an average grade < 7 get class 0, other reviews get class 1. Now for clustering the documents we base the clusters on the combination of provider and class: for each provider, we create two clusters, one for the negative reviews and one for the positive reviews.

After we have computed the scores with these new categories, we can input a provider. For both classes, we select the 10 terms with the highest final score and plot the results. Two examples of the output are shown in the first two figures in Appendix C. We see that in some cases a term occurs as a keyword in both classes. In such a case, we get little information from the plot, as people are both positive and negative about the subject. We also see that some words occur as a keyword for many different providers. This is because the corpus level score is not dependent on the category. If a term has a very high corpus level score, its final score can be high enough to occur as keywords for many providers. For these words, we also get little information.

### 10.1 Using the syntactic dependency tree

We see that using this analysis, a single keyword often needs more context to be useful. We would therefore like to provide some words that are informative and occur within the context of the keyword. SpacyR is able to provide the dependency tree of the sentences as well. When using this, it adds `token_id`, `head_token_id`, and `dep_rel` values for each term. We can use these relations to add context to the keywords, by adding terms with a high final score that occur as a dependency in one of the sentences of the keyword. For each keyword, we extract in which sentences the word occurs within the same category. For each sentence, we extract all descendants of the keyword in the dependency tree. We then select the  $n$  dependencies with the highest final score. This way we select words that

Table 4: POS-weight evaluation results for `testset_strict`, ordered by decreasing F1. The POS weight values are for the tags NOUN, VERB, ADJ, ADV.

Precision	Recall	F1	POS weights
0.32988	0.73162	0.42235	1, 0,5, 1, 0,5
0.31784	0.70087	0.40750	1, 0,5, 0,8, 0,25
0.31784	0.70087	0.40750	1, 0,5, 0,8, 0,25
0.31560	0.70146	0.40532	1, 0,5, 0,8, 0,7
0.30428	0.72504	0.39814	1, 1, 1, 1
0.30210	0.71827	0.39439	1, 0,8, 0,8, 0,8

Table 5: Keywords per province on ‘positive’ reviews. The keywords are ordered based on the decreasing final score.

Gelderland	Noord-Holland	Zuid-Holland	Noord-Brabant	Fryslân	Limburg	Overijssel	Drenthe	Utrecht	Flevoland	Zeeland	Groningen
duidelijk	luisteren	tevreden	tevreden	luisteren	duidelijk	tevreden	luisteren	tb	duidelijk	tevreden	tevreden
prettig	snel	behandelen	luisteren	tevreden	uitleg	prettig	lymph	tevreden	tevreden	duidelijk	luisteren
uitleg	prettig	luisteren	duidelijk	duidelijk	tevreden	luisteren	tevreden	luisteren	prettig	prettig	duidelijk
luisteren	duidelijk	snel	behandelen	behandelen	snel	behandelen	behandelen	duidelijk	dag	snel	snel
zorg	tevreden	duidelijk	snel	prima	luisteren	duidelijk	prettig	tevreden	uitleg	uitleg	behandelen
tevreden	behandelen	dag	zorg	zorg	behandelen	snel	medewerker	behandelen	dr.	zorg	prettig
snel	uitleg	zorg	uitleg	uitleg	dag	zorg	snel	prettig	snel	behandelen	uitleg
dag	nemen	uitleg	dag	dag	behelpzaam	prima	uitleg	dag	luisteren	dag	zorg
behandelen	ervaring	ervaring	fijn	prettig	zorg	ervaring	prima	snel	prima	prima	ervaring
patiënt	dag	prettig	prettig	patiënt	prettig	uitleg	dag	medewerker	medewerker	medewerker	medewerker

Table 6: Keywords per province on ‘negative’ reviews. The keywords are ordered based on the decreasing final score.

Gelderland	Noord-Holland	Zuid-Holland	Noord-Brabant	Fryslân	Limburg	Overijssel	Drenthe	Utrecht	Flevoland	Zeeland	Groningen
dag	dag	dag	dag	dag	dag	dag	dag	dag	patiënt	dag	dag
patiënt	slecht	slecht	week	patiënt	patiënt	patiënt	patiënt	patiënt	dag	patiënt	slecht
week	week	patiënt	slecht	luisteren	slecht	slecht	slecht	slecht	slecht	uur	week
slecht	patiënt	maken	patiënt	slecht	week	week	luisteren	luisteren	uur	slecht	jaar
luisteren	uur	uur	luisteren	ervaring	uur	uur	week	week	week	week	behandelen
uur	luisteren	week	uur	keer	luisteren	keer	keer	uur	pijn	jaar	patiënt
behandelen	keer	luisteren	krijgen	jaar	behandelen	behandelen	uur	behandelen	afdeling	duidelijk	klacht
pijn	behandelen	keer	behandelen	onderzoek	keer	ervaring	klacht	keer	keer	pijn	keer
keer	pijn	nemen	ervaring	week	klacht	pijn	ervaring	informatie	2	zorg	ervaring
ervaring	ervaring	krijgen	klacht	mens	informatie	jaar	behandelen	klacht	echt	klacht	pijn

are both informative and occur in the context of the keyword. Two sample plots of the new analysis are shown in the last two figures of Appendix C.

We see that the dependencies provide useful context. For many keywords, it becomes clear what aspects are experienced as positive or negative. However, the fact remains that some words occur in both classes. We also see that some words with a positive sentiment occur as a dependency in the negative class, and vice versa. The main problem is that we classify a complete review into one class, based on the average grade. However, in many cases, a review is positive about some aspects and negative about others. This distinction is not made in this analysis, which is why the results sometimes look weird.

It is hard to draw conclusions from this analysis. A word occurring in the positive class for a certain provider does not necessarily mean that all people are positive about that topic, likewise for the negative class. We also see some words that seem not too informative, especially without their dependencies. As for future work, it might be an idea to pre-define a list of topics we’d like to score providers on. We might be able to find out which topics occur often in which classes for which providers, and thereby conclude that the provider scores good or bad on that topic. Another option is to select keywords for providers in a more sophisticated way. The algorithm selects keywords per document, but these do not have to be keywords for the providers. We might for example select all keywords of all reviews in the category, and reorder them based on their occurrence within the category. If we then take the top 10 words, these are words that hold information and occur often within the category. In any case, the issue that remains is that we won’t be able to know how well the algorithm performs unless we create a testset for this task as well. However, this was outside the scope of this project, as we are mainly showing a proof of concept.

## 11 Proof of concept: Geographical analysis

As another proof of concept, we would like to apply the keywords in the context of geographical analysis. As mentioned, the address of a provider is also stated on ZorgkaartNederland. We scraped all the addresses and used nlgeocoder<sup>13</sup> to parse the strings into locations. For this analysis, we subset the data, by only selecting reviews of the provider type “Hospital”. For a start, we checked for major differences between the grades given over the different provinces in the Netherlands. A pie chart for each province is shown in Figure 9. We see that the grades do differ per province, but not significantly.

Now similar to the previous section, we divide the dataset into new categories. For each review, we base the category on the combination of the province and the grade class. Hence we get 24 categories since we have 12 provinces. We compute the scores and select the top 10 as keywords for each category. The results are shown in Tables 5 and 6.

As with the sentiment analysis, we again see many overlapping words over the different provinces. This again shows us that we miss the context. The scores are excluded in the tables for the sake of readability, but the scores are all very close to each other. The final score of term 10 in the list is very

<sup>13</sup><https://cran.r-project.org/web/packages/nlgeocoder/index.html>

similar to the term on position 100. By taking only the first 10 words, we probably limit ourselves too much. In this representation, the keywords per province and grade class provide us with little information. However, we do see for some of the keywords that they clearly belong to the grade class they are assigned to. This shows that at least some of the desired information is captured.

For future work, we might want to extend this analysis. We could for example select much more keywords per class, for example 1000 each. We can then try to cluster the keywords by means of a clustering algorithm. A cluster consists of similar keywords, and clusters with a high average IKE score are important within the class. Clusters of terms provide more information than single keywords, and especially more context. This would probably make the analysis much more useful. Using this approach we are also more certain of more diversity between classes, which makes the result more informative. In our opinion, the approach of using keywords for geographical analysis could be very useful, but it needs more research.

## 12 Reflection & Future work

The internship has been very educative and fun. We have encountered many different aspects and challenges of NLP. We have created a dataset ourselves, by scraping the website of ZorgkaartNederland. We then built and tuned an algorithm to extract keywords from the texts of those reviews. The idea of those words is that we are able to capture most of the information in the text, by only selecting a few keywords. We are then able to represent the dataset by using solely these keywords only, making it much more compact, and easier to represent. The main challenge in this project was the task of evaluation, because of two reasons. The first reason is that we did not have a testset containing reviews with their corresponding keywords. We solved this problem by annotating part of the dataset with our team at [REDACTED], hereby creating a labeled testset of about 350 reviews. The second and more fundamental problem is the subjectivity around keywords. There is no strict rule set with which a term must comply to become a keyword, and humans definitely do not always agree about the keyword selection. When we evaluated our algorithm on the annotated testset, we clearly see that this is a problem. In many cases, the output of the algorithm does not align with the manual annotations, although both options often seem feasible.

For future work, it would be nice to investigate whether one can use IKE to answer questions regarding the topics discussed in Section 4. The hyperparameters of the algorithm probably need to be tweaked for different problems. One also might want to include other information, as we did with the dependency tree information in sentiment analysis. As we saw in the sentiment/topic analysis, we need to think of a clever way to generalize keywords to higher levels. Terms that are important in a document, are not necessarily informative when looking at the grade-provider combination of that document. In any case, for evaluating performance one will probably need to annotate a labeled testset manually. Testing the algorithm on other but alike datasets would also be very interesting. For example the reviews of [REDACTED] application SamenGezond. If the dataset is large enough and consists of short documents, keyword extraction using IKE should work quite well.

To conclude, I would like to thank [REDACTED] for the supervision at [REDACTED]. Our regular discussions have been very helpful in the process, and I really enjoyed my time at [REDACTED]. I would also like to thank Faegheh Hasibi for supervising on behalf of Radboud University.

## References

- [1] Boer, D. de, Wiegers, T., Bos, N., Springvloet, L., Jong, J. de, and Friele, R. De Transparantiemonitor 2018. ZorgkaartNederland: Hoe draagt ZorgkaartNederland bij aan transparantie en de best passende zorg voor patiënten?, 2019.
- [2] Alex Brandsen, Suzan Verberne, Milco Wansleeben, and Karsten Lambers. Creating a Dataset for Named Entity Recognition in the Archaeology Domain. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4573–4577, Marseille, France, May 2020. European Language Resources Association.
- [3] Olga Damman. Public reporting about healthcare users’ experiences: The Consumer Quality Index. *Journal of Stroke & Cerebrovascular Diseases - J STROKE CEREBROVASC DIS*, January 2010.
- [4] Louise Deleger, Qi Li, Todd Lingren, Megan Kaiser, Katalin Molnar, Laura Stoutenborough, Michal Kouril, Keith Marsolo, and Imre Solti. Building Gold Standard Corpora for Medical Natural Language Processing Tasks. *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, 2012:144–153, November 2012.
- [5] Rosanne Geesink. Increasing importance of patient ratings. *A study on the validity and reliability of ZorgkaartNederland. nl. Amsterdam: Vrije Universiteit Amsterdam, Policy and Organization of Healthcare*, pages 1–47, 2013.
- [6] S. M. Kleefstra. *Hearing the patient’s voice: The patient’s perspective as outcome measure in monitoring the quality of hospital care*. 2016.
- [7] Mika Timonen, Timo Toivanen, Melissa Kasari, Yue Teng, Chao Cheng, and Liang He. *Keyword Extraction from Short Documents Using Three Levels of Word Evaluation*, volume 415. January 2013. Journal Abbreviation: Communications in Computer and Information Science Pages: 146 Publication Title: Communications in Computer and Information Science.
- [8] Mika Timonen, Timo Toivanen, Yue Teng, Chao Chen, and Liang He. *Informativeness-based Keyword Extraction from Short Documents*. January 2012. Journal Abbreviation: KDIR 2012 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval Pages: 421 Publication Title: KDIR 2012 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval.
- [9] Mika Juhani Timonen. Categorization of Very Short Documents. In *Proceedings of the 4th International Conference on Knowledge Discovery and Information Retrieval (KDIR 2012)*, pages 5–16, October 2012.

## A Samples of TF-IDF scores

Table 7: Keywords by highest-scoring TF-IDF values of a sample review. We clearly see the TF=1 problem. The review text is "Ik ben goed geholpen en kon altijd met mijn vragen terecht"

<b>Term</b>	<b>TF</b>	<b>IDF</b>	<b>TF-IDF</b>
terecht	0.09091	3.50825	0.31893
kon	0.09091	3.05620	0.27784
vragen	0.09091	2.83390	0.25763
geholpen	0.09091	2.51011	0.22819
altijd	0.09091	1.98128	0.18012

Table 8: Keywords by highest-scoring TF-IDF values of a sample review. We clearly see the problem with parsing/spelling errors. The review text is "in 2017 geholpen aan staar, beide ogen hebben een behandeling gehad.Totaal geen napijn, negatieve reactie, behandeling is verlopen zoals kenbaar gemaakt door de mensen in de kliniek.Geweldige ervaring weer scherp te kunnen zien, mede door de mensen van Bergman"

<b>Term</b>	<b>TF</b>	<b>IDF</b>	<b>TF-IDF</b>
kliniek.geweldige	0.02564	13.37362	0.34291
gehad.totaal	0.02564	12.27500	0.31474
kenbaar	0.02564	7.44669	0.19094
scherp	0.02564	7.39473	0.18961
napijn	0.02564	7.20820	0.18483

Table 9: Keywords by highest-scoring TF-IDF values of a sample review. We see that if a word happens to have a  $TF > 1$ , it is very likely to be a keyword. The review text is "Prima ziekenhuis met een uitstekende behandeling en zorg. Ze denken met de patiënt mee. Dit omdat er twee operaties na elkaar werden uitgevoerd door twee verschillende artsen."

<b>Term</b>	<b>TF</b>	<b>IDF</b>	<b>TF-IDF</b>
twee	0.07407	3.65165	0.27049
operaties	0.03704	6.18520	0.22908
uitgevoerd	0.03704	4.82959	0.17887
uitstekende	0.03704	4.53000	0.16778
denken	0.03704	4.52971	0.16777



## B Sample outputs of IKE versus the manually annotated test-set

Table 10: Sample output of the algorithm vs the annotations. It seems the output of the algorithm is better. The review text is "De zeer ervaren orthomanueel therapeuten bieden uitstekende rugzorg. In drie behandelingen is mijn rug recht gezet en dat zonder kraken! Deze behandelmethode is absoluut aan te bevelen boven reguliere manuele therapie."

Output	Annotations
behandeling	ervaren
therapeut	
uitstekend	

Table 11: Another sample output of the algorithm vs the annotations. It seems the manual annotations are better, although it only captures the sentiment. The review text is "Was opgeroepen voor klachten op de borst. Werd in 1 ochtend door het hele team onderzocht (bloedprikken, hart filmpje, inspannings test en voorbespreking en na bespreking. Alles gebeurde op tijd. Personeel zeer vriendelijk en uitleg van tests en eindbespreking zeer duidelijk. Wil een groot compliment maken naar het hele team"

Output	Annotations
klacht	tijd
borst	vriendelijk
	duidelijk
	compliment

## C Example outputs of sentiment analysis

Figure 5: Example output of sentiment analysis on a sample provider.

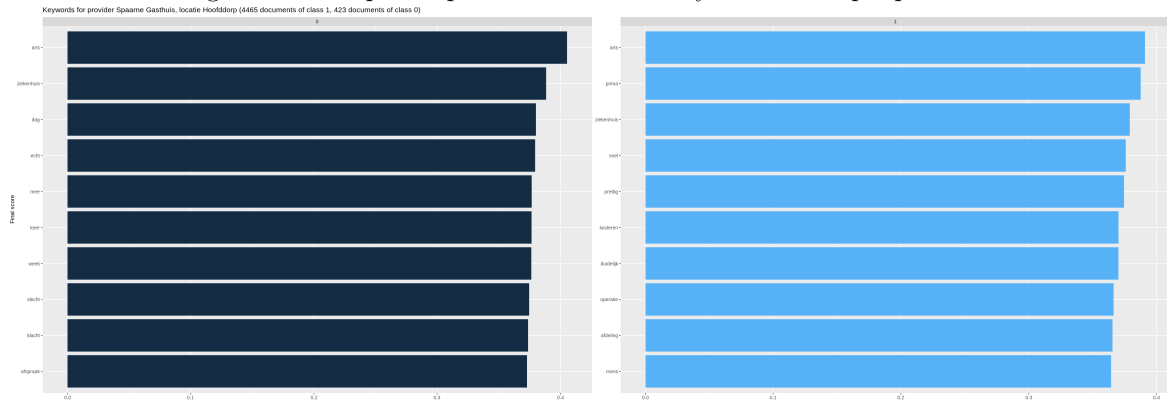


Figure 10 consists of two horizontal bar charts, (a) and (b), showing the F1 score for 12 keywords. The x-axis for both charts represents the F1 score, ranging from 0.0 to 0.4. The y-axis lists the keywords: sport, price, item, event, location, vintage, gift, artist, book, painting, and costume.

Chart (a) shows the F1 score for class 0 (dark blue bars). The F1 scores are approximately: sport (0.38), price (0.37), item (0.36), event (0.35), location (0.34), vintage (0.33), gift (0.32), artist (0.31), book (0.30), painting (0.29), and costume (0.28).

Chart (b) shows the F1 score for class 1 (light blue bars). The F1 scores are approximately: sport (0.39), price (0.38), item (0.37), event (0.36), location (0.35), vintage (0.34), gift (0.33), artist (0.32), book (0.31), painting (0.30), and costume (0.29).

[illegible][illegible]

## D Geographical analysis

Figure 9: Distributions of the grades per province of the Netherlands.



## E Examples of reviews posted on ZorgKaartNederland



Figure 10: A review with a low grade and short text.

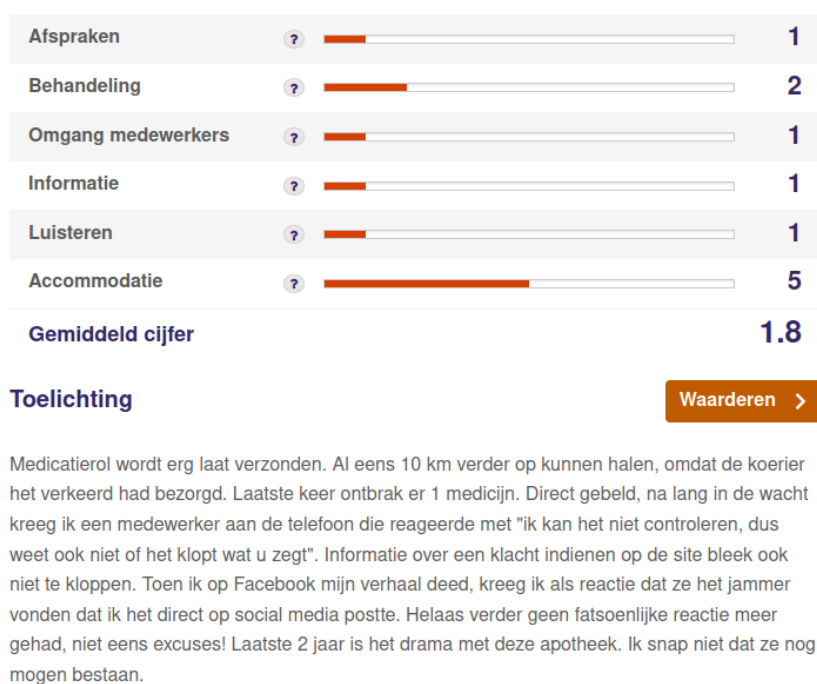
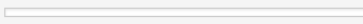



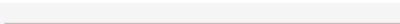



Figure 11: A review with a low grade and a more extensive text.

Afspraken		N.v.t.
Behandeling		N.v.t.
Omgang medewerkers		10
Informatie		8
Luisteren		10
Accommodatie		10
<b>Gemiddeld cijfer</b>		<b>9.5</b>
Aandoening / behandeling Vitamine B12		

### Toelichting

Waarderen >

Ik wilde een waardering plaatsen, omdat ik erg blij ben met de apotheek en hun klantvriendelijkheid. Een van de apothekersassistentes weet zelfs mijn naam uit haar hoofd, terwijl ik niet eens vaak langskom. Ze is ook ontzettend vriendelijk en behulpzaam. We hebben laatst een praatje gemaakt en dat was erg fijn. Ook de andere assistentes zijn aardig en behulpzaam, het contact is meestal wel kort natuurlijk (recept ophalen en dan weer weg).

Figure 12: A review with a high grade and a more extensive text.

Luisteren		9
Uitleg		9
Deskundigheid		9
Voor- en nadelen		9
Samen beslissen		9
Samenwerking		9
Behandeling		9
<b>Gemiddeld cijfer</b>		<b>9.0</b>

### Toelichting

Waarderen >

Prachtige groene omgeving, makkelijk parkeren. Kleine maar goed geoutilleerde praktijk. Geweldig team dat je op je gemak weet te stellen. Ontspannen sfeer. Arts die niet alleen naar je luistert, maar ook hoort wat je zegt. Bereid om buiten de box te denken. Je wordt overal bij betrokken, zo kun je meekijken met de echo en wordt de brief met eindconclusie tijdens het intypen meteen met je doorgenomen. Ik kwam voor een second opinion en heb uitgebreid antwoord gekregen op al mijn vragen. Zelfs nog wat extra tips. Was niet bij ons in de buurt, maar de extra kilometers die we hebben gereden meer dan waard.

Figure 13: A review with a high grade and new grade distribution.