Infrastructural Damage Information Retrieval from Microblogs

B.Tech Project Report
Submitted in Partial Fulfillment of
the Requirements for the Degree of

Bachelor of Technology

by

Gopal Kumar
(1401CS17)

under the guidance of

Dr. Joydeep Chandra

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY PATNA
PATNA – 800013, BIHAR

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Infrastructural Damage Information Retrieval from Microblogs**" is a bonafide work of **Gopal Kumar** (**Roll No. 1401CS17**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Patna under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Joydeep Chandra**

Assistant Professor,

May, 2018

Department of Computer Science & Engineering,

Patna.

Indian Institute of Technology Patna, Bihar.

Acknowledgements

I take this opportunity to express a deep sense of gratitude towards my guide Dr. Joydeep Chandra, for providing excellent guidance, encouragement and inspiration throughout the project work. I would also like to thank Doctoral candidate Shalini Priya for helping out whenever there was a scope of improvement in the project.

Abstract

During disaster situations people tend to use microblogging site like twitter to express their concern, views or updates about the infrastructural damage. So it is very important to have suitable Information Retrieval (IR) systems that retrieve relevant information from microblogs. The primary aim of this project is to develop information retrieval models which can help collect microblogs relevant to infrastructural damage so that necessary relief operations can be carried out. We present four different models of retrieval out of which two of the models are novel in this project. The first two models are based on Rocchio expansion and word2vec expansion of the query terms. Binary Independence technique is used for developing the rest two models using Croft Harper estimation and Grieff estimation. A comparison among the four models is carried out at the end of the project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Microblogging sites like Twitter and Weibo are crucial sources of information during situations such as an earthquake of a flood [MIV15], [IVS13] . But the drawback of these microblogging sites is people tend to use it for stating opinion and posting sentiments. Thus, the required relevant information can go unnoticed. And since time is vital for carrying out relief operations after a disaster, it is essential to have information retrieval models which can cash in on the information given by these microblogs. In this project the focus is more on infrastructural damage related microblogs (tweets) so that necessary relief operations can be sent to such places or areas in real time. An effective model integrated with the relief operations can help save many lives.

There have been efforts to extract specific types of microblogs (tweets) during disaster situations (see Section 2). The contribution of this project is to build on the prior works and to suggest some more models so that an efficient and apt model can be selected for retrieving the required information. For the same, we collected a set of about 50,000 distinct microblogs posted during disaster events and carried out the our research on the same.

To summarize, the present work has three contributions. First, we develop a test collection for evaluating microblog retrieval methodologies in the context of a disaster

situation. Second, we develop the retrieval models based on the prior works and our present work. Third, we carry out the evaluation based on the test collection and the retrieval models.

1.1 Organization of The Report

- In Chapter 2, we are going to discuss about the prior works and the literature reviewed before taking this project. We will also discuss about the models that are built using prior works.
- In Chapter 3, we are going to discuss about mathematical and theoretical study over cosine similarity and binary independence model.
- In Chapter 4, we will discuss about the background work done before the implementing the models.
- In Chapter 5, the methodologies of all the four information retrieval models that we have evaluated in this project are going to be discussed.
- In Chapter 6, we are going to evaluate the results hence obtained from the models on our dataset.
- In Chapter 7, a conclusion is drawn at the end for the aptness of the model.

Chapter 2

Review of Prior Works

The following two research paper were reviewed before starting with this project work. In both the papers, the retrieval is being done using two models namely Rocchio expansion (section 4.1) and word2vec expansion (section 4.2). We have used these models in our research work alongwith the two models that we proposed.

2.1 Identifying Post-Disaster Resource Needs and Availabilities from Microblogs (Ghosh *et al.* [MBG17b])

In post-disaster relief operations situations, identifying needs and availabilities of various types of resources is critical for effective coordination of the relief operations. This work focuses on the problem of automatically identifying tweets that inform about needs and availabilities of resources, termed as need-tweets and availability-tweets respectively. Traditionally, pattern matching techniques are adopted to identify such tweets. This work presents novel retrieval methodologies, based on word embeddings, for automatically identifying need-tweets and availability-tweets. Experiments over tweets posted during two disaster events show that the proposed methodologies outperform prior pattern-matching techniques

2.2 Microblog Retrieval in a Disaster Situation: A New Test Collection for Evaluation (Ghosh *et al.* [MBG17a])

During disaster situations it is important to design and evaluate Information Retrieval (IR) systems that retrieve information from microblogs. The primary contribution of this paper is to develop a test collection for evaluating IR systems for microblog retrieval in disaster situations. The collection consists of about 50,000 microblogs posted during the Nepal earthquake in April 2015, a set of five topics (information needs) that are practically important during a disaster, and the gold standard annotations of which microblogs are relevant to each topic. This work also presents some IR models that can be suitable in this evaluation setup, including a standard language model based retrieval, and word embedding based retrieval. This work finds that the term embedding based retrieval performs better for short, noisy microblogs.

Chapter 3

Theoretical Study

3.1 Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0° is 1, and it is less than 1 for any other angle in the interval [0,2π). It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. Cosine similarity is particularly used in positive space, where the outcome is neatly bounded in [0,1]. The name derives from the term "direction cosine": in this case, note that unit vectors are maximally "similar" if they're parallel and maximally "dissimilar" if they're orthogonal (perpendicular). This is analogous to the cosine, which is unity (maximum value) when the segments subtend a zero angle and zero (uncorrelated) when the segments are perpendicular.

These bounds apply for any number of dimensions, and cosine similarity is most commonly used in high-dimensional positive spaces. In information retrieval and text mining, each term is notionally assigned a different dimension and a document is characterised by a vector where the value of each dimension corresponds to the number of times that term appears in the document. Cosine similarity then gives a useful measure of how similar two

documents are likely to be in terms of their subject matter.

Since, we have to find the cosine similarity between the vectors in our models it was suitable to explain what is the basis of the term. It is accomplised in python language by using a package called *sklearn.metrics.pairwise.cosine_similarity.*

3.2 Binary Independence Technique

The Binary Independence Model (BIM)[MS08] introduces some simple assumptions, which make estimating the probability function $P(R|d,q)$ practical. Here, "binary" is equivalent to Boolean: documents and queries are both represented as binary term incidence vectors. That is, a document d is represented by the vector $\vec{x} = (x_1, \ldots, x_M)$ where $x_t = 1$ if term t is present in document d and $x_t = 0$ if t is not present in d. With this representation, many possible documents have the same vector representation. Similarly, we represent q by the incidence vector $\vec{q}$. "Independence" means that terms are modeled as occurring in documents independently. The model recognizes no association between terms.

Under the BIM, we model the probability $P(R|d,q)$ that a document is relevant via the probability in terms of term incidence vectors $P(R|\vec{x},\vec{q})$. Then, using Bayes rule, we have:

$$P(R=1|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=1,\vec{q})P(R=1|\vec{q})}{P(\vec{x}|\vec{q})} \quad (1)$$

$$P(R=0|\vec{x},\vec{q}) = \frac{P(\vec{x}|R=0,\vec{q})P(R=0|\vec{q})}{P(\vec{x}|\vec{q})} \quad (2)$$

Here, $P(\vec{x}|R=1,\vec{q})$ and $P(\vec{x}|R=0,\vec{q})$ are the probability that if a relevant or nonrelevant, respectively, document is retrieved, then that document's representation is $\vec{x}$.

Given a query q, we wish to order returned documents by descending $P(R=1|d,q)$. Under the BIM, this is modeled as ordering by $P(R=1|\vec{x},\vec{q})$. Rather than estimating this probability directly, because we are interested only in the ranking of documents, we work with some other quantities which are easier to compute and which give the same ordering of

documents. In particular, we can rank documents by their odds of relevance (as the odds of relevance is monotonic with the probability of relevance). This makes things easier, because we can ignore the common denominator in Rxq-bayes, giving:

$$O(R|\vec{x},\vec{q}) = \frac{P(R=1|\vec{x},\vec{q})}{P(R=0|\vec{x},\vec{q})} = \frac{\frac{P(R=1|\vec{q})P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|\vec{q})}}{\frac{P(R=0|\vec{q})P(\vec{x}|R=0,\vec{q})}{P(\vec{x}|\vec{q})}} = \frac{P(R=1|\vec{q})}{P(R=0|\vec{q})} \cdot \frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})} \tag{3}$$

The left term in the rightmost expression of Equation 3 is a constant for a given query. Since we are only ranking documents, there is thus no need for us to estimate it. The right-hand term does, however, require estimation, and this initially appears to be difficult: How can we accurately estimate the probability of an entire term incidence vector occurring? It is at this point that we make the Naive Bayes conditional independence assumption that the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):

$$\frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})} = \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})} \tag{4}$$

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})} \tag{5}$$

Since each x_t is either 0 or 1, we can separate the terms to give:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=1} \frac{P(x_t=1|R=1,\vec{q})}{P(x_t=1|R=0,\vec{q})} \cdot \prod_{t:x_t=0} \frac{P(x_t=0|R=1,\vec{q})}{P(x_t=0|R=0,\vec{q})} \tag{6}$$

Henceforth, let $p_t = P(x_t = 1|R = 1, \vec{q})$ be the probability of a term appearing in a document relevant to the query, and $u_t = P(x_t = 1|R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document.

Let us make an additional simplifying assumption that terms not occurring in the query

are equally likely to occur in relevant and nonrelevant documents: that is, if $q_t = 0$ then $p_t = u_t$. Then we need only consider terms in the products that appear in the query, and so,

$$O(R|\vec{q},\vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0,q_t=1} \frac{1-p_t}{1-u_t} \tag{7}$$

The left product is over query terms found in the document and the right product is over query terms not found in the document. We can manipulate this expression by including the query terms found in the document into the right product, but simultaneously dividing through by them in the left product, so the value is unchanged. Then we have:

$$O(R|\vec{q},\vec{x}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t} \tag{8}$$

The left product is still over query terms found in the document, but the right product is now over all query terms. That means that this right product is a constant for a particular query, just like the odds $O(R|\vec{q})$. So the only quantity that needs to be estimated to rank documents for relevance to a query is the left product. We can equally rank documents by the logarithm of this term, since log is a monotonic function. The resulting quantity used for ranking is called the Retrieval Status Value (RSV) in this model:

$$RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)} \tag{9}$$

So everything comes down to computing the RSV. Define c_t:

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{(1-p_t)} + \log \frac{1-u_t}{u_t} \quad (10)$$

Now, df_t is the number of documents that contain term t. Under the assumption that relevant documents are a very small percentage of the collection, it is plausible to approximate statistics for nonrelevant documents by statistics from the whole collection. Under this assumption, u_t (the probability of term occurrence in nonrelevant documents for a query) is df_t/N and

$$\log[(1-u_t)/u_t] = \log[(N-df_t)/df_t] \approx \log N/df_t (11)$$

Our next two models are based on this technique and on the estimated values of the variables in calculating the Retrieval Status Value of the tweets.

Chapter 4

Background and Theoretical Study

4.1 Developing Test Collection

The first part of the project is to develop a test collection for microblog retrieval methodologies. For the same, we collected tweets related to the events related to disaster, through the Twitter Search API [6]. In total, we collected around 50,000 English tweets (as identified by Twitters own language detection mechanism) posted during or after the time of the disasters.

4.1.1 Removal of duplicates

Often most of the users on Twitter re-post (retweet) the same information, possibly in slightly different language, leading to duplicate and near-duplicate tweets. It is recommended that the duplicate documents should not be there in test collections for IR evaluation, as they can lead to over-estimation of the performance of IR methodologies, and also create information overload for human annotators (for developing the gold standard). Therefore, we removed duplicates / near-duplicates.

4.1.2 Developing gold standard for retrieval

We used human annotators to develop the gold standard. The annotators were proficient in English and were a frequent user of Twitter. The annotators were given the set of tweets and was asked to identify all tweets relevant to the topic of infrastructural damage.

The test collection developed as described above contains around 50,000 tweets and the gold standard specifying which tweets are relevant to the topic of infrastructural damage.

4.2 Initial Query Generation

Query terms related to topic infrastructural damages	destruction infrastructure damage resources structures dams houses mobile tower communication roads runways railway electricity mobile Internet connectivity buildings destroy

Table 4.1 Manual Initial Query Generation

The first and foremost part of information retrieval is to build up a query to begin testing the dataset for relevant tweets, i.e., tweets related to infrastructural damage. For a given topic, we consider the query to be a set of terms (unigrams) extracted from the text of the topic. In this project, we have generated the initial query by manual query generation. In this method, a human selected specific terms from the text of the topic, i.e., terms related to infrastructural damages, which are intuitively important and likely to be present in tweets relevant to the topic. The resulting query terms are stated as in given in table 4.1.

4.3 Preprocessing of tweets

We further carried out cleaning and preprocessing of our dataset in each model. Since this is a common step in all the models, it being explained here in detail. The preprocessing

and cleaning of the dataset can be carried out by keeping following in mind:

- **Escaping HTML characters:** Data obtained from web usually contains a lot of html entities like < > & which gets embedded in the original data. It is thus necessary to get rid of these entities. One approach is to directly remove them by the use of specific regular expressions. Another approach is to use appropriate packages and modules (for example htmlparser of Python), which can convert these entities to standard html tags.

- **Decoding data:**This is the process of transforming information from complex symbols to simple and easier to understand characters. Text data may be subject to different forms of decoding like Latin, UTF8 etc. Therefore, for better analysis, it is necessary to keep the complete data in standard encoding format. UTF-8 encoding is widely accepted and is recommended to use.

- **Removal of Stop-words:**When data analysis needs to be data driven at the word level, the commonly occurring words (stop-words) should be removed. One can either create a long list of stop-words or one can use predefined language specific libraries.

- **Removal of Punctuations:** All the punctuation marks according to the priorities should be dealt with. For example: ".", ",","?" are important punctuations that should be retained while others need to be removed.

- **Removal of Expressions:** Textual data (usually speech transcripts) may contain human expressions like [laughing], [Crying], [Audience paused]. These expressions are usually non relevant to content of the speech and hence need to be removed.

- **Split Attached Words:** We humans in the social forums generate text data, which is completely informal in nature. Most of the tweets are accompanied with multiple attached words like RainyDay, PlayingInTheCold etc. These entities can be split into their normal forms using simple rules and regex.

- **Apostrophe Lookup:** When apostrophes are used, chances of disambiguation increases. For example "it's is a contraction for it is or it has". All the apostrophes should be converted into standard lexicons. One can use a lookup table of all possible keys to get rid of disambiguates.

- **URLs and hyperlinks:** URLs and hyperlinks in text data like comments, reviews, and tweets should be removed.

4.4 Term vector generation

A Word2vec[TMZ13] based retrieval model that is suitable for short documents like tweets. We first train Word2vec on the tweets. For training Word2vec, the following parameter values were used Vector size: 2000, Context size: 5, Learning rate: 0.05.5 The Word2vec model gives a term-vector for each distinct term in the dataset which encompasses the context in which the term has been used in the dataset.

Chapter 5

Information Retrievals Models

Once we develop our test collection for information retrieval the next step is to formulate models for the retrievals. As we studied in the earlier sections, past works have been carried out in this regard of retrievals. Building on the past work we will be assessing two models from previous works with the two novel models formulated in this work.

5.1 Model 1: Rocchio Expansion

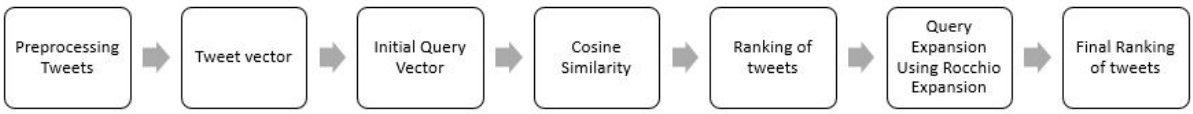

Fig. 5.1 Model 1 (Rocchio expansion) methodology

The main steps involved in the methodology[5] of this model is expained as follows:

1. **Preprocessing tweets:** The given dataset of tweets are first of all preprocessed using the rules as stated above to get rid of unnecessary terms.

2. **Tweet vector:** For each tweet (pre-processed), we construct a tweet-vector by adding the term-vectors of all terms contained in the tweet and then dividing the vector sum

by the number of terms in the tweet.

3. **Initial query vector:** For the given query, we construct a query-vector by performing vector addition of the term-vectors of all terms in the query, and then dividing the vector sum by the number of words in the query.

4. **Cosine similarity:** The cosine similarity of the initial query vector and each tweet vector is hence calculated and stored.

5. **Ranking of tweets:** The tweets are ranked in the descending order of the score of cosine similarity as calculated with the initial query vector.

6. **Query expansion using Rocchio Expansion:**The motivation of the query expansion phase is to add (to the query) some dataset-specific or event-specific terms, so that more relevant tweets can be retrieved. We apply the Rocchio expansion scheme for determining the candidate expansion terms from the top-ranked tweets retrieved using the initial query. Here, after documents are retrieved using a particular (initial) query, the top-ranked k (a small number) documents are assumed to be relevant, and certain terms are selected from the top retrieved documents to expand the query. Specifically, for each distinct term in the k = 10 top-ranked tweets retrieved by the original query, we compute the tf idf Rocchio scores, where tf is the frequency of the term among the 10 top-ranked tweets, and idf is the inverse document frequency of the term over the entire dataset. The top p = 5 terms in the decreasing order of Rocchio scores are selected for expanding the query

7. **Final ranking of tweets:** With the new query, steps 3,4,5 are redone in order to achieve a final list of tweets in descending order of cosine scores calculated with the tweet vector and the newly expanded query vector.

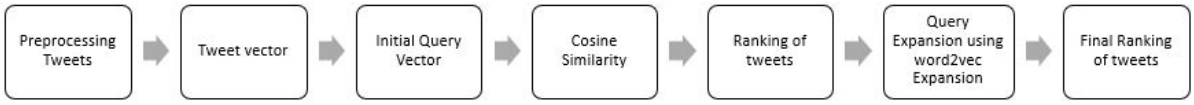

Fig. 5.2 Model 1 (Word2vec expansion) methodology

5.2 Model 2: Word2vec Expanion

The main steps involved in the methodology of this model is expained as follows:

1. **Preprocessing tweets:** The given dataset of tweets are first of all preprocessed using the rules as stated above to get rid of unnecessary terms.

2. **Tweet vector:** For each tweet (pre-processed), we construct a tweet-vector by adding the term-vectors of all terms contained in the tweet and then dividing the vector sum by the number of terms in the tweet.

3. **Initial query vector:** For the given query, we construct a query-vector by performing vector addition of the term-vectors of all terms in the query, and then dividing the vector sum by the number of words in the query.

4. **Cosine similarity:** The cosine similarity of the initial query vector and each tweet vector is hence calculated and stored.

5. **Ranking of tweets:** The tweets are ranked in the descending order of the score of cosine similarity as calculated with the initial query vector.

6. **Query expansion using Word2vec Expansion:**To expand the initial query using word2vec, we compute the cosine similarity of the query-vector with the term-vector of every distinct term in the dataset, and select those p = 5 terms for which the termvector has the highest cosine similarity with the query-vector.

7. **Final ranking of tweets:** With the new query, steps 3,4,5 are redone in order to achieve a final list of tweets in descending order of cosine scores calculated with the tweet vector and the newly expanded query vector.

5.3 Model 3: Croft and Harper estimation

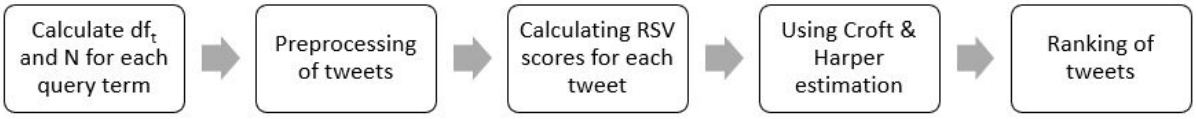

Fig. 5.3 Model 3 (Croft and Harper estimation) methodology

The main steps involved in the methodology of this model is expained as follows:

1. **Calculating df_t and N for query terms:** N is the total number of distinct documents. It would be constant for the complete dataset, here N would be the number of distinct tweets. For Calculating df_t we have to calculate the number of documents that contain the query term.

2. **Preprocessing tweets:** The given dataset of tweets are first of all preprocessed using the rules as stated above to get rid of unnecessary terms.

3. **Calculating RSV Scores for each tweet:** For caculating the RSV score of a tweet we would first calculate the value of c_t (equation 11) for each term and then have the summation over all the terms in the tweet to get the final RSV score of the tweet. The final expression of RSV is given by equation 12 where all the values for a given term are known except p_t which we would estimate in the next step.

$$RSV_d = \sum_{t:x_t=q_t=1} \log \frac{p_t}{(1-p_t)} + \log N/\mathrm{df}_t \qquad (12)$$

4. **Using Croft and Harper estimation:** p_t is constant over all terms x_t in the query and that $p_t = 0.5$. This means that each term has even odds of appearing in a relevant document, and so the p_t and $(1 - p_t)$ factors cancel out in the expression for RSV. Combining this method with our earlier approximation for u_t, the document ranking is determined simply by which query terms occur in documents scaled by their idf weighting.

5. **Ranking of tweets:** The tweets are ranked in the descending order of the score of Retrieval Status Value (RSV) as calculated above.

5.4 Model 4: Grieff Estimation

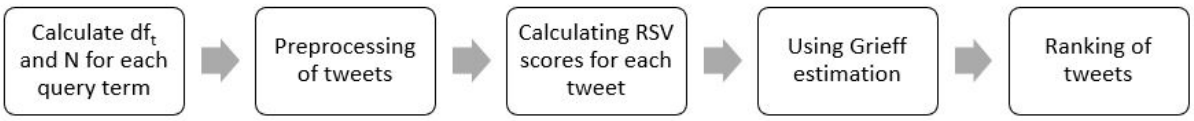

Fig. 5.4 Model 4 (Grieff estimation) methodology

The main steps involved in the methodology of this model is expained as follows:

1. **Calculating df_t and N for query terms:** N is the total number of distinct documents. It would be constant for the complete dataset, here N would be the number of distinct tweets. For Calculating df_t we have to calculate the number of documents that contain the query term.

2. **Preprocessing tweets:** The given dataset of tweets are first of all preprocessed using the rules as stated above to get rid of unnecessary terms.

3. **Calculating RSV Scores for each tweet:** For caculating the RSV score of a tweet we would first calculate the value of c_t (equation 11) for each term and then have the summation over all the terms in the tweet to get the final RSV score of the tweet.

The final expression of RSV is given by equation 12 where all the values for a given term are known except p_t which we would estimate in the next step.

$$RSV_d = \sum_{t:x_t=q_t=1} \log \frac{p_t}{(1-p_t)} + \log N/\mathrm{df}_t \tag{12}$$

4. **Using Grieff estimation:** As might be expected, p_t is shown to rise with df_t. Based on Grieff's data analysis, a plausible proposal would be to use the estimate $p_t = \frac{1}{3} + \frac{2}{3} df_t/N$.

5. **Ranking of tweets:** The tweets are ranked in the descending order of the score of Retrieval Status Value (RSV) as calculated above.

Chapter 6

Evaluations and Conclusions

The outputs that we obtained have to be evaluated to check out the competency and efficiency of the model hence structured. Evaluation measures for information retrieval were used for the same to get an insight to efficiency of the models.

Evaluation measures for an information retrieval system are used to assess how well the search results satisfied the user's query intent. Such metrics are often split into kinds: online metrics look at users' interactions with the search system, while offline metrics measure relevance. Offline metrics are generally created from relevance judgment sessions where the judges score the quality of the search results. Both binary (relevant/non-relevant) and multi-level (e.g., relevance from 0 to 5) scales can be used to score each document returned in response to a query. In practice, queries may be ill-posed, and there may be different shades of relevance. We would be using three different parameters of Offline metrics of evaluation measures to compare the results:

6.1 Precision

Precision is the fraction of the documents retrieved that are relevant to the user's information need.

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|} \quad (13)$$

In binary classification, precision is analogous to positive predictive value. Precision takes all retrieved documents into account. It can also be evaluated at a given cut-off rank, considering only the topmost results returned by the system. This measure is called precision at n or P@n.

6.2 Recall

Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|} (14)$$

In binary classification, recall is often called sensitivity. So it can be looked at as the probability that a relevant document is retrieved by the query.

It is trivial to achieve recall of 100% by returning all documents in response to any query. Therefore, recall alone is not enough but one needs to measure the number of non-relevant documents also, for example by computing the precision.

6.3 F-Measure

The weighted harmonic mean of precision and recall, the traditional F-measure or balanced F-score is:

$$F = \frac{2 \cdot \text{precision} \cdot \text{recall}}{(\text{precision} + \text{recall})} (15)$$

This is also known as the F_1 measure, because recall and precision are evenly weighted.

6.4 Results

Following results were obtained when we calculated precision@100 and recall@100. The cutoff k=100 was selected for top 100 tweets in all the four models and the gold standard once framed was used in calculation of the values of the evaluation measures. The following results are obtained which are noted down in the table. In the gold standard there were

Eval. Measures	Model 1	Model 2	Model 3	Model 4
Precision	0.08	0.29	0.32	0.31
Recall	0.009	0.032	0.036	0.035
F-Measure	0.163	0.0591	0.0653	0.0632

Table 6.1 Evaluation Results of the models

880 tweets relevant to infrastructural damage and the top 100 tweets were having 8,29,32 and 31 relevant tweets respectively.

Fig. 6.1 Comparison between all the models

Model 3 (Croft and Harper) performs the best by achieving both reasonable precision as well as reasonable recall, leading to significantly better F-score values. Model 2(Word2vec) and Model 4 (Grieff) were comparatively with the latter having a good F-score as well. Model 1 (Rochhio) could not give satisfactory results. The reason for this might be lower cutoff value of 100, a larger value might give a better insight at the standings of all the four models and their competencies.

Chapter 7

Conclusion

We examined a huge dataset of tweets in order to retrieve tweets which may give us idea about the damages related to infrastructure. The resulting tweets were very close and exact to what we had aimed for in the beginning. However, the low F-scores indicate that there is still plenty of room for improving the techniques, and we look forward to exploring more effective techniques in future. This work makes available to the community a novel test collection method, some novel models for information retrievals and some efficient evaluations techniques for microblog retrieval strategies for information needs pertaining to infrastructural damage in a disaster situation.

References

[6] Twitter search api.

[IVS13] K. Torisawa C. Hashimoto K. Ohtake T. Kawai J.-H. Oh I. Varga, M. Sano and S. D. Saeger. Aid is out there: Looking for help from tweets during a large scale disaster. In *ACL*, 2013.

[MBG17a] K. Ghosh S. Bandyopadhyay M. Basu, A. Roy and S. Ghosh. Microblog retrieval in a disaster situation: A new test collection for evaluation. In *Workshop on Exploitation of Social Media for Emergency Relief and Preparedness (SMERP) co-located with European Conference on Information Retrieval*, 2017.

[MBG17b] S. Das R. Dey S. Bandyopadhyay M. Basu, K. Ghosh and S. Ghosh. Identifying post-disaster resource needs and availabilities from microblogs. In *ASONAM*, 2017.

[MIV15] F. Diaz M. Imran, C. Castillo and S. Vieweg. Processing social media messages in mass emergency: A survey. *ACM Computing Surveys*, 47(4):167, jun 2015.

[MS08] P.; Manning, C. D.; Raghavan and Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[TMZ13] W. Yih T. Mikolov and G. Zweig. Linguistic regularities in continuous space word representations. In *NAACL HLT*, 2013.