Implementing various tokenization strategies for Biomedical Information Retrieval

Mallipeddi Akshay,
201301216@daiict.ac.in
DA-IICT

***Abstract*— The main challenge involved when dealing with biological names in biomedical text, is appropriate tokenization for biomedical information retrieval. In this report I will briefly explain various tokenizing strategies. In this work, I have conducted a systematic evaluation of different methods on the available TREC(2014) biomedical text collections for ad-hoc document retrieval.The main idea is that different types of text require different tokenization methods.**

I. INTRODUCTION

The unusual usage in biomedical text, such as constant occurrences of gene symbols and their lexical variants,raise many challenges in biomedical information retrieval.The main purpose of tokenization is to break down the text into terms, that are small units of meaningful text,such that a match between a term in the query and a term in a document increases our confidence that the document is relevant to that query.Let's see at the challenges involved while handling biomedical text.The following are the challenges involved:-

- Frequent occurrences of gene symbols on the biomedical text.
- Use of inconsistent lexical variants of same gene symbols.For example:
  1. MIP-1-alpha,
  2.MIP-1 alpha,
  3.(MIP)-1 alpha and
  4.MIP-1alpha.
- The text also contains various names involving genes,proteins and chemicals.

Tokenization could be done in many ways, one simple way would be to consider whitespace as delimiter, or using non-alphanumeric characters as delimiters. In biomedical text, however, the content words include not only English words, but also many special terms such as the names of genes, proteins and chemicals. We now realize that a simple tokenizer doesn't help us in tokenizing the text given.

Let us look at an example, where we demonstrate two tokenizers which can only identify some variants individually.[Table-1]

We see in Table-1 that first tokenizer could identify Variant-3 but couldn't identify the first two variants, whereas the second tokenizer could identify first two variants and couldn't identify Variant-3.There is no ideal tokenizer that could identify all kinds of variants in biomedical text.

II.THE DATA SET USED

The data used in the evaluation is taken from CDS(Clinical Decision Support)that was introduced in TREC 2014.The data contains 7,33,138 documents and 30 topics.Each topic consists a case report and one of the three clinical question types ('diagnosis','treatment','test').

TABLE 1:-Effect of two tokenizers on lexical variants of MIP-1-alpha

Variant	**Original text**	**Tokenized text**			
		Tokenizer 1	**Match?**	**Tokenizer 2**	**Match?**
Query	MIP-1-alpha	mip 1 alpha	N/A	mip1alpha	N/A
Variant-1	MIP-1alpha	mip 1alpha	No	mip1alpha	Yes
Variant-2	(MIP)-1alpha	mip 1alpha	No	mip1alpha	Yes
Variant-3	MIP-1 alpha	mip 1 alpha	Yes	mip1 alpha	No

III.TERRIER

Terrier , **Terabyte Retriever**, is a project that was initiated at the University of Glasgow,for development of Information Retrieval applications.Like any other search engine Terrier also provides:

- Indexing:-Terrier can efficiently index large corpora of documents, and of documents, and provides multiple indexing strategies.Real-time indexing of document streams are also supported by index structures.[Figure-1]

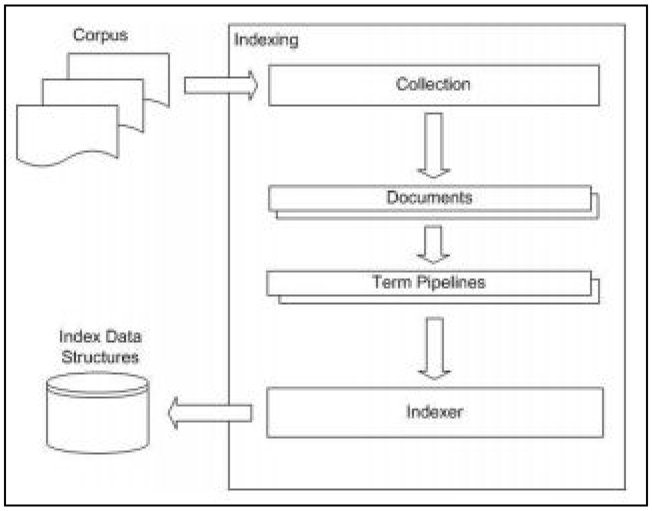
- Retrieval:-Uses various models for supervised ranking via built-in plugins, which help in retrieval of documents.Some of the models used are Divergence from Randomness, BM25F,term dependency proximity models etc.

Terrier index consists of 4 main data structures:-

- Lexicon
- Inverted Index
- Document Index
- Direct Index

Terrier also provides many predefined weighting models(BM25, In expC2,TF-IDF,BB2 etc).I have used TF-IDF model.

Figure-1:-: This figure shows the indexing architecture of Terrier.A corpus of documents is controlled by a plugin,which produces a stream of document objects.Each document in turn generates a stream of terms,which are converted by a series of pipeline components, after which Indexer writes to disk.



IV.TOKENIZATION HEURISTICS

We define the following heuristics rules to remove non-functional characters:-

- Replace the following characters with spaces: ! ” # $ % & * < = > ? @ \ | ~
- Remove the following characters if they are followed by a space: . : ; ,
- Remove the following pairs of brackets if the open bracket is preceded by a space and the close v=bracket is followed by a space:( ) [ ]
- Remove the single quotation mark if it is preceded by a space or if it is followed by a space: ‘
- Remove ‘s and ‘t if they are followed by a space.
- Remove slash / if it is followed by a space.

The third rule defined above removes brackets when they are not inside geen names, because when brackets occur inside gene names, either the open bracket or close bracket is inside the text, such as in (MIP-1-alpha).

An example demonstrating the above rules:

TABLE-2: An example showing the application of above mentioned heuristic rules

Original Text	Modified Text
(HbA(1c) >=6% or GO >=1.26 g/L),	HbA(1c) 6 or GO 1.26 g/L
(‘ppm’)	ppm

So we see that even after applying these rules to the text,the following characters may still exist : (,), [,], -, /,., :, ;, , , ‘, and +.

So, now we define a two of special characters that left out after applying the heuristic rules.

TABLE-3: Two set of special characters that are left after applying the rules

Special character set 1	Special character set 2
( ) [ ] - /	. : ; , ’ +

IV.BREAK POINTS

Besides the characters shown in Table-3, there can also other places where we can further breakdown the text.These places are where an alphabetical character changes to number of changes its case(Upper/Lower).Let’s define these places as “hidden places”.The following are the rules where we can further break the text:

- Between an alphabetical character on the left(right) and a number on the right(left).As in, between report and 123 in report123.
- A place where there is change in the case.As in, HEL and lo in HELlo
- Between an upper case letter on the left and lower case letter on the right, unless the upper case is preceded by space.As in ,INORM and ation in INFORMation.

Now we define three set of break points.Break points are places where a name can be broken down into further smaller components, such that after connecting those smaller units we get the original name.So,these are following three set of break points:

- **Break Point Set 1(BP1):-** Consists of special characters set -1 shown in table-3
- **Break Point Set 2(BP2):-** Consists of both set of special characters shown in table-3.
- **Break Point Set 3(BP3) :-**Consists of all special characters in BP2 and all hidden breakpoints defined by the 3 rules.

V.BREAK POINT NORMALIZATION

Now we can normalize the different lexical variants of same entity by normalizing break points into same representation. We can replace the break points by hyphen, and identify the breakpoints by hyphen in the modified text.Thus,MIP 1 alpha would convert to MIP-1-alpha.

Another to normalize is to replace all the breakpoints with space. For example, MIP-1-alpha,MIP-1alpha will become three tokens:MIP,1,alpha.

Method one replaces break points with hyphens, we call this method as Hyphen-Normalization method, or H-Norm.Method two replaces break points with spaces, we call this method as Space-Normalization method, or S-Norm.Method three removes all breakpoints(or meaning it doesn’t do anything). we call this method as Join-Normalization method, or J-Norm.

VI.STEMMING AND STOPWORD REMOVAL

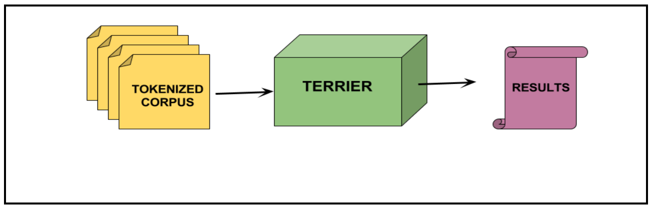
We now consider three stemmers:-

- **Porter stemmer:**An algorithm for suffix stripping
- **Lovins stemmer:**Removes longest possible string of characters from a word(using external stopword list, we use stop word list from PubMed)
- **S stemmer:** Removes the common word endings.

VII.APPROACH

The data set which was mentioned above (TREC 2014 CDS) will first be tokenized into various sets based on the method used. Then the tokenized data will be sent to terrier for indexing(as we saw in Figure-1,as corpus) and retrieval and the final results are obtained.

FIGURE-2:-Showing the approach followed



VIII.EVALUATION

There are a total of 30 topics in TREC 2014 CDS dataset. The topics in TREC 2014 are medical case narratives created by expert topic developers.The information may describe patient's medical history, the patient's current symptoms, tests performed by a physician to diagnose the patient's condition ,the patient's eventual diagnosis etc. The topics are divided as shown in table-4.

TABLE-4 :-Table showing TREC 2014 topics descriptions.

Type	Generic Clinical Question	Number of Topics
Diagnosis	What is the patient's diagnosis?	10
Test	What tests should the patient receive?	10
Treatment	How should the patient be treated?	10

- Format of the topics:-

```
<topics>
 <topic number="1" type="diagnosis">
  <description>Description of topic 1</description>
  <summary>Summary of topic 1</summary>
 </topic>
 ...
</topics>
```

➔ **First trial run:-**In the first run,I considered a sample data from the TREC 2014 CDS data,which contains approximately 50,000 documents and 30 topics.The results are as follows:-

TABLE-5:-Showing the results of first trial run

		Normalization methods		
Stemmer used		**H-NORM**	**S-NORM**	**J-NORM**
Porter	MAP	0.0085	0.0086	0.0091
Lovins	MAP	0.0065	0.0062	0.0059
S	MAP	0.0090	0.0090	0.0093

The baseline value used was **MAP:** 0.0075 (Default Tokenization)

The results in the first trial run was very bad and no conclusion could be drawn on the methods used because even the data set used was very limited.Based on the results in table-5 , S stemmer performed better, as it gave MAP more than the baseline value.Porter stemmer also performed better while Lovins stemmer was below the baseline.

➔ **Second final run:-** In the second run, I considered the whole TREC 2014 CDS data which contains approximately 7,50,000 documents and 30 topics.The results are as follows:-

TABLE-6:-Showing the results of second final run

		Normalization methods		
Stemmer used		**H-NORM**	**S-NORM**	**J-NORM**
Porter	MAP	0.0953	0.0954	0.0940
Lovins	MAP	0.0673	0.0632	0.0613
S	MAP	0.1001	0.1000	0.0976

The baseline values used was **MAP:** 0.0954 (Default Tokenization)

The results in the final run were better than the first run and S stemmer performed better, as it gave MAP more than the baseline value.And Porter stemmer performed equally well with the baseline, while Lovins stemmer was below the baseline.

IX.CONCLUSION

Because of the irregular variants of names in biomedical text,tokenization is an important preprocessing step in biomedical information retrieval.In the report, we saw the tokenization strategies including a non-functional character removal step,a break point normaliztion step with possible normalization mwthods and possible break pioints.The evaluation was conducted on available TREC biomedical information retrieval test collections.Results show that tokenizations could improve the performance.

REFERENCES

[1] http://terrierteam.dcs.gla.ac.uk/publications/ounis06terrier-osir.pdf

[2] [Lovins 1968] Lovins, J. (1968). Development of a stemming algorithm. Mechanical Translation and Computational Linguistics.

[3] [Porter 1980] Porter, M. F. (1997). An algorithm for suffix stripping. Program, 14(3).

[4] [Zhai 2001a] Zhai, C. (2001). Notes on the Lemur TFIDF model. http://www.cs.cmu.edu/ lemur/1.1/tfidf.ps.

[5] http://sifaka.cs.uiuc.edu/czhai/pub/ir-tok.pdf