# M. TECH. THESIS

On

# Increasing Retrieval Efficiency In Noisy Corpora

Author

**Riya Roy**

**M.TECH in COMPUTER SCIENCE**

**INDIAN STATISTICAL INSTITUTE, KOLKATA**

**July 2018**

# Increasing Retrieval Efficiency In Noisy Corpora

**A Thesis**

*Submitted in partial fulfillment of the*
*requirements for the award of the degree*

***of***
**MASTER OF TECHNOLOGY**
**in**

**COMPUTER SCIENCE**

*Submitted by:*
**RIYA ROY**

*Guided by:*
**Mandar Mitra**

**INDIAN STATISTICAL INSTITUTE, KOLKATA**
**July 2018**

# CANDIDATE'S DECLARATION

I hereby declare that the project entitled **Increasing Retrieval Efficiency In Noisy Corpora** submitted in requirement for the award of the degree of Masters of Technology in Computer Science and Engineering completed under the supervision of Dr. Mandar Mitra

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

**Signature and name of the student with date**

---

**CERTIFICATE by Guide**

It is certified that the above statement made by the student is correct to the best of my knowledge.

**Signature of Guide with dates and their designation**

---

**Acknowledgement**

First and foremost I offer my sincerest gratitude to my supervisor, Dr. Mandar Mitra, who has supported me throughout my dissertation with his patience and knowledge whilst allowing me the room to work in my own way. One simply could not wish for a better or friendlier supervisor.
All the members of Information Retrieval Laboratory of CVPR Unit, ISI made it a convivial place to work. In particular, I would like to thank Dwaipayan Roy, Suchana Dutta and Ayan Bandapadhyay for their help and admiration in the past one year.
My deepest gratitude goes to my father, mother and sister's support throughout my life. I would also like to thank my friends for giving me a wonderful time to spend with.

**Abstract**

In this thesis we tried to catch the word variations in the noisy corpus. Initially we tried to solve the problem using string similarity and context similarity in the Generalized Language Model. But then this model was unable to improve the retrieval performance as seen experimentally. On delving into the depth of the problem as to why the model was not performing well we came up with a simple and effective approach to solve the problem. This is a simple Query Expansion based method which is used to increase the retrieval performance.

# Contents

# Chapter 1

# Introduction

## 1.1 Noisy Corpora

Noisy Corpora contains Noisy text. Noisy text is a text with differences between the surface form of a coded representation of the text and the intended, correct, or original text. The noise may be due to typographic errors or colloquialisms always present in natural language and usually lowers the data quality in a way that makes the text less accessible to automated processing by computers, including natural language processing. The noise may also have been introduced through an extraction process (e.g., transcription or OCR) from media other than original electronic texts. [1]

**These days it is very common to come across such type of noise in text. The social media like twitter, sms proliferates with such type of noisy text. Moreover these noises are very common in Legal Documents which are misread by OCR.**

## 1.2 Motivation

1. The process of *word normalization* enhances retrieval performance at query time
2. The stemming algorithm in the paper *Combining Local and Global Word Embeddings for Microblog Stemming CIKM 2017* is ad hoc
3. Goal is to see if a simpler approach can yield equal retrieval effectiveness
4. **Challenges In Microblog Data**
   (a) Tweets needs to be of 140 characters and hence they are abbreviated : *iloveindia*

[1] The definition has been taken from wikipedia

(b) There are lot spelling variations example : building is written as *bdlng*.

5. **Challenges In OCR Data**

   The ocr data is very poorly tampered and the data not only conatins wrong spelling of the words but it also includes non alphanumeric characters. So to retrieve from such a collection is really a challenging work.

Our aim is to come up with a simplified approach to solve the problem related to noisy corpus in an effective way. It is a huge challenge to solve this problem because the most of the time it becomes difficult to find out which valid word it is referring to after being corrupted.

**Effective information retrieval from noisy corpus is a very practical problem. Hence we have chosen our problem as *Increasing Retrieval Efficiency In Noisy Corpora***

## 1.3 Our Approach

We tried using only context similarity and only string similarity to catch the word variation in noisy corpus. Later we combined the string similarity and context similarity in a linear combination to perform the same. This was a document expansion approach with contextual words or with semantically similar words. We experimentally found out that this approach was unable to produce the effective result we were looking for. Hence we changed our approach from document expansion to query expansion methodology. In this approach we used string similarity measures as well as context based methods independently to catch the word variations. Every time we found out that the string similarity based approach was doing better than the context similarity based approach. Our method is able to produce good result in more than 4 data set so we presume it is a presentable work.

## 1.4 Thesis Outline

Here is a outline of what each of the chapter is about. The first chapter i.e **chapter 1** is a introductory chapter which is about the motivation behind choosing the topic and a very brief outline of what our approaches were. **chapter 2** is the about the details of IR and also about the tools and terminolgies used throughout the thesis work. **chapter 3** is the related work which is about the two baseline paper we have used and their algorithms. **chapter 4** is our proposed methodology 1 and **chapter 5** is our proposed methodology 2. The algorithms and details are explained here. **chapter 6** is about the results of the experiments. **chapter 7** is about the future work and conclusion.

# Chapter 2

# Preliminaries

## 2.1 Definition

Information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on metadata or on full-text.[1]

Information retrieval is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

## 2.2 Document, Collection And Query

A document is a any textual content containing significant information about a topic. Examples of documents are web pages, email, books, news, stories, scholarly papers, text message .

A set of documents of same type is called Collection.

An information retrieval process begins when a user enters a query into the system. Queries are formal statements of information needs, for example search strings in web search engines. In information retrieval a query does not uniquely identify a single object in the collection. Instead, several objects may match the query, perhaps with different degrees of relevancy.[2]

[1] Certain portions were taken from previous thesis on Information Retrieval
[2] Certain portions were taken from previous thesis on Information Retrieval

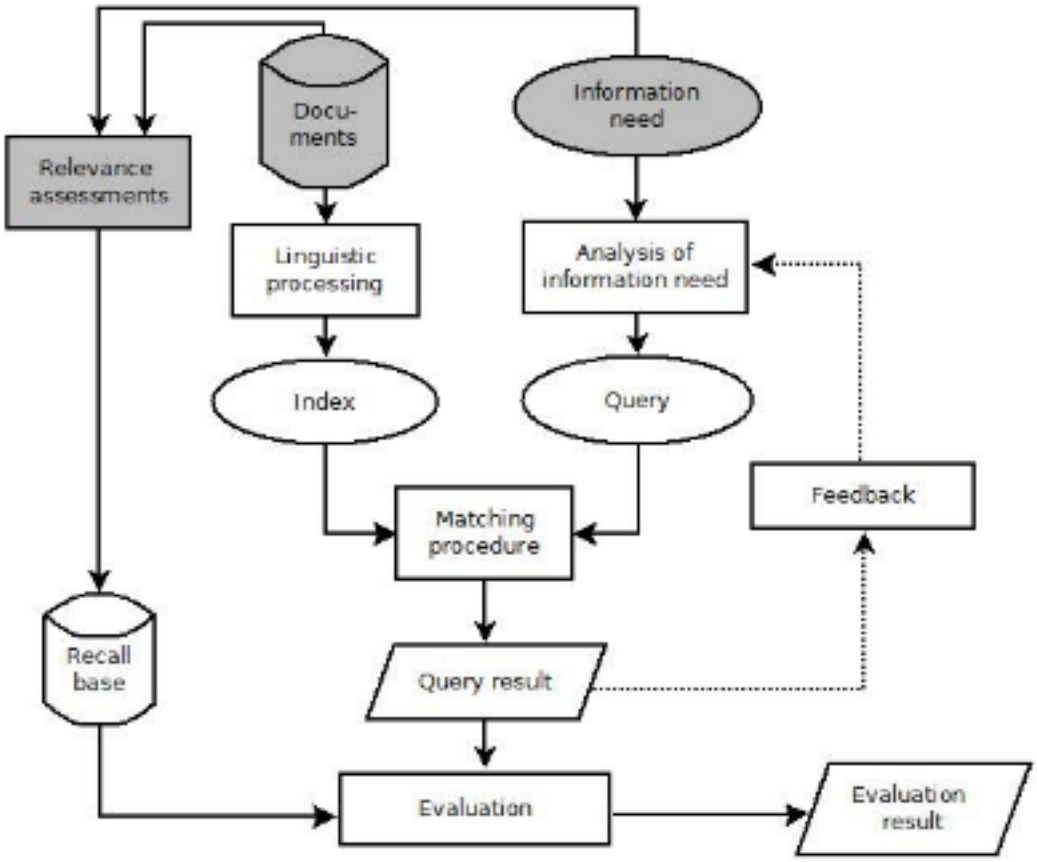


Figure 2.1: Overview Of IR Model.

[3]

## 2.3 Cranfield Paradigm

This paradigm gives an overview of how an overall IR model works. This overview is explained diagramatically in the above figure.

## 2.4 Vector Space Model

In vector space model documents and queries are represented as vectors. This model is very commonly used.

$$d_j = (w_{1_j}, w_{2_j}, w_{3_j}, ....w_{t_j})$$
$$q = (w_{1_q}, w_{2_q}, w_{3_q}, ....w_{n_q})$$

Each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. Several different ways of computing these values, also known as (term) weights, have been developed. One of the best known schemes is tf-idf weighting. TF means term frequency which is the number of times a term occurs in a document. IDF means inverse document frequency which is the log of the ratio of collection size and number of documents containing that term. The definition of term depends on the application. Typically terms are single words, keywords, or longer phrases. If the words are chosen to be the terms, the dimensionality of the vector is the number of words in the vocabulary (the number of distinct words occurring in the corpus). Vector operations can be used to compare documents with queries.

## 2.5 Language Modelling

In language model based retrieval methods, for a given query $Q$, documents are scored based on the probability $P(Q|d)$. $d$ is taken to be a language model i.e a probability distribution over words. $P(Q|d)$ represents the probability of generating $Q$ from $d$. $P(Q|d)$ is the probability of observing the query Q in the document d.

$$\begin{aligned}
\text{Score}(Q,d) &= p(Q|\mathcal{D}) \\
&= \prod_{t \in Q} p(t|d) \\
&= \prod_{t \in Q} \lambda \hat{p}(t|d) + (1-\lambda)\hat{p}(t|C) \\
&= \prod_{q \in Q} \lambda \frac{tf(t,d)}{|d|} + (1-\lambda)\frac{cf(t)}{|C|} \qquad (2.1)
\end{aligned}$$

In Equation 2.1, the set $C$ represents a universe of documents (commonly known as the *collection*), $\hat{p}(t|d)$ and $\hat{p}(t|C)$ denote the maximum likelihood estimated probabilities of generating a query term $t$ from the document $d$ and the collection $C$ respectively. The probabilities of these two (mutually exclusive) events are denoted by $\lambda$ and $1-\lambda$ respectively. The notations $tf(t,d)$, $|d|$, $cf(t)$ and $|C|$ denote the term frequency of term $t$ in document $d$, the length of $d$, i.e total number of words or terms contained in $d$, collection frequency of the term $t$, and the collection size respectively.

## 2.6 Retrieval And Evaluation Procedure

### 2.6.1 Retrieval

Before actual retrieval begins, the documents within the collection must be indexed. Indexing involves processing each document in a collection and building a data structure of indexed documents. Following are the steps of indexing :

- 1 . Reading and parsing a document.
- 2 . Stopword removal and stemming of each term in the document.
- 3 . Inserting each term in the data structure of indexed documents.
- 4 . Using some retrieval model to index the data set.

  The first step of retrieval is to assign a score to each document according to its relevance with the given query.

### 2.6.2 Evaluation

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

$$Precision = \frac{\text{Number Of Relevant Documents Retrieved}}{\text{Number Of Documents Retrieved}}$$

- Recall

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

$$Recall = \frac{\text{Number Of Relevant Documents Retrieved}}{\text{Number Of Relevant Documents In The Collection}}$$

- Mean Average Precision(MAP)

  Average Precision Mean of the precision scores for a single query after each relevant document is retrieved, where relevant documents not retrieved have P of zero.MAP is the mean of average precisions for a query batch.

- Bpref

  The bpref measure is designed for situations where relevance judgments are known to be far from complete. It was introduced in the TREC 2005 terabyte track. Bpref computes a preference relation of whether judged relevant documents are retrieved ahead of judged irrelevant documents. Thus, it is based on the relative ranks of judged documents only.

## 2.7 Stemming

In linguistic morphology and information retrieval, stemming is the process of reducing inflected ( or sometimes derived ) words to their word stem, base or root form generally a written word form. The stem need not be identical to the morphological root of the word , it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root.

**Stemming is a vital step employed to improve retrieval performance through efficient unification of morphological variants of a word**

## 2.8 Different Stemmers Used

In our proposed work we have tried out various stemmers :

- **Porter Stemmer**

  The porter stemming a ( language dependent ) algorithm is a process of removing suffices from words in English. Removing suffixes automatically is an operation which is specially useful in the field of information retrieval. In a typical IR system a document is represented by a vector of words or terms. Terms with a common stem usually have same meaning.

- **Yass**

  Yet Another suffix Stripper ( Yass ) is a statistical stemmer. In this stemmer the distance between two words X and Y are measured using the distance formula :

$$D_3(X,Y) = \begin{matrix} \frac{n-m+1}{m} * \sum_{i=m}^{n} 2^{i-m} \text{ if m} > 0 \\ 0 \text{ otherwise} \end{matrix}$$

  The distance functions defined previously are used to cluster words into homogeneous groups. Each group is expected to represent an equivalence class consisting of morphological variants of a single root word. The words within a cluster are stemmed to the central word in that cluster. Since number of cluster is not known from before hand graph theoretic clustering algorithm is used. Therefore the complete linkage method of clustering is used to cluster the words.

- **Gras**

  It is a novel graph based language independent computationally inexpensive stemming algorithm for information retrieval. In this algorithm first the words are clustered based on the condition that they have some common prefix of length l. Now for any two words in the cluster, the suffix pairs should be obtained. In this way the count of the suffix pair will be obtained. A graph is constructed from the words of the lexicon and the weight of the edges being the frequency of suffix pair. A vertex is selected with the maximum degree and considering all its adjacent or neighbouring vertices the ones having strong cohesion value are kept rest are deleted. The new graph obtained is devoid of the vertices removed in the first step. Now again the process of finding a vertex with maximum degree is found and process of is repeated. So basically in every step we are getting the classes of the words in the algorithm.

## 2.9 String Similarity

[4]

- **Longest Common Subsequence (LCS) Similarity**

  Given a sequence X = ( $x_1$ , $x_2$ ,..., $x_m$ ) a sequence Z = ( $z_1$ , $z_2$ ,..., $z_k$ ) ( k $\leq$ m ) is called a subsequence of X if there exists a strictly increasing sequence $i_1$ , $i_2$ ,..., $i_k$ of indices of X such that for all j = 1,2,..., k , we have $x_{i_j} = z_j$ . Now, given two sequences X and Y , we say that Z is a common subsequence of X and Y if Z is a subsequence of both X and Y. A common subsequence of X and Y that has the longest possible length is called a longest common subsequence or LCS of X and Y.

$$\text{LCSSimilarity}(\ w_1\ ,\ w_2\ )\ = \frac{StringLength(\ LCS(\ w_1,w_2)\ )}{Maximum(\ StringLength(\ w_1)\ ,StringLength(\ w_2)\ )}$$

- **Edit Distance Based Similarity**

  Edit distance is a popular measure for measuring distance between two strings. Edit distance between two words is the minimum number of single character edits, i.e., insertions, deletions or substitutions, required to change one word into the other. Let ED denotes edit distance and ES denotes edit similarity.

$$\text{ES}(\ w_1\ ,\ w_2\ )\ = 1 - \frac{ED(\ w_1,w_2)}{Maximum(\ StringLength(\ w_1)\ ,StringLength(\ w_2)\ )}$$

[4]The defintions of these metrics were taken from baseline paper.

## 2.10 Data Set

### 2.10.1 Microblog

Microblogging sites ( e.g., Twitter, Weibo ) have become important sources of information on various topics and events. On such forums, the user generated content is usually written in cavalier, informal ways without adhering to standard grammar rules. Also , due to strict limitation on the length of tweets ( 140 characters ) individual words are often shortened in non standard ways or multiple words are clubbed together. These factors pose unique challenges to researchers looking to perform Information Retrieval (IR) from such texts. The data set which was used here is TREC 2016 Microblog Data set. This is a twitter data which has lot of word inflections.

### 2.10.2 Fire Risot Bangla Collection

Bangla original is the clean or error free version created from Anandabazar Patrika. Bangla OCRed is the scanned and OCRed version of the same. A document in the original version and its OCRed version had the same unique document identification string so that the original-OCRed pairs can be easily identified. The number of unique terms in Bangla original corpus is 396968 while the same number in its OCRed version is 466867.

### 2.10.3 Fire Risot Hindi Collection

Hindi original is the clean or error free version created from Anandabazar Patrika. Hindi OCRed is the scanned and OCRed version of the same. The number of unique terms in Hindi original corpus and its OCRed version are 242047 and 264240 respectively. This discrepancy is caused by OCR errors. Most of the inflations are caused by misrecognition ( as multiple candidates ) . We will have a more detailed discussion on this issue in a subsequent section. The Bangla collection has 66 topics and the Hindi collection has 28 topics. These topics were created for previous FIRE Ad Hoc tasks. A subset of the Ad Hoc topics was selected for the RISOT task.

### 2.10.4 IIT CDIP

IIT CDIP 1.0 (Illinois Institute of Technology Complex Document Information Processing Test Collection, version 1.0) is a data set supporting research in information retrieval, document analysis, computational linguistics, data mining, and related fields. IIT CDIP Records 1.0: 6,910,192 XML records describing documents that were released in various lawsuits against the US tobacco companies and research institutes.IIT CDIP Assessments 1.0: Relevance judgments on the 40 topics against

pooling-based samples of the records. IIT CDIP collection lacks the clean-text original version which is useful in error-modelling.

### 2.10.5 Trec 5 Confusion Track

This is the document set used in the TREC-5 confusion track. It consists of the 1994 edition of the Federal Register. The United States Government Printing Office (GPO) prints the Federal Register as a record of the transactions of the government. One issue is published each business day and contains notices to Federal agencies and organizations, executive orders and proclamations, proposed rules and regulations, etc. The Federal Register was selected for these experiments because it is a large collection for which both hardcopy and electronic versions are readily available. The corpus contains 395MB of text divided into approximately 55,600 documents.

## 2.11 Word2Vec

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space. Word2vec can utilize either of two model architectures to produce a distributed representation of words **continuous bag of words** or **continuous skip gram**.
In the **continuous bag of words** architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag of words assumption) .
In the **continuous skip-gram** architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words . [5]

## 2.12 Fasttext

fastText is a library for learning of word embeddings and text classification created by Facebook's AI Research (FAIR) lab. The model is an unsupervised learning algorithm for obtaining vector representations for words. FastText differs in the sense that word vectors a.k.a word2vec treats every single word as the smallest unit whose vector representation is to be found but FastText assumes a word to be formed by a

[5]The details were being taken from wikipedia.

n-grams of character. [6]

## 2.13 Retrieval Methods

### 2.13.1 LM Jelinek Mercer Smoothing

The first approach we can do is to create a mixture model with both distributions:

$$P(q|d, C) = \lambda P(q|d) + (1 - \lambda)P(q|C)$$

where q is the Query .d is the document and C is the collection in which the documents are present.

This model tries to find out the probability of finding the query terms in the documents and in case it misses that it smoothes the model using the probabilistic occurence in its collection.
High value of $\lambda$ : conjunctive like search tends to retrieve documents containing all query words.
Low value of $\lambda$: more disjunctive, suitable for long queries

### 2.13.2 LM Dirichlet Smoothing

This model in general performs better than LM Jelinek Mercer in terms of retrieval performance.

$$P_\mu(w|d, C) = \frac{|d|}{|d|+\mu} \frac{c(w,d)}{|d|} + \frac{\mu}{|d|+\mu} P(w|C)$$

where w is the query word , d is the document ,$|d|$ is the length of the document , $\mu$ is a parameter and $c(w, d)$ is the count of the word w in the document d.

[6]The details were being taken from wikipedia.

# Chapter 3

# Related Work

## 3.1 Combining Local and Global Word Embeddings for Microblog Stemming

The above paper (CLGWEMS) [?, ?, ?, ?] illustrates a method to improve retrieval performance through efficient unification of morphological variants of a word. They have proposed an unsupervised, context-specific stemming algorithm for microblogs, based on both local and global word embeddings, which is capable of handling the informal, noisy vocabulary of mi-croblogs .

Word context plays a crucial role in the case of microblogs where non-standard word representations are frequent. Here,a combination of common prefix and context is essential in identifying semantically related variants. In this work, they have used word embedding tools word2vec for local embeddings, and GloVe in `https://nlp.stanford.edu/projects/glove/` for global embeddings to capture the context of word variants, in amalgamation with common prefix length and other string similarity measures, to identify morphological variants of words.

### 3.1.1 Two Measures Of Similarity

- **String Similarity :** This is achieved in two ways. First, if the two words have a common prefix of length p which is a positive integer, they are similar in the sense that they are likely to be morphological variants of each other. We consider p to be very short for tweets where mismatches can occur very early in the word. We choose p $\leq 3$ for noisy , informal microblogs, since a common prefix of length 3 was used for formally written text. Second, we compute the length of the Longest Common Subsequence LCS of the residual suffixes, say $w_{suff}$ , $w'_{suff}$ of the two words $w$ and $w'$ respectively , after ignoring the common prefix

of length p. This LCS similarity is denoted by $LCS_{length}(w_{suff}, w'_{suff})$ .

- **Contextual Similarity :** There is two types of contextual similarity . One is local context and other global context .
  - **Local Context :**
    To get the corpus specific embedding of the words they have trained word2vec over the set of tweets. This will construct the embeddings of words which are present specifically in the cor- pus and in the queries. The word2vec model provides a vector for each term in the corpus, which we call the term-vector. Let $w^{"}$ and $w^{"}_{\prime}$ be the word2vec term-vectors of $w$ and $w_{\prime}$ respectively. The term vector is designed to arrest the context in which a word appears in the corpus . Hence, we measure the contextual similarity of the two words as the cosine similarity of the corresponding $w^{"}$ and $w^{"}_{\prime}$ term-vectors, denoted as cossim ( $w^{"}$,$w^{"}_{\prime}$ ) . So, we consider $w_{\prime}$ to be a possible variant of $w$ only if they have high string similarity as well high contextual similarity as observed in the corpus.
  - **Global Context :** To incorporate more robust context embeddings for the general words, they used the a pretrained embedding for tweets produced by the GloVe algorithm which is pretrained on 2 billion random tweets available at `https://nlp.stanford.edu/projects/glove/`. They used the 50 dimensional word vectors for the exper- iments. The global context is used to refine the final clusters of word variants, as described later in the section. Then computing the global contextual similarity between two words $w$ and $w_{\prime}$ as the $w_g$ and $w'_g$ cosine similarity between the word vectors of the GloVe model.

### 3.1.2 Algorithm

For given tweets let $L$ be the lexicon or set of words of the corpus which are lower case folded excluding the stopwords, URLs, user mentions, email ids, and other non alpha numeric words.

- **Grouping :** Identification of possible variants by string matching. The purpose of this step is to identify the possible variants of words using string similarity.For each word $w \in L$ , a set $L_w$ is formed containing all the words $w_{\prime} \in L$ satisfying the following three conditions:
  - 1 : $w_{\prime}$ has the same common prefix of length p as w, where $p \leq 3$

- 2 : $|w_{'}|, |w| \lneq$ p + 2 , ( where $|s|$ denotes the length of string s ) i.e., $w_{'}$ , $w$ are of length not less than p+2. The need for this condition is explained below.
- 3 : The length of the Longest Common Subsequence of characters between the suffixes $w_{suff}$ and $w^{'}_{suff}$ follows the condition , LCS l length ( $w_{suff}$ , $w^{'}_{suff}$ ) $\lneq \alpha *$ MAX ( $|w_{suff}|, |w^{'}_{suff}|$ ) where $\alpha \in [0, 1]$ is a parameter of the algorithm, and MAX ( a,b ) denotes the larger value between a and b.

The condition ( 3 ) described above captures suffix match beyond the common prefix. The condition ( 2 ) above ensures the LCS of the suffixes ( after ignoring the common prefix ) may be too small to ensure a reasonable comparison.

- **Filtering :** Identify possible variants by contextual matching . In this step, it is aimed at selecting the variants of w from the set $L_w$ that have considerable contextual similarity with w. First the word w $\in L_w$ is found , w $\neq w_{'}$ such that cossim( w , $w_{'}$ ) > cossim( w , $w_{"}$ ) for all $w_{"}$ ( $\neq$ w , $w_{'}$ ) $\in L_w$ that is $w_{'}$ is a word in $L_w$ that is semantically most similar to group head w ( measured by the cosine similarity of the word2vec vectors of the two terms) . $\gamma = \lambda *$ cossim( w , $w_{'}$ ) where $\lambda$ is between 0 to 1 .This threshold is used to calibrate the relative similarity of all other words in $L_w$ with respect to the maximum similarity of any word in $L_w$ with group head w.
$L_w^{refined} = \{ w_s : w_s \in L_w$ and cossim ( w , $w_s$ ) $\leq \gamma \}$ . $L_w^{refined}$ is expected to contain those words which are both string and semantic simialarity with $w$ .

- **Selection :** Now $L_w^{refined}$ is formed for each w but this $L_w^{refined}$ have overlaps . So for a word $w_{'}$ belonging to more than one $L_w^{refined}$ , it is kept in only that particular $L_w^{refined}$ which has highest semantic simialrity with root word w.

- **Refinement :** Now $L_w^{refined}$ is a modified set . The similarity between two words in $L_w^{refined}$ is that of glove similarity. Now the words in $L_w^{refined}$ are clustered according to the given condition. Two words $w_1$ and $w_2 \in L_w^{refined}$ are in same cluster if they satisfy the below condition.
  - 1 : Among all the words in $L_w^{refined}$ either $w_1$ has the highest glove similarity with $w_2$ or vice versa.
  - 2 : Glove similarity of the 2 words are more than $\beta$ times the maximum glove similarity among all pair similarity of the words in $L_w^{refined}$.
  - 3 : If $w_1$ or $w_2$ are absent in the glove model than by default they are put in the same cluster.

Now each of the $L_w^{refined}$ is splitted into several cluster . The cluster which contains w is choosen as the final cluster . In that cluster the word of the shortest length is found as the stem of the cluster.

## 3.2 Improving Information Retrieval Performance On OCRed Text in the Absence of Clean Text Ground Truth

The above paper proposes a novel language independent approach for detecting OCR errors and improving retrieval performance from the erroneous corpus in a situation where training samples are not available to model errors. In this paper a method has been proposed that automatically identifies erroneous term variants in the noisy corpus, which are used for query expansion, in the absence of clean text. An effective combination of contextual information and string matching techniques has been used here .

### 3.2.1 Algorithm

This has two major parts :

- Agglomeration
- Melding

**Agglomeration**

- Segregation

  Let D denotes document collection and L denotes the set of lexicon or the set of all unique words in the documentents D. Let q $\in$ Q be a query . A set $L_{w_i}^{\alpha}$ = { w $\in$ L : string similarity( w, $w_i$ ) $> \alpha$ } where $\alpha$ is a threshold value lying in the interval ( 0, 1) . So this set has all the words in L which has string similarity with the query word $w_i$ .

- Graph Formation

  Now with the words in $L_{w_i}^{\alpha}$ a graph is formed where an edge between the words are placed if the co-occur in a document.

- Pruning

  The maximum edge weight of the graph is $max_{ew}$ . Those edges of the graph are removed which has edge weight $\beta * max_{ew}$ where $\beta$ is a parameter. Since this

can produce wrong result the pruning is done only when maximum document frequency of nodes of the connected sub graph is greater than $\gamma$ .

- Congregation

  We now cluster the vertices of graph G based on edge weights . Basically $v_1$ is the strongest neighbour of $v_2$ if of all the neighbouring vertices of $v_2$ , the weight of the edge joining $v_1$ and $v_2$ is maximum. The clustering is done according to the following condition .

  Two vertices $v_1$ and $v_2$ will belong to the same cluster if

  - 1: Either $v_1$ is the strongest neighbour of $v_2$
  - 2: Or $v_2$ is the strongest neighbour of $v_1$

## Melding

Let $C = \{Cl_1 , Cl_2 , ....Cl_k \}$ be the cluster formed from $L^{\alpha}_{w_i}$ . A word $w_{closet}$ is found out in all the cluster which has the maximum string similarity with $w_i$ . The cluster containing $w_{closet}$ is chosen as the error variant of the query word . If there are more than one cluster having that variant then the word is not expanded.

# Chapter 4

# Proposed Scheme 1

## 4.1 A Generalized Language Model

In this section, we propose the generalized language model (GLM) that models term dependencies using the embedding of terms. In Information Retrieval, generally it is assumed that the occurrence of two terms $t_i$ and $t_j$ in a document is independent of each other. Firstly it looks for a contribution of a term $t_i$ in the document and then in the collection as in the equation 2.1. This is the smoothing technique.

### 4.1.1 Term Transformation Events

As per Equation 2.1, terms in a query are generated by sampling them independently from either the document or the collection. We propose the following generalization to the model. Instead of assuming that terms are mutually independent during the sampling process, we propose a generative process in which a noisy channel may transform (mutate) a term $t'$ into a term $t$. More concretely, if a term $t$ is observed in the query corresponding to a document $d$, according to our model it may have occurred in three possible ways :

1. **Direct term sampling**: Standard LM term sampling, i.e. sampling a term $t$ (without transformation) either from the document $d$ or from the collection.
2. **Transformation via Document Sampling**: Sampling a term $t'(t' \neq t)$ from $d$ which is then transformed to $t$ by a noisy channel. We denote the probability of this happening as $P(t, t'|d)$. This is given by the below expression.

$$P(t, t'|d) = \sum_{t \in d} tf(t', d).\frac{sim(t, t')}{\sum_{t'' \in C} sim(t', t'')} \tag{4.1}$$

3. **Transformation via Collection Sampling**: Sampling the term $t'$ from the collection which is then transformed to $t$ by the noisy channel. We denote the probability of this happening as $P(t, t'|C)$. The expression is given in the below equation.

$$P(t, t'|C) = \pi . \sum_{t' \in C} P(t', C) . \frac{sim(t, t')}{\sum_{t'' \in C} sim(t', t'')} \tag{4.2}$$

**1. Direct Term Sampling**

This is the standard language modelling term sampling; a term $t$ in the query can be generated from either the document or from the collection following 2.1

**2. Transformation via Document Sampling**

Let $P(t, t'|d)$ denote the probability of generating a term $t'$ from a document $d$ and then transforming this term to $t$ in the query.

$$P(t, t'|d) = P(t|t', d)P(t'|d) \tag{4.3}$$

$$P(t|t', d) = \frac{sim(t, t')}{\sum_{t'' \in d} sim(t, t'')} \tag{4.4}$$

In Equation 4.3, $P(t'|d)$ can be estimated by maximum likelihood with the help of the standard term sampling method as shown in Equation 2.1. For the other part, i.e. transforming $t'$ to $t$, we make use of the cosine similarities between the two embedded vectors corresponding to $t$ and $t'$ respectively. More precisely, the probability that the language model corresponding to document d generates t via the term $t^{'}$ contained in d is taken to be $P(t'|d)$ is proportional to the similarity of $t$ with $t'$. This is shown in Equation 4.4, where $sim(t, t')$ is the cosine similarity between the vector representations of $t$ and $t'$.

Consequently, we can write Equation 4.3 as

$$P(t, t'|d) = \frac{sim(t', t)}{\sum_{t'' \in d} sim(t, t'')} \frac{tf(t', d)}{|d|} \tag{4.5}$$

Words that are used in similar contexts with respect to the query term $t$ over the collection, as given by the vector embeddings, are more likely to contribute to the term score of $t$.

## 3. Transformation via Collection Sampling

Let the probability of event of transforming a term $t'$, sampled from the collection instead of a particular document, to the observed query term $t$ be denoted by $P(t, t'|C)$. This can be estimated as follows.

$$P(t, t'|C) = P(t|t', C).P(t'|C) = P(t|t', C).\frac{cf(t')}{cs} \tag{4.6}$$

Now $P(t|t', C)$ can be estimated in a way similar to computing $P(t|t', d)$, as shown in Equation 4.4. However, instead of considering all $(t, t')$ pairs in the vocabulary for computation, it is reasonable to restrict the computation to a small neighbourhood of terms around the query term t, say $N_t$ because taking too large a neighbourhood may lead to noisy term associations. $cf(t')$ is the collection frequency of the term $t^{'}$ and $cs$ is the collection size i.e it is the total number of terms or words in the entire collection.

This is shown in Equation 4.7.

$$P(t|t', C) = \frac{sim(t, t')}{\sum_{t'' \in N_t} sim(t, t'')} = \frac{sim(t, t')}{\Sigma(N_t)} \tag{4.7}$$

Therefore, 4.6 can be written as

$$P(t, t'|C) = \frac{sim(t, t')}{\Sigma(N_t)}.\frac{cf(t')}{cs} \tag{4.8}$$

### 4.1.2 Combining the Events

Finally, to place all the events together in the LM generation model, let us assume that the probability of observing a query term $t$ without the transformation process (as in standard LM) is $\lambda$. Let $\alpha$ denote the probability of sampling the query term $t$ via a transformation through a term $t'$ sampled from the document $d$, with and let $\beta$ be probability of $t'$ sampled from the collection as shown schematically in the diagram below.

The LM term generation probability can thus be written as shown in Equation 4.9. This is a generalized version of the standard LM, which we henceforth refer to as *generalized language model* (GLM). Note that the GLM degenerates to standard LM by setting $\alpha$ and $\beta$ to zero, i.e. not using the transformation model in the term generation process. The equation 4.9 gives the modified score of the document.

$$\begin{aligned} Score(t, d) &= \lambda P(t|d) + \alpha \sum_{t' \in d} P(t|t', d)P(t'|d) + \beta \sum_{t' \in N_t} P(t|t', C)P(t'|C) + (1 - \lambda - \alpha - \beta)P(t|C) \\ &= \lambda \frac{tf(t, d)}{|d|} + \alpha \sum_{t' \in d} \frac{sim(t, t')}{\Sigma(d)} \frac{tf(t', d)}{|d|} + \beta \sum_{t' \in N_t} \frac{sim(t, t')}{\Sigma(N_t)}.\frac{cf(t')}{cs} + (1 - \lambda - \alpha - \beta)\frac{cf(t)}{cs} \end{aligned} \tag{4.9}$$

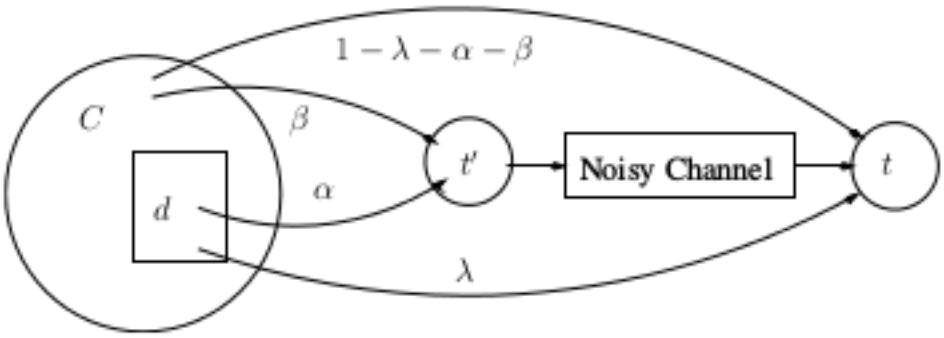


Figure 4.1: Schematics of generating a query term t in our pro- posed Generalized Language Model (GLM). GLM degenerates to LM when $\alpha = 0$ $\beta = 0$.

### 4.1.3 Modifications In The GLM Model

. **Term Transformation Events : via Document And Collection Sampling**

In noisy corpus a term (e.g., 'cat') may have been transformed into another word (e.g., 'oat') because the documen creation process (e.g., ocr) may randomly change a few letters of any word. To address this problem, we considered both the **string similarity** via yass and **contextual similarity** via word embeddings obtained from word2vec or fasttext.

$P(t|t', d)$ is the probability of term t given that the terms $t^{'}$ is seen in document d. Previously we were only considering the contextual similarity but now for computing $P(t|t', d)$ we have also incorporated the string similarity. We have used a linear combination of string similarity and contextual similarity to calculate $P(t|t', d)$ as shown in the equation below.

$$P(t|t', d) = \frac{\gamma Word2VecSim(t, t') + (1-\gamma)YassSim(t, t')}{\sum_{t'' \in d} \gamma Word2VecSim(t, t'') + (1-\gamma)YassSim(t, t'')} \tag{4.10}$$

1. Yass [Yet Another Suffix Stripper] is a stemming algorithm which clusters words using Yass Distance $D_3$ to find their stem

2. Two strings are similar to each other if they have lesser Yass distance .

$$D_3(X, Y) = \frac{n-m+1}{m} * \sum_{i=m}^{n} 2^{i-m} \text{ if m} > 0$$
$$0 \text{ otherwise}$$

3. To convert Yass distance to similarity.

$$YassSim(X,Y) = 1 - \frac{D_3(X,Y)}{len(X) * len(Y)} \tag{4.11}$$

$$P(t|t',C) = \frac{\gamma Word2VecSim(t,t') + (1-\gamma)YassSim(t,t')}{\sum_{t'' \in N_t} \gamma Word2VecSim(t,t'') + (1-\gamma)YassSim(t,t")} \tag{4.12}$$

Similarly $P(t|t',C)$ is defined as above equation, shown in eqaution 4.12.
$P(t,t'|C)$ is a collection sampling event and we have used the yass similarity for (string similarity) and the word2vec similarity for contextual similarity. The denominator is the nearest neighbours union of both yass similarity and word2vec similarity of the words.

### 4.1.4 Implementation Outline

The implementation of the GLM term weighting can be achieved with a two pass indexing. After the first pass stores the LM term weights, the second pass reads every document at a time, reweights the terms and adds new terms by making use of the term embeddings. An efficient approach to get the neighbours of a given term is to store a pre-computed list of nearest neighbours in memory for every word in the vocabulary. After this step, for each document $d$ in the collection, we iterate over term pairs $(t, t')$ and assign a new term-weight to the term $t$ representing the document sampling transformation according to Equation 4.5. Then we iterate again over every term $t$ in $d$ and use $N_t$, pre-computed nearest neighbours of $t$ to compute a score for the collection sampling transformation, as shown in Equation 4.8.

To account for the fact that these transformation probabilities are symmetrical, we add the term $t'$ to $d$. Note that it is not required to add the term $t'$ in case of the document sampling transformation event because $t'$ is already present in $d$.

In lucene retrieval system it is not possible to store term weights easily such that it will take into account the weights while scoring the documents. Therefore it is required to store the term weights separately in the index. We implemented this in term indexing format.

**For each term in the document there is a three tuple field stored. One is the document id , another is the term and third is the is the weight corresponding to $P(t|d)$ , $P(t,t'|d)$ and $P(t,t'|C)$ all normalized by $P(t|C)$ which are separated by pipe .**

**During retrieval process a query term is fetched from the index and each of the documents which are retrieved are score according to the equation**

**4.10 . Each of the score of the individual query terms is then combined for a particular document. Then the documets are sorted according to the combined score for each of the documents and reported in res files**

### 4.1.5 BaseLine Methods For Comparing With GLM

- **LM-JM (None) :**

  In this method, the $\alpha$ and $\beta$ values in GLM are set to 0. This baseline thus corresponds to the original LM approach of Jelinek Mercer smoothing.

- **LM-JM using (3.1) :**

  In this method, all terms are stemmed with the CIKM 2016 Stemmer and then the retrieval is done using LM-JM language model .

- **LM-JM using (2.8) :**

  In this method, all terms are stemmed Porter stemmer and then the retrieval is done using LM-JM language model .

- **LM-JM using (2.9) :**

  In this method, all terms are stemmed using Yass stemmer and then the retrieval is done using LM-JM language model .

### 4.1.6 OverAll FlowChart Of The Entire Process

This diagram 4.2 is the overall pipeline of the entire process flow in the Generalized Language Model FrameWork. After the corpus were stemmed(unstemmed), using the stemmers and then the embeddings are generated which are used to find the nearest neighbours. The nearest neighbours are used to expand the documents. Then the documents are scored according to the doc similarity(document similarity score ) and the nearest neighbour score. Then the retrieval is performed on the Jelinek Mercer Framework. There are various variant tried out in this framework which are explained

### 4.1.7 Various Combination Tried Out With GLM

- **LM-JM (Word2Vec) :**

  In this method for computing $P(t, t'|d)$ only the word2vec similarity is cosidered and not the string similarity from yass. Similary for $P(t, t'|C)$ the documents

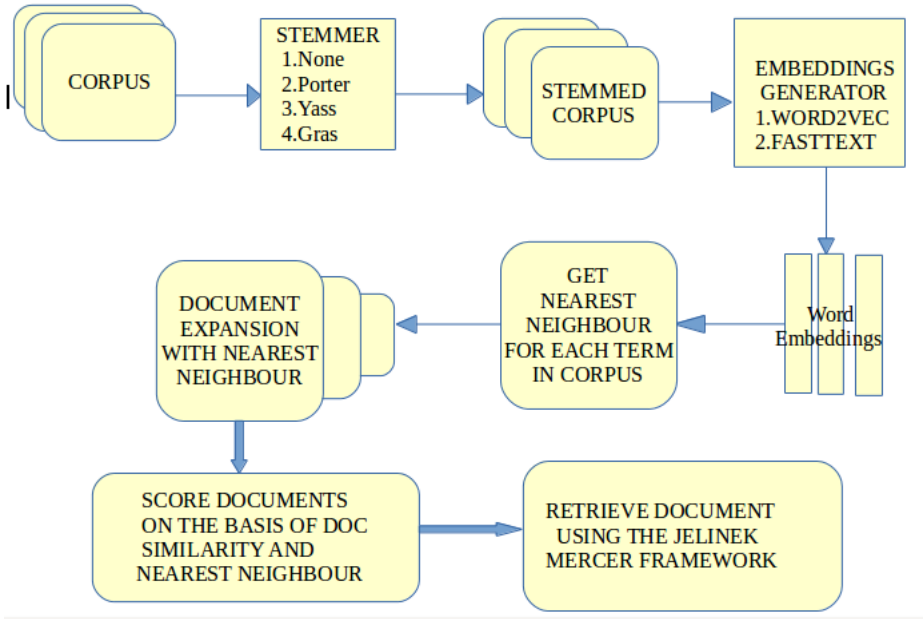


Figure 4.2: Flow Chart Of The Process

are expanded from the word2vec neighbour of each terms.

- **LM-JM (Yass) :**

  In this method for computing $P(t, t'|d)$ only the yass similarity is cosidered and not the semantic similarity from word2vec. Similary for $P(t, t'|C)$ the documents are expanded from the yass neighbour of each terms.

- **LM-JM (Word2vec and Yass) :**

  In this method for computing $P(t, t'|d)$ the yass similarity and word2vec similarity are combined in a linear combination and used . Similary for $P(t, t'|C)$ the documents are expanded from the union of yass and word2vec neighbour of each terms.

- **GLM With Yass Stemmed :**

  Intially the documents were stemmed with yass and nearest neighbiurs are computed for each stemmed word using word2vec. In this method for computing $P(t, t'|d)$ word2vec similarity are computed for two words in the documents.

Similary for $P(t, t'|C)$ the documents are expanded from the neighbour of each terms.

- **GLM With Porter Stemmed :**

  Intially the documents were stemmed with porter and nearest neighbiurs are computed for each stemmed word using word2vec. In this method for computing $P(t, t'|d)$ word2vec similarity are computed for two words in the documents. Similary for $P(t, t'|C)$ the documents are expanded from the neighbour of each terms.

- **GLM With Yass Stemmed And Fasttext :**

  Intially the documents were stemmed with yass and nearest neighbiurs are computed for each stemmed word using Fasttext. In this method for computing $P(t, t'|d)$ Fastext similarity are computed for two words in the documents. Similary for $P(t, t'|C)$ the documents are expanded from the Fasttext neighbour of each terms.

- **GLM With Porter Stemmed And Fasttext :**

  Intially the documents were stemmed with porter and nearest neighbiurs are computed for each stemmed word using Fasttext. In this method for computing $P(t, t'|d)$ Fasttext similarity are computed for two words in the documents. Similary for $P(t, t'|C)$ the documents are expanded from the Fasttext neighbour of each terms.

# Chapter 5

# Proposed Scheme 2

This approach was mainly developed from a failure analysis of the previous method. The nearest neighbours of the words were observed and it was found out that a stemmed corpus might yield better retrieval results than an unstemmed corpus . This is because an unstemmed corpus has lot of variations of a word and hence either the document or query size keeps increasing on adding additional terms in the documents or query respectively.

Another major observation behind coming up with the algorithm is the query terms variants seemed to be more appropriate rather than expanding a documents with every terms in it. So basically a documents also contains many non query words and on expanding those non query words, it was misguiding the retrieval procedure. This was also making the index size huge unnecesarily. There is an eamples of a neighbour in 5.1 of a query word. Now in the document the exact query word may not be present because it was tampered so while quering during retrieval we would never get a match of the root word.

## 5.1 Algorithm

- **Step 1 :**

  Intially the documents and queries are stemmed with a standard stemmer. For this step we tried the following stemmers yass, gras and porter.

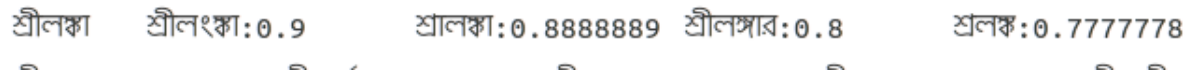


Figure 5.1: Top four neighbour of a query word in Risot Collection

- **Step 2 :**

  The stemmed query words are searched in the collection. If there is no match for the query words, then it means those words are absent either they are misrecognized during the OCR process or they are genuinely not present in the corpus. Thus we obtain the out of vocabulary words from the corpus.

- **Step 3 :**

  Now for each of the query words, top $k$ LCS neighbours or Edit distance neighbours are calculated with respect to the stemmed words in the collection. These neighbours are used to expand the query.

- **Step 4 :**

  Now for each query word, a minimum of $x$ of its LCS or Edit Neighbours are considered for expansion of the query.

- **Step 5 :**

  Each query word is expanded based on the following condition:

  - First it is checked whether the query word is present in the corpus or not. In case the query word is present in the collection and its neighbour's LCS score or the Edit Similarity score is above a certain threshold value $th$ then the query word is expanded. For in vocabulary word $th$ is generally kept high. So it may happen that a query word will not be expanded at all due to the $th$ value.
  - In case it is a out of vocabulary word then it could have been inflected due to noise in the corpus. In such case also we exand the query word if its neighbour's LCS score or the Edit Similarity score is above a certain threshold value $th$. But here we keep a low $th$ value to get a variant of the missing query word.

- **Step 6 :**

  In this step the query terms are given score in the following ways.

  - If the number of expanded terms for a query word is zero and also it's neighbours has LCS score or Edit distance score then it implies it is not a out of vocabulary word and that due to a high $th$ value the query could not be expanded. Thus in such case the query term is given score 1.

- If the number of expanded terms for a query word is greater than zero then in that case the original query word is given the $Qmix$ score(0.1-0.9) or query mix score which is a parameter. The neighbours or the expanded terms are given score as (1-$Qmix$)∗LCS score or (1-$Qmix$)∗edit score normalized by the sum total score of all the expanded terms.
- **3.** If it is a out of vocabulary word then the normalization procedure remains same as the step 2 except that the $Qmix$ score is 0 for the out of vocabulary word.

- **Step 7:**

  Now when the query is ready the documents are retrieved using a retrieval model, in our case it was mainly LM Dirichlet.

### 5.1.1 Methods tried in this model

- In addition to LCS and Edit neighbours we have tried with the neighbours which Fasttext returns. This is for finding out how the query expansion with contextual similarity in our model.
- We also used this model on unstemmed data. The neigbours were unstemmed and query were expanded using unstemmed terms.

**Note :The documents were initally preproceed and removed of all possible noise**

### 5.1.2 Various preprocessing techniques

- The stop words are removed.
- The words are converted to lower case.
- The numbers or digits are removed.
- If any non alphabetic character is present in between a word then those alphanumeric characters are removed and the word is squeezed it.
- In one of the variant we have let the non alphabetic character be present in between the words as it is. This is because it may penalize the stemming procedure due to false match.
- In another variant we have removed those words which have more than 60 percent non alphabetic character in them. This is done because stemming these words would keep it as it is.

Out of the two variants tried out the one with stemmed documents and query expanded by LCS or Edit Neighbours performed best in atleast 4 dataset. There is another dataset IIT-CDIP where we are unable to show the final result after stemming since the dataset is huge and it was taking plenty of computation time.

# Chapter 6

# Performance evaluation and experimental results

### 6.0.3 Results of the baseLine paper as reported in :Combining Local and Global Word Embedding for Microblog Stemming

**The results were reported for Trec 2016 Microblog Data Set(RTS)**

| Algorithm | Precision@20 | MAP | Bpref |
|---|---|---|---|
| Unstemmed | 0.0292 | 0.006 | 0.008 |
| Porter | 0.2114 | 0.0964 | 0.1101 |
| Yass | 0.1077 | 0.0411 | 0.0547 |
| Proposed | **0.2125** | **0.1043** | **0.1162** |

| Algorithm | Precision@20 | MAP | Bpref |
|---|---|---|---|
| Unstemmed | 0.1455 | 0.0870 | 0.1050 |
| Porter | **0.1911** | **0.1011** | **0.1386** |
| Yass | 0.1616 | 0.0923 | 0.1243 |
| Proposed | 0.1437 | 0.0675 | 0.1031 |

**Note:The retrieval model used here is Jelinek Mercer and the queries were title only query.**

The above results were obtained when we implemented the baseline method. To our surprise we found that it is the porter stemmer performing the best which could be because the queries were plain english words.

**Note: The results of the Fire 2016 data set were not reported in the thesis because there were only 7 queries in the data set and also the queries were very vague.**

## 6.1 Results of Generalised Language Model(GLM) and its variants

**The results were reported for Trec 2016 Microblog Data Set(RTS)**

| Method | Stemming | GLM-Neighbours | Precision@20 | MAP | Bpref |
|---|---|---|---|---|---|
| GLM | None | word2vec | 0.0679 | 0.1286 | 0.1171 |
| GLM | None | Yass | 0.1455 | 0.0909 | 0.1226 |
| GLM | None | word2vec and Yass | 0.1616 | 0.0961 | 0.1331 |
| GLM | Yass | word2vec | 0.1545 | 0.0975 | 0.1347 |
| GLM | Porter | word2vec | 0.175 | 0.1001 | 0.1416 |
| GLM | Porter | Fasttext | **0.1821** | **0.1043** | **0.1461** |
| GLM | Yass | Fasttext | 0.1518 | 0.0897 | 0.1232 |

**Intially we were experimenting on unstemmed variant but then since the stand only stemmed version were producing better results , we experimented with stemmed data using the GLM framework.**

**Though our results were comparable to the baseline paper as well as to Porter stemmer , it was not significant improvement over the Porter Stemmer stand alone. Hence we could find that the simplier approaches was performing better than the complicated model like GLM or the Proposed Stemmer in the paper.**

**We were also tried the above methods on RISOT corpus (Bengali) Version , and to our surprise it was not performing well enough to beat raw baseline in baseline paper 2 : *Improving Information Retrieval Performance On OCRed Text in the Absence of Clean Text Ground Truth***

## 6.2 Results of the baseLine paper : Improving Information Retrieval Performance On OCRed Text in the Absence of Clean Text Ground Truth

**The results were reported for RISOT Bengali Version**

| Algorithm | MAP |
|---|---|
| Proposed Method | **0.2231** |
| Original | **0.2567** |

**The results were reported for RISOT Hindi Version**

| Algorithm | MAP |
|---|---|
| SIGIR 2015 | **0.1685** |
| Proposed Method | 0.1672 |
| Original | **0.2551** |

**The results were reported for IIT CDIP Data Set**

| Algorithm | MAP | Recall@1000 |
|---|---|---|
| Proposed Method | **0.1011** | **0.4056** |

**The results were reported for Confusion Track at 5 percent Degradation**

| Algorithm | MAP |
|---|---|
| Original | **0.7653** |
| Proposed Method | **0.6446** |

**The results were reported for Confusion Track at 20 percent Degradation**

| Algorithm | MAP |
|---|---|
| Original | **0.7653** |
| Proposed Method | **0.4619** |

## 6.3 Results of the prosposed method 2 on the above data set

**Though different Query Expansion (QE) neighbours were tried out like edit , lcs and fasttest but the results which performed the best were reported only.**
**Note : The retrieval model used was LM-Dirichlet**

**The results were reported for RISOT Bengali Version .**

| Algorithm | QE-Neigbours | MAP |
|---|---|---|
| Proposed Method | Edit-Neighbour | **.2459** |

**These results were obtained at $\mu = 900$ , which is a dirichle parameter , at threshold 1 = .75 and threshold 2 = .80 , k=5 where k is the number of expansion terms and Qmix=0.5**

**The results were reported for RISOT Hindi Version .**

| Algorithm | QE-Neigbours | MAP |
|---|---|---|
| Proposed Method | Edit-Neighbour | **.1742** |

**These results were obtained at $\mu = 300$ , which is a dirichle parameter , at threshold 1 = .71 and threshold 2 = .85 , k=5 where k is the number of expansion terms and Qmix=0.5**

**The results were reported for IIT-CDIP . This is a huge collection and the process of stemming was taking lot of time so for now we have reported the results on the same model using unstemmed collection.**

| Algorithm | MAP | Recall@1000 |
|---|---|---|
| Proposed Method | **0.0860** | **.3584** |

**These results were obtained at $\mu = 900$ , which is a dirichle parameter , at threshold 1 = .75 and threshold 2 = .80 , k=5 where k is the number of expansion terms and Qmix=0.5 .**

**The results were reported for Confusion Track at 5 percent Degradation**

| Algorithm | QE-Neigbours | MAP |
|---|---|---|
| Proposed Method | Lcs-Neighbour | **.7503** |

**These results were obtained at $\mu = 1200$ , which is a dirichlet parameter , at threshold 1 = .77 and threshold 2 = .85 , k=5 where k is the number of expansion terms and Qmix=0.5 .**

**The results were reported for Confusion Track at 20 percent Degradation**

| Algorithm | QE-Neigbours | MAP |
|---|---|---|
| Proposed Method | edit-Neighbour | **.3387** |

**These results were obtained at $\mu = 2100$ , which is a dirichle parameter , at threshold 1 = .77 and threshold 2 = .85 , k=3 where k is the number of expansion terms and Qmix=0.5 .**

**The results were pretty bad for this data set for stemmed collection. Therefore we thought of delving deeper into why it is under performing, and we tried the variant into unstemmed corpus.**

| Algorithm | QE-Neigbours | MAP |
|---|---|---|
| Proposed Method(Unstemmed) | edit-Neighbour | **.4364** |

**These results were obtained at $\mu = 4000$ , which is a dirichle parameter , at threshold 1 = .77 and threshold 2 = .87 , k=5 where k is the number of expansion terms and Qmix=0.5 .**

**Still the underline baseline could not be beaten . Since the unstemmed data set was giving a better result we tried with 60 percent pruning method. In this method we have kept only those words which has 60 percent alphabets , otherwise we have ignored the words. Then we have stemmed it with gras and fitted it in the proposed model.**

| Algorithm | QE-Neigbours | MAP |
|---|---|---|
| Proposed Method | edit-Neighbour | **.4761** |

**These results were obtained at $\mu = 2100$ , which is a dirichlet parameter ,**

**at threshold 1 = .75 and threshold 2 = .85 , k=2 where k is the number of expansion terms and Qmix=0.7 .**

Using this method we are able to solve the Confusion Track 20 percent degradation data set problem.

# Chapter 7

# Future work and conclusion

- While computing the lcs and edit neighbours many a times we have obtained similar score for various words. How to break a tie between neighbours which share similar score is a challenge. For now we have thought about a method of breaking the tie. After we obtain the top 30 lcs or edit neighbours, we can rearrange the ones with the same score with their context similarity obtained from word2vec or fasttext.
- Stemmers like Yass and Gras are unable to handle large data like IIT CDIP (44GB). So we would like to explore ways to make the stemmer code more efficient for quick computation.

We conclude that so far we are able to improve the retrieval efficiency to certain extent but may be not be across all data set mostly due computation inefficieny in large data set like IIT-CDIP.

## Biblography


1. Moumita Basu, Anurag Roy, Kripabandhu Ghosh, Somprakash Bandyopadhyay, and Saptarshi Ghosh. 2017. A Novel Word Embedding Based Stemming Approach for Microblog Retrieval During Disasters. In Proc. ECIR.

2. D. Ganguly, D. Roy, M. Mitra, and G. J. Jones. Word embedding based generalized language model for information retrieval. In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval.

3. A.Roy , T.Ghosh , K.Ghosh ,S.Ghosh .2017 Combining Local and Global Word Embeddings for Microblog Stemming . In CIKM .

4. Ghosh , Chakraborty , Parui Improving Information Retrieval Performance On Ocred Text in the Absence Of Clean Text Ground Truth.

5. http://library.isical.ac.in:8080/jspui/bitstream/123456789/6294/1/DIS S-124.PDF

6. http://library.isical.ac.in:8080/jspui/bitstream/123456789/6453/1/DISS-296.pdf

7. http://library.isical.ac.in:8080/jspui/bitstream/123456789/6403/1/Diss-246.PDF

8. https://www.wikipedia.org/