An Empirical Study of Tokenization Strategies for Biomedical Information Retrieval

Jing Jiang ChengXiang Zhai [*]

Abstract

Due to the great variation of biological names in biomedical text, appropriate tokenization is an important preprocessing step for biomedical information retrieval. Despite its importance, there has been little study on the evaluation of various tokenization strategies for biomedical text. In this work, we conducted a careful, systematic evaluation of a set of tokenization heuristics on all the available TREC biomedical text collections for ad hoc document retrieval, using two representative retrieval methods and a pseudo relevance feedback method. We also studied the effect of stemming and stop word removal on the retrieval performance. As expected, our experiment results show that tokenization can significantly affect the retrieval accuracy; appropriate tokenization can improve the performance by up to 96%, measured by mean average precision (MAP). In particular, it is shown that different query types require different tokenization heuristics, stemming is effective only for certain queries, and stop word removal in general does not improve the retrieval performance on biomedical text.

Keywords: biomedical information retrieval, tokenization, stemming, stop word.

[*]Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N Goodwin Ave, Urbana, IL 61801. Email: {jiang4, czhai}@cs.uiuc.edu

1 Introduction

Recently, the growing amount of scientific literature in genomics and related biomedical disciplines has led to an increasing amount of interest in and need for applying information retrieval as well as other text management techniques to access the biomedical textual data. The special language usage in biomedical literature, such as the frequent occurrences of gene symbols and the use of inconsistent lexical variants of the same genes, has raised many new challenges in the field of biomedical information retrieval.

In previous work on biomedical information retrieval, while many efforts have been put into query expansion and synonym normalization, little attention has been paid to tokenization and other text preprocessing steps that transform the documents and the queries into the bag-of-word representation. Although tokenization is not a critical step for retrieval in English text in general domains, it is not a trivial task for languages in special domains (sometimes referred to as sublanguages), largely due to the domain-specific terminologies.

Information retrieval methods generally rely on term matching. The purpose of tokenization is therefore to break down the text into *tokens* (or *terms*), which are small units of meaningful text, such that a match between a token in the query and a token in a document can in general increase our confidence that the document is relevant to the query. It is often also desirable during the preprocessing stage to normalize tokens that look similar and convey the same meaning into a single canonical form. By normalizing terms with slight differences into canonical forms, we can improve the recall of a retrieval system, although we may risk some decrease in precision.

For English text in general domains, individual English words are naturally used as tokens. Tokenization can be done by simply using white spaces as delimiters, or in a slightly more sophisticated way, by using all non-alphanumerical characters as delimiters. Stemming is often used to normalize the morphological variants of the same base word. 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. These names often contain special

Table 1: An example of the effect of two tokenization strategies for matching the query "MIP-1-alpha" with three variants and a mismatch

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
Mismatch	*MIP-1 beta*, *IFN-alpha*	mip 1 beta ifn alpha	Yes	mip1beta ifnalpha	No

characters such as numerals, hyphens, slashes and brackets, and the same entity often has different lexical variants. Clearly, a simple tokenizer for general English text cannot work well in biomedical text. If all non-alphanumerical characters inside a named entity are used as delimiters to separate the name into several tokens, the proximity of these tokens is lost in the bag-of-word representation, which may result in a loss of the semantic meaning of the tokens and cause mismatches. Moreover, breaking named entities into fragments may affect the tf.idf weighting of the tokens in an unwanted way. On the other hand, if all the non-alphanumerical characters are kept, it is hard to capture minor variations of the same name.

Table 1 shows an example taken from Topic 38 of the ad hoc retrieval task of the TREC 2003 Genomics Track, where two tokenizers produce different matching results. The original query contains the gene symbol *MIP-1-alpha*. There are three lexical variants of this symbol that appear in the judged relevant documents, as shown in the "Original Text" column of three rows, "Variant 1", "Variant 2", and "Variant 3", in Table 1. The bottom row shows a piece of text that can possibly become a mismatch. If no special tokenization strategy is used, none of the three variants will match with the query. Now we consider two special tokenization strategies: Tokenizer 1 uses all non-alphanumerical characters as token delimiters; Tokenizer 2 removes special characters such as hyphens and brackets. The "Tokenized Text" column shows the tokenized text. We can see that Tokenizer 1 captures one lexical variant, but misses the other two variants and generates one mismatch. Tokenizer 2, however, captures two variants and ignores the mismatch. Thus Tokenizer 2 is superior to Tokenizer 1 in this particular example. Because many queries in biomedical information retrieval contain gene names and symbols like the one in the above example, it is important to find a suitable tokenization strategy in order to generate the best retrieval results.

Despite its importance, to the best of our knowledge, there has not been any work devoted to a systematic comparison of different tokenization strategies for biomedical information retrieval. In this paper, we present a set of tokenization heuristics that are generalized from previous work on biomedical information retrieval, and we conduct a systematic evaluation of these heuristics. In particular, we define three sets of *break points*, three break point normalization methods, and a Greek alphabet normalization method. We also study the effects of stemming and of stop word removal for biomedical text as these preprocessing steps may also affect the retrieval performance. The goal of our study is to provide a set of standard tokenization heuristics that are in general suitable for biomedical information retrieval. With such standard heuristics, we can presumably have a strong baseline method for biomedical information retrieval, on which more advanced techniques can be applied, and against which any new method should be compared.

We evaluate the tokenization heuristics on the data from the TREC 2003, TREC 2004 and TREC 2005 Genomics Track. Thus, the data we use includes all available TREC biomedical information retrieval test collections for ad hoc document retrieval. We use two representative retrieval methods. We also apply a pseudo relevance feedback method on top of one of the retrieval methods we use. Results from both retrieval methods and the pseudo relevance feedback method show that tokenization strategies can affect the retrieval performance significantly; good tokenization can improve the performance by up to 96%. For different types of queries, different sets of tokenization heuristics are preferred in order to achieve the optimal performance. In particular, for queries with only gene symbols, removing a set of special characters and replacing Greek letters with Latin letters are shown to be effective. In contrast, for queries with only full gene names and for verbose queries that also contain English words to describe the information need, replacing special characters with spaces produces the best results. In addition, for verbose queries, stemming further improves the performance. Stop word removal does not help retrieval in general.

The rest of the paper is organized as follows: In Section 2, we survey the tokenization strategies explored in previous work on biomedical information retrieval. In Section 3 and Section 4, we generalize the various tokenization strategies into a set of heuristics, and explain the rationale behind each heuristic. We evaluate the different tokenization heuristics in Section 5. In Section 6, we discuss the scope and limitation of our study, and conclude our work by recommending a set of suitable tokenization heuristics based on the query types.

2 Related Work

Most previous work on biomedical information retrieval appeared in the Genomics Track in TREC 2003, TREC 2004 and TREC 2005. To address the prevalent name variation problem in biomedical text, most groups focused on query expansion, either by using external knowledge bases such as LocusLink to find gene synonyms [Buttcher et al. 2004, Fujita 2004], or by generating lexical variants of the gene names in the queries using heuristic rules [Buttcher et al. 2004, Huang et al. 2005]. Tokenization of the corpus was not seriously studied, and most groups did not describe their tokenization strategies in detail.

Among the work that mentioned special tokenization techniques, Tomlinson [2003] pointed out that allowing a token to contain both alphabetical and numerical characters was a little better than separating them. In contrast, Pirkola and Leppanen [2003] and Crangle et al. [2004] chose to separate alphabetical and numerical strings in order to handle hyphenation of the alphabetical and the numerical parts in a gene name. But Pirkola and Leppanen [2003] also imposed proximity search to ensure that the separated components were close together in the retrieved documents. Dayanik [2003] allowed the following special characters to be part of a token provided that they were not the first or the last character of the token: "(", ")", "[", "]", "'", "-", "," and "/". Thus names such as *1,25-dihydroxyvitamin* and *dead/h* would be considered single tokens.

Some work also considered combining adjacent words that were separated by spaces into single terms. Song et al. [2003] used a simple rule to combine a short-length word and its adjacent non-short-length word into a single keyword. They reported a 15% improvement over the baseline using this simple heuristic. Ando et al. [2005] combined adjacent alphabetical chunks and numerical chunks into token bigrams.

Existing work has also explored different ways to tokenize the queries and to generate alternative gene names using heuristic rules. Huang et al. [2005] defined *break-points* in tokens where hyphens and spaces can be inserted or removed without changing the meaning of the token. Buttcher et al. [2004] and Huang et al. [2005] also considered replacing Greek letters with their Latin equivalents to generate lexical variants of gene names in the queries.

Although the Porter stemmer [Porter 1980] was the most commonly used English stemmer, a few groups reported experiments with different stemmers on biomedical text. Song et al. [2003] showed that the Porter stemmer could decrease the performance while the Lovins stemmer [Lovins 1968] improved the performance. Savoy et al. [2003] showed that the S stemmer [Harman 1991] sometimes was advantageous over the Lovins stemmer. Many groups also removed stop words from the document collection using an external stop word list [Song et al. 2003, Carpenter 2004, Fujita 2004]. A commonly used stop word list for biomedical text is the one from PubMed [Carpenter 2004]. There has not been any work comparing the retrieval performance with and without stop word removal.

3 Tokenization Heuristics

In this section, we generalize the various tokenization strategies in previous work into a set of organized heuristics. Such generalization allows us to better understand the rationale behind each strategy, and to systematically evaluate the strategies.

Table 2: Heuristic rules to remove non-functional characters

1) replace the following characters with spaces: ! ” # $ % & $*$ $<$ $=$ $>$? @ \ | ~
2) remove the following characters if they are followed by a space: . : ; ,
3) remove the following pairs of brackets if the open bracket is preceded by a space and the close bracket is followed by a space: ( ) [ ]
4) remove the single quotation mark if it is preceded by a space or if it is followed by a space: ′
5) remove ′s and ′t if they are followed by a space
6) remove slash / if it is followed by a space

For all the tokenization strategies we consider, we assume that the last step of tokenization is to change all upper case letters into lower cases. This case normalization is done in the very end because some tokenization heuristics rely on the cases of letters to determine where to break the text. Also, the tokenization strategies that we consider do not include the ones that combine adjacent words originally separated by spaces into bigram tokens.

Before we go into the various tokenization heuristics, we first define a *naive method* as follows: The naive method uses only white space characters as delimiters to break down the text into tokens. We use the naive method as the very basic baseline to compare against.

3.1 Removal of Non-Functional Characters

In biomedical text, although non-alphanumerical characters are frequently used to help represent various kinds of entities and other biomedical information, the most important special characters that make retrieval difficult are those that frequently occur in gene names and protein names, such as hyphens, slashes, and brackets. Many other non-alphanumerical characters such as ‘=’ and ‘#’ usually do not occur inside an entity name or convey any important semantic meaning. These “non-functional” characters can thus be excluded from the tokens. Previous work that had special handling of non-alphanumerical characters also only focused on a subset of non-alphanumerical characters. Therefore, as a first step, we manually identified a set of special characters that we believe can be safely discarded in certain context, and we defined a set of rules in the form of regular expressions to remove these special characters. Table 2 lists the heuristic rules we defined to remove such non-functional characters. The first rule presumably removes characters that we believe rarely occur in

Table 3: Example texts before and after removal of non-functional characters

Original Text	**Modified Text**	**Rule(s) Applied**
(HbA(1c) >=6% or GO >=1.26 g/L),	HbA(1c) 6 or GO 1.26 g/L	(1), (2), (3)
(′ppm′)	ppm	(3), (4)
(Langerhan′s cells)	Langerhan cells	(3), (5)

Table 4: The counts of all non-alphanumerical characters in 4000 randomly chosen gene names and symbols

Character	**Count**	**Character**	**Count**	**Character**	**Count**
-	1042	_	90	;	9
,	396	.	88	[	3
:	230	/	65	]	3
)	209	+	18	@	2
(	208	′	16	*	1

gene names or symbols. The second rule presumably removes periods, colons, semi-colons and commas when they are indeed used as punctuation marks rather than some meaningful characters inside gene names or symbols. Similarly, the third rule presumably removes round brackets and square brackets when they are not inside gene names, because usually when brackets occur inside gene names, either the open bracket or the close bracket has no adjacent space, such as in *(MIP)-1alpha*. The fourth and the fifth rules deal with single quotation marks and apostrophes. And the last rule deals with slashes when they are not inside gene names or symbols. Table 3 shows some example texts before and after the non-functional characters are removed.

After applying these heuristic rules to raw text, the following non-alphanumerical characters may still occur in the modified text: “(”, “)”, “[”, “]”, “-”, “_”, “/”, “.”, “:”, “;”, “,”, “′”, and “+”. To see whether indeed these are the non-alphanumerical characters that may occur inside gene names and symbols, we randomly chose 4000 gene names and symbols from LocusLink, and counted all the non-alphanumerical characters in these names. Table 4 shows the counts. Comparing the table with the list of characters above, we can conclude that (1) the special characters we have removed indeed do not occur in gene names and symbols often, and (2) the special characters that remain in the text are all possible to occur in gene names and symbols. Exactly how much the retrieval performance is affected by

Table 5: Two sets of special characters that are left in the text after removal of the non-functional characters

special character set 1	**special character set 2**
( ) [ ] - _ /	. : ; , ' +

removing these non-functional characters is an empirical question and will be examined in our experiments.

3.2 Break Points

After the non-functional characters are removed, the remaining text consists of alphanumerical characters and the special characters listed in Table 5. We divided these special characters into two sets. The special characters in set 1 are often used to separate the several components of an entity name, such as in *(MIP)-1alpha*, *pRB/p105*, and *TrpEb_1*. The special characters in set 2 are not very frequently used for gene names, but rather mostly used inside numbers like *0.20* and *20,000*, inside chemical formulas like the ions *Ca2+* and *Na+*, or inside names of chemical compounds and DNA sequences to describe the structure of those entities, like in *1,2,3,4-TeCDD*, *2,3,7,8-TeCDD* and *2',5'-linked 3'-deoxyribonucleotides*. In most of these cases, it is not necessary to divide the two alphanumerical strings around those special characters in set 2 into different components.

Besides these special characters listed in Table 5, there are also other places within some strings where we should consider breaking the strings into smaller tokens. These are the places where an alphabetical character changes to a numerical character and vice versa, or where a sequence of upper case letters changes to a sequence of lower case letters and vice versa. Formally, following the work in [Huang et al. 2005], we use the following rules to define three kinds of "hidden places" where strings can be further broken down:

1. Places between an alphabetical character on the left(right) and a numerical character on the right(left). For example, between *p* and *105* in *p105*.

2. Places between a lower case letter on the left and an upper case letter on the right.

For example, between *Trp* and *Eb_1* in *TrpEb_1*.

3. Places between an upper case letter on the left and a lower case letter on the right, unless the upper case letter is preceded by a space, or by a numerical character or another lower case letter (in which case the upper case letter will be separated from its previous lower case letter by rule (2)). For example, between *MIP* and *alpha* in *MIPalpha*, but not between *E* and *b* in *TrpEb_1*.

We regard these places as "hidden" because these places are not marked by a single special character, but by the characters to the left and to the right of these places. With these rules, gene symbols such as *MIP-1alpha* and *MIP-1-alpha* can both be broken down into *MIP 1 alpha*, making it possible to match them.

Borrowing the term from [Huang et al. 2005], we refer to all the special characters listed in Table 5 and the three kinds of hidden places defined above as *break points*. The break points are the places where an entity name can be potentially broken down into smaller components such that if we connect these smaller components differently, we could form a lexical variant of the original name. Because of the different degrees to which we believe these break points should be used, we consider three sets of break points.

- **Break Point Set 1 (BP1)** consists of the special characters in special character set 1 in Table 5.
- **Break Point Set 2 (BP2)** consists of both special character set 1 and special character set 2 in Table 5.
- **Break Point Set 3 (BP3)** consists of all special characters in BP2 and the hidden break points defined by the three rules.

3.3 Break Point Normalization

With the break points we have defined in the last section, we can now normalize the different lexical variants of the same entity by normalizing the break points into the same representation. There are different ways to normalize the break points. One way is to replace all the break points with a single special character, such as a hyphen, and keep these break points designated by hyphens in the final tokens. Thus, if we use BP3, gene symbols *MIP-1-alpha*, *MIP-1alpha* and *(MIP)-1alpha* will all become the same single token *MIP-1-alpha.* Another way to normalize the break points is to replace all of them with spaces. Since spaces are used as token delimiters in the very end, replacing the break points with spaces is the same as splitting the text into tokens by these break points. For example, when BP3 is used, gene symbols *MIP-1-alpha*, *MIP-1alpha* and *(MIP)-1alpha* will all become three tokens: *MIP*, *1*, and *alpha.* There are advantages and disadvantages of both normalization methods. For the first one, the proximity of the components of a gene name is preserved, ensuring high precision in matching entity names. However, it could not handle the case when one lexical variant contains a space while another lexical variant has a break point in the place of the space, such as in *MIP-1 alpha* and *MIP-1-alpha.* The second normalization method can handle this case well because all break points are replaced with spaces. However, proximity of the components of the name is lost, which may cause mismatches.

There is another problem with the two normalization methods described above. Sometimes a hidden break point cannot be captured by the three rules we defined in Section 3.2. For example, the topics in TREC 2003 Genomics Track contain these gene alias symbols: *Pkca* and *Prkca* (for the gene "*Protein kinase C, alpha*"), *Tcra* and *Tcralpha* (for the gene "*T-cell receptor alpha*"), and *Ifnb2* (for the gene "*Interleukin 6 (interferon, beta 2)*"). We can see that the Greek letters *alpha* and *beta*, or their Latin equivalents *a* and *b*, cannot be clearly distinguished from the rest of the text in these symbols. Thus if a hyphen is inserted into such a hidden break point, this hyphenated variant cannot be matched with the one without the hyphen by either of the normalization methods we described above. Because such hidden break points are too hard to detect by any simple regular expressions, one way to solve the problem is to normalize all variants into the form without hyphens or any other special characters. We have not seen such an approach in any previous work, but we think this approach is a reasonable solution to the problem with undetectable break points.

To summarize, we consider three methods to normalize the break points. Method one replaces all break points with hyphens (or inserts hyphens into hidden break points). We call this method the *Hyphen-Normalization* method, or *H-Norm*. Method two replaces all break points with spaces (or inserts spaces into hidden break points). We call this method the *Space-Normalization* method, or *S-Norm*. Method three removes all break points (or does nothing to hidden break points). We call this method the *Join-Normalization* method, or *J-Norm*. After normalization, we can then simply use the white space characters to split the text into tokens. Note that all three break point normalization methods can be used in conjunction with any of the three sets of break points, except that when J-Norm is used, BP3 becomes essentially the same as BP2.

3.4 Greek Alphabet Normalization

Another heuristic that has been previously explored is to replace Greek letters with their Latin equivalents. In biomedical text, entity names often contain Greek letters such as *alpha*, *beta*, etc. Sometimes these Greek letters are abbreviated as *a*, *b*, etc., but there are no consistent rules as to when the Greek letters should be abbreviated. A simple method to tackle this problem is to replace all occurrences of Greek letters with the Latin letters that are equivalent to them.

Note that sometimes a Greek letter can be embedded in an alphabetical string and hard to detect, such as in *Tcralpha*. We cannot distinguish the occurrence of *alpha* in *Tcralpha* from that in *alphabet*, which should not be considered an embedded Greek letter. Since it is hard to distinguish these two cases, we do not attempt to do so, and we follow a simple strategy as follows. We check each maximum span of consecutive alphabetical characters in the text, and replace the ones that are in the Greek alphabet. Thus, both the *alpha* in *Tcralpha* and that in *alphabet* will be replaced by *a*. We call this Greek letter replacement strategy the *Greek-Normalization* heuristic, or *G-Norm*. Note that while the three normalization methods described in Section 3.3 are mutually exclusive, *G-Norm* is orthogonal to those three normalization methods, and thus can be applied on top of any of them.

4 Stemming and Stop Word Removal

After tokenization, stemming is an optional step to further normalize the tokens. Based on previous work that explored stemming algorithms for biomedical information retrieval, we consider three stemmers in our evaluation: the Porter stemmer [Porter 1980], the Lovins stemmer [Lovins 1968], and the S stemmer [Harman 1991]. The S stemmer only removes a few common word endings. The Lovins stemmer is more aggressive than the Porter stemmer, which in turn is more aggressive than the S stemmer.

We also consider two stop word removal methods. One method uses an external stop word list. In our experiments, we use the stop word list from PubMed. Since stop words are essentially the most frequent words in a document collection, the second method we consider uses a stop word list generated from the document collection itself by extracting the most frequent k tokens.

5 Evaluation

In this section, we show our empirical evaluation of the set of tokenization heuristics we have described in Sections 3 and 4. Specifically, our goal is as follows: For removal of the non-functional special characters, intuitively it should improve the performance, because most of the noise caused by punctuation such as periods and commas is removed by this heuristic. The purpose of the evaluation of this heuristic is thus to see whether this non-functional character removal step is safe for most of the queries. For the three sets of break points we defined, BP1, BP2, and BP3, the goal of the evaluation is to see which set gives the best retrieval performance when it is used in conjunction with break point normalization. Similarly, for the three break point normalization methods, the goal is the find the best normalization method for retrieval. For Greek alphabet normalization, we want to see whether this replacement can improve the retrieval performance. Lastly, for stemming and stop word removal, we want to see whether stemming improves the performance and which stemmer performs the best, and whether removing stop words improves the performance.

We also need to make our evaluation of tokenization strategies independent of the retrieval method being used so that the best tokenization strategies we find can be used for any standard information retrieval method. All the state-of-the-art retrieval formulas are based on the bag-of-word representation and share similar retrieval heuristics [Fang et al. 2004]. We thus expect them to be affected by the tokenization method in a similar way. In our study, we choose two representative retrieval methods to use in our evaluation: a TF-IDF retrieval method with BM25 term frequency weighting, and the KL-divergence retrieval method [Lafferty and Zhai 2001], which represents the language modeling approach. We call the first method "TFIDF" and the second method "KL" in this section. The details of the TFIDF method are explained in [Zhai 2001a].

Both retrieval methods are implemented in the Lemur Language Modeling Toolkit[1], which we use for our experiments. We tune the parameters in our experiments because the parameters are sensitive to the query type and to the tokenization heuristics used.

Previous studies have shown that pseudo relevance feedback can often improve the retrieval performance. Usually pseudo relevance feedback works by introducing useful new terms to the queries. To see whether the choice of the best tokenization strategies is affected by whether pseudo relevance feedback is used, we also applied the model-based feedback method [Zhai and Lafferty 2001b] in our experiments. This method is based on the KL-divergence retrieval method, and is also implemented in the Lemur Toolkit. The number of

[1]http://www.lemurproject.org/

feedback documents to use is set to 5 in all our experiments. The other feedback parameters are tuned in the experiments. We refer to this pseudo relevance feedback method as "KL-FB" in the rest of this section.

In all our experiments, for each set of queries, we use the mean average precision (MAP) measure as the evaluation metric. MAP has so far been the standard measure used to evaluate ad hoc retrieval results and has also been used in the TREC Genomics Track evaluation [Hersh et al. 2004, Hersh et al. 2005]. Compared with some other performance measures such as "precision at 10" and "R-precision", MAP has the advantage of being sensitive to the rank of *every* relevant document, thus it reflects well the overall ranking accuracy.

5.1 Document Collections and Queries

The document collections and the queries we use for evaluation are from the ad hoc retrieval task of the TREC 2003, TREC 2004 and TREC 2005 Genomics Track[2]. The document collection used in the TREC 2003 Genomics Track contains 525,938 MEDLINE records between April 2002 and April 2003. The collection used in 2004 and 2005 is a 10-year subset of MEDLINE records from 1994 to 2003.

The topics used in the three years' Genomics Track represent different types of queries and different information need. The 50 topics from TREC 2003 each consist of a gene and an organism name with the specific retrieval task formally stated as follows: For gene X, find all MEDLINE references that focus on the basic biology of the gene or its protein products from the designated organism. Basic biology includes isolation, structure, genetics and function of genes/proteins in normal and disease states. Because this information need is very broad but at the same time centered around the topic genes, we use only the gene names to form keyword queries. We do not use the names of the organisms because our preliminary experiment results show that including the organism names may hurt the performance. For

[2]http://ir.ohsu.edu/genomics/

the gene in each topic, several types of names are given, including the official name, the official symbol, the alias symbols, the product, etc. These types of names fall into two categories: the gene names and gene products are usually long, descriptive names, such as *chemokine (C-C motif) ligand 3*, and the gene symbols are short, symbolic names, such as *CCL3* and *MIP1A*. We thus form two groups of keyword queries from the 2003 topics. For each 2003 topic, we use the union of the topic gene's descriptive names to form a *name query*, and we use the union of the gene's symbolic names to form a *symbol query*. In the end, we get 50 keyword name queries and 50 keyword symbol queries from the 2003 topics. This separation presumably captures two possible types of real-world queries from biology researchers.

The 50 topics from TREC 2004 each consist of a title field, an information need field, and a context field. These topics may or may not contain a gene or protein name. Our preliminary experiment results show that using only the information need field to form queries gives the best retrieval performance. We thus use the text in the information need field only to form 50 verbose queries. These queries often contain common English words as background words, such as *about* in the query "Find articles about Ferroportin-1, an iron transporter, in humans."

The 50 topics from TREC 2005 are structured topics with templates. There are 5 templates representing 5 kinds of information need. For example, one template is "Provide information about the role of the gene $\underline{\text{X}}$ involved in the disease $\underline{\text{Y}}$." We exclude those template background words such as *provide* and *information* when forming the queries. After removing the background words, most queries still contain more than 5 words, including some English words, so we still consider them verbose queries. We further divide the queries into two groups: queries that involve at least one gene (queries belonging to Templates 2, 3, 4 and 5), and queries that do not involve any gene (queries belonging to Template 1). We exclude Topic 135 because it does not have any judged relevant document. We thus get 39 *gene queries* and 10 *non-gene queries* from the 2005 topics.

To summarize, we use 5 sets of queries for evaluation: 50 keyword gene symbol queries from TREC 2003, 50 keyword gene name queries from TREC 2003, 50 verbose mixed queries from TREC 2004, 39 verbose gene queries from TREC 2005, and 10 verbose non-gene queries from TREC 2005. The 2004 queries are more verbose than the 2005 queries. We do not consider query expansion using external knowledge bases because the goal of our study is not to improve the absolute retrieval performance, but rather to compare the tokenization strategies. Once we find the best set of tokenization strategies, we can establish a strong baseline method for biomedical information retrieval by using the best tokenization strategies. Presumably, any more sophisticated technique tailored for biomedical information retrieval is orthogonal to this strong baseline, and therefore can be applied on top of the strong baseline.

5.2 Tokenization Heuristics

To evaluate the tokenization heuristics, we first compare the retrieval performance before and after the removal of non-functional characters. Concluding that removing the non-functional characters is safe, we then further apply the three break point normalization methods in conjunction with the three sets of break points. This gives us 9 sets of experiments. We then evaluate the Greek alphabet normalization heuristic by applying it on top of each break point normalization method in conjunction with the best set of break points. All experiments are run on each set of queries.

5.2.1 Removal of Non-Functional Characters

Table 6 shows the comparison of the retrieval performance between the naive tokenization method, which is defined in Section 3, and the tokenization method that removes the non-functional characters. We refer to the latter method as the baseline method because it is applicable to general English text as well. Recall that the naive method uses only white space characters as delimiters. The "% Diff." rows show the relative difference between the baseline method and the naive method for each set of queries. An asterisk indicates that the

Table 6: Comparison of the naive and the baseline methods

Keyword Queries

03 Symbol			
Method	TFIDF	KL	KL-FB
Naive	0.1548	0.1451	0.1668
Baseline	0.1659	0.1523	0.1817
% Diff.	**+7.17%**	**+4.96%**	**+8.93%**

03 Name			
Method	TFIDF	KL	KL-FB
Naive	0.0919	0.0833	0.1047
Baseline	0.0958	0.0891	0.0975
% Diff.	**+4.24%***	**+6.96%***	**-6.88%**

Verbose Queries

04			
Method	TFIDF	KL	KL-FB
Naive	0.1736	0.1671	0.2377
Baseline	0.2695	0.2687	0.2972
% Diff.	**+55.24%***	**+60.80%***	**+25.03%***

05 Gene			
Method	TFIDF	KL	KL-FB
Naive	0.1929	0.1921	0.2041
Baseline	0.2198	0.2260	0.2343
% Diff.	**+13.95%***	**+17.65%***	**+14.80%***

05 Non-gene			
Method	TFIDF	KL	KL-FB
Naive	0.1166	0.1302	0.1864
Baseline	0.1416	0.1560	0.2297
% Diff.	**+21.44%**	**+19.82%**	**+23.23%***

The numbers shown here are the MAP measures. The **% Diff.** rows show the relative difference between the two methods. Asterisks indicate differences that are statistically significant.

difference is statistically significant at the 95% confidence level. We can see that in almost all cases, the baseline method outperforms the naive method.

The improvement is especially substantial with the verbose queries. Note that the main purpose of removing the non-functional characters is to normalize the words connected with punctuation marks into their canonical forms, i.e. the same words without any connected punctuation marks. For example, "gene," will be normalized into "gene". The difference between the verbose queries and the keyword queries suggests that the common English words in the queries are more affected by the removal of the non-functional characters than the gene names or symbols in the queries. Indeed, compared with gene names or symbols, common English words are more likely to be connected with punctuation marks such as periods and commas.

5.2.2 Break Points

Table 7 shows the comparison among the three sets of break points when one of H-Norm, S-Norm and J-Norm is used in conjunction. For each set of queries and each normalization method, the performance of BP1 is shown in the first row, followed by the performance of BP2 and of BP3. The "% Diff." rows show the relative difference between BP2 or BP3 and

Table 7: Comparison of the three break point sets

Keyword Queries

03 Symbol

Norm	BP	TFIDF	KL	KL-FB
H	**BP1**	0.1653	0.1548	0.1911
	BP2	0.1636	0.1537	0.1818
	% Diff.	**-1.03%**	**-0.71%**	**-4.87%***
	BP3	0.1615	0.1608	0.1873
	% Diff.	**-2.30%**	**+3.88%**	**-1.99%**
S	**BP1**	0.1528	0.1466	0.1726
	BP2	0.1481	0.1434	0.1779
	% Diff.	**-3.08%**	**-2.18%**	**+3.07%**
	BP3	0.1038	0.1276	0.1515
	% Diff.	**-32.07%***	**-12.96%***	**-12.22%**
J	**BP1**	0.1755	0.1758	0.1920
	BP2/ BP3	0.1740	0.1746	0.1905
	% Diff.	**-0.85%**	**-0.68%**	**-0.78%**

03 Name

Norm	BP	TFIDF	KL	KL-FB
H	**BP1**	0.0978	0.0917	0.0997
	BP2	0.0940	0.0881	0.0972
	% Diff.	**-3.89%**	**-3.93%**	**-2.51%***
	BP3	0.0904	0.0880	0.0971
	% Diff.	**-7.57%**	**-4.03%**	**-2.61%**
S	**BP1**	0.1070	0.1061	0.1132
	BP2	0.1045	0.1078	0.1120
	% Diff.	**-2.34%***	**+1.60%**	**-1.06%**
	BP3	0.0900	0.0969	0.1096
	% Diff.	**-15.89%***	**-8.67%**	**-3.18%**
J	**BP1**	0.0930	0.0916	0.1005
	BP2/ BP3	0.0894	0.0882	0.0978
	% Diff.	**-3.87%**	**-3.71%**	**-2.69%**

The numbers shown here are the MAP measures. The **% Diff.** rows show the relative difference between BP2 or BP3 with BP1. Asterisks indicate differences that are statistically significant. When J-Norm is used, BP2 and BP3 become the same.

Verbose Queries

04

Norm	BP	TFIDF	KL	KL-FB
H	**BP1**	0.2702	0.2693	0.3027
	BP2	0.2715	0.2702	0.2992
	% Diff.	**+0.48%**	**+0.33%**	**-1.16%**
	BP3	0.2698	0.2678	0.2955
	% Diff.	**-0.15%**	**-0.56%**	**-2.38%**
S	**BP1**	0.3201	0.3015	0.3329
	BP2	0.3122	0.3006	0.3258
	% Diff.	**-2.47%**	**-0.30%***	**-2.13%***
	BP3	0.2339	0.2406	0.2967
	% Diff.	**-26.93%**	**-20.20%**	**-10.87%**
J	**BP1**	0.2823	0.2741	0.3033
	BP2/ BP3	0.2838	0.2749	0.3054
	% Diff.	**+0.53%***	**+0.29%**	**+0.69%***

05 Gene

Norm	BP	TFIDF	KL	KL-FB
H	**BP1**	0.2424	0.2432	0.2606
	BP2	0.2421	0.2431	0.2604
	% Diff.	**-0.12%***	**-0.04%**	**-0.08%**
	BP3	0.2442	0.2447	0.2605
	% Diff.	**+0.74%**	**+0.62%**	**+1.69%**
S	**BP1**	0.2757	0.2808	0.3024
	BP2	0.2745	0.2793	0.3017
	% Diff.	**-0.44%**	**-0.53%***	**-0.23%**
	BP3	0.2578	0.2757	0.2941
	% Diff.	**-6.49%**	**-1.82%**	**-2.74%**
J	**BP1**	0.2471	0.2476	0.2689
	BP2/ BP3	0.2469	0.2475	0.2692
	% Diff.	**-0.08%***	**-0.04%**	**+11.00%**

05 Non-gene

Norm	BP	TFIDF	KL	KL-FB
H	**BP1**	0.1415	0.1563	0.2260
	BP2	0.1415	0.1563	0.2303
	% Diff.	**0.00%**	**0.00%**	**+1.90%**
	BP3	0.1414	0.1562	0.2266
	% Diff.	**-0.71%**	**-0.06%**	**+0.27%**
S	**BP1**	0.1706	0.1858	0.2352
	BP2	0.1650	0.1835	0.2408
	% Diff.	**-3.28%**	**-1.24%**	**+2.38%**
	BP3	0.1654	0.1795	0.2205
	% Diff.	**-3.05%**	**-3.39%**	**-6.25%**
J	**BP1**	0.1416	0.1565	0.2268
	BP2/ BP3	0.1417	0.1563	0.2270
	% Diff.	**+0.07%**	**-0.13%**	**+0.09%**

BP1. An asterisk indicates a statistically significant difference. Although the pattern is not consistent, we can see that BP1 performs better than BP2 and BP3 in most of the cases, especially when S-Norm is used, or when J-Norm is used for the keyword queries. As we will show next, S-Norm and J-Norm are preferred over H-Norm. We thus choose BP1 as the best set of break points to use.

5.2.3 Break Point Normalization

Table 8 shows the comparison among the three break point normalization methods when each set of break points is used in conjunction. For the keyword symbols queries, we put the performance of J-Norm in the first row, followed by the performance of H-Norm and of S-Norm. For all the other query sets, we put S-Norm in the first row, followed by H-Norm and J-Norm. The "% Diff." rows show the relative difference between the previous row and the first row in that section. In another word, for the keyword symbol queries, the "% Diff." rows show the relative difference between H-Norm or S-Norm and J-Norm; for the other queries, the "% Diff." rows show the relative difference between H-Norm or J-Norm and S-Norm. An asterisk indicates a statistically significant difference.

It is very clear from Table 8, especially when BP1 or BP2 is used, that for the keyword symbol queries, J-Norm performs the best, and for the keyword name queries and the verbose queries, S-Norm performs the best. This suggests that for gene symbols that are combinations of alphabetical characters, numerical characters and special characters such as hyphens, removing the special characters is the most effective way to normalize different lexical variants of the same name. For keyword name queries and verbose queries, however, most of the query words are not gene symbols. Replacing special characters such as hyphens and slashes with spaces is the most effective normalization method. This is probably not only because S-Norm may help normalize the gene names but also because it effectively separates hyphenated compound words such as *CCAAT/enhancer-binding* and *azaserine-induced*, which are better to be separated for retrieval purposes.

Table 8: Comparison of break point normalization methods

Keyword Queries

03 Symbol

BP	**Norm**	TFIDF	KL	KL-FB
BP1	**J**	0.1755	0.1758	0.1920
	H	0.1653	0.1548	0.1911
	% Diff.	**-5.81%**	**-11.95%**	**-0.47%**
	S	0.1528	0.1466	0.1726
	% Diff.	**-12.93%**	**-16.61%**	**-10.10%**
BP2	**J**	0.1740	0.1746	0.1726
	H	0.1636	0.1537	0.1818
	% Diff.	**-5.98%**	**-11.97%**	**-4.57%**
	S	0.1481	0.1434	0.1779
	% Diff.	**-14.89%**	**-17.87%**	**-6.61%**
BP3	**J**	0.1740	0.1746	0.1905
	H	0.1615	0.1608	0.1873
	% Diff.	**-7.18%**	**-7.90%**	**-1.68%**
	S	0.1013	0.1276	0.1515
	% Diff.	**-40.34%***	**-26.92%***	**-20.47%***

03 Name

BP	**Norm**	TFIDF	KL	KL-FB
BP1	**S**	0.1070	0.1061	0.1132
	H	0.0978	0.0917	0.0997
	% Diff.	**-8.60%***	**-13.57%**	**-11.93%**
	J	0.0930	0.0916	0.1005
	% Diff.	**-13.08%***	**-13.67%**	**-11.22%**
BP2	**S**	0.1045	0.1078	0.1120
	H	0.0940	0.0881	0.0972
	% Diff.	**-10.05%***	**-18.27%**	**-13.21%**
	J	0.0894	0.0882	0.0978
	% Diff.	**-14.45%***	**-18.18%**	**-12.68%**
BP3	**S**	0.0900	0.0969	0.1096
	H	0.0904	0.0880	0.0971
	% Diff.	**+0.44%**	**-9.18%**	**-11.41%**
	J	0.0894	0.0882	0.0978
	% Diff.	**-0.67%**	**-8.98%**	**-10.77%**

The numbers shown here are the MAP measures. The "**% Diff.**" rows show the relative difference between the previous row and the first row in the current section. Asterisks indicate differences that are statistically significant.

Verbose Queries

04

BP	**Norm**	TFIDF	KL	KL-FB
BP1	**S**	0.3201	0.3015	0.3329
	H	0.2702	0.2693	0.3027
	% Diff.	**-15.59%***	**-10.68%***	**-9.07%**
	J	0.2823	0.2741	0.3033
	% Diff.	**-11.81%***	**-9.09%***	**-8.89%**
BP2	**S**	0.3122	0.3006	0.3258
	H	0.2715	0.2702	0.2992
	% Diff.	**-13.04%***	**-10.11%**	**-8.16%**
	J	0.2838	0.2749	0.3054
	% Diff.	**-9.10%***	**-8.55%**	**-6.26%**
BP3	**S**	0.2339	0.2406	0.2967
	H	0.2698	0.2678	0.2955
	% Diff.	**+15.35%**	**+11.31%**	**-0.40%**
	J	0.2838	0.2749	0.3054
	% Diff.	**+21.33%**	**+14.26%**	**+2.93%**

05 Gene

BP	**Norm**	TFIDF	KL	KL-FB
BP1	**S**	0.2757	0.2808	0.3024
	H	0.2424	0.2432	0.2606
	% Diff.	**-12.08%***	**-13.39%***	**-13.82%***
	J	0.2471	0.2476	0.2689
	% Diff.	**-10.37%***	**-11.82%***	**-11.08%***
BP2	**S**	0.2745	0.2793	0.3017
	H	0.2421	0.2431	0.2604
	% Diff.	**-11.80%***	**-12.96%***	**-13.69%***
	J	0.2469	0.2475	0.2692
	% Diff.	**-10.05%***	**-11.39%***	**-10.77%***
BP3	**S**	0.2578	0.2757	0.2941
	H	0.2442	0.2447	0.2650
	% Diff.	**-5.28%***	**-11.24%***	**-9.89%***
	J	0.2469	0.2475	0.2692
	% Diff.	**-4.23%**	**-10.23%***	**-8.47%***

05 Non-gene

BP	**Norm**	TFIDF	KL	KL-FB
BP1	**S**	0.1708	0.1858	0.2352
	H	0.1415	0.1563	0.2260
	% Diff.	**-17.06%**	**-15.88%**	**-3.91%**
	J	0.1416	0.1656	0.2268
	% Diff.	**-17.00%**	**-15.77%**	**-3.57%**
BP2	**S**	0.1650	0.1835	0.2408
	H	0.1415	0.1563	0.2303
	% Diff.	**-14.24%**	**-14.82%***	**-4.36%**
	J	0.1417	0.1563	0.2270
	% Diff.	**-14.12%**	**-14.82%**	**-5.73%**
BP3	**S**	0.1654	0.1795	0.2205
	H	0.1414	0.1562	0.2266
	% Diff.	**-14.51%**	**-12.98%**	**+2.77%**
	J	0.1417	0.1563	0.2270
	% Diff.	**-14.33%**	**-12.92%**	**+2.95%**

Table 9: The effect of Greek alphabet normalization

Keyword Queries				
03 Symbol				
Norm	**G-Norm**	TFIDF	KL	KL-FB
H	**No**	0.1653	0.1548	0.1911
	Yes	0.1659	0.1541	0.1886
	% Diff.	**+0.36%**	**-0.45%**	**-1.31%**
S	**No**	0.1528	0.1466	0.1726
	Yes	0.1534	0.1471	0.1724
	% Diff.	**+0.39%**	**+0.34%**	**-0.12%**
J	**No**	0.1755	0.1758	0.1920
	Yes	0.1783	0.1789	0.1943
	% Diff.	**+1.60%**	**+1.76%**	**+1.19%**
03 Name				
Norm	**G-Norm**	TFIDF	KL	KL-FB
H	**No**	0.0978	0.0917	0.0997
	Yes	0.0951	0.0899	0.0980
	% Diff.	**-2.76%***	**-1.96%***	**-1.71%**
S	**No**	0.1070	0.1061	0.1132
	Yes	0.1033	0.1034	0.1094
	% Diff.	**-3.46%***	**-2.54%***	**-3.36%***
J	**No**	0.0930	0.0916	0.1005
	Yes	0.0896	0.0897	0.0976
	% Diff.	**-3.66%***	**-2.07%***	**-2.89%***

Verbose Queries				
04				
Norm	**G-Norm**	TFIDF	KL	KL-FB
H	**No**	0.2702	0.2693	0.3027
	Yes	0.2688	0.2671	0.2992
	% Diff.	**-0.52%**	**-0.82%***	**-1.16%**
S	**No**	0.3201	0.3015	0.3329
	Yes	0.3186	0.2997	0.3308
	% Diff.	**-0.47%**	**-0.60%**	**-0.63%***
J	**No**	0.2823	0.2741	0.3033
	Yes	0.2832	0.2740	0.3015
	% Diff.	**+0.32%**	**-0.04%**	**-0.59%**
05 Gene				
Norm	**G-Norm**	TFIDF	KL	KL-FB
H	**No**	0.2424	0.2432	0.2606
	Yes	0.2416	0.2428	0.2602
	% Diff.	**-0.33%**	**-0.16%**	**-0.15%**
S	**No**	0.2757	0.2808	0.3024
	Yes	0.2751	0.2814	0.3029
	% Diff.	**-0.22%**	**+0.21%**	**+0.17%**
J	**No**	0.2471	0.2471	0.2689
	Yes	0.2465	0.2476	0.2680
	% Diff.	**-0.24%***	**-0.20%***	**-0.33%**
05 Non-gene				
Norm	**G-Norm**	TFIDF	KL	KL-FB
H	**No**	0.1415	0.1563	0.2260
	Yes	0.1415	0.1563	0.2177
	% Diff.	**0.00%**	**0.00%**	**-3.67%**
S	**No**	0.1706	0.1858	0.2352
	Yes	0.1705	0.1858	0.2343
	% Diff.	**-0.06%**	**0.00%**	**-0.38%**
J	**No**	0.1416	0.1565	0.2268
	Yes	0.1416	0.1565	0.2267
	% Diff.	**0.00%**	**0.00%**	**-0.04%**

The numbers shown here are the MAP measures. The "**% Diff.**" rows show the relative difference between the previous row and the first row in the current section. Asterisks indicate differences that are statistically significant.

5.2.4 Greek Alphabet Normalization

In Table 9, we show the effect of applying G-Norm on top H-Norm, S-Norm and J-Norm when the best set of break points, BP1, is used. For each set of queries and each normalization method, the "% Diff." row shows the relative difference between the methods with and without G-Norm. An asterisk indicates a statistically significant difference. Although the pattern is not very clear in the comparison, we can see that in most cases, Greek alphabet normalization does not improve the performance. However, for keyword symbol queries, when J-Norm is used, G-Norm does improve the performance a little bit. Since J-Norm is the best break point normalization method for keyword symbol queries, we conclude that we should apply G-Norm on top of J-Norm when the queries are keyword symbol queries.

Table 10: The effect of stemming

Keyword Queries			
03 Symbol			
Method	TFIDF	KL	KL-FB
Best Tokenization	0.1783	0.1789	0.1943
+Porter	0.1743	0.1762	0.1870
% Diff.	**-2.24%***	**-1.51%***	**-3.76%***
+Lovins	0.1728	0.1713	0.1871
% Diff.	**-3.08%***	**-4.25%***	**-3.71%**
+S	0.1749	0.1768	0.1905
% Diff.	**-1.91%***	**-1.17%***	**-1.96%**
03 Name			
Method	TFIDF	KL	KL-FB
Best Tokenization	0.1070	0.1061	0.1132
+Porter	0.1080	0.1029	0.1118
% Diff.	**+0.93%**	**-3.02%***	**-1.24%**
+Lovins	0.1054	0.1030	0.1110
% Diff.	**-1.50%**	**-2.92%***	**-1.94%**
+S	0.1070	0.1020	0.1117
% Diff.	**0.00%**	**-3.86%**	**-1.33%**

The numbers shown here are the MAP measures. The "**% Diff.**" rows show the relative difference between each stemmer and the best tokenization method without stemming. Asterisks indicate statistically significant differences.

Verbose Queries			
04			
Method	TFIDF	KL	KL-FB
Best Tokenization	0.3201	0.3015	0.3329
+Porter	0.3297	0.3285	0.3357
% Diff.	**+3.00%**	**+8.96%***	**+0.84%**
+Lovins	0.3088	0.3058	0.2969
% Diff.	**-3.53%**	**+1.43%**	**-10.81%**
+S	0.3240	0.3219	0.3472
% Diff.	**+1.22%**	**+6.77%***	**+4.30%**
05 Gene			
Method	TFIDF	KL	KL-FB
Best Tokenization	0.2757	0.2808	0.3024
+Porter	0.2859	0.2892	0.3119
% Diff.	**+3.70%**	**+2.99%***	**+3.14%***
+Lovins	0.2964	0.2935	0.3251
% Diff.	**+7.51%***	**+4.52%***	**+7.51%***
+S	0.2849	0.2830	0.3058
% Diff.	**+3.34%***	**+0.78%***	**+1.12%***
05 Non-gene			
Method	TFIDF	KL	KL-FB
Best Tokenization	0.1706	0.1858	0.2352
+Porter	0.1607	0.1964	0.2385
% Diff.	**-5.80%**	**+5.71%**	**+1.40%**
+Lovins	0.1788	0.2005	0.2378
% Diff.	**+4.81%**	**+7.91%**	**+1.11%**
+S	0.1714	0.1949	0.2408
% Diff.	**+0.47%**	**+4.90%**	**+2.38%**

5.3 Stemming and Stop Word Removal

5.3.1 Stemming

In this section, we compare non-stemming and stemming performance, and compare the different stemming algorithms. We apply three stemmers, the Porter stemmer, the Lovins stemmer, and the S stemmer, on top of the best tokenization strategy for each set of queries. The performance is shown in Table 10. For each set of queries, the performance of the best tokenization method is shown in the first row, followed by the performance when each stemmer is applied on top of the best tokenization method. The "% Diff." rows show the relative difference between each stemmer and the best tokenization method. Asterisks indicate statistically significant differences. We can see that for keyword queries, all three stemmers decrease the performance in most cases. However, for verbose queries, in most cases all three stemmers improve the performance. There is no clear conclusion about which

Table 11: The effect of stop word removal

Keyword Queries

03 Symbol			
Stop Word List	TFIDF	KL	KL-FB
No	0.1783	0.1786	0.1943
Pubmed (132)	0.1783	0.1753	0.1903
% Diff.	**0.00%**	**-2.01%**	**-2.06%**
Collection-5	0.1783	0.1778	0.1928
% Diff.	**0.00%**	**-0.61%**	**-0.77%**
Collection-10	0.1783	0.1771	0.1939
% Diff.	**0.00%**	**-1.01%**	**-0.21%**
Collection-20	0.1783	0.1768	0.1915
% Diff.	**0.00%**	**-1.17%**	**-1.44%**
Collection-100	0.1783	0.1771	0.1865
% Diff.	**0.00%**	**-1.01%**	**-4.01%**

03 Name			
Stop Word List	TFIDF	KL	KL-FB
No	0.1070	0.1061	0.1132
Pubmed (132)	0.1071	0.0953	0.1155
% Diff.	**+0.09%**	**-10.18%**	**+2.03%**
Collection-5	0.1070	0.1035	0.1119
% Diff.	**0.00%**	**-2.45%**	**-1.15%**
Collection-10	0.1070	0.1015	0.1118
% Diff.	**0.00%**	**-4.34%**	**-1.24%**
Collection-20	0.1070	0.1006	0.1116
% Diff.	**0.00%**	**-5.18%**	**-1.41%**
Collection-100	0.0990	0.0861	0.0905
% Diff.	**-7.48%***	**-18.85%***	**-10.05%***

Verbose Queries

04			
Stop Word List	TFIDF	KL	KL-FB
No	0.3297	0.3285	0.3357
Pubmed (132)	0.3346	0.3219	0.3300
% Diff.	**+1.49%***	**-2.01%***	**-1.70%***
Collection-5	0.3298	0.3248	0.3286
% Diff.	**+0.03%**	**-1.13%***	**-2.11%***
Collection-10	0.3298	0.3229	0.3263
% Diff.	**+0.03%***	**-1.40%***	**-2.80%***
Collection-20	0.3288	0.3216	0.3289
% Diff.	**-0.27%**	**-2.10%***	**-2.03%***
Collection-100	0.3216	0.3099	0.3206
% Diff.	**-2.46%***	**-5.66%***	**-4.50%***

05 Gene			
Stop Word List	TFIDF	KL	KL-FB
No	0.2859	0.2892	0.3119
Pubmed (132)	0.2840	0.2916	0.3177
% Diff.	**-0.66%**	**+0.83%**	**+1.86%***
Collection-5	0.2860	0.2937	0.3159
% Diff.	**+0.04%***	**+1.56%***	**+1.28%***
Collection-10	0.2860	0.2926	0.3178
% Diff.	**+0.03%**	**+1.18%***	**+1.89%***
Collection-20	0.2817	0.2920	0.3166
% Diff.	**-1.47%**	**+0.97%***	**+1.51%**
Collection-100	0.2735	0.2761	0.3000
% Diff.	**-4.34%**	**-4.53%**	**-3.82%**

05 Non-gene			
Stop Word List	TFIDF	KL	KL-FB
No	0.1607	0.1964	0.2385
Pubmed (132)	0.1594	0.1852	0.2296
% Diff.	**-0.81%**	**-5.70%**	**-3.73%**
Collection-5	0.1608	0.1942	0.2490
% Diff.	**+0.06%**	**-1.12%***	**+4.40%**
Collection-10	0.1609	0.1932	0.2344
% Diff.	**+0.12%**	**-1.63%**	**-1.72%**
Collection-20	0.1508	0.1730	0.2199
% Diff.	**-6.16%**	**-11.91%***	**-7.80%**
Collection-100	0.1392	0.1497	0.1874
% Diff.	**-13.38%***	**-23.78%***	**-21.43%***

The numbers shown here are the MAP measures. "**Collection-*k***" means the collection-based stop word list with the most frequent k tokens from the collection. The "**% Diff.**" rows show the relative difference brought by removing a certain number of stop words. Asterisks indicate statistically significant differences.

stemmer is the best to use for the verbose queries.

5.3.2 Stop Word Removal

In this section, we compare the two stop word removal methods. Method one uses the PubMed stop word list, which consists of 132 common English words. Method two uses a collection-based stop word list, i.e. the most frequent k words in the same document collection for retrieval. We use different values of k to see how this cutoff number affects the performance. Table 11 shows the performance on each query set before and after stop word removal. The "% Diff." rows show the relative difference brought by removing a certain number of stop words. Asterisks indicate statistically significant differences. We can see that stop word removal either does not improve the performance, or only slightly improves the performance. When we use collection-based stop word list, it is also hard to decide the cutoff number k. We thus conclude that in general we should not apply stop word removal for biomedical information retrieval, simply because we do not know when it will help.

5.4 Improvement Summary

From the above comparisons, we can draw the following conclusions. First, we can safely remove those non-functional characters as defined in Section 3.1. Second, BP1 is the best set of break points to use for break point normalization. Third, for keyword symbol queries, J-Norm is the most effective break point normalization method, and for keyword name queries and verbose queries, S-Norm is the most effective normalization method. Fourth, Greek alphabet normalization in general is not effective except when J-Norm is used for the keyword symbol queries. And last, for verbose queries, we should perform stemming.

In Table 12, we show the relative improvement brought by each tokenization heuristic and the overall improvement over the naive method. Except for the overall improvement, the percentage of improvement shown in the table is the improvement with respect to the previous row. Asterisks indicate statistically significant differences. The gene queries and the non-gene queries from 2005 are combined. Besides the MAP measure, here we also report another performance metric, precision at 10 (Pr@10).

We can see from the table that when a set of suitable tokenization heuristics are used for each type of queries, the MAP performance measure can improve by at least 8% for all sets of queries. The improvement is mostly substantial for the 2004 and 2005 queries, which are verbose queries. For 2004 queries, the improvement is mostly brought by the removal of the non-functional characters. The reason may be that 2004 queries contain more background English words than the 2005 queries. It therefore suggests that the more verbose

Table 12: The relative improvement brought by each tokenization heuristic and stemming

Keyword Symbol Queries

	03 Symbol					
	TFIDF		KL		KL-FB	
Method	**MAP**	**Pr@10**	**MAP**	**Pr@10**	**MAP**	**Pr@10**
Naive	0.1548	0.1320	0.1451	0.1220	0.1668	0.1420
Baseline	0.1659	0.1280	0.1523	0.1200	0.1817	0.1520
% Impr.	+7.17%	-3.03%	+4.96%	-1.64%	+8.93%	+7.04%
BP1 + J-Norm	0.1755	0.1420	0.1758	0.1440	0.1920	0.1640
% Impr.	+5.79%	+10.94%	+15.43%	+20.00%*	+5.67%	+7.89%
+G-Norm	0.1783	0.1500	0.1789	0.1520	0.1943	0.1740
% Impr.	+1.60%	+5.63%	+1.76%	+5.56%	+1.19%	+6.10%
Overall % Impr. over Naive	+15.18%	+13.64%	+23.29%	+24.59%*	+16.49%	+22.54%*

Keyword Name Queries

	03 Name					
	TFIDF		KL		KL-FB	
Method	**MAP**	**Pr@10**	**MAP**	**Pr@10**	**MAP**	**Pr@10**
Naive	0.0919	0.0860	0.0833	0.0780	0.1047	0.0880
Baseline	0.0958	0.0740	0.0891	0.0680	0.0975	0.0760
% Impr.	+4.24%*	-13.95%	+6.96%*	-12.82%*	-6.88%	-13.64%
BP1 + S-Norm	0.1070	0.0800	0.1061	0.0920	0.1132	0.0820
% Impr.	+11.69%*	+8.11%	+19.08%	+35.29%*	+16.10%	+7.89%
Overall % Impr. over Naive	+16.43%*	-6.98%	+27.37%*	+17.95%	+8.12%	-6.82%

Verbose Queries

	04					
	TFIDF		KL		KL-FB	
Method	**MAP**	**Pr@10**	**MAP**	**Pr@10**	**MAP**	**Pr@10**
Naive	0.1736	0.3960	0.1671	0.3920	0.2377	0.4240
Baseline	0.2695	0.4420	0.2687	0.4700	0.2972	0.4980
% Impr.	+55.24%*	+11.62%*	+60.80%*	+19.90%*	+25.03%*	+17.45%*
BP1 + J-Norm	0.3201	0.4920	0.3015	0.4860	0.3329	0.45020
% Impr.	+18.78%*	+11.31%*	+12.21%*	+3.40%	+12.01%*	+0.80%
+Porter	0.3297	0.5300	0.3285	0.5040	0.3357	0.4860
% Impr.	+3.00%	+7.72%	+8.96%*	+3.70%	+0.84%	-3.19%
Overall % Impr. over Naive	+89.92%*	+33.84%*	+96.59%*	+28.57%*	+41.23%*	+14.62%*

Verbose Queries

	05					
	TFIDF		KL		KL-FB	
Method	**MAP**	**Pr@10**	**MAP**	**Pr@10**	**MAP**	**Pr@10**
Naive	0.1773	0.3592	0.1795	0.3653	0.2005	0.3816
Baseline	0.2039	0.3531	0.2117	0.3694	0.2334	0.3898
% Impr.	+15.00%*	-1.70%	+17.94%*	+1.12%	+16.41%*	+2.15%
BP1 + J-Norm	0.2543	0.3735	0.2614	0.4122	0.2887	0.4347
% Impr.	+24.72%*	+5.78%	+23.48%*	+11.59%*	+23.69%*	+11.52%
+Porter	0.2603	0.4082	0.2703	0.4367	0.2969	0.4571
% Impr.	+2.36%	+9.29%	+3.40%*	+5.94%*	+2.84%*	+5.15%
Overall % Impr. over Naive	+46.81%*	+13.64%	+50.58%*	+19.55%*	+48.08%*	+19.79%*

a query is, i.e. the more background English words a query contains, the more important it is to carefully remove the non-functional characters when doing tokenization. For 2003 name queries and 2005 queries, however, the break point normalization step contributes more than the removal of the non-functional characters to the final improvement. This suggests that for queries containing gene names, normalization of break points is important in tokenization.

The change of Pr@10 in general follows the same pattern as that of MAP. However, when some heuristic is applied, especially when removal of non-functional characters is applied to the keyword queries, Pr@10 decreases while MAP increases, which is not very surprising given that normalization of terms is generally expected to increase recall at the risk of decreasing the precision of top-ranked documents.

We compared the best performance we obtained with the performance of the TREC Genomics Track official runs. For TREC 2004, the best automatic run achieved a MAP of 0.4075, and the average MAP of all runs is 0.2074 [Hersh et al. 2004]. The best MAP we obtained is 0.3357, which is better than the fifth best automatic run in TREC 2004. For TREC 2005, the best automatic run achieved a MAP of 0.2888 [Hersh et al. 2005]. The best MAP we obtained is 0.2969, which is higher than the best run at TREC. For TREC 2003, since we separated the gene symbols and the gene names in the queries, our results are not comparable to those reported at TREC. This comparison shows that by using good tokenization strategies, we can have a baseline method for biomedical information retrieval that is competitive with the state-of-the-art systems.

6 Discussion and Conclusions

Because of the irregular forms of entity names and their lexical variants in the biomedical text, appropriate tokenization is an important preprocessing step in biomedical information retrieval. In this paper, we systematically evaluated a set of tokenization strategies generalized from existing work, including a non-functional character removal step, a break point normalization step with three possible normalization methods and three possible sets of break points, and a Greek alphabet normalization step. We also empirically studied the effect of stemming and stop word removal for biomedical information retrieval. Our evaluation was conducted on all the available TREC biomedical information retrieval test collections, and we employed two representative retrieval methods as well as a pseudo relevance feedback method. Results from all experiments show that tokenization can significantly affect the retrieval performance, as we expected; appropriate tokenization can improve the retrieval performance by up to 96%.

Based on our experiment results, the general recommendations are: First, non-functional characters should be removed from the text using a set of heuristic rules. Second, for different types of queries, different tokenization heuristics should be applied. For queries that contain only gene symbols, removing brackets, hyphens, slashes and underlines in the tokens and replacing Greek letters with their Latin equivalents are useful. For queries that contain only full gene names and for verbose queries that also contain English words, replacing brackets, hyphens, slashes and underlines with spaces should be used. Numerical characters should not be separated from alphabetical characters. Third, for verbose queries, stemming can be used to further improve the performance. Finally, stop word removal in general should not be performed to avoid the risk of hurting the performance.

Our study still has some limitations. One limitation is that not all text collections we use are well suited for this evaluation. In particular, we are aware that the relevance judgments for the TREC 2003 Genomics Track data set were automatically generated from GeneRIF [Hersh et al. 2003], which are incomplete; some relevant documents are not included in the relevance judgments. This problem with the data clearly may affect our evaluation, and the conclusions we draw from the keyword queries, which are all from TREC 2003, may need further testing.

It is hard, if not impossible, to enumerate all the possible tokenization strategies. Our philosophy has been to take a "bottom-up" approach and systematically enumerate the most basic tokenization strategies, with the goal being to develop a set of effective, but conservative tokenization strategies. However, there are still some other tokenization strategies that we did not include in our evaluation. For example, we did not consider the option of not performing case normalization. In biomedical text, sometimes the same string may convey different meanings when it is capitalized and when it is all in lower case. However, we believe that such cases are not common, and if we do not perform case normalization, the improvement we gain for these special cases is likely to be outweighed by the decrement resulted from other cases where case normalization is harmful. Nevertheless, we should test this option in our future work. Another strategy that we did not try is to combine some adjacent words into single tokens. For example, "MIP 1" may be transformed into "MIP-1" with this strategy, and hence be matched with "MIP-1". We did not test this strategy for two reasons. (1) One of our break point normalization methods, S-Norm, already addresses the same problem in a different way. "MIP 1" and "MIP-1" will both be transformed into "MIP 1" under S-Norm, and thus be matched. (2) It is hard to define a set of reasonable rules to combine adjacent tokens without compromising our philosophy of being conservative as this heuristic may affect term frequencies and document lengths in an unpredictable way. However, it would still be interesting to further explore this heuristic in the line of Song [2003].

Acknowledgments

This work was in part supported by the National Science Foundation under award numbers 0425852 and 0428472.

References

[Ando et al. 2005] Ando, R. K., Dredze, M., & Zhang, T. (2005). TREC 2005 genomics track experiments at IBM Watson. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005).*

[Buttcher et al. 2004] Buttcher, S., Clarke, C. L. A., & Cormack, G. V. (2004). Domain-specific synonym expansion and validation for biomedical information retrieval (MultiText experiments for TREC 2004). In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004).*

[Carpenter 2004] Carpenter, B. (2004). Phrasal queries with LingPipe and Lucene: ad hoc genomics text retrieval. In *Proceedings of Thirteenth Text REtrieval Conference (TREC 2004).*

[Crangle et al. 2004] Crangle, C., Zbyslaw, A., Cherry, J. M., & Hong, E. L. (2004). Concept extraction and synonym management for biomedical information retrieval. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004).*

[Dayanik et al. 2003] Dayanik, A., Nevill-Manning, C. G., & Oughtred, R. (2003). Partitioning a graph of sequences, structures and abstracts for information retrieval. In *Proceedings of the Twelfth Text REtreival Conference (TREC 2003).*

[Fang et al. 2004] Fang, H., Tao, T., & Zhai, C. (2004). A formal study of information retrieval heuristics. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 49–56).

[Fujita 2004] Fujita, S. (2004). Revisiting again document length hypotheses TREC-2004 genomics track experiments at Patolis. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004).*

[Harman 1991] Harman, D. (1991). How effective is suffixing? *Journal of the American Society for Information Science, 42*(1), 7–15.

[Hersh et al. 2003] Hersh, W. R., & Bhuptiraju, R. T. (2003) TREC genomics track overview. In *Proceedings of the Twelvth Text REtrieval Conference (TREC 2003).*

[Hersh et al. 2004] Hersh, W. R., Bhuptiraju, R. T., Ross, L., Johnson, P., Cohen, A. M., & Kraemer, D. F. (2004) TREC 2004 genomics track overview. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004).*

[Hersh et al. 2005] Hersh, W., Cohen, A., Yang, J., Bhuptiraju, R. T., Roberts, P., & Hearst, M. (2005) TREC 2005 genomics track overview. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005).*

[Huang et al. 2005] Huang, X., Zhong, M., & Si, L. (2005). York University at TREC 2005: genomics track. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC 2005).*

[Lafferty and Zhai 2001] Lafferty, J., & Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 111–119).

[Lovins 1968] Lovins, J. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics, 11*, 22–31.

[Pirkola and Leppanen 2003] Pirkola, A., & Leppanen, E. (2003) TREC 2003 genomics track experiments at UTA. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003).*

[Porter 1980] Porter, M. F. (1997). An algorithm for suffix stripping. *Program, 14*(3), 130–137.

[Savoy et al. 2003] Savoy, J., Rasolofo, Y., & Perret, L. (2003). Report on the TREC 2003 experiment: genomic and web searches. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003).*

[Song et al. 2003] Song, Y-I., Han, K-S., Seo, H-C., Kim, S-B., & Rim, H-C. (2003). Biomedical text retrieval system at Korea University. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003).*

[Tomlinson 2003] Tomlinson, S. (2003). Robust, web and genomics retrieval with Hummingbird SearchServer at TREC 2003. In *Poceedings of the Twelfth Text REtrieval Conference (TREC 2003).*

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

[Zhai and Lafferty 2001b] Zhai, C., & Lafferty, J. (2001). Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Management* (pp. 403–410).