Universiteit Leiden
The Netherlands

Opleiding Informatica

Improving Twitter Stream Filtering using Pseudo-Relevance Feedback

Bart de Zoete

Supervisors:
Suzan Verberne & Frank Takes

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

9/07/2020

Abstract

Researchers often create Twitter datasets by searching for a few search terms using the Twitter API. Using a few search terms can, however, cause a large portion of topic relevant data to be lost. We experimented with the usage of Pseudo-Relevance Feedback (PRF) to relieve this issue. PRF is a query expansion technique that can find more Tweets than would be found otherwise. For our experiments we created a custom dataset about the 2019 Coronavirus outbreak, using 23 search terms. Our experiments showed that PRF is often highly effective at finding additional topic relevant Tweets. For queries consisting of two terms that individually find few Tweets, PRF obtained an F1 score between 0.4 and 0.8 in 50% of these combinations. For queries consisting of two terms that by themselves find a large number of Tweets, 50% got F1 scores between 0.79 and 0.83 after applying PRF. These results were obtained and validated on two different, random subsets of the dataset.

Contents

1 Introduction

Twitter's numerous users generate a lot of data which contains valuable information. Hence, Twitter has naturally become a common subject in the field of social media analytics. The nature of the data makes it challenging to use: user generated data can be of low quality, and Tweets are short causing them to contain little data.
When researchers work with Twitter data, they are most commonly interested in just a certain topic. This is for example often the case in sentiment analysis, where researchers try to categorize opinions about a certain topic. In order to handle the large amount of available Twitter data, the Twitter API is often used to filter the Twitter stream. This can be done by performing search queries with the API. If proper search terms are used, only topic relevant Tweets will be found.

This thesis is part of the RISE_SMA project[1]. RISE_SMA is an international collaboration that aims to improve the usage of social media analytics (SMA) to aid in society and crisis communication. In this project, it was observed that researchers often use just a single or a few search terms in their queries for the Twitter API.
Since Twitter has many users, topics are often discussed by a large number of different people. As a result, topics tend to also be discussed through various different phrases. Because of this, when just a few search terms are used, a large portion of topic relevant Tweets can be missed. Single Tweets contain little information due to the 280 character limit. Therefore, a large amount of data is required for datasets. Having a dataset that is not only large, but also more complete, will make the task for which a dataset was collected more reliable.

In this thesis, we experiment with a *query expansion* technique called "Pseudo-Relevance Feedback", or PRF (see Section 3.3). In short, PRF first performs the initial search query. Within the results, the most relevant terms relating to the query are found. By adding these terms to the query, the new query should allow to find more topic relevant Tweets. We have experimented with PRF to find additional topic relevant Tweets when a query of just a few terms is provided. Our research question is thus: "When starting with an initial query of just a few search terms, how effective is Pseudo-Relevance Feedback for finding supplementary topic relevant Tweets?"

1.1 Thesis overview

In Section 2, we discuss background information that is relevant to our research, including prior research. All methodologies that we used are explained in detail in Section 3. Next in Section 4 we first state in detail how and which experiments we have performed. We then go on to present and interpret the results of the experiments. A discussion of our research and proposals for further research can be found in Section 5. To end the thesis we sum up what our research showed and answer our research question in Section 6.

[1] https://social-media-analytics.org/

2 Background

In this Section, we describe background information that is relevant to this thesis. First, we provide more information about the problem that this thesis addresses. Secondly, we look at previous research and how it differs from what we will research in this thesis. Thirdly, we provide a brief overview of the tools which we used in our research.

2.1 Problem Background

Twitter is a popular microblogging service. Users post microblogs, so-called Tweets, which are shown to the people whom follow this user. Tweets can be retweeted by other users, showing the same post to their Followers. Data generated on Twitter has some interesting characteristics. For example, Tweets are of short length, and of the symbols # and @ are used regularly. The former is used to indicate a "hashtag" which is used to categorize Tweets. The latter is used to indicate a "mention" which is a way of tagging other users to a post.
A lot of data is generated on Twitter. Researchers such as social scientists often create Twitter datasets using the *Twitter API* (see Section 2.3). To this aim, they generally search for one or two search terms. However, Twitter users frequently use different phrases and hashtags to refer to the same topic. When creating datasets with a few search terms this can lead to significant data loss as only Tweets containing the specific search terms are found. In our dataset, just 77% of topic relevant Tweets can be found using the top two search terms.
As individual Tweets contain little data, numerous Tweets are needed to make a dataset useful. Moreover, having a more complete dataset can make whichever task is performed on the data more reliable. It would thus be useful to be able to find additional topic relevant Tweets that would not be found when using just a few search terms. In this thesis, we research how well we can find additional Tweets using the technique *Pseudo-Relevance Feedback*.

2.2 Related Work

One issue when working with the Twitter API (see Section 2.3), is that it returns a sample of all Tweets. Twitter also provides the *Firehose*, which is an API that returns all Tweets. Morstratter et. al. [MPLC13] studied the differences in data returned by the Firehose and the regular Twitter API. They showed that the two APIs can return nearly identical, but also quite differing results. This depends on the type of analysis for which the data is collected. One of their results was that the coverage of the Twitter API is lower when more Tweets match a query. We created a dataset about a topic which was highly popular at the time (see Section 3.1.2). Due to the high cost of the Firehose, we had to use the Twitter API, and thus we likely have not found all relevant Tweets. For simplicity, we will not take the sampling of the API into account. Instead we act as if the Twitter API returns all Tweets.
Massoudi et al. [MTDRW11] studied the effects of query expansion on Twitter data. They did this by taking textual and microblog specific quality indicators into account. Expansion terms were chosen from Tweets that were sent within a specific time from the time that the query was submitted. Our research differs from this as we do not take any quality measures into account. We also find search terms in the most relevant documents, instead of the most recent documents at the time of performing the query.

Others such as [ZLS16] and [WHF17] use external knowledge bases to improve query expansion for Twitter data. Tweets contain little information which can make it challenging to find adequate expansion terms. To overcome this issue, more information to find expansion terms can be found in an external dataset regarding the queries subject. We do not use this approach as it cannot be used when no external knowledge bases exist for a topic. This can for example be the case when Tweets need to be found regarding a current event. As we want our method to work for such topics, it has to work using only data from Tweets.
The effects of PRF as a query expansion method on Twitter data has been studied only little. One frequently stated problem with PRF is that expansion terms that are found can be harmful [CNGR08], causing irrelevant documents to be found. Most papers on PRF for Twitter data aim to relieve this issue or improve the usage of PRF in other ways. Miyanishi et al. [MSU13] attempt to prevent harmful expansion terms from being added by manual Tweet selection. They handpick a single Tweet from the initial search results that is certainly relevant to the topic. This Tweet is then used for the basis of the PRF. We, however, want to use PRF via a fully automated process. To prevent harmful search terms from being added, we have attempted to optimize the k and m parameters of PRF.
Our research specifically focuses on the problem described above in Section 2.1. We start with a few search terms and aim to find as many Tweets relating to the topic as possible. For this, we use PRF with optimized settings to try and strike a good balance between recall and precision (see Section 3.4). Different seed search terms are used to see how the choice of seed terms impacts the performance of PRF.

2.3 Tools

For the data collection and experimentation in this thesis, we made use of two tools.

Twitter API
Datasets on Twitter data are often created using the Twitter API[2]. This official API from Twitter allows for programmatic retrieval and analysis of Twitter data. In order to get access to the API, a Twitter Developer account is required. This has to be obtained through a process where Twitter manually checks if your usage of the API is legitimate. Once the account is acquired, authentication keys can be generated which programs can use to access the API.
Working with the API can be quite tricky as Twitter puts many limitations on the API. For example, there are limits to how many API requests can be performed per 15 minutes. Working with these restrictions, and other aspects of the API can be made easier by using a wrapper. Several wrappers are available for different programming languages.
To create the dataset for this thesis, we used the Python wrapper *Tweepy*[3]. Our choice went to Tweepy for several reasons:

- Tweepy provides all the features that were needed for this project, mostly relating to performing automatic search requests.
- Documentation is extensive and up-to-date.

[2] https://developer.twitter.com/en/docs
[3] https://www.tweepy.org/

- It has many users, making it so that a lot of other developers could easily work with our code.
- Twitter API limitations can be handled automatically.

Indri

Experimentation with PRF is often done by researchers using the open-source search engine *Indri*[4], which is also the tool that we have used. Indri is part of the *Lemur Project* which develops several different tools for various applications, such as research in information retrieval and text mining. Indri provides several methods for improving search performance, such as built-in PRF functionalities. To work with Indri, a *search index* must first be created for the desired dataset. A search index is a type of data structure that is designed for fast retrieval, ~~whilst keeping accuracy high.~~

[4] https://www.lemurproject.org/indri/

3 Methods

In this Section, we first discuss the dataset that was used for our experiments. After that, we describe what preprocessing we performed on the data. Then we describe the methodology for the experiments themselves. We give a more detailed description of the method which we used to find additional Tweets, state how methods are evaluated, what baseline method we use, and discuss the methods which we experimented with. All programming is done in Python 3.

3.1 Dataset

In this Section, we discuss why we created a custom dataset instead of using an existing dataset, how the dataset was collected, and then we sum up some statistics of the dataset.

3.1.1 Dataset selection criteria

For this project we needed a dataset that met the following requirements:

- The dataset contains a large number of Tweets.
- All the Tweets in the dataset originate from a certain time span.
- A subset of the Tweets are about a certain topic and labeled as such.

Using a dataset that satisfies these requirements, we can start with seed search terms and attempt to retrieve as many of the topic relevant Tweets as possible.
There are two problems with existing datasets that we came across in our research.
Firstly, datasets often only contain Tweets that are about a certain topic. With these datasets, every single Tweet that we would find will always be about the topic, which is not useful in this research. We could add unrelated Tweets to such a dataset, but since the Twitter API limits searches to only the past seven days, we could not add Tweets from the same time span.
Secondly, there are datasets that contain Tweets about several different topics. Here it would be possible to search for Tweets about a certain topic and also have topic unrelated Tweets. However, in these datasets it is not labeled which topic belongs to which Tweet.
Because existing datasets did not satisfy our requirements, we created a new dataset.

3.1.2 Coronavirus

The Twitter API limits searches to just the last seven days. Because of this, we had to create a dataset about a topic that was relevant at the time. The 2019 Coronavirus outbreak which started in late 2019 and causes the disease COVID-19 provided a good topic to build a dataset around. Online coverage of the outbreak was vast, so a lot of topic relevant Tweets could be gathered. Users also used a large number of different hashtags in their Tweets, so not all Tweets could be found using just a few search terms. All this together made the outbreak a well-suited topic for a dataset.

3.1.3 Data Collection

To create the dataset, we first had to gather as many Tweets as possible. Then we had to find Tweets about the 2019 Coronavirus outbreak that were sent within the same time frame. Due to

API limitations, we had a total of 7 days to gather all the data. Both of these datasets had to consist of a continuous stream of Tweets.
One thing to note is that it is not possible to find *all* the Tweets that were sent in some time frame. Only Tweets that are available through the Twitter API can be found. This excludes Tweets sent from private accounts and Tweets that were deleted since being sent. Furthermore, the Twitter API also provides only a sample of all Tweets. The Twitter Firehose can be used to analyze all Tweets, but this is expensive and highly resource consuming. More on this can be found in Section 2. Nevertheless, we can collect the vast majority of the Tweets that are sent.

Creating a continuous stream
Making sure that a dataset is a continuous stream is not trivial. The Twitter API only returns a maximum of one hundred Tweets per API request. By making good use of the parameters that can be set for search requests, several requests can be used to form a continuous stream.
Setting the `result mode` parameter to "recent" makes it so that only the most recent Tweets are returned. Combining this with the `max_id` parameter, a continuous stream of Tweets can be made. `max_id` specifies that we only want to find Tweets with an ID lower than the chosen value. Newer Tweets have higher IDs, so using `max_id` we only find Tweets that are older than the chosen ID. Thus by setting the `max_id` to an initial value and only requesting the most recent Tweets, we can request the hundred most recent Tweets that were sent before that initial value. From the results, we can now take the ID of the oldest Tweet and use this as the new `max_id`. This way we essentially go back in time, finding all the available Tweets in batches of a hundred. We used this method to find all the Tweets for the dataset.

Filtering Retweets
For both datasets, we filtered out Retweets whilst searching. This can be achieved by adding the text "filter:-retweets" to the search query.
We filtered Retweets out because a Tweet that was popular and retweeted often would likely appear often in our dataset as well, but we want to avoid such repeats. It can be argued that these Tweets are more important and thus they should appear more often. We, however, accounted for the importance by storing the number of Retweets and Favorites that a Tweet has gotten. Those values were collected about a week after the Tweets were sent. If these values would be stored during the collection of the dataset, some Tweets would be so recent that they could not possibly have gotten retweeted or favorited yet.
Another reason to filter Retweets, is that the Twitter API makes it impossible to retrieve the full text of Retweets. This is possible for original Tweets which makes them preferable.

Collecting topic unrelated Tweets
The first step in creating our dataset was to collect as many Tweets as possible with no particular topic. This can be done by providing the search query '*' when making the API request. The star is interpreted as a regular expression, so all Tweets are matched and found with this search query. Initially, we set the starting value of `max_id` to the ID of the Tweet that was last send when the dataset creation started. Using the method above we collected Tweets for about five days so that we would have two days left to find Tweets about the Coronavirus outbreak. We ended up with a dataset of 23,824,607 Tweets total.

Collecting Coronavirus related Tweets

For the next step, we had to find Tweets about the Coronavirus outbreak that were sent in the same time span as the full dataset. We gathered a list of 23 search terms that would guarantee to only find Tweets that relate to the topic and find as many as possible. This list was created using the input of three people who are not related to the research in order to minimize bias. To avoid finding Tweets outside the time frame, the search parameter called `since_id` was set to the ID of the oldest Tweet in the full dataset. This way we would not be able to find Tweets that were sent before that Tweet. `max_id` was initially set to the ID of the newest Tweet in the full dataset. Altogether we collected 1,471,597 Tweets about the Coronavirus outbreak. Many Tweets found by one search term were also found by others. When removing duplicate Tweets (by turning the list of all found IDs into a set) we were left with 803,249 Tweets about the Coronavirus outbreak. This dataset is our ground truth data.

3.1.4 Dataset Statistics

Table 1 shows how many total Tweets and how many Coronavirus related Tweets our dataset contains. The first Tweet in our dataset was posted on March 30, 2020, at 12:06. Nearly five hours later was the last Tweet posted, on March 30, 2020, at 16:57.

Total	23,824,607
Coronavirus related	803,249

Table 1: Statistics of the dataset.

In Table 2, we show how many Tweets were found by each of our 23 search terms. A percentage of how many of the ground truth Tweets can be found using that search term is also provided. Since the searches were performed separately and Tweets can contain several search terms, Tweets were often found several times. Because of this the percentages do not add up to 100%. The percentages do, however, give a good indication of how useful a search term is.

Search term	# of results	% of GT	Search term	# of results	% of GT
coronavirus	373,103	46.45	coronaoutbreak	1,129	0.14
covid	294,377	36.65	ncov	1,067	0.13
covid-19	250,321	31.16	coronaalert	578	0.07
covid_19	250,266	31.16	corana	241	0.03
covid19	147,622	18.38	2019ncov	151	0.02
corona	107,043	13.33	covid-2019	146	0.02
corona virus	26,960	3.36	coranavirus	140	0.02
coronavirusoutbreak	5,539	0.69	ncov2019	104	0.01
covid2019	5,006	0.62	wuhanflu	82	0.01
chinesevirus	4,278	0.53	chinaflu	46	0.01
wuhanvirus	1,836	0.23	chineseflu	39	0.00
coronavirusupdates	1,523	0.19			

Table 2: Statistics for the 23 search terms with which Coronavirus related Tweets were found.

As can clearly be seen, some search terms are much more useful than others. Also interesting to note, is how similar the results are for the "covid-19" and "covid_19" search terms. This could be because Twitter's indexing method interprets the underscore and dash symbols similarly, causing many of the same Tweets to be found.

3.2 Data Preprocessing

All the Tweets were preprocessed using the same method. First, all punctuation was removed with exception of a few characters. Underscores (_) and dashes (-) were kept as they are sometimes used in hashtags to separate words, for example in the hashtag "#COVID_19". Hashtags are kept since these are very important in Twitter data, and @'s are kept to remove the mentions of usernames later on. All other punctuation as found in Python's `string.punctuation`[5] are removed.
After punctuation removal, the Tweets were converted to lowercase. Then, by looping over the words in each Tweet, unwanted words are discarded. Such unwanted words are:

- Mentions: mentions of usernames start with an @ and are used to send Tweets to specific users. A certain user being mentioned could help to identify the Tweet as being topic relevant but we think that this effect will be minimal. Thus we exclude usernames.
- URLs: these do not provide valuable information and are thus removed by discarding words starting with "https".
- Stop words: These are words that occur commonly such as "the" or "a". Experimentation showed that our baseline method performed slightly better when not using stop word removal. However, not removing stop words also caused the PRF method to consistently add stop words in the expanded query. Indri does not provide an option to ignore stop words during PRF. When stop words are chosen as expansion terms, the precision of the results will be extremely low as all Tweets, topic related or not, are likely to contain stopwords. For this reason, all stop words are removed from the dataset. This is done by matching each word to the list of stop words found in `nltk.corpus.stopwords`[6]. If there is a match, the word is discarded.

3.3 Pseudo-Relevance Feedback (PRF)

For this thesis, we have used Pseudo-Relevance Feedback (PRF) to find supplementary topic relevant Tweets. PRF is a technique for *query expansion*. Query expansion is simply the altering of a search query to make it better suited for finding more relevant documents. PRF starts with an initial search query and finds topic relevant terms which are added to this initial query. The results that are found by this new query are then returned. Several different versions of how PRF performs the query expansion exist. We will describe the PRF model which is used by the *Indri* search engine since this is the tool that we use in this thesis (as described in Section 2.3).
Indri implements PRF based on a model proposed by [LC01]. A summary of Indri's implementation can be found on their *SourceForge* page[7] under header "Pseudo-Relevance Feedback".

[5] https://docs.python.org/3/library/string.html#string.punctuation
[6] http://www.nltk.org/nltk_data/
[7] https://sourceforge.net/p/lemur/wiki/Indri%20Retrieval%20Model/

Note that in the context of this thesis a "document" is a Tweet.
First, the initial query is performed. To rank the search results the formula

$$P(I \mid D)$$

is used. Here D is the document and I is the *information need node*. The latter combines probabilities from an underlying network into a single probability. This network is a Bayesian network which contains terms as variables, and probabilities. These probabilities represent matters such as that certain query terms occurred in certain documents. I thus represents all this information into one probability. Also, note that the network structure is dependent on the content of the search query. Altogether, the formula $P = (I \mid D)$ represents the probability of the document D matching the query. Documents for which this probability is higher are ranked higher.
Secondly, a *relevance model* is computed. This model is a probability of the form

$$P(r \mid I)$$

which is computing using the formula:

$$P(r \mid I) = \frac{\sum_D P(r|D)P(I|D)P(D)}{P(I)}$$

where the summation goes over documents D which are the top k documents found by the previous step. Here, r is a *representation concept* which is some word from the top documents. In the numerator we essentially see in how many of the top documents we find word r. More weight is given to r if it occurs in better-matched documents. By dividing this by $P(I)$ we get a model that returns the probability of word r being related to a search query, given the k most related documents. Furthermore, k is a parameter that can be chosen by the user.
Thirdly, after calculating the probability of each word being related to the query, we sort the words based on these probabilities . The m most relevant words are kept as the found expansion terms, this number must also be chosen by the user.
Fourthly, the new, expanded query is created with the weight for each expansion term being equal to the probability found by the relevance model.
Lastly, we search for the expanded query and return the matching documents.

This method used in Indri follows the general structure of the PRF technique. First, the initial search query is performed. It is assumed that the top k results are relevant. Within these results, the top m most relevant terms are found and added to the initial search query. By adding the most relevant terms from the most topic relevant documents, we can presumably find more topic relevant documents than with the initial query. The values k and m are two parameters which must set by the user. Most often, values for these parameters are chosen after testing several different values.

3.4 Evaluation

The effectiveness of PRF in retrieving Coronavirus related Tweets is evaluated by computing the following three measures for each search:
The *recall* is the portion of the Coronavirus related Tweets that we retrieved with some query:

$$\text{recall} = \frac{\text{tweets found in ground truth}}{\text{total tweets in ground truth}}$$

The *precision* is the portion of Tweets found by a query that is about the Coronavirus outbreak:

$$\text{precision} = \frac{\text{tweets found in ground truth}}{\text{total tweets found}}$$

Lastly, the *F1 score* is used for finding a good balance between the recall and precision. The F1 score is the harmonic mean of the recall and precision which is computed by:

$$\text{F1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

We use these measures as they match our goal in the research. Namely, seek to find as many topic relevant Tweets as possible. To measure how well we do this, we look at the recall. However, we also want to make sure that we do not find too many unrelated Tweets, for which the precision is used. The F1 score then simply allows us to look at these two measures at the same time.

3.5 Searching with Indri

Queries using Indri make use of the *Indri Query Language*[8]. Important to our research is that we can search for several search terms at once using the `#combine()` operator. This operator combines given search terms, making it so that documents containing at least one of the search terms are returned. We also make use of the `#1()` operator. Search terms are space separated in the Indri Query Language. This operator allows us to search for literal phrases, such as "corona virus". We also use this operator for search terms that contain underscores. Indri interprets underscores as spaces, so "covid_19" is interpreted as "covid 19". These instances can still be retrieved by searching for "#1(covid 19)".

When we perform an Indri search, we must provide our search index in which to search, and a *parameter file*. This parameter file must be written in TREC format[9]. In it we state all the parameters of the search, such as the desired number of results. Each search that we perform with Indri requests 25 million Tweets. Our full dataset consists of about 24 million Tweets, so requesting more than this total makes it so that as many as possible Tweets are returned.

Twitter and Indri have slightly different indexing methods. This can cause the same search query to retrieve different documents. With Indri we retrieve 88% of the topic relevant Tweets that we found using the Twitter API, using the same search terms. From all the results found with Indri, 97% were also found with the Twitter API. As both these recall and precision are high, using Indri gives a sufficiently accurate representation of what results we would get when directly using the Twitter API.
Figure 1 illustrates how the recall of 0.88 is reached. For this Figure, we added one word to a search query at a time and recorded the recall after each search. Words were added in order of highest recall as seen in Table 1. Entirely on the left, the search query is just "coronavirus". Next to that, the query was "#combine(coronavirus #1(covid 19))". This way we see that initially adding search terms increases the recall greatly. As more words are added, fewer new Tweets are found. We can also clearly see that we cannot get a recall of more than 0.88 due to the differences in indexing. In order to come close to this recall, at least the 4 best search terms must be used.

[8] https://www.lemurproject.org/lemur/IndriQueryLanguage.php
[9] http://www.cs.cmu.edu/~lemur/LemurGuide.html#data

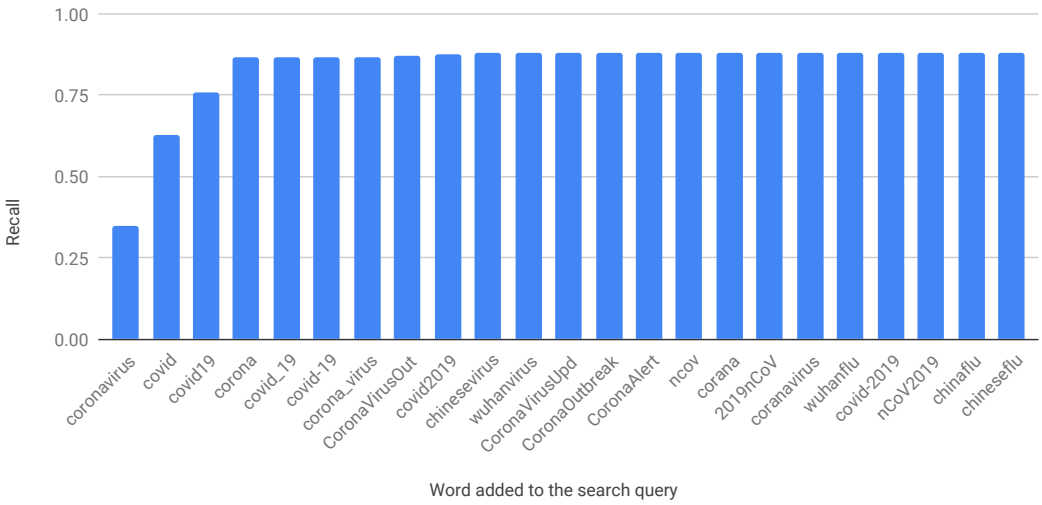

Figure 1: Recall converging to a limit of 0.88 due to a difference in indexing between Indri and Twitter.

3.5.1 Baseline

The baseline method is a simple Indri search without any methods of finding more documents such as PRF. This means that in the parameter file, we specify just the minimally necessary information. Such a parameter file contains the following information:

- The retrieval model, for which we have chosen *indri*. Different models such as *tf-idf* are also available. However, when using Indri, it's own retrieval model is most commonly used, and we found results of other retrieval models to differ only little.
- The desired query. This must be provided in the format of the Indri Query Language as described at the start of this Section.
- A count of the desired number of results.

For the baseline, we searched for each of the 23 search terms with which we also found Coronavirus related Tweets using the Twitter API.

3.5.2 Indri PRF

To improve the baseline method we attempt to find more Coronavirus related Tweets using PRF via Indri. For PRF, as described in Section 3.3, two parameters must be chosen: k for the number of expansion documents and m for the number of expansion terms. Using Indri we must set these parameters in the parameter file. When these parameters are provided, Indri will automatically perform PRF.
Indri always adds the terms from the initial search query to the expanded query. So, if the original query consists of 2 terms, the expanded query will only contain $m - 2$ new search terms. When reporting the number of expansion terms, we only consider the newly added terms.

4 Experiments and Results

This Section first describes all the experiments that we have performed. After that, the results of each experiment are presented and interpreted.

4.1 Experiments

Each experiment is performed on a development dataset, which contains half of the entire dataset. We later validate our final results on the other half, which is our validation dataset.
Splitting the dataset was done randomly. We first took the Tweet IDs of the full 24 million Tweets dataset and put these in a list. This list was placed in random order. Next, the first half of this shuffled list was split off from the second half, after which both splits were indexed into separated search indices via Indri. One of these halves is considered the development dataset, the other half makes up the validation dataset.
For the evaluation of individual searches, we need to compute the recall, precision, and F1 score. Each of these requires a ground truth dataset, and as we split our dataset, the two search indices have separate ground truth sets. We retrieve these by going through each Tweet ID in the separate search indices and finding which of them were in the original ground truth dataset. This way we obtain two new ground truth datasets, one for each dataset split.

4.1.1 PRF Settings

For our first experiment, we determine an adequate setting for PRF that we will use for the other experiments. To determine good settings we test several combinations for the number of expansion documents k and the number of expansion terms m. For k we test values in [100, 200, ..., 1500], and for m we test values in [2, 3, ..., 16]. As stated in Section 3.5.2, Indri adds terms from the original query to the expanded query. Because our initial search queries always contain 1 word for this experiment, we actually add 1 through 15 new expansion terms.

We use two measures to decide which parameter setting we use for other experiments: the F1 score and the precision.
Each combination of parameter settings is tested on all 23 Coronavirus related search terms. Over these 23 searches, we calculate the average F1 score and the average precision.
We test each combination of parameter settings on all these 23 search terms as our research focuses on a scenario where a few initial search terms are used. It can be the case that these search terms are well suited for PRF or not. As we want to find good PRF settings that are well suited no matter the initial search query, we take all Coronavirus related search terms into account.

A high F1 score is mainly our target. However, in our baseline, most search terms have a recall so small that even with a perfect precision the F1 score will be low. This can make it so that the F1 score is greatly improved via PRF easily. One risk that occurs then, is that the precision can become so low that PRF is not practically useful. This is not noticed when just looking at the F1 score. To ensure that the precision remains acceptable, we take the average precision into account when deciding which PRF settings to use.

Furthermore, to get more insight into the information that the averages provide us with, we also show the standard deviations for both the F1 scores and the precision. These are again calculated for each combination of PRF parameters over the results of the 23 search terms.

4.1.2 PRF on Individual Search Terms

In the experiment described in Section 4.1.3, we explore the effects of PRF when using several seed search terms. For this, we need to know what search terms are well and ill-suited to PRF.
With this experiment, we will take a detailed look at how PRF impacts individual search terms. This includes looking at the recall, precision, and F1 score after performing PRF. We do this for each of the 23 Coronavirus related search terms. For some of the most interesting results, we will also look at what expansion terms were added, and how these affected the F1 scores.

Using the gained knowledge we determine which search terms are well and which are ill-suited to PRF. This information is used in the following experiment.

4.1.3 PRF on Combinations of Seed Search Terms

Our research focuses on a realistic scenario in which a researcher filters the Twitter stream based on a few search terms. Up to this point, we have only researched how well PRF performs when a single seed search term is used. However, if PRF works well for single search terms, this does not necessarily mean that it will work well when several seed terms are used.
With this experiment, we explore the effects of PRF when several search terms are used. Specifically, we consider combinations of two search terms.
First, we experiment with combinations of search terms based on the effectiveness of PRF for the two individual terms. Later, we look at combinations based on how well they perform in the baseline.

Combinations based on PRF performance
When we have two search terms that are ill-suited to PRF, this does not imply that combining them will also lead to poor results. As stated in Section 3.3, PRF finds expansion terms in the Tweets retrieved by the initial search terms. The combination of the two search terms thus gives us more and different Tweets in which to find expansion terms. It can thus occur that other, perhaps better search terms are added. Likewise, it could also be the case that PRF performs poorly when combining two search terms that individually are well suited to PRF.
In this experiment, we explore if there is a connection between the quality of PRF for the individual terms and their combination. From the experiment described in Section 4.1.2 we found search terms for which PRF worked exceptionally well, and for which it worked poorly. We look at three types of combinations between these two groups, and observe the performance of PRF:

- PRF is poor for both search terms.
- For one search term PRF is poor. For the other it is effective.
- We use two terms for which PRF works well.

Each time we test all the possible combinations.

Combinations based on baseline performance

In the previous experiment, we looked at combinations of search terms based on PRF performance. However, in practice we can only determine the effectiveness of PRF after a dataset is created. We must therefore also test if PRF is effective when not taking PRF performance of individual terms into account. This will show how PRF will work in a realistic scenario.

For this, we do consider the baseline performance of search terms. Specifically, we consider the number of documents that can be retrieved by a term in the baseline. We use this measure rather than the F1 score, as it is more intuitive. Additionally, most F1 scores in the baseline are near 0, making it difficult to choose sensible cut-off points.
Search terms that find a total of fewer than 1,000 Tweets in the baseline we consider to be poor. 1,000 was chosen as a cut-off point, as this means that a search term can retrieve very few Tweets, less than one eighthundredth of our ground truth data.
If a search term finds more than 100,000 Tweets we consider it to be good. We chose 100,000 as a cut-off point, so that search terms are only considered good if they can retrieve a large portion of the ground truth data, over 10 percent.
Table 3 shows the search terms that are considered to be poor and good based on the above criteria.

Poor baseline term	Tweets found in baseline
chineseflu	30
chinaflu	43
nCoV2019	49
covid-2019	67
wuhanflu	75
coranavirus	111
2019nCoV	161
corana	233
ncov	443
CoronaAlert	579

Good baseline term	Tweets found in baseline
corona	100,468
covid19	136,725
covid	253,758
coronavirus	287,998

Table 3: Search terms which perform well and poorly in the baseline. Left are those that performed poorly. Search terms on the right performed well.

We distinguish three types of combinations of search terms between these groups. These are based on how many Tweets the search terms can find together, without PRF.

- Both search terms perform poorly without PRF. This can be seen as a sort of worst-case scenario, as without PRF we are certain to find few Tweets. Realistically speaking this combination is unlikely to occur. However, if a researcher is not aware of the most crucial aspects of a topic, it is plausible.

  This experiment will show how effective PRF is when there is the most room for improvement, but the least data to work with.

- One of the search terms performs poorly without PRF and the other does perform well. In this scenario one of the search terms already finds a large portion of the data, whilst the

other does not contribute much additional data. Whether or not this affects the performance of PRF is interesting to experiment with.

It is likely in practice, that if one good search term is chosen, that the researchers were also able to choose another adequate search term. Though perhaps unlikely, the scenario described above can still occur. Also, it shows how the small increase of information impacts the results of PRF, which can be valuable knowledge.

- Both search terms perform well without PRF. This can be seen as a best-case scenario, in the sense that even if PRF does not work well, we will still be able to retrieve a large number of the topic relevant Tweets. With this experiment, we determine if and how well PRF can find additional Tweets if a large portion of relevant Tweets is already found.

  In practice, it is quite likely that researchers can correctly identify the most topic-relevant terms. This experiment will show how well PRF performs in such a scenario.

For each scenario, we test all possible combinations.
We believe that these experiments give an accurate representation of how well PRF will perform in practice. Because of this, we will validate our results from the development set, by also performing the experiment on the validation dataset that we made, as explained in Section 4.1. All the results that we found will be presented separately for both data splits. We will also dive into more detail than in the previous experiment.

4.2 Results

In this Section, we present, interpret, and discuss the results of the experiments which we performed.

4.2.1 PRF Settings

Our first experiment aimed to help us choose adequate PRF settings. These settings will be used throughout the remainder of this thesis.
This experiment was performed on the development dataset.

F1 score comparison
In Figure 2 and Figure 3, we compare different settings for PRF. The number of expansion documents is shown on the horizontal axis, and the lines represent the used number of expansion terms. In Figure 2, the vertical axis displays the average F1 score. This is the standard deviation in Figure 3. We plot the baseline as a straight line on the appropriate height. For the baseline, 0 expansion documents are considered, but by plotting the value for each number of expansion documents, it is easier to compare settings to the baseline.

Looking at Figure 2, we see that generally using more expansion documents leads to improved results. The benefits are most significant in the range of 100 to 500 expansion documents. After that, the average F1 score is not impacted much by the number of expansion documents.
Standard deviations appear slightly higher in the range of 300 to 500 expansion documents. Other than that, the number of expansion documents appears unrelated to the standard deviation.

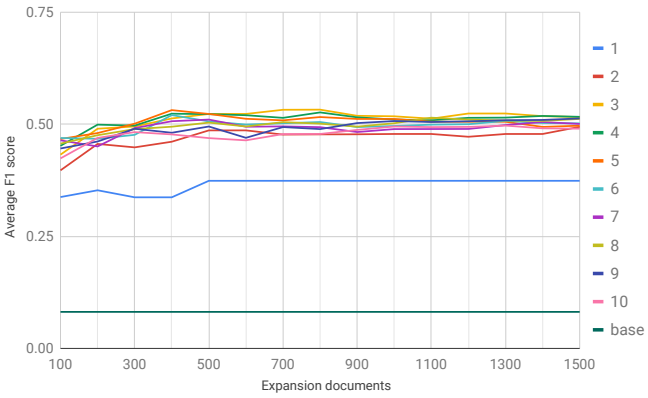

Figure 2: Average F1 score over 23 seed search terms for various PRF settings. Each line represents a different number of expansion terms that was used.

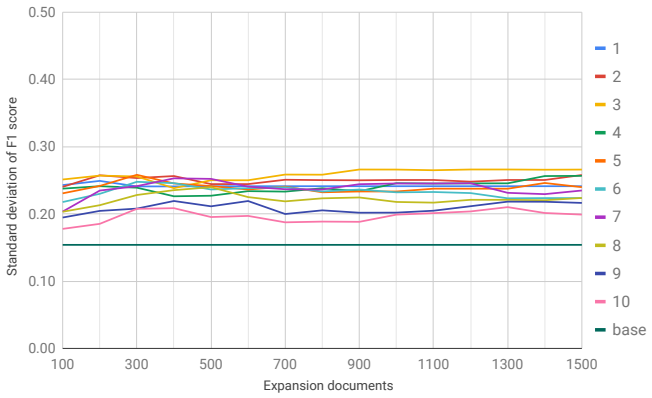

Figure 3: Standard deviation of the F1 scores over 23 seed search terms for various PRF settings. Each line represents a different number of expansion terms that was used.

The number of expansion terms has a more noticeable effect on both the F1 scores and the standard deviations.
Improving the baseline in terms of the F1 score is achieved easily with PRF. Even by adding just a single expansion term, the average F1 score is over four times as high. Using a second expansion term again increases the average F1 score largely. After that, adding more expansion terms does not seem to impact the average F1 scores by much. This creates a group of close lines that are hard to distinguish from one another. We do observe a spike in the average F1 score for 5 expansion terms at 400 expansion documents.

The standard deviations are rather high which is not surprising. PRF performs much better on some search terms than on others, causing there to be a large difference in F1 score between searches. Also, we can see that the standard deviations are lower when more expansion terms are used. An explanation for this could be that as more expansion terms are added, the recall and precision converge to some point as no new Tweets can be found. If these convergence points are similar for the 23 different seed search terms, then the standard deviation over these terms will be lower when more expansion terms are used.

Precision comparison
In Figure 4 and Figure 5, we compare how settings for PRF impact the precision of searches. The Figures are much the same as before, only now we show the average precision and the standard deviation of the precision on the vertical axis.

In Figure 4, we observe first that the baseline precision is very high, about 97%. By using PRF no matter the settings we are unable to get an average precision quite that high. Moreover, as we can tell with Figure 5, the standard deviation within the 23 searches is much higher when using PRF, than when it is not used. From that, we can conclude that the precision on some searches becomes low when using PRF, whilst on other searches it remains high.
We also see that the standard deviation is highly dependent on how many expansion terms we add.

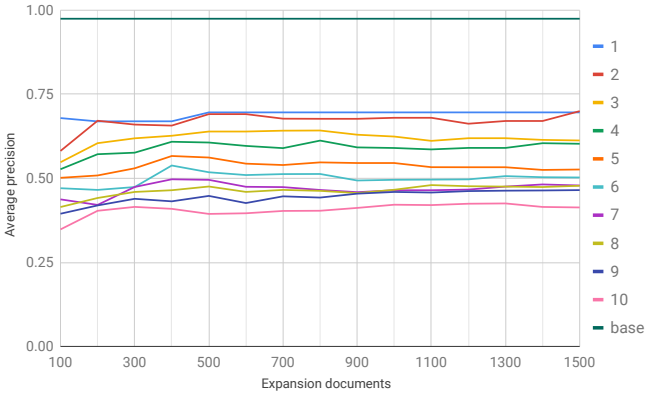

Figure 4: Average precision over 23 seed search terms for various PRF settings. Each line represents a different number of expansion terms that was used.

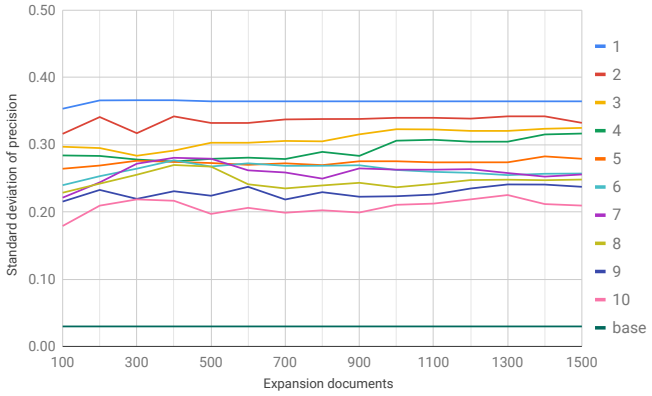

Figure 5: Standard deviation of the average precision over 23 seed search terms for various PRF settings. Each line represents a different number of expansion terms that was used.

The more we add the lower the standard deviation. This is, again, likely since the precision scores converge when many expansion terms are used.

Figure 4 also shows us that using more expansion documents has a similar effect on the precision as it does on the F1 scores. Again, the main contributing factor to the average scores is the number of added expansion terms. Using 1 or 2 expansion terms results in almost the same average precision. When using more than 2 expansion terms the precision declines fairly consistently.

Chosen Settings
As stated before, there was a spike in the F1 scores on the development set at 5 expansion terms with 400 expansion documents.
The average precision at this point is rather low. However, as the precision scores fluctuate so much between the 23 searches, we deem the precision to be adequate for our following experiments. Thus, we decided to use 5 expansion terms and 400 expansion documents for the remainder of our experiments.

4.2.2 PRF on Individual Search Terms

In Figure 6, we look at the F1 scores for the 23 Coronavirus related search terms after PRF is applied. The search terms are sorted on the baseline F1 score. So, the further a search term is on the right, the higher the effectiveness of that search term, without PRF.

This Figure is rather interesting, as the quality of PRF appears unrelated to how many documents we can initially retrieve. In fact, the initially fourth-best search term, "corona" is actually the third-worst after applying PRF. "2019nCoV" is one of the best search terms when using PRF, but it is among the worst when PRF is not used.
It also seems like most search terms that are well suited for PRF, initially have a mediocre recall. Again though, there are exceptions such as the search term "chinesevirus".

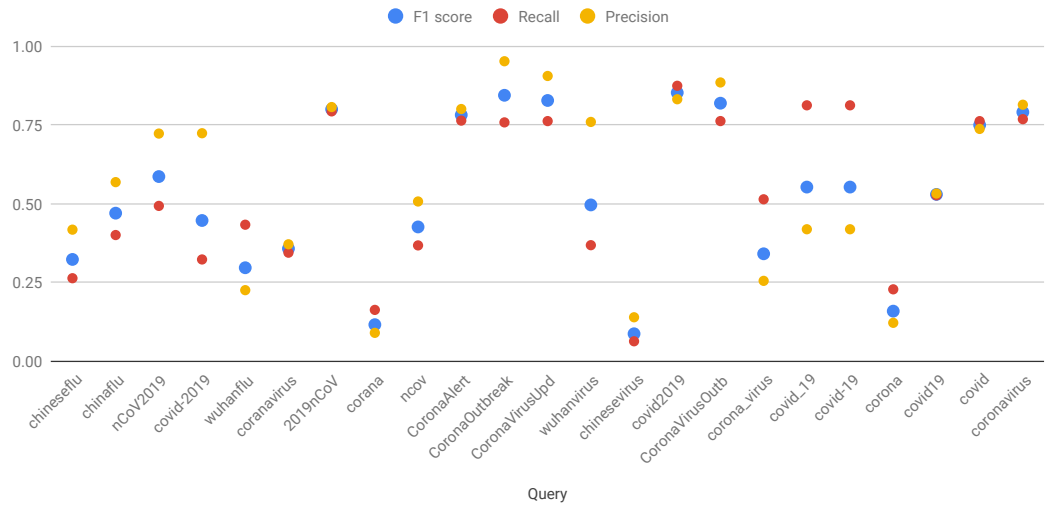

Figure 6: F1 score and its components after applying PRF to each of the 23 Coronavirus related search terms.

Most of all, this Figure shows us that the quality of PRF appears rather unpredictable, but that it is generally good.

Just the F1 scores after PRF do not adequately show the quality of PRF. For that, we must also make a comparison with the F1 scores without PRF. Figure 7 shows for each of the 23 Coronavirus related search terms the F1 score without and after applying PRF.

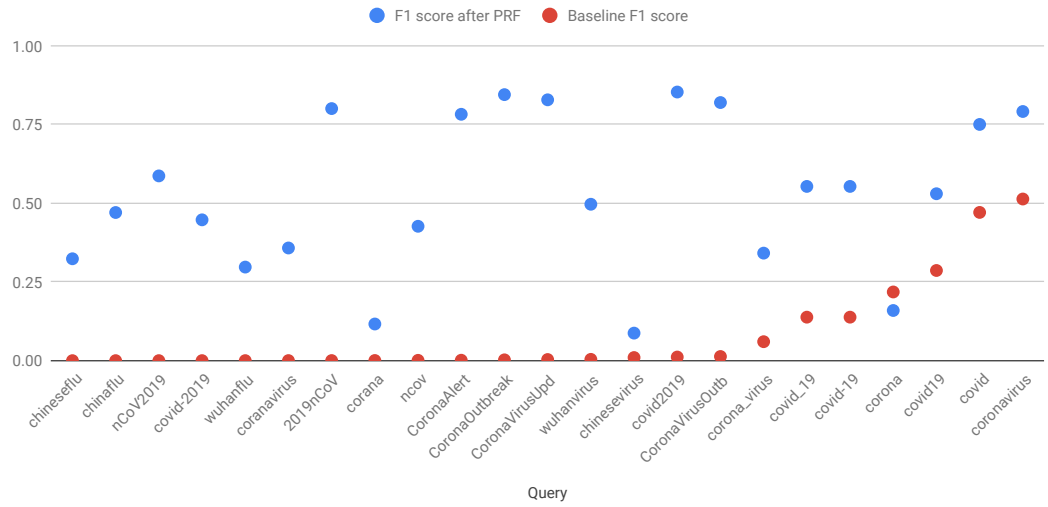

Figure 7: F1 score before and after applying PRF for each of the 23 Coronavirus related search terms.

With this Figure we can see that PRF is greatly effective at improving the F1 score of almost all search terms. “corona” is the only search term where the F1 score is decreased by PRF. On average

the F1 scores are improved by 0.45. Such great improvements are largely as most search terms have an initially small F1 score of nearly 0. When starting with such a poor F1 score it is not surprising that we can improve it. However, the extent to which we see improvements is tremendous for nearly all search terms. Even for those with an initially high F1 score we see a high increase, except for "corona". From this we conclude that PRF is highly effective at finding more topic relevant Tweets.

Breakdown of bad PRF search terms

Three of the 23 search terms get exceptionally poor F1 scores after applying PRF. These are: "corona", "chinesevirus", and "corana". In Table 4, we see how the F1 scores of these three search terms change as more expansion terms are added. The first rows state the initial queries.

Term added	F1 score
corona	0.22
virus	0.23
go	0.17
vip	0.17
loaded	0.17
people	0.16

Term added	F1 score
Chinesevirus	0.01
china	0.05
world	0.08
chinese	0.09
chinesevirus19	0.09
wuhanvirus	0.09

Term added	F1 score
corana	0.00
virus	0.10
fuck	0.09
people	0.11
stay	0.12
sir	0.12

Table 4: Development of F1 scores as more expansion terms are added for the three worst performing search terms.

We see here that many of the added expansion terms are harmful. Terms such as "china" and "go" will plainly find many topic unrelated Tweets. Nevertheless, after harmful terms are added, useful search terms can still be found. Namely, in the middle Table, we find that the last two expansion terms "chinesevirus19" and "wuhanvirus" are clearly topic relevant. However, they likely do not find enough Tweets to significantly increase the F1 score. We thus conclude that PRF can add harmful search terms. As a result, the F1 scores can become poor, but even then useful expansion terms might still be added.

Breakdown of good PRF search terms

Eight of the 23 search terms have a particularly high F1 score after PRF is applied. For the best three of these, we will look more in-depth at how PRF impacted their F1 scores. These three search terms are: "covid2019", "CoronaVirusUpdates", and "CoronaOutbreak". In Table 5, we see how the F1 scores of these three search terms change as more expansion terms are added.

With these tables, we can easily tell how these search terms got such good results; each of them has the same first three expansion terms: "coronavirus", "covid", and "covid19". These are the three search terms that have the highest baseline recall. By adding these we unsurprisingly see that the F1 scores grow rapidly. The other search terms which are added can also be linked to the Coronavirus outbreak. "19" is likely added because it is often used in conjunction with "Covid". Both times that it is added, we see that the F1 score remains the same as before, so it is certainly not harmful. Overall, these tables show us that PRF can identify the most important search terms to a topic. When this is the case the results can be profound.

Term added	F1 score
covid2019	0.01
coronavirus	0.52
covid	0.77
covid19	0.86
lockdown	0.79
corona	0.85

Term added	F1 score
CoronaVirusUpdates	0.00
coronavirus	0.51
covid	0.77
covid19	0.85
19	0.85
cases	0.83

Term added	F1 score
CoronaOutbreak	0.00
coronavirus	0.51
covid	0.77
covid19	0.85
19	0.85
indiafightscorona	0.85

Table 5: Development of F1 scores as more expansion terms are added for the three best performing search terms.

Breakdown of other interesting results

The search term "2019nCoV" is interesting, because it finds a total of only 161 search results in the baseline. But, the search term has one of the highest F1 scores after PRF. In Table 6, we see which expansion terms were added and how these impacted the F1 score. We observe again that the three best search terms in the baseline are added. This shows that the best way to get a high F1 score is to find the search terms that are most relevant to the topic. Whilst this is unsurprising, it is interesting that all these three terms can be identified with just 161 available expansion documents.

Term added	F1 score
2019nCoV	0.00
coronavirus	0.51
covid19	0.65
covid	0.85
19	0.85
virus	0.80

Table 6: F1 score development for the seed term "2019nCoV".

Term added	F1 score
chineseflu	0.00
china	0.05
coronavirustruth	0.06
lockdown	0.11
covid19	0.30
pandemic	0.32

Term added	F1 score
chinaflu	0.00
coronavirus	0.51
china	0.49
says	0.47
chinavirus	0.47
pandemic	0.47

Term added	F1 score
nCoV2019	0.00
covid19	0.29
coronavirus	0.65
sarscov2	0.65
2019ncov	0.65
safe	0.59

Table 7: Development of F1 scores for the three worst baseline search terms.

In Table 7, we see how the F1 score changes as search terms are added to the three worst search terms in the baseline. For each of these search terms, less than 50 expansion documents were available.

On the left, the increase in the F1 score is gradual. It is notable that even without the successful search term "covid19" the F1 score can be increased substantially.

In the two tables on the right, the F1 score increases rapidly first and then stay quite stable.

Considering how few expansion documents were available, and that a single Tweet contains little information, it is rather surprising that we obtained such good results here. Even if some bad search terms are added, the majority of added search terms are beneficial and sensible regarding the topic of the Coronavirus outbreak.

Overall findings
In this experiment, we took a more detailed look at how PRF affects the F1 scores for the 23 individual search terms. From the experiment, we conclude that PRF can be highly successful for Twitter data. There is no correlation between how many initial documents are found and how well PRF performs. Most important for the success of PRF seems to be the ability to find and add expansion terms that find many topic-relevant documents. When many such search terms are added, the F1 score will likely be high.
For most searches, PRF is highly effective at adding terms that will improve the F1 score. Often some of the top baseline search terms are added. Sometimes harmful terms are found, but this does not directly cause the F1 score to be poor. We also find that many expansion terms are topic relevant, but that their addition leads to improvements too small to notice when looking at the F1 scores. The majority of the added search terms are topic relevant. This is even the case when only little information is available.

We decided that we consider search terms that got an F1 score of over 0.75 to be well-suited to PRF.
Search terms with an F1 score of under 0.25 are considered to be poor regarding the PRF performance. We also consider search terms that have a high recall but a low precision to be poor. For that, we choose a cut-off point of 0.5 for the precision, and of 0.5 for the recall. In this case, a large portion of the ground truth dataset is found, but more than half of the found Tweets are not in the ground truth. Perhaps depending on the research, this might be acceptable, but we assume here that it is not.
Table 8 shows which search terms are well and which are ill-suited to PRF.

Terms ill-suited to PRF	Terms well-suited to PRF
corana	2019nCoV
chinesevirus	CoronaAlert
corona_virus	CoronaOutbreak
covid_19	CoronaVirusUpdates
covid-19	covid2019
	CoronaVirusOutbreak
	covid
	coronavirus

Table 8: The search terms which perform well and poorly when applying PRF. Terms on the left have poor results after PRF is applied. On the right the performance of the terms is good.

4.2.3 PRF on Combinations of Seed Search Terms

With our final two experiments, we explore the effectiveness of PRF when several seed search terms are used.

Combinations based on PRF performance

In Figure 8, we see the results of applying PRF to different combinations of search terms. Here we combined search terms based on their PRF performance. For that we consider search terms for PRF works well or poorly, as can be seen in Table 8.

Left we see the results when two terms are used that are both ill-suited to PRF. Here a total of 10 combinations were made. The middle box plot shows combinations where one of the search terms is ill-suited and the other is well-suited to PRF. 40 such combinations were made. On the right, we have combinations of two search terms that are both well suited to PRF. For that we made 28 combinations.

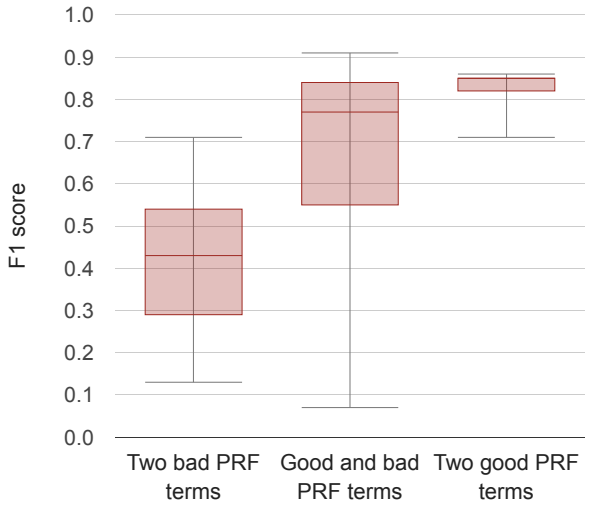

Figure 8: The results of combining two search terms based on their individual PRF performance.

Combinations on the left generally have poor F1 scores. We had 10 combinations here, 3 of which got an F1 score higher than 0.5. All of these contained either the term "covid_19" or "covid-19". These terms individually also got F1 scores that were quite high, but their precision was too low. We found that of all combinations with these terms, 2 got a precision of 0.72, and the others were still below 0.5. This shows that combining search terms can be beneficial for improving the precision, but that this is certainly not a given.

In the middle, we get mixed results. Half of the combinations got an F1 score of over 0.77 which is high, but a fourth was below 0.55, which is quite poor.

Each combination contains one search term that individually gets an F1 score of over 0.75 with PRF. About half the total combinations have an F1 score below 0.75. It thus seems that there is an even chance that expansion terms are chosen so that results are worse than when only the individually well-performing search term is used.

Clearly, on the right PRF is highly effective. Even the lowest observed F1 score is high. The third quartile and the median have the same value, causing them to overlap. Indeed this means that at least 25% of all combinations have an F1 score of about 0.85. All the combinations that have an F1 score of about 0.85 share the expansion terms "coronavirus", "covid", and "covid19". We can thus

conclude that if these three search terms are added, that the F1 score will be close to 0.85.

Overall, we conclude that the quality of PRF on combinations of search terms is dependent on the PRF quality of the individual terms. If search terms are combined that themselves are well-suited to PRF, their combination will also be well-suited. Combining individually effective and ineffective search terms mostly leads to improved results, but results can also be poor. F1 scores are predominantly low if both search terms are ill-suited to PRF.

Combinations based on baseline performance

The results of experimentation with combinations of search terms based on their baseline performance can be found in Figure 9. We made several combinations of the best and worst search terms in the baseline. Table 3 showed earlier which search terms belong to these categories.

This experiment aims to give a realistic depiction of how PRF will perform in practice. To make sure that our results do not depend on what subset of our data we use, the experiment was also performed on the validation set. Results of both the development and the validation datasets are shown.

In left box plots, both search terms find little Tweets in the baseline, for that we had a total of 45 combinations. In the middle are combinations where one term finds few Tweets in the baseline and the other finds many. There we made 40 different combinations. Both Figures show on the right combinations of two search terms which both find many Tweets in the baseline. Here we could only make 6 combinations.

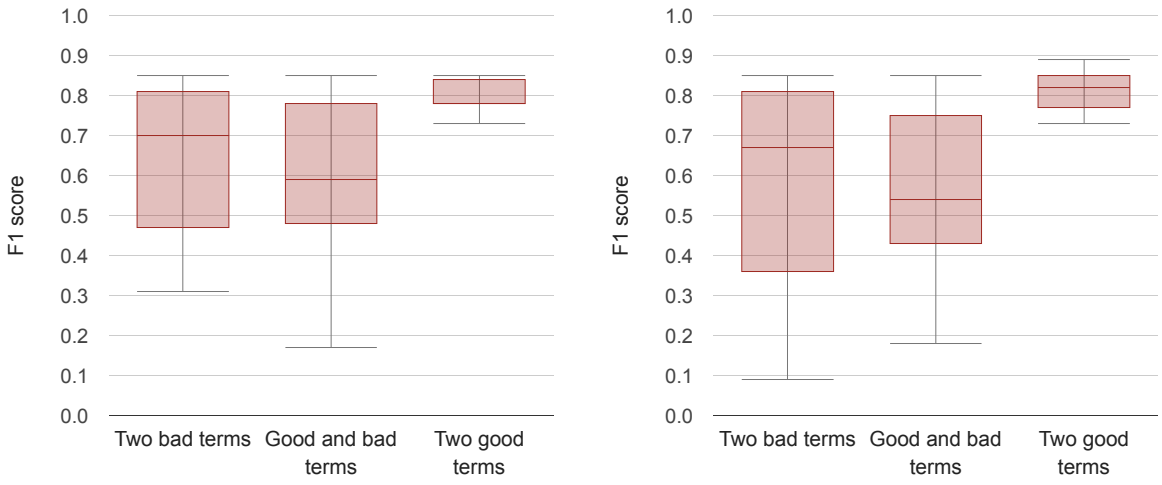

Figure 9: The results of combining two search terms based on their individual baseline performance for both dataset splits. Left are results of the development set. Right are results from the validation set.

Both Figures show on the right that PRF is highly effective. Part of this success is that combining two search terms that are good in the baseline always leads to a high recall. However, the F1 scores observed here are even higher. When we have so many potential expansion documents, it seems like PRF is highly effective at finding the relevant documents and expansion terms.

As we observe similar levels of success on both dataset splits, we are certain that they are not coincidental and that this pattern is not dataset dependent. Furthermore, we only had 6 available data

points, but as we observe the same pattern on both dataset splits this does not impact our conclusion.

The middle box plots show more varying results. About half the data points get high F1 scores, whereas the other half get poor scores. By inspecting the combinations that got poor results we see a pattern also seen before; topic irrelevant terms were added, and not enough relevant terms were added to compensate.
When combining the best baseline terms with the worst, these worst terms provide little additional information. Table 9 shows how this additional information impacts the effectiveness of PRF. We do this by looking at the average F1 score that was obtained with combinations for each of the best baseline terms. These scores are compared to the F1 scores that these terms individually got after applying PRF.

Combinations with	Individual F1 score	Avg. F1 score of combinations	
corona	0.16	0.51	0.51
coronavirus	0.79	0.60	0.61
covid	0.75	0.64	0.59
covid19	0.53	0.61	0.51

Table 9: Impact on the effectiveness of PRF when combining the best baseline terms. The left average F1 score comes from the development dataset. Right the score from the validation set is shown.

The Table shows us that the little additional information can greatly impact the effectiveness of PRF. We also see that the search results for the term "corona" are improved substantially, on average. Furthermore, though some results are decreased, we still on average have relatively good F1 scores.
Results for the different dataset splits are again consistent.

The left box plots are quite surprising. When the least information is available we see that PRF performs relatively well. In fact, the results are often better than in the middle box plots where more information is available. We use Table 10 to help understand how this comes. This is done by showing how often particular expansion terms were added. The rightmost columns show in what percentage of the combinations the expansion terms were added.

Search term	Count	% of combinations
coronavirus	40	86.67
covid	29	62.22
covid19	26	55.56
2019	17	35.56
china	11	22.22

Search term	Count	% of combinations
coronavirus	33	73.33
covid	23	51.11
covid19	20	44.44
2019	16	35.56
virus	12	26.67

Table 10: Top 5 expansion terms as two search terms are combined that both obtain poor baseline results. Left are results for the development dataset. Right are those of the validation dataset.

Search term	Count	% of combinations
coronavirus	22	61.11
covid	19	52.78
virus	13	36.11
covid19	12	33.33
19	12	33.33

Search term	Count	% of combinations
coronavirus	18	45.00
covid	17	42.50
covid19	15	37.50
virus	13	32.50
china	13	32.50

Table 11: Top 5 expansion terms for combinations of one well and one poorly performing term in the baseline. Results from the development dataset are shown left. Right are those of the validation set.

Table 11 shows the same information, but for combinations of one good and one poor baseline term. In Table 10 we see that the best baseline terms are added much more frequently than in Table 11. This shows us how it comes that PRF performs better in the right box plot. Why this occurs is not entirely clear to us. It might be related to how the quality of PRF is distributed over the individual search terms.
We also note that, again, the results are similar for the two dataset splits.
In the worst-case scenario described in Section 4.1.3, PRF can be useful for finding additional topic relevant Tweets.

All and all we conclude that PRF is effective for finding supplementary Tweets when search terms of differing baseline performances are combined. Especially when the combined terms have a high recall, PRF will likely find almost all topic relevant Tweets. PRF performance is generally still adequate, albeit less consistent, in other scenario's.
Results are highly similar for the two dataset splits. From that, we conclude that the general patterns that we observed are not dependent on the dataset which is used.

5 Discussion

In this Section, we will discuss some limitations of the research which we have conducted. We will also propose ideas for further research.

We used the Indri search engine for experiments, mostly for it's PRF functionalities. Using PRF without Indri could open up new possibilities.
By not having to use Indri we could more easily hook our PRF functionalities up to the Twitter API. Researchers can then provide their query, and from the first X results returned by the API, we can determine expansion terms. To do this with Indri it would require the additional step of building a search index. This can be quite time and resource consuming. Section 4.1.2 and Section 4.2.3 both showed that PRF can be effective even when little information is available. Because of this, we suspect that X can be quite low.
We saw in Section 4.2.1, that we obtained optimal PRF performance when 400 expansion documents were used. Even search terms with fewer total available documents were improved by PRF, so we suspect that using the first $X = 400$ Tweets might be enough for highly effective results. This suspicion must, however, be confirmed by further research.
Being able to use PRF without Indri will also allow us to create a PRF functionality custom-tailored to Twitter data. For example, it could be valuable to look at hashtags specifically for finding expansion terms.

An important aspect of our research was the custom made dataset about the 2019 Coronavirus outbreak. When using such a dataset there always is the possibility that results are dependent on the dataset. By splitting our dataset, we confirmed that the patterns that we observed are not subset dependent. However, a different dataset can still give different results.
Our dataset was made for a topic that got a lot of online attention. Because of this, numerous different topic relevant terms were available. For smaller topics, there might well be less adequate potential expansion terms available. In that case, PRF could be less effective.
Research on the effectiveness of PRF on smaller topics could thus be valuable.

One way that we think PRF can be used is as a tool that suggests expansion terms. A researcher would provide their query and PRF would be applied. However, we would not perform a search with the expanded search query. The found expansion terms will be shown to the user and they can choose which terms to use.
A major reason that our research found poor PRF results was due to poor expansion terms being added. By putting the user in control of what expansion terms are used, we suspect that precision will always remain high. We also found that for nearly every search, some topic related terms are found by PRF, even if most other expansion terms are not helpful. This suggests that we will also be able to improve the recall. Together this should cause a major increase in PRF performance.
Whether or not it is practical to ask users for feedback depends on the use case. For the RISE_SMA project, PRF will possibly be used in this manner.

Section 4.2.3 showed that when we consider several seed terms, the quality of PRF for the combination is dependent on the PRF quality of the individual terms. We also found that PRF quality is not correlated to the baseline performance of search terms in Section 4.2.2. This makes it

so that we currently cannot predict beforehand if PRF will be effective. Only when we have several search terms that individually find many Tweets can we be certain that PRF will be effective. In other cases, it can occur that a large number of topic irrelevant Tweets will be found by PRF. Whilst this occurs infrequently, it does make it impractical to automatically apply PRF without performing some manual quality checking.

Further research might focus on improving PRF to make it more consistently effective. Most important for this is to avoid the addition of topic unrelated expansion terms. We think that it is unreasonable to expect a model to never add such expansion terms. However, it might be possible to decrease the frequency with which harmful expansion terms are added.

In Figure 4 we found the average precision to be highest when few expansion terms are added. Because of this, it might be better for the precision to first perform PRF and add just one expansion term. We can then use the expanded query as our new starting query on which to apply PRF. This method has the potential of causing a smaller decrease in precision.

6 Conclusions

This thesis focused on finding supplementary topic relevant Tweets. For this we used a custom dataset about the 2019 Coronavirus outbreak. Using this dataset and the Indri search engine, we experimented with the query expansion technique PRF. The aim was to explore how effective PRF would be at finding supplementary Tweets in a scenario where a researcher starts with a few seed search terms.

Our research question was: "When starting with an initial query of just a few search terms, how effective is Pseudo-Relevance Feedback for finding supplementary topic relevant Tweets?"
Our experiment that most accurately shows PRF performance in practice had promising results. 50% of queries which have very poor performance without PRF, will get an F1 score of between 0.4 and 0.8 after applying PRF. This range is 0.79 and 0.83 when the initial query is highly capable of retrieving topic relevant Tweets without PRF.
This shows that PRF can be highly effective for this goal, but that its effectiveness is not always consistent.
Low precision was found to be the main factor when PRF did not perform well. This happens when harmful, topic unrelated, expansion terms are chosen. These harmful effects can be mitigated when the most topic-relevant terms are also added by PRF.

More research could be focused on tuning PRF to more consistently preventing a decrease in precision. Performing our experiments on different datasets could also lead to interesting insights. We think that PRF can be most valuable when it is directly linked to the Twitter API, and perhaps functions as an expansion term suggestion tool.

References

[CNGR08] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson. Selecting good expansion terms for pseudo-relevance feedback. *SIGIR*, page 243–250, 2008.

[LC01] V. Lavrenko and W. B. Croft. Relevance-based language models. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127, 2001.

[MPLC13] F. Morstatter, J. Pfeffer, H. Liu, and K. M. Carley. Is the sample good enough? comparing data from twitter's streaming api with twitter's firehose. *Proceedings of the Seventh International AAAI Conference on Weblogs and Social Media*, pages 400–408, 2013.

[MSU13] T. Miyanishi, K. Seki, and K. Uehara. Improving pseudo-relevance feedback via tweet selection. *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 439–448, October 2013.

[MTDRW11] K. Massoudi, M. Tsagkias, M. De Rijke, and W. Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, pages 362–367, April 2011.

[WHF17] Y. Wang, H. Huang, and C. Feng. Query expansion based on a feedback concept model for microblog retrieval. *Proceedings of the 26th International Conference on World Wide Web*, pages 559–568, April 2017.

[ZLS16] M. A. Zingla, C. Latiri, and Y. Slimani. Short query expansion for microblog retrieval. *KES*, pages 225–234, January 2016.