Albert-Ludwigs-Universität

Freiburg

Master Project

Wikification

Author:
Ragavan Natarajan

Supervisor:
Prof. Dr. Hannah Bast

This report is submitted in partiful fulfillment for the master project

at the

Chair of Algorithms and Data Structures
Department of Computer Science

June 2013

Abstract

This report provides an insight into the lessons learned, challenges faced, and the solutions created during the implementation of a wikification engine. It takes a detailed look at the task of extracting the knowledge-base from the Wikipedia database dump which forms a crucial part of the process. Additionally, it also discusses, in enough detail, the part that does the actual wikification, and offers insight into the results and concludes with a note on the future work.

Contents

Chapter 1

Introduction

This chapter introduces the term ***wikification*** and provides a broad classification of the stages involved in the wikification of a document.

1.1 Wikification: definition

Wikification is the process of identifying the important phrases in a document and linking each of them to appropriate articles on *Wikipedia* based on their context of occurrences. The important phrases in a document are also called ***keyphrases***, somewhat similar to the term *keyword*, but unlike a keyword , a keyphrase can consist of one or more words.

1.2 The two stages

The entire task can be broadly classified into the following two stages.

1. **Knowledge-Base Preparation:**

   In this stage, a knowledge-base is prepared from the Wikipedia database dump file which is later used by the engine, for the wikification of a document.

2. **Wikification:**

   In this stage, keyphrases in the input document are recognized and are linked to appropriate articles on Wikipedia based on their context of occurrences.

The content of this report is organized as follows. Chapter 2 addresses the knowledge-base creation process in detail, and chapter 3 elaborates the process of wikification. The quality

metrics are stated along with an insight into the result in Chapter 4. Chapter 5 discusses the future work in this topic, including several ways of improving the quality of the wikifier and Chapter 6 concludes the report.

Chapter 2

Knowledge-Base Creation

A knowledge-base is necessary to help the *wikifier*[1] make entity-linking decisions. This chapter addresses the task of creating a knowledge-base, covering everything from obtaining the Wikipedia database dump file to parsing it and extracting the data from it which constitutes the knowledge-base, and also the challenges involved in it and ways to increase the size of the extracted keyphrases vocabulary.

Once created, a knowledge-base could be considered static, i.e., it needs to be recreated only when there is a significant amount of change in the number of articles in *Wikipedia*. Moreover, since the updated Wikipedia database dump files are made available once per month, an available knowledge-base has a lifetime of at least one month, after which, if required, it may be recreated.

The importance of a knowledge-base cannot be understated as the working of a wikifier depends entirely upon the correctness of the data in it. Since it was understood that Wikipedia articles are unsuitable for functioning as knowledge-base for *high-recall*[2] entity-linking tasks because of the sparse annotation of *name-mentions* (hereinafter referred by the word *keyphrases*) in them [1], special efforts have been made to extract keyphrases from unannotated entities too, in an attempt to overcome this drawback. Doing so has helped identify ca. 10.5 million keyphrases from the database dump.

The wikifier imposes several restrictions upon what constitutes a keyphrase. Doing so, guarantees uniformity in the process of identifying keyphrases in an input document, as well as while extracting them from the Wikipedia database dump, for knowledge-base creation.

[1] A program that does wikification.
[2] *Recall* is a metric explained later.

2.1 Obtaining the Wikipedia database dump

The Wikipedia database dump file containing the latest pages and articles could be downloaded from http://download.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2. It contains only the current revisions and no talk or user pages − but it is sufficient. The size of the database dump as on April 2013 is approximately 9.06GB compressed, 42GB uncompressed.

The Perl module *MediaWiki::DumpFile::FastPages*, available in CPAN[3], has been used to parse the uncompressed database dump. It enables simultaneous iteration over the title and content of each page in the database dump in a sequential manner.

2.2 Types of pages in Wikipedia

Three different types of pages of interest could be identified based on the page-title of a page in the Wikipedia database dump. They are mentioned below.

1. **Disambiguation pages:**

   The content of one such page on Wikipedia contains one ore more links to existent articles or other disambiguation pages. Such a page could be identified by the presence of the phrase *(disambiguation)* in its title text. For example, if the phrase *Austin (disambiguation)* is the title text of a page, then its content would be a list of one or more links as mentioned above.

2. **Redirect pages:**

   The content of one such page contains a single link that causes Wikipedia to redirect to a page (hereinafter referred to as the *target page*) referred by that link. It is possible that the target page in the dump file is also of type *redirect page*. Wikipedia uses the term *double redirects* to denote such pages. In order to prevent infinite loops, Wikipedia stops the chain after the first redirect[2]. Hence, at the time of knowledge-base creation double redirects are ignored.

3. **Articles:**

   A page of this type is an article written about the topic contained in its title-text. This is what an user gets to see when they search for an existent article on Wikipedia. For example, following the URL https://en.wikipedia.org/wiki/Chennai, takes to a page on Wikipedia, which is of type *article*.

[3] http://www.cpan.org/

2.3 Free Links in Wikipedia

In Wikipedia, free links are used to produce internal links between pages, which enable users to access information related to the article they are reading. They are created by enclosing a page-title in double square brackets. Thus, enclosing the word *Chennai* in double square brackets, like this, [[*Chennai*]], will cause the text to be anchored with a hyperlink to the respective Wikipedia article, http://en.wikipedia.org/wiki/Chennai, in this case. Optionally a vertical bar '|' could be added to customize the anchor text. For example, typing [[*Chennai* | *the capital city of Tamilnadu*]] would result in the text *the capital city of Tamilnadu* being anchored to the respective Wikipedia article, which is, in this case, the aforementioned article.

2.4 Format of the data

The Wikipedia database dump is an XML file containing both metadata and data. As mentioned earlier, the Perl module *MediaWiki::DumpFile::FastPages*, available in CPAN, is used to process the database dump. Pages are written in **Wikitext language**[4], a lightweight markup language for writing Wikipedia pages.

2.5 Keyphrase extraction

Keyphrase extraction is one of the major tasks involved in the knowledge-base creation. This section begins by defining certain rules on what constitutes a keyphrase. As mentioned before, a keyphrase can be composed of one or more words, but for reasons mentioned later in this section further restrictions are imposed on keyphrases.

2.5.1 Composition of a keyphrase

The following set of rules restrict what a keyphrase can be composed of.

1. A keyphrase can be composed of only alphanumeric characters. However, it is not just limited to alphanumeric characters in the ASCII representation.
2. Every non-alphanumeric character found in the keyphrase should be replaced with a single whitespace character.
3. It cannot contain punctuation of any form.

[4] http://en.wikipedia.org/wiki/Wikitext

4. All the letters should be case-folded to lowercase.
5. In case of the phrase containing multiple words, the words are to be separated by a single whitespace character.
6. The number of words in a keyphrase is limited to a maximum of 10.

These restrictions on the keyphrase offer the following benefits:

1. The set of target entities of two keyphrases, which are essentially the same but are under different case foldings could then be combined. Consider two keyphrases *austin* and *Austin* each having their own set of referred entities. Let the referred entities of the keyphrase *austin* be *Austin Island* and *Austin College*, and let that of *Austin* be *Austin College* and *Austin Motor Company*.

   Applying the rules, the sets of entities of the two keyphrases could then be merged with the case-folded *austin* acting as the keyphrase with referred entities being *Austin Island*, *Austin College* and *Austin Motor Company*.

2. Eliminating the non-alphanumeric characters in the keyphrases provides an uniform way of recognizing the keyphrases in an input document at the time of wikification. Consider, for example, the following excerpt in an input document.

   ... Alaska's residents grapple with changing climate ...

   The rules for keyphrase extraction when applied to the input document case-folds it to lowercase and replaces each of the non-alphanumeric characters in it with a single whitespace, causing the above excerpt to become as follows.

   ... alaska s residents grapple with changing climate ...

   This makes it possible to recognize the word *alaska* in the document if such a keyphrase existed in the keyphrase vocabulary. Imagine, without these rules, the entire word would be *Alaska's* which may not contain a matching keyphrase in the keyphrase vocabulary of the knowledge-base.

2.5.2 Extracting from articles

An *article* is a special type of Wikipedia page as mentioned earlier. Keyphrases are extracted from the *free links* of articles, described in section 2.3.

- For free links without a vertical bar, i.e. without '|', the enclosed text would act both as the keyphrase and the *target entity*[5]. For example, for the free link [[*Chennai*]], the extracted keyphrase and target entity would be the following.

  chennai ⟶ *Chennai*

- For free links with a vertical bar, the text before the bar, i.e. the entity text, acts as the target entity whereas the text that follows it acts as the keyphrase. In addition to that, the entity text is also used as a keyphrase, just like it was used in a free link without vertical bar. For example, for the free link [[*Chennai*|*the capital city of Tamilnadu*]], the following *keyphrase - target entity* pairs are extracted.

  the capital city of tamilnadu ⟶ *Chennai*
  chennai ⟶ *Chennai*

  Doing so, helps improve the keyphrase vocabulary, which in turn could increase the possibility of important phrases in a document being recognized at the time of wikification.

2.5.3 Extracting from page titles

Limiting the keyphrase extraction to free links in articles would severely limit the keyphrase vocabulary leaving several non-linked article titles orphan. In other words, if there was an article on Wikipedia, not linked from any other article, then it will be left unlinked if the keyphrase extraction procedure is limited to extracting keyphrases from the body of the articles, as done above.

2.5.3.1 Extracting from article titles

As the parser iterates the article titles, keyphrases are extracted from the titles and the titles are added to the set of target entities of those keyphrases. Any information enclosed in parenthesis is discarded. For example, if a page title is *Casablanca (film)*, then the following *keyphrase – target entity* pair is extracted from it.

casablanca ⟶ *Casablanca (film)*

If another page title, such as *Casablanca (volcano)*, is encountered, then another *keyphrase – target entity* pair is generated with the same keyphrase, i.e. *casablanca* causing it to have two different target entities.

casablanca ⟶ *Casablanca (film)*, *Casablanca (volcano)*

[5] A *target entity* is one of the several articles on Wikipedia that a keyphrase links to.

It is necessary to understand why the text within parentheses is discarded. If it were not, then the generated keyphrase would, for example, be *casablanca film*. It is very unlikely that an input document has this exact word sequence. By discarding the text within parentheses, the chances of a phrase in an input document being found in the keyphrase vocabulary increases. Since there are algorithms to disambiguate entities in the later stages of wikification, this does not cause any problem. Even if the word sequence *casablanca film* appeared in an input document, the wikifier would still be able to identify the phrase *casablanca* in it.

2.5.3.2 Extracting from redirect titles

The keyphrase obtained from the title of a **redirect page** could not use the title as a target entity, since the title redirects to another page on Wikipedia. Therefore, the keyphrase is assigned to the title in the body of the redirect page (hereinafter referred to as the *redirect target*). Additionally, a keyphrase is extracted from the redirect target and assigned to it. For example, the title *Air Transport* on Wikipedia redirects to the title *Aviation*. In this case, the following *keyphrase − target entity* pairs are extracted.

air transport ⟶ *Aviation*
aviation ⟶ *Aviation*

However, in order to avoid chain-redirects, if a *redirect target* redirects to another page, no *keyphrase − target entity* pairs are obtained.

2.5.3.3 Extraction from disambiguation titles

A keyphrase obtained from the title of a disambiguation page has, as its target, each of the articles available in the page. For example, the title *Casablanca (disambiguation)* is that of a *disambiguation page* on Wikipedia, containing the following entities.

[[*Casablanca (volcano)*]]
[[*Casablanca Records*]]
[[*Casablanca (film)*]]

As done before, the keyphrase *casablanca* is extracted from the disambiguation title, and each of the entities in the disambiguation page is assigned to be target of this keyphrase, as shown below.

casablanca ⟶ *Casablanca (volcano)*, *Casablanca Records*, *Casablanca (film)*

2.5.3.4 Nested disambiguations

It is possible for an entry in a Wikipedia disambiguation page to link to another disambiguation page. For example, the page *Austin (disambiguation)* on Wikipedia, contains the following entry in its content, which is the title of another disambiguation page.

[[*Austin Station (disambiguation)*]]

A disambiguation entity cannot be assigned as the target of a keyphrase. Therefore, the page of the nested disambiguation entity is fetched and all the non-disambiguation entities in its content are assigned to this keyphrase. For example, if the page *Austin Station (disambiguation)* contained the following content,

[[*Austin (Amtrak station)*]]
[[*Austin (CTA Blue Line)*]]
[[*Austin (CTA Green Line)*]]

all of these entities are assigned to be the target of the keyphrase *austin*, as shown below.

austin ⟶ *Austin (Amtrak station)*, *Austin (CTA Blue Line)*, *Austin (CTA Green Line)*

In addition to that, obviously, as done before, the keyphrase keyphrase *austin station* also gets all these target entities.

austin station ⟶ *Austin (Amtrak station)*, *Austin (CTA Blue Line)*, *Austin (CTA Green Line)*

2.5.3.5 Chained disambiguations

It is possible for a nested disambiguation page to further contain links to disambiguation pages. Such entries are ignored to avoid having to go down a possibly infinite chain.

2.6 Additional extracted information

In addition to extracting the keyphrases, several other information are extracted from the database dump. A list of all articles titles and their contents are extracted. Similar to what was done during the keyphrase extraction, all non-alphanumeric characters are removed from the article contents at the time of extraction. The content of an article is used to compute its similarity with an input document, as discussed in the later chapters.

Moreover, for each article, a set of **incoming links** is also maintained. The incoming links of an article is the set of all articles in Wikipedia that have free-links to this article. This information is used to compute the semantic relatedness between two articles on Wikipedia.

2.7 Storage and retrieval

The amount of data in the knowledge-base is summarized below.

- ⋆ 7 million articles
- ⋆ 10.5 million distinct keyphrases
- ⋆ 33 million distinct words

Each of the 7mn articles have a title and content and the total size of the contents is approximately 27GB uncompressed. Each of the 10.5mn keyphrases have at least one target entity and every word in the vocabulary has its *inverse document frequency* score.

All this information are required at the time of wikification, and therefore, it is important to design a strategy for its efficient retrieval. The content of an article, as mentioned earlier, is necessary to compute its similarity with the input document. However, storing, for example, 27GB of data in the main memory, is avoided for the following reasons.

1. Reading such huge amount of data from secondary storage into main memory during the start of wikification engine, is still a time-consuming operation.
2. Only a very small proportion of the data that resides in the memory is required at any time by the wikification engine. Therefore, the rest of the data occupies precious space in the main memory.
3. On a server where several applications run simultaneously such a wastage of main-memory adversely impacts the server's scalability.

2.7.1 Indexing the knowledge base

This section addresses, how the aforementioned problem could be solved, by indexing the text files. When a knowledge base is created, each distinct component of it is stored one per line. In other words, the keyphrase vocabulary is stored in a file with each line containing one keyphrase. Pairs of data are stored in separate files but in the same line number in both files. Consider, for example, the following keyphrase and its set of target entities.

austin station ⟶ *Austin (Amtrak station)*, *Austin (CTA Blue Line)*

If the keyphrase *austin station* is stored at, say, line number 10 of the file *keyphrases.txt*, then its target entities *Austin (Amtrak station)* and *Austin (CTA Blue Line)* are stored in the same

line number, i.e. line number 10 of the file, say, *entities.txt*. Doing so, enables the entities of a keyphrase to be accessed from the file based on the line number at which the keyphrase occurs. The same holds for article titles and their respective contents.

Now, the problem of accessing a random line in a text file needs to be addressed. Issuing an UNIX command such as `sed -n <line>p <file>` takes forever to fetch a line near the end of the file for very large files, as the complexity of this approach is $\mathcal{O}(s)$, where s is the size of the file in bytes (The command has to skip that many *newline* characters before the desired line number is reached).

2.7.1.1 Indexing the lines in the file

File systems are capable of randomly addressing data very efficiently (usually in logarithmic time, with a large base), by using disk-based data structures, such as, for example, B^*- *trees*. The idea presented here, exploits both the the static nature of the knowledge-base and the ability of the file system to read the data at a random line-number from a file very efficiently.

For each line in the input file, an index is created, which contains the byte location at which the line starts. For example, the index for the first line in the file would be 0. In this fashion, for a file having N number of lines, N number of indices are created and stored in another file, namely, the index file.

Given the static nature of the information in the knowledge-base, the index seldom needs to be updated. Now any line in the file could be read efficiently with the help of the index. In Java, the class *java.io.RandomAccessFile* has a *seek*(long) method to randomly address a location in the file, specified in its argument.

The size of the index file of the article contents is ca. 80MB, which is only 0.28% of the size of the article contents file, and therefore, could be easily stored in the main memory, in an array, for example. On a machine running Core i7 QM-3720 with Sata 6 SSD as the secondary storage, reading 1000 articles randomly from the disk using this approach took about 0.5ms, with an average size of about 4KB per article.

Chapter 3

Wikification

Wikification it involves the identification of *keyphrases* in the input document and linking them to appropriate articles on Wikipedia. The wikification engine begins by reading the knowledge base into main memory and waits for a document at its input. Before an overview of the wikification procedure is presented, the following section discusses how the keyphrases in the input document and their candidate entities are ranked.

3.1 Ranking keyphrases

The **keyphraseness**[3] of a phrase, measures the significance of the phrase in any document. This is the approach currently followed by the wikifier for ranking the keyphrases, whereas, Chapter 5 presents some more ideas. The keyphrases found in the input document are ranked in decreasing order of their keyphraseness scores. The keyphraseness of a phrase p is defined as follows.

$$\text{keyphraseness}(p) = \frac{|p_{\text{link}}|}{|\text{DF}(p)|}$$

Where, $|p_{\text{link}}|$ is the number of articles in Wikipedia in which the phrase p appears as a link, and $\text{DF}(p)$ is the document frequency[1] of p. It holds that $|p_{\text{link}}| \leq \text{DF}(p)$, and hence,

$$0 \leq \text{keyphraseness}(p) \leq 1$$

A phrase with keyphraseness of 1 would mean that it is linked wherever it appears, and hence, must be a significant phrase, whereas a phrase with lower keyphraseness would mean that it is seldom linked, and hence, is a keyphrase of less significance. Therefore, the phrases in the input document are sorted in descending order of their respective keyphraseness scores.

[1] Appendix B contains the definition of *document frequency*

3.2 Ranking entities

A keyphrase can refer to several target entities. The entities are ranked based on their **local compatibility** scores. The compatibility between a keyphrase k and an entity e, denoted by $\text{CP}(k, e)$ is defined as follows.

$$\text{CP}(k, e) = \frac{\vec{m} \cdot \vec{e}}{|\vec{m}||\vec{e}|}$$

Where, $\vec{m}$ is a vector containing the $tf \times idf$ scores[2] of the words in the *local context* of the keyphrase, $\vec{e}$ is a vector of $tf \times idf$ scores of the words in the entity, and $|\vec{m}|$ and $|\vec{e}|$ denote the normalization of the vectors $\vec{m}$ and $\vec{e}$, respectively.

3.3 Overview of the wikification procedure

For every input document, the wikifier goes through the following procedure.

1. The input document is case folded to all lowercase.
2. All non-alphanumeric characters in the input document are replaced with blank space.
3. The wikifier computes n-grams[3] of up to 10 words in the modified document.
4. All the n-grams, not present in the set of keyphrases in the knowledge-base, are discarded.
5. The remaining n-grams (keyphrases) are ranked based on their *keyphraseness* scores, and only the top-ranking N number of keyphrases are retained.
6. The *local context* of each of the retained keyphrases is obtained. The local context of a keyphrase is the words surrounding it for a predetermined window size. The window size is set to 50 according to the results published in Pedersen et al.[4]
7. For each of the keyphrases, its set of target entities is fetched from the knowledge-base and the *local compatibility* between each of article's content and the keyphrases' local context is computed.
8. The entity that has the maximum cosine similarity with the local context of the keyphrase wins.
9. The keyphrases are then anchored to their corresponding winning articles on Wikipedia □.

[2] $tf \times idf$ score is explained in Appendix B
[3] Appendix A discusses n-grams in detail

Chapter 4

Results

This chapter begins by introducing the metrics for evaluating the result of wikificaiton. A sample document is wikified and the result is discussed in detail. Later, in this chapter, a generalization of the result for different categories of input documents, of different sizes each, is provided.

4.1 Precision and Recall

Precision and **recall** are widespread measures for evaluating the quality of the result. Let K_r be the number of keyphrases linked correctly, K_i be the number of irrelevant identified phrases, and let K_u be the number of significant but unidentified phrases in the result. Then the precision and recall are defined as follows.

$$\text{Precision, } \mathcal{P} = \frac{K_r}{K_r + K_i}$$

and,

$$\text{Recall, } \mathcal{R} = \frac{K_r}{K_r + K_u}$$

Both precision and recall are measured in percentage. The quality of a wikifier could be consider perfect if both P and R are 100%. In practice, however, attempt to increase either of the two may adversely affect the other. For example, $\mathcal{R}$ could be increased by attempting to recognize many number of phrases in the input document. This may increase the possibility of all relevant and significant phrases in the document being identified and linked, thereby increasing $\mathcal{R}$, but it may also inadvertently increase the amount of irrelevant phrases being identified and linked, causing $\mathcal{P}$ to plummet. Therefore, the aim should be to strike a balance between these two quality metrics.

4.2 Sample document and result

In this section, the result of wikification of a sample document, is presented, and its quality, based on the aforementioned metrics, is discussed. In the first part, the wikifier was set to identify and link the top 15 keyphrases in the sample document, and in the second part, top 30 keyphrases.

Significant phrases that are *identified and linked correctly* are highlighted in green, and both insignificant phrases and significant but incorrectly linked phrases are highlighted in red. Significant phrases that are left unrecognized are highlighted in blue and the color yellow marks a keyphrase that was identified and linked correctly, but that was not necessary for the document.

Note: Significant phrases in the document for which no relevant articles exist in Wikipedia at all, are not considered to be unrecognized, for obvious reasons.

4.2.1 Result: Top 15 keyphrases

In this run, the wikifier was set to link only the top 15 keyphrases.

> Delhi Police has said it arrested Sreesanth, Ajit Chandila and Ankeet Chavan - all Rajasthan Royals bowlers - for the alleged fulfilling of promises made to bookmakers during this year's IPL. Neeraj Kumar, the commissioner of Delhi Police, provided a detailed explanation of its investigation and the charges against the players but said it had no evidence to suggest any other player, administrator or team owner was involved. It has registered cases under Indian Penal Code section 420 and 120B, which deal with fraud, cheating and criminal conspiracy.
>
> The police has identified the three matches where the alleged spot-fixing happened: against Pune Warriors on May 6, Kings XI Punjab on May 9 and Mumbai Indians on May 15. Kumar said the deal was for the bowlers to concede a specified minimum number of runs in a pre-decided over. He explained in detail how the deals were struck, how the players allegedly indicated to the bookmakers that the deal was on, and how they went on to concede those number of runs. He said the police has recordings of those tapped phone conversations.

4.2.1.1 Analysis of the result

For the above result, $K_r = 13$, $K_i = 1$, and $K_u = 1$.

$$\text{Precision, } \mathcal{P} = \frac{13}{13+1} = 92.8\% \quad ; \quad \text{Recall, } \mathcal{R} = \frac{13}{13+1} = 92.8\%$$

The quality metric suggests that the wikifier has done a fairly reasonable job of identifying and correctly linking the significant phrases in the document (high *precision*), and that it has not left too many significant phrases unrecognized (high *recall*).

Of the keyphrases in the result, marked in green, out of twelve, five had more than one contending entity to choose from. The wikifier has done a good job of linking to the correct entity out of the several contending entities. For example, the keyphrase *IPL* had 21 contending entities to choose from, some of which are listed below.

Indian Premier League
Information Processing Language
Intense pulsed light
IBM Public License
Indianapolis Power & Light
Inner plexiform layer
Iran Pro League
...

Out of the contention, the wikifier has correctly picked *Indian Premier League* to be the target entity for the keyphrase *IPL*, based on its context of occurrence.

4.2.2 Result: Top 25 keyphrases

In this run, the wikifier was set to link the top 25 keyphrases.

Delhi Police has said it arrested Sreesanth, Ajit Chandila and Ankeet Chavan - all Rajasthan Royals bowlers - for the alleged fulfilling of promises made to bookmakers during this year's IPL. Neeraj Kumar, the commissioner of Delhi Police, provided a detailed explanation of its investigation and the charges against the players but said it had no evidence to suggest any other player, administrator or team owner was involved. It has registered cases under Indian Penal Code section 420 and 120B, which deal with fraud, cheating and criminal conspiracy.

The police has identified the three matches where the alleged spot-fixing happened: against Pune Warriors on May 6, Kings XI Punjab on May 9 and Mumbai Indians on May 15. Kumar said the deal was for the bowlers to concede a specified minimum number of runs in a pre-decided over. He explained in detail how the deals were struck, how the players allegedly indicated to the bookmakers that the deal was on, and how they went on to concede those number of runs. He said the police has recordings of those tapped phone conversations.

4.2.2.1 Analysis of the result

For the above result, $K_r = 14$, $K_i = 10$, and $K_u = 0$.

$$\text{Precision, } \mathcal{P} = \frac{14}{14+10} = 58.3\% \quad ; \quad \text{Recall, } \mathcal{R} = \frac{14}{14+0} = 100\%$$

As mentioned earlier, increasing the number of keyphrases to be recognized in the input, has caused $\mathcal{R}$ to rise and $\mathcal{P}$ to plummet. The quality metric now suggests that, while the wikifier has not left too many significant phrases unrecognized (high *recall*), it has identified several insignificant phrases in the document (moderate *precision*) too.

4.2.3 Results summary

Table 4.1 provides a summary of the results of wikification, for input documents belonging to different categories, randomly chosen from the Internet. In two runs, the wikifier was set to identify the top 15 and top 30 keyphrases.

TABLE 4.1: Summary of the results of wikification

Category	Words	N=15					N=25				
		K_r	K_i	K_u	$\mathcal{P}(\%)$	$\mathcal{R}(\%)$	K_r	K_i	K_u	$\mathcal{P}(\%)$	$\mathcal{R}(\%)$
Nat. geo.	205	6	6	5	50	54.5	8	12	2	40	80
Sports (Cricket)	235	13	1	2	92.8	86.6	15	8	0	65.2	100
Stock Market	273	11	2	4	84.6	73.3	12	7	3	63.1	80
Politics	297	9	4	2	69.2	81.8	9	10	2	47.3	81.8
Sports (Football)	319	7	6	3	53.8	70	8	10	2	44.4	80
Total	1329	46	19	16	**70.7**	**74.1**	52	47	9	**52.5**	**85.2**

Each of the text was chosen arbitrarily from different news channels. From the table, it is clear that increasing the number of keyphrases to be recognized in the input document, makes *recall* better, while decreasing the *precision*.

However, it is possible to increase both recall and precision by trying to avoid recognizing insignificant phrases. Chapter 5 discusses few techniques to address this problem.

4.3 Memory requirement

The wikifier occupies a relatively less amount of space in the main memory. The present version, implemented in Java, takes less than 3GB in the RAM, since a major part of the knowledge-base is kept in, and accessed from, the secondary storage. The wikifier needs 30GB

of secondary storage space for storing the knowledge-base, a whopping 90% of which contains the article contents.

The present version of the wikifier, takes, on average, a little over 100ms for the wikification of a document of about 500 words, in a laptop running Core i7-3720QM, 16GB RAM, SATA 6 SSD. However, it is expected to considerably become faster in the future versions, given the number of optimizations planned for, some of which are discussed in the next chapter.

Chapter 5

Future Work

Even though the wikifier did a reasonable job of identifying keyphrases and linking them to appropriate entities, it was seen how the *precision* of the wikifier got affected every time it was set to recognize more keyphrases for achieving better *recall*. The knowledge base has a wealth of keyphrases for recognizing many, if not, all of the significant phrases in an input document. However, the keyphrase ranking method had to rank those phrases correctly, in order not to recognize insignificant keyphrases. From the way the precision plummeted every time, it is not hard to notice that the phrase ranking method is not perfect. The following section discusses some different approaches to ranking the phrases.

5.1 Ranking the phrases

This section provides an alternate approach to ranking the phrases. A phrase having more number of words is ranked higher than a phrase with less number of words. For example, if a document contained the phrase *National Stock Exchange*, and if two phrases could be recognized in the knowledge base, namely, *national stock exchange* and *stock exchange*, then it sounds reasonable to rank *national stock exchange* higher, even if it had lower keyphraseness score. A sample document, wikified by ranking the phrases in this method, is shown below. The color codes have the same meaning as in the last chapter. The wikifier was run to identify only the **top 15** keyphrases. This apporoach to ranking keyphrases, is then compared with the conventional approach of ranking using the keyphraseness feature.

5.1.1 Comparison of the ranking approaches

5.1.1.1 Ranking by word-count

Rising for the third straight session, the BSE benchmark Sensex on Thursday rose by over 34 points to 20,247.33 to hit a fresh 28-month high on buying mainly in realty, healthcare, oil and gas and banking stocks, amid strong foreign capital inflows. Brokers said the market was bullish ever since reports of easing inflation have raised hopes that RBI will go for rate cuts to boost economic growth. In the 30-BSE index components, 14 stocks gained led by Hindalco, Cipla, RIL, Dr. Reddys, Sterlite Industries and SBI.

$$\mathcal{P} = 66\% \quad ; \quad \mathcal{R} = 22\%$$

5.1.1.2 Ranking by keyphraseness

Rising for the third straight session, the BSE benchmark Sensex on Thursday rose by over 34 points to 20,247.33 to hit a fresh 28-month high on buying mainly in realty, healthcare, oil and gas and banking stocks, amid strong foreign capital inflows. Brokers said the market was bullish ever since reports of easing inflation have raised hopes that RBI will go for rate cuts to boost economic growth. In the 30-BSE index components, 14 stocks gained led by Hindalco, Cipla, RIL, Dr. Reddys, Sterlite Industries and SBI.

$$\mathcal{P} = 87.5\% \quad ; \quad \mathcal{R} = 77.8\%$$

5.1.1.3 Discussion of the result

It is clear, that while some of the significant phrases are identified by means of ranking the keyphrases by word-count, it has also identified and linked many **insignificant** phrases. Some phrases of interest, such as, RBI, BSE and so on, were also not considered by the first approach, for being short. On the other hand, while ranking by means of *keyphraseness* has helped recognize the short keyphrases, it has left some of the keyphrases, recognized by the former method, unrecognized. Combining these two approaches appears to be a good solution to increasing the precision and recall. In this example, combining these two approaches yield a precision of 80% and recall of 88.8%. However, experimental study of this approach of combining both the keyphraseness score and the length of a keyphrase for ranking needs to be made.

Additionally, measures are to be taken to leave the undesired phrases (those that were marked in yellow) unrecognized. It is difficult to say what are undesired phrases as there is always a subjective element involved in it. A method is to be found, so that the phrases of less importance do not mask significant phrases while ranking.

5.2 Disambiguating the entities

At present, the entities are disambiguated based on a similarity score computed with the input document. However, much better ideas are currently available in the literature. Han et al.[1] provide a graph-based **Collective Entity Linking** method which models the global interdependence between different entity linking decisions, minimizing the possibility of a keyphrase being linked to the wrong entity. Future versions of the wikifier will be modified to use this approach.

5.3 Possible code-level improvements

Some code optimizations could be made to possibly improve the performance of the present tool. Even though, at present, the wikifier consumes only a little less than 3GB of main memory, a major part of it is consumed by storing the keyphrases in a Java HashMap, whose values contain the indices of their target entities. Given the static nature of the knowledge-base, the keyphrases could be stored in a *perfect hash table* causing the methods **get**(...) and **containsKey**(...) to run in time $\mathcal{O}(1)$, with a far smaller constant-factor than before.

Also, since *java.util.HashMap* stores native types as objects, more space is consumed in the main memory than what is possibly required. So a *perfect hashmap*, capable of storing native types as they are, could be developed. Since the keyphrases themselves are not too long, and also since most of them have common prefixes, another way of of efficiently storing the keyphrases would be a **compressed trie** data structure. Even though it is theoretically a better choice, the practical side of it needs to be witnessed for an implementation written in Java.

Chapter 6

Conclusion

This report presented the task of wikification in two stages, wherein the first stage elaborated the task of extracting the knowledge-base from the wikipedia database dump, and the second stage addressed the task of keyphrase identification and entity disambiguation. It was discussed, that by employing certain approaches an extensive keyphrase vocabulary could be constructed overcoming the problem of poor-recall mentioned in Han et al.[1] In the later chapters, keyphrase identification and disambiguation algorithms were presented and the quality-metrics and results were discussed. Finally, in the chapter on future work, several ideas to improve quality of the wikifier were also presented.

Appendix A

Computing the n-grams

In was mentioned earlier that n-grams of up to 10 words are computed from the input document, and those that are not present in the keyphrase vocabulary are discarded. An n-gram of words is a contiguous sequence of n words in a given text. For example, for the text *The quick brown fox jumps over the lazy dog*, all its 4-grams are listed below.

The quick brown fox
quick brown fox jumps
brown fox jumps over
fox jumps over the
jumps over the lazy
over the lazy dog

The total number of n-grams in a text with M number of words, is $M - n + 1$. A 1-gram of a given text, is simply the list of all words in the text. For the input document, n-grams for each value of n from 1 to 10, are computed.

Appendix B

$tf \times idf$ score

The **term frequency** $\times$ **inverse document frequency** score is a measure of the importance of a word for a given document in a collection.

Term Frequency. The term frequency of a word w in a document d is simply the number of times w occurrs in d.

Document Frequency. Given a collection C containing a set of documents, the document frequency of a word w in C, written as $\mathrm{DF}(w, C)$, is the total number of documents in C, in which w occurs. Mathematically,

$$\mathrm{DF}(w, C) = |\{d | d \in C \wedge w \in d\}|$$

Inverse Document Frequency. The inverse document frequency of a word w in a collection C, is defined as,

$$\mathrm{IDF}(w, C) = \log_2 \left(\frac{|C|}{\mathrm{DF}(w, C)} \right)$$

The $tf \times idf$ score of a word, is the product of its term frequency and inverse document frequency.

Bibliography

[1] Xianpei Han, Le Sun, and Jun Zhao. Collective entity linking in web text: a graph-based method. pages 765–774, 2011. doi: 10.1145/2009916.2010019. URL http://doi.acm.org/10.1145/2009916.2010019.

[2] Wikipedia:double redirects. URL http://en.wikipedia.org/wiki/Wikipedia:Double_redirects.

[3] Rada Mihalcea and Andras Csomai. Wikify!: linking documents to encyclopedic knowledge. pages 233–242, 2007. doi: 10.1145/1321440.1321475. URL http://doi.acm.org/10.1145/1321440.1321475.

[4] Ted Pedersen, Amruta Purandare, and Anagha Kulkarni. Name discrimination by clustering similar contexts. pages 226–237, 2005. doi: 10.1007/978-3-540-30586-6_24. URL http://dx.doi.org/10.1007/978-3-540-30586-6_24.