Comparing Different Text Similarity Methods

JunPeng Bao
baojp@mail.xjtu.edu.cn
Department of Computer Science & Technology, Xi'an Jiaotong University,
Xi'an 710049, P.R. China

Caroline Lyon
c.m.lyon@herts.ac.uk

Peter C. R. Lane
peter.lane@bcs.org.uk

Wei Ji
w.1.ji@herts.ac.uk

James A. Malcolm
j.a.malcolm@herts.ac.uk
School of Computer Science, University of Hertfordshire, Hatfield, AL10 9AB, UK

Abstract

This paper reports experiments on a corpus of news articles from the Financial Times, comparing different text similarity models. First the Ferret system using a method based solely on lexical similarities is used, then methods based on semantic similarities are investigated. Different feature string selection criteria are used, for instance with and without synonyms obtained from WordNet, or with noun phrases extracted for comparison. The results indicate that synonyms rather than lexical strings are important for finding similar texts. Hypernyms and noun phrases also contribute to the identification of text similarity, though they are not better than synonyms. However, precision is a problem for the semantic similarity methods because too many irrelevant texts are retrieved.

1 Preamble

The work described in this report originated partly in the UK, partly in China. All the texts used are in English, but we show how some language processing methods can be applied in a similar way to these very different languages, whilst other methods do not transfer.

The task addressed in these experiments is to take the description of a topic and find documents that are relevant to it out of a large set.

The first experiments used the well tried Ferret system to find relevant documents. This method, originally developed to detect plagiarism, is based on lexical similarity between short strings of words: when two documents have a proportion of matching strings above a threshold they are similar. Ferret is very effective in finding plagiarism or copied passages in English and Chinese, but this approach does not turn out to be useful in finding semantically similar texts written independently. However, it is an example of a language processing technology that can be transferred from English to Chinese. In both cases the method is based on processing consecutive tokens; in English these tokens are words, in Chinese they are characters or strings of characters [1, 2].

The main experiments described in this report use different methods to identify the semantic content of a text and then compare it with other texts. The concept originated in considering Chinese language, where written words are composed of one, two or more characters. A passage

暴力抗议已导致国家的**司法**系统瘫痪。两名**法官**上周已辞职。

Violent protests have brought the country’s **judicial** system to a standstill. Two **judges** resigned last week.

香港及其它亚洲市场的**法规制定者**被督促收紧**立法**，以保护投资者的意愿不被冲淡。

Regulators in Hong Kong and other Asian markets are being urged to tighten **legislation** to protect investors from being diluted against their will.

Figure 1: Example where semantically related words are lexically different in English but have a matching character in Chinese.

on a given topic may often have repeated characters that constitute part of a word even though the words as a whole do not match. Some examples are shown in Figure 1, with sentences taken from a recent issue of the Financial Times. They show how words with a common semantic sense, such as “judicial” and “judges” (first example), or “regulators” and “legislation” (second example) include characters that are the same, while the words themselves differ in both Chinese and English.

The occurrence of repeated Chinese characters can be used to help detect the subject of a text, but as the figures illustrate, English does not display the characteristic of repeated words to anything like the same extent that Chinese has repeated symbols. Words with similar or related semantic content are not necessarily lexically similar. Two topics on similar subjects may contain words which have related meanings, but to match these up an ontology or thesaurus will have to be used. The first experiments described below look for *lexical* matches; we then go on to look for *semantic* matches.

2 Introduction

We report on algorithms that are based on different models of text similarity and compare their performance. The core concept behind the experiments described here is to move from word matching algorithms to methods that use semantic matching.

Different criteria are used to select the characteristic features of a text. For instance word strings are selected with and without synonyms obtained from WordNet [9], or with noun phrases extracted for comparison, using the Stanford Parser [10]. The results indicate that synonyms rather than lexical strings are important for finding similar texts, when we are looking for texts relevant to a given topic. Hypernyms and noun phrases also contribute to the identification of text similarity, though they are not better than synonyms.

Section 3 gives an overview of the data used. The experimental methods are described in Section 4 and results reported in Section 5. Programs are written in Java. The appendix to this report contains notes relating to the program code, which is obtainable from the first author. Our experimental specifications were variants of those run by TREC.

The main set of experiments is supplemented by others based on sets of single key words rather than phrases, but also using WordNet. This simple method, described in Section 6 was developed to provide some benchmarks for the main results.

Precision is a problem for the semantic similarity methods because too many irrelevant texts are typically retrieved. It can be improved by reducing the number of synonyms, at the cost of lowering recall. In these experiments we take the standard F-score as an evaluation measure, based on precision and recall (see Section 5.1). However, there are occasions when the user of an information retrieval system will want to attach more weight to recall (e.g.some forensic applications) or to precision (e.g. some web searches).

3 Data

All data are derived from a corpus of news articles from the Financial Times (FT). These were collected by TREC. As well as the raw corpus there is an associated list of topics, of which 50 are used (topics 351 to 400). For each topic a set of texts has been put together, and each one evaluated by TREC assessors, (for details see Section 5.2). These texts are marked as relevant or not relevant to a given topic (the qrels data), so they can be used both to develop and to evaluate our methods.

Each topic has a title, a description (desc) of a few lines and a longer narrative (narr).

This is used as the seed document (i.e. the query text), and the aim is to find articles relevant (similar or related) to the topic from from the set of documents that have been assessed and are given to search. The sets contain between 500 and 1500 texts each, details are given in Table 7. The number of words in a text is typically between 500 and 600.

As a preliminary step we use a small subset of the data to test all methods in Experiment 1 (see Section 5.1). The main experiments are described in Section 5.2.

4 Text Similarity Methods

There are 11 text similarity methods tested in the main experiments. Their codes are 00, 10, 11, 13, 20, 21, 23, 30, 31, 40, and 41. All these methods have two basic stages. Firstly, feature strings ("fingerprints") are extracted from each text in different ways, denoted by the second digit of the method code. Secondly, the similarity of a text pair is measured by comparing the fingerprints of the texts, using different models, denoted by the first digit of the method code. Five similarity models are tested in the experiments:

- 0: Ferret model [5, 6, 7], which looks for lexical matches. Each text is converted to a set of 3 word trigrams, and matching trigrams are found. Though there are usually some matches, above a certain threshold the texts are similar. The comparison is based on the Jaccard coefficient value [8, page 299]
- 1: String matching model, which counts those words or word sequences that occur in both texts (omitting stop words) and calculates the Jaccard coefficient value to determine the similarity between 2 texts. A feature string may contain one or more words.
- 2: Meaning matching model, which is developed from the string matching model. The difference is that meanings rather than lexical strings are matched between texts. The meanings of a string include the synonyms and hypernyms of the whole string as well as that of each word in the string. WordNet is applied to look up word meanings.
- 3: Semantic sequence kin model (SSK) [4], which extracts semantic sequences from a text as its feature strings, and then takes into account both word and word position when 2 semantic sequences are compared.
- 4: Common semantic sequence model (CSSM) [3], which is similar to semantic sequence kin model, but uses another formula to calculate similarity of semantic sequences, in which the word position is not considered.

The second digit of the method code, indicates how fingerprints are extracted. For Ferret, there is only one way to extract fingerprints (trigrams), but for the other methods the fingerprints can be extracted in the following different ways:

- 0: Word based semantic strings. If a word is repeated locally it is taken to have semantic significance. Stop words are omitted; "local" is a parameter that can be varied. The first step identifies locally frequent words. The second step extracts phrases consisting of adjacent frequent words. These constitute the semantic sequences, which can be one or more words long, the fingerprints of the text.

  An example follows. Frequent words (more than one occurrence) are emphasised. Text is preprocessed to exclude stop words; in this case no stemming (lemmatization) is done, so "engines" and "engine" will not match. The definition of "local" is a parameter of the system, in this example it is taken as the whole paragraph.

> *Meta tags* are used to supply information for *search* engines that will not be seen by the *web* surfer unless they were to view the HTML of your *web site*. In the past, *meta tags* were a primary way for your *site* to be recognized by *web* spiders, but the internet community abused the *meta tags* to artificially increase their ranking in the *search engine* databases. Nevertheless, you should still include *meta* for those *search* bots that do recognize them, to allow your *site* to be included in their *search engine*.

  From this the phrases *meta tags* and *search engine* are extracted. The phrase *web site* is also extracted, because although it only occurs once the individual words occur frequently.

- 1: Synonym based semantic strings. In this case we look for frequent meanings rather frequent words, using WordNet to get synonyms. The first step identifies locally frequent meanings, the second step extracts phrases consisting of adjacent frequent meanings.

  Using the above example WordNet produces:

  meta tags: no synonyms
  search engine: program, programme, computer program, computer programme
  web site: website, internet site, site, computer, computing machine, computing device, data processor, electronic computer, information processing system

  (To search for a phrase on WordNet link the individual words with an underscore.) WordNet has many shortcomings, but is arguably the best available English language ontology.

- 3: Noun phrases. Every sentence in a text is parsed by the Stanford sentence parser (Stanford Lexicalized Parser v1.5.1 [10]) as a preliminary procedure. The fingerprints are noun phrases taken from low level components of the parsed text. For example, in the sentence in Figure 1 the noun phrase

  ```
  Regulators in Hong Kong and other Asian markets
  ```

  would not be taken as a whole, but would produce the 3 noun phrases *Regulators, HongKong, other Asian markets*. A noun phrase can be a single word or a sequence of words.

Code 2 was a method to extract semantic sequences based on relative meanings rather than repeated words or meanings. But it is not tested in the experiments because it is too slow.

Putting together the explanations above of the first and second digit of the method code we have the following: in each case the seed document, the topic description text, is compared to the others.

- 00: Ferret method (implemented by Java class Ferret).

- 10: A string matching method for semantic sequences. The first step identifies locally frequent words; the second step extracts phrases consisting of adjacent frequent words; the third step looks for matches. A semantic sequence can be a phrase or a single word. (The method is implemented by the Java class StringMatchBasedSimilar with the class SemSeqBasedTopicStrings and the class SameWord_SS.)

- 11: A string matching method for semantic sequences in which we look for frequent meanings rather than frequent words, using Wordnet to get synonyms. The first step identifies locally frequent meanings; the second step extracts phrases consisting of adjacent frequent meanings; the third step looks for matches. (The method is implemented by the Java class StringMatchBasedSimilar with the class SemSeqBasedTopicStrings and the class SameMeaning_SS.)

- 13: A string matching method in which the semantic strings are derived from noun phrases of the text, extracted by the Stanford sentence parser. (The method is implemented by the class StringMatchBasedSimilar with the class NounPhraseBasedTopicStrings.)
- 20: This starts in the same way as method 10. For the third step WordNet is used. For each identified phrase the meaning is found in WordNet, synonyms and hypernyms will be used for matching. This process is repeated on individual words. This differs from method 11, where Wordnet is used in step 1, and only synonyms are employed. (The method is implemented by the class WordNetBasedSimilar with the class SemSeqBasedTopicStrings and the class SameWord_SS.)
- 21: WordNet is used twice in this method. First, at step 1, as in 11, then at the end as in 20. (The method is implemented by the class WordNetBasedSimilar with the class SemSeqBasedTopicStrings and the class SameMeaning_SS.)
- 23: This starts the same way as method 13, but then WordNet is used to produce other noun phrases using WordNet synonyms and hypernyms. (The method is implemented by the class WordNetBasedSimilar with the class NounPhraseBasedTopicStrings.)
- 30: Semantic Sequence Kin method (SSK). Like 10, the first step is to identify locally frequent words. These frequent words are compared for the two texts. The metric used also takes into account the distance between frequent words. (The method is implemented by the class SSK with the class SameWord_SS.)
- 31: As 30, but based on locally frequent meanings derived from WordNet synonyms. (The method is implemented by the class SSK with the class SameMeaning_SS.)
- 40: Common Semantic Sequence Model (CSSM). As 30, but employing a different formula to measure the similarity score. (The method is implemented by the class CSSM with the class SameWord_SS.)
- 41: Common Semantic Sequence Model (CSSM). As 31, but employing the formula as used in 40. (The method is implemented by the class CSSM with the class SameMeaing_SS.)

All methods use the default parameters in this experiment. Please see Appendix B for details.

5 Results of Experiments

5.1 Experiment 1

We first ran a small scale preliminary experiment. The small data is run on two topics (topic 351 and 352) with only 32 texts. 8 texts are relevant to topic 351, another 8 texts relevant to topic 352, the remaining 16 texts are not relevant to either topic.

The similarities between the query text (the topic description)and the others are measured by different methods, described above. If the similarity is greater than or equal to a threshold, then the file being processed is considered relevant to the query text.

The standard precision p, recall r and f_1 score are then measured. They are defined in the following way, [8, page 267], and TREC 2006 note on Common Evaluation Measures [11]. Precision is a measure of the ability of the method to return only relevant texts.

$$precision = \frac{number\ of\ relevant\ texts\ retrieved}{total\ number\ of\ texts\ retrieved} \tag{1}$$

Recall is a measure of the ability of the method to return all relevant texts

$$recall = \frac{number\ of\ relevant\ texts\ retrieved}{total\ number\ of\ relevant\ texts} \tag{2}$$

T^+, true positives, denotes the number of texts correctly found to be relevant, as marked by the TREC assessors.
F^+, false positives, denotes texts that were marked relevant when they were not.
F^- denotes texts that were not marked relevant when they should have been.

$$p = \frac{T^+}{T^+ + F^+} \tag{3}$$

$$r = \frac{T^+}{T^+ + F^-} \tag{4}$$

The f score combines precision and recall into a single performance measure. Otherwise there can be a trade off between them: if every text is marked relevant then recall is complete but precision would be very low. If p and r are equally weighted then

$$f_1 = \frac{2 \times p \times r}{p + r} \tag{5}$$

In fact, this is the harmonic mean. Note these formulae should not be applied if there are no relevant documents to retrieve: in this case a perfect performance would give a score of zero for precision and an indeterminate result for recall.

This experiment was run to validate the process. The data is too limited to enable any conclusions to be drawn. Table 7 in Appendix A lists the best results for each method on their optimal thresholds.

5.2 Experiment 2

In this experiment, all methods are run on all 50 topics with all marked texts from the Financial Times corpus. Table 1 gives the number of marked texts for each topic and other details about the texts. Table 2 lists precision for each method on their optimal threshold in the range [0.01, 0.5]. Table 3 gives recall and Table 4 lists the f_1 scores. Note that for topic 379 no texts were considered relevant by the assesors, so it is excluded from the calculations (see comment at end of Section 5.1).

Further details can be found in Appendix A. Table 8 lists the false positive and true positive values of methods 1*(10,11,13) and 2*(20,21,23). Table 9 lists that of the other methods.

5.2.1 Timings

Table 5 lists the mean time of processing a text in milliseconds, using different methods. Figure 2 shows a comparison of average times in a histogram. See Appendix A for Table 10 for further details.

Table 1: Texts in the Financial Times corpus

Topic ID	Num. of texts	Num. of similar(related) texts
351	743	28
352	1747	249
353	621	54
354	848	80
355	537	2
356	1052	23
357	538	85
358	452	2
359	931	39
360	597	14
361	852	2
362	391	5
363	871	7
364	580	3
365	584	11
366	855	21
367	592	42
368	580	5
369	601	1
370	607	57
371	808	2
372	535	15
373	722	14
374	506	79
375	577	20
376	1300	30
377	569	15
378	1485	96
379	685	0
380	582	1
381	676	7
382	470	6
383	725	88
384	432	4
385	518	36
386	651	7
387	508	13
388	485	17
389	858	171
390	705	71
391	1058	188
392	685	66
393	977	9
394	836	5
395	759	110
396	617	12
397	695	8
398	624	22
399	510	13
400	437	50
Total	35,574	1,905
Total distinct texts	30,069	1,847

*Texts in topics may have overlap.

Table 2: Precision using different methods on their optimal threshold

Topic	Method code										
ID	00	10	11	13	20	21	23	30	31	40	41
351	0.38	0.17	0.06	0.18	0.12	0.08	0.14	0.00	0.09	0.43	0.12
352	0.00	0.00	0.08	0.08	0.18	0.16	0.18	0.00	0.06	0.00	0.06
353	0.00	0.00	0.34	0.51	0.09	0.09	0.12	0.00	0.13	0.00	0.13
354	0.25	0.00	0.03	0.00	0.25	0.10	0.10	0.00	0.06	0.00	0.12
355	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.33	0.00	0.04
356	0.20	0.00	0.00	1.00	0.02	1.00	1.00	0.00	0.57	0.00	0.23
357	0.20	0.29	0.25	0.33	0.18	0.22	0.23	1.00	0.22	0.75	0.26
358	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
359	0.00	0.08	0.07	0.02	0.06	0.05	0.05	0.00	0.09	0.00	0.05
360	0.00	0.21	1.00	0.04	0.50	0.04	0.04	0.00	0.15	0.00	0.10
361	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
362	0.00	0.00	0.02	1.00	0.00	0.06	0.50	0.00	0.00	0.00	1.00
363	0.00	0.00	0.14	0.04	0.50	0.04	0.03	0.00	0.04	0.00	0.03
364	0.00	0.00	0.00	1.00	0.01	0.01	0.01	0.00	0.00	0.00	0.00
365	0.00	0.29	0.75	0.50	0.27	0.20	0.12	1.00	0.21	1.00	0.33
366	0.00	0.00	0.04	0.00	0.02	0.03	0.03	0.00	0.00	0.00	0.33
367	0.00	0.00	0.06	0.07	0.02	0.06	0.08	0.00	1.00	0.00	0.22
368	0.00	0.00	0.00	0.00	0.02	0.09	0.05	0.00	0.00	0.00	0.00
369	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
370	0.00	0.00	0.13	0.43	0.00	0.13	0.28	0.00	0.19	0.00	0.17
371	0.00	0.00	0.02	0.20	0.00	0.11	0.04	0.00	0.25	0.00	0.20
372	0.00	0.16	0.14	0.00	0.27	0.05	0.13	0.00	0.16	1.00	0.44
373	0.00	1.00	0.05	0.05	1.00	0.11	0.04	0.00	0.05	0.50	0.05
374	0.00	0.60	0.23	0.50	0.20	0.19	0.20	0.33	0.29	0.67	0.32
375	1.00	0.00	0.09	0.00	0.00	0.28	0.07	0.00	0.25	0.00	0.38
376	1.00	0.00	0.01	0.11	0.03	0.03	0.03	0.00	0.50	0.11	0.48
377	0.00	1.00	0.06	0.00	0.10	0.03	0.04	0.00	0.00	0.00	0.00
378	0.00	0.00	0.06	0.16	0.00	0.07	0.06	0.00	0.14	0.00	0.16
379											
380	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
381	0.00	0.00	0.00	0.00	0.00	0.08	0.03	0.00	0.04	0.00	0.02
382	0.00	0.30	0.33	0.50	0.25	1.00	0.29	0.00	0.01	0.38	0.06
383	0.00	0.00	0.10	0.07	0.00	0.14	0.13	0.00	0.15	0.00	0.13
384	1.00	0.10	0.50	0.17	0.67	1.00	1.00	0.00	0.08	0.07	0.13
385	0.00	0.00	0.80	0.10	0.00	0.12	0.10	0.00	0.37	0.00	0.33
386	0.00	0.03	0.02	0.01	0.04	0.01	0.01	0.00	0.00	0.00	0.00
387	0.00	0.00	0.04	0.05	0.00	0.04	0.03	0.00	0.09	0.00	0.08
388	0.00	0.00	0.06	0.19	0.00	0.07	0.07	0.00	0.38	0.00	0.25
389	0.00	0.00	0.04	0.02	0.26	0.21	0.20	0.33	0.04	0.08	0.04
390	0.00	0.06	0.09	0.08	0.13	0.12	0.12	1.00	0.13	1.00	0.11
391	0.25	0.33	0.30	0.12	0.24	0.20	0.18	0.43	0.40	0.54	0.34
392	0.00	0.00	0.31	0.00	0.00	0.10	0.10	1.00	0.07	1.00	0.07
393	0.00	0.00	0.33	1.00	0.17	0.50	1.00	0.00	0.27	0.00	0.40
394	0.00	0.00	0.04	0.01	0.01	0.01	0.01	0.00	0.05	0.00	0.03
395	0.00	0.00	0.26	0.11	0.00	0.15	0.15	0.00	0.15	0.00	0.13
396	0.00	0.57	0.38	0.83	0.17	0.07	0.07	0.00	0.19	1.00	0.64
397	0.00	0.00	0.00	0.00	0.03	0.09	0.10	0.00	0.00	0.00	0.00
398	0.00	0.00	0.00	0.23	0.06	0.29	0.14	0.00	0.05	0.00	0.15
399	0.00	0.00	0.03	0.13	0.03	0.08	0.05	0.00	0.09	0.00	0.10
400	0.00	0.22	0.37	0.33	0.20	0.21	0.15	0.00	0.35	0.00	0.26

Table 3: Recall using different methods on their optimal threshold

Topic	Method code										
ID	00	10	11	13	20	21	23	30	31	40	41
351	0.11	0.14	0.36	0.71	0.29	0.50	0.46	0.00	0.11	0.11	0.25
352	0.00	0.00	0.14	0.04	0.87	0.94	0.88	0.00	0.07	0.00	0.08
353	0.00	0.00	0.19	0.39	0.28	0.91	0.76	0.00	0.13	0.00	0.17
354	0.01	0.00	0.03	0.00	0.03	0.84	0.80	0.00	0.03	0.00	0.06
355	0.00	0.00	0.00	0.00	0.50	1.00	1.00	0.00	0.50	0.00	0.50
356	0.04	0.00	0.00	0.04	0.35	0.04	0.04	0.00	0.17	0.00	0.22
357	0.01	0.14	0.39	0.24	0.75	0.81	0.68	0.04	0.14	0.07	0.31
358	0.00	0.00	0.00	0.00	0.00	0.50	1.00	0.00	0.00	0.00	0.00
359	0.00	0.54	0.64	0.03	0.72	0.95	0.92	0.00	0.18	0.00	0.23
360	0.00	0.29	0.07	0.21	0.07	0.36	0.43	0.00	0.36	0.00	0.29
361	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
362	0.00	0.00	0.20	0.20	0.00	0.60	0.20	0.00	0.00	0.00	0.20
363	0.00	0.00	0.29	0.43	0.14	0.14	0.14	0.00	0.57	0.00	0.14
364	0.00	0.00	0.00	0.33	0.67	1.00	1.00	0.00	0.00	0.00	0.00
365	0.00	0.36	0.27	0.09	0.36	0.18	0.45	0.09	0.27	0.36	0.55
366	0.00	0.00	0.10	0.00	0.33	0.48	0.62	0.00	0.00	0.00	0.10
367	0.00	0.00	0.05	0.02	0.07	0.83	0.81	0.00	0.02	0.00	0.10
368	0.00	0.00	0.00	0.00	0.20	0.20	0.20	0.00	0.00	0.00	0.00
369	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00
370	0.00	0.00	0.28	0.05	0.00	0.70	0.33	0.00	0.18	0.00	0.21
371	0.00	0.00	0.50	0.50	0.00	0.50	0.50	0.00	0.50	0.00	0.50
372	0.00	0.27	0.53	0.00	0.20	0.53	0.13	0.00	0.20	0.07	0.47
373	0.00	0.07	0.36	0.14	0.07	0.07	0.86	0.00	0.21	0.07	0.43
374	0.00	0.04	0.42	0.01	0.54	0.73	0.82	0.01	0.13	0.05	0.48
375	0.05	0.00	0.35	0.00	0.00	0.25	0.45	0.00	0.15	0.00	0.25
376	0.07	0.00	0.07	0.70	0.63	0.80	0.80	0.00	0.37	0.03	0.33
377	0.00	0.07	0.27	0.00	0.27	1.00	0.53	0.00	0.00	0.00	0.00
378	0.00	0.00	0.09	0.23	0.00	0.92	0.99	0.00	0.32	0.00	0.50
379											
380	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00	0.00	0.00
381	0.00	0.00	0.00	0.00	0.00	0.14	0.14	0.00	0.14	0.00	0.14
382	0.00	0.50	0.17	0.17	0.50	0.17	0.33	0.00	0.17	0.50	0.50
383	0.00	0.00	0.05	0.01	0.00	0.77	0.91	0.00	0.15	0.00	0.30
384	0.25	0.25	0.25	0.75	0.50	0.25	0.25	0.00	0.75	0.25	0.50
385	0.00	0.00	0.11	0.06	0.00	0.72	0.86	0.00	0.19	0.00	0.14
386	0.00	0.14	0.43	0.14	0.14	1.00	1.00	0.00	0.00	0.00	0.00
387	0.00	0.00	0.23	0.23	0.00	0.85	1.00	0.00	0.62	0.00	0.69
388	0.00	0.00	0.24	0.18	0.00	0.71	0.71	0.00	0.18	0.00	0.24
389	0.00	0.00	0.05	0.01	0.85	0.99	1.00	0.01	0.02	0.01	0.04
390	0.00	0.07	0.18	0.10	0.75	0.92	0.62	0.01	0.08	0.01	0.10
391	0.02	0.16	0.57	0.13	0.67	0.87	0.98	0.02	0.62	0.07	0.77
392	0.00	0.00	0.08	0.00	0.00	0.97	0.98	0.02	0.03	0.02	0.06
393	0.00	0.00	0.11	0.11	0.11	0.11	0.11	0.00	0.33	0.00	0.22
394	0.00	0.00	0.40	0.20	1.00	0.60	0.60	0.00	0.20	0.00	0.20
395	0.00	0.00	0.45	0.04	0.00	0.93	1.00	0.00	0.07	0.00	0.09
396	0.00	0.33	0.25	0.42	0.42	0.75	0.17	0.00	0.58	0.25	0.58
397	0.00	0.00	0.00	0.00	0.13	0.13	0.63	0.00	0.00	0.00	0.00
398	0.00	0.00	0.00	0.45	0.59	0.09	0.14	0.00	0.05	0.00	0.23
399	0.00	0.00	0.23	0.08	0.38	0.23	0.85	0.00	0.15	0.00	0.23
400	0.00	0.08	0.58	0.10	0.54	0.64	0.92	0.00	0.22	0.00	0.42

Table 4: f_1 scores using different methods on their optimal threshold

Topic	Method code										
ID	00	10	11	13	20	21	23	30	31	40	41
351	0.17	0.16	0.10	0.29	0.17	0.15	0.21	0.00	0.10	0.17	0.16
352	0.00	0.00	0.10	0.05	0.29	0.28	0.30	0.00	0.07	0.00	0.07
353	0.00	0.00	0.24	0.44	0.13	0.17	0.21	0.00	0.13	0.00	0.15
354	0.02	0.00	0.03	0.00	0.05	0.17	0.18	0.00	0.04	0.00	0.08
355	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.40	0.00	0.07
356	0.07	0.00	0.00	0.08	0.04	0.08	0.08	0.00	0.27	0.00	0.22
357	0.02	0.19	0.30	0.28	0.30	0.35	0.35	0.07	0.17	0.13	0.28
358	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00
359	0.00	0.14	0.13	0.02	0.12	0.10	0.10	0.00	0.12	0.00	0.08
360	0.00	0.24	0.13	0.06	0.13	0.07	0.07	0.00	0.21	0.00	0.15
361	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
362	0.00	0.00	0.03	0.33	0.00	0.11	0.29	0.00	0.00	0.00	0.33
363	0.00	0.00	0.19	0.07	0.22	0.06	0.05	0.00	0.08	0.00	0.05
364	0.00	0.00	0.00	0.50	0.02	0.01	0.01	0.00	0.00	0.00	0.00
365	0.00	0.32	0.40	0.15	0.31	0.19	0.19	0.17	0.24	0.53	0.41
366	0.00	0.00	0.05	0.00	0.03	0.06	0.07	0.00	0.00	0.00	0.15
367	0.00	0.00	0.05	0.04	0.03	0.12	0.14	0.00	0.05	0.00	0.13
368	0.00	0.00	0.00	0.00	0.03	0.13	0.07	0.00	0.00	0.00	0.00
369	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
370	0.00	0.00	0.18	0.09	0.00	0.21	0.31	0.00	0.18	0.00	0.19
371	0.00	0.00	0.03	0.29	0.00	0.18	0.07	0.00	0.33	0.00	0.29
372	0.00	0.20	0.23	0.00	0.23	0.09	0.13	0.00	0.18	0.13	0.45
373	0.00	0.13	0.09	0.07	0.13	0.09	0.08	0.00	0.08	0.13	0.09
374	0.00	0.07	0.30	0.02	0.29	0.31	0.33	0.02	0.18	0.09	0.38
375	0.10	0.00	0.14	0.00	0.00	0.26	0.12	0.00	0.19	0.00	0.30
376	0.13	0.00	0.02	0.20	0.05	0.06	0.06	0.00	0.42	0.05	0.39
377	0.00	0.13	0.10	0.00	0.15	0.06	0.07	0.00	0.00	0.00	0.00
378	0.00	0.00	0.07	0.19	0.00	0.13	0.12	0.00	0.20	0.00	0.24
379											
380	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00	0.00
381	0.00	0.00	0.00	0.00	0.00	0.11	0.05	0.00	0.06	0.00	0.03
382	0.00	0.38	0.22	0.25	0.33	0.29	0.31	0.00	0.02	0.43	0.10
383	0.00	0.00	0.06	0.02	0.00	0.23	0.22	0.00	0.15	0.00	0.18
384	0.40	0.14	0.33	0.27	0.57	0.40	0.40	0.00	0.14	0.11	0.21
385	0.00	0.00	0.20	0.07	0.00	0.21	0.18	0.00	0.25	0.00	0.20
386	0.00	0.04	0.04	0.03	0.07	0.02	0.02	0.00	0.00	0.00	0.00
387	0.00	0.00	0.07	0.08	0.00	0.07	0.06	0.00	0.16	0.00	0.15
388	0.00	0.00	0.09	0.18	0.00	0.13	0.13	0.00	0.24	0.00	0.24
389	0.00	0.00	0.05	0.01	0.39	0.34	0.34	0.01	0.03	0.01	0.04
390	0.00	0.06	0.12	0.09	0.22	0.22	0.21	0.03	0.10	0.03	0.11
391	0.03	0.22	0.39	0.13	0.35	0.33	0.31	0.03	0.49	0.13	0.47
392	0.00	0.00	0.12	0.00	0.00	0.18	0.18	0.03	0.04	0.03	0.07
393	0.00	0.00	0.17	0.20	0.13	0.18	0.20	0.00	0.30	0.00	0.29
394	0.00	0.00	0.07	0.01	0.02	0.02	0.02	0.00	0.07	0.00	0.05
395	0.00	0.00	0.33	0.06	0.00	0.26	0.26	0.00	0.10	0.00	0.11
396	0.00	0.42	0.30	0.56	0.24	0.12	0.10	0.00	0.29	0.40	0.61
397	0.00	0.00	0.00	0.00	0.05	0.11	0.17	0.00	0.00	0.00	0.00
398	0.00	0.00	0.00	0.31	0.11	0.14	0.14	0.00	0.05	0.00	0.18
399	0.00	0.00	0.05	0.10	0.06	0.12	0.09	0.00	0.11	0.00	0.14
400	0.00	0.12	0.45	0.15	0.29	0.32	0.26	0.00	0.27	0.00	0.32

.

Table 5: Mean time of processing a text in milliseconds. Experiments done on machines with 1024MB RAM, processor 2.4GHz

Method code	00	10	11	13	20	21	23	30	31	40	41
Mean time	16.68	2.56	70.95	18.82	57.73	80.80	79.57	2.50	71.39	2.56	71.42

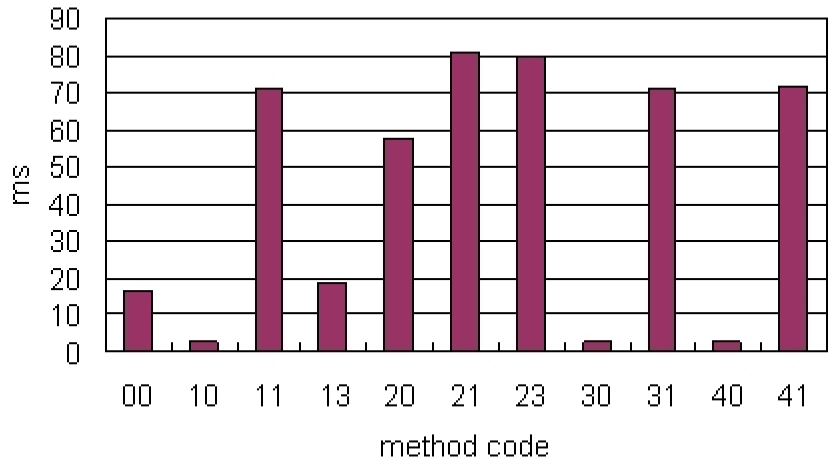

Figure 2: Mean time of processing a text in milliseconds. Experiments done on machines with 1024MB RAM, processor 2.4GHz

5.2.2 Collating results

The average performance is evaluated in 3 ways, micro average macro average and weighted average. The figures 4 and 3 and 5 illustrate them in histograms.

The micro average first finds the precision, recall and f score for each topic. Then each of these measures are averaged over all topics. No account is taken of the different size of document sets for different topics, as shown in Table 1.

- Micro average

  The precision, recall of each topic are measured first as in the formula 3, and formula 4, then they are summed up to get a mean value as shown in the formula 6, and formula 7. The f_1 is still the harmonic mean of precision and recall as in the formula 8.

$$p_{micro} = \frac{1}{n} \sum p_i \tag{6}$$

$$r_{micro} = \frac{1}{n} \sum r_i \tag{7}$$

$$f_{1micro} = \frac{2 \times p_{micro} \times r_{micro}}{p_{micro} + r_{micro}} \tag{8}$$

- Macro average The macro average finds the cumulative score over all topics and then calculates the precision, recall and f score for the whole set taken together.

  Macro average means that the number of found similar (relevant) texts in each topic are summed up first, and then precision, recall, and f_1 are measured as in the formula 9, formula 10, and formula 11.

$$p_{macro} = \frac{\sum T_i^+}{\sum (T_i^+ + F_i^+)} \tag{9}$$

Table 6: Average performance of different methods

Method code	00	10	11	13	20	21	23	30	31	40	41
p_{micro}	0.09	0.11	0.15	0.20	0.12	0.15	0.15	0.10	0.15	0.17	0.17
r_{micro}	0.01	0.07	0.20	0.15	0.28	0.57	0.64	0.00	0.18	0.04	0.24
f_{1micro}	0.02	0.09	0.17	0.17	0.17	0.24	0.24	0.00	0.16	0.06	0.20
p_{macro}	0.34	0.16	0.12	0.12	0.11	0.10	0.09	0.58	0.16	0.43	0.16
r_{macro}	0.01	0.05	0.24	0.11	0.43	0.80	0.80	0.01	0.18	0.02	0.26
f_{1macro}	0.02	0.08	0.16	0.11	0.18	0.18	0.16	0.02	0.17	0.04	0.20
p_{weight}	0.05	0.09	0.15	0.14	0.12	0.14	0.14	0.08	0.16	0.13	0.16
r_{weight}	0.01	0.05	0.23	0.10	0.41	0.74	0.61	0.01	0.17	0.02	0.24
$f_{1weight}$	0.02	0.06	0.18	0.12	0.19	0.24	0.23	0.02	0.16	0.03	0.19

$$r_{macro} = \frac{\sum T_i^+}{\sum (T_i^+ + F_i^-)} \tag{10}$$

$$f_{1macro} = \frac{2 \times p_{macro} \times r_{macro}}{p_{macro} + r_{macro}} \tag{11}$$

- Weighted average

These metrics can be weighted in various ways. Some users will want to attach more weight to precision, others to recall. The metrics can also be weighted to take account of different corpora sizes. The type of weighting used here compensates for the different number of relevant documents in different topics. Weighted average precision is calculated by summing the precision for each topic modified by a weight (equation 12). The weight is the proportion of the topic's similar(related) texts to all topics' similar(related) texts (equation 15). Weighted average recall is calculated in an analogous way (equation 13). The f_1 score is the harmonic mean of precision and recall (equation 14).

$$p_{weight} = \sum (w_i \times p_i) \tag{12}$$

$$r_{weight} = \sum (w_i \times r_i) \tag{13}$$

$$f_{1weight} = \frac{2 \times p_{weight} \times r_{weight}}{p_{weight} + r_{weight}} \tag{14}$$

$$w_i = \frac{number\ of\ relevant\ texts\ to\ the\ ith\ topic}{sum\ of\ relevant\ texts\ to\ each\ topic} \tag{15}$$

Table 6 shows the average performance of different methods on the whole set of topics, in graphical form in Figures 3, 4 and 5.

Whichever measure is used, the average performance of methods 11, 20, 21, 23, 31, and 41 are better than the others, though their average running times are longer. It indicates that the use of an ontology such as WordNet improves performance.

Comparing methods 10 and 11, the difference between them is whether a word alone is identified, or a word together with its synonyms. Using WordNet to get synonyms increases the f_1 score 2 to 3 times - at a price of more than 27 times greater running time. Comparing methods 30 and 31, we see that method 30 performs very poorly, its average f_1 score is only

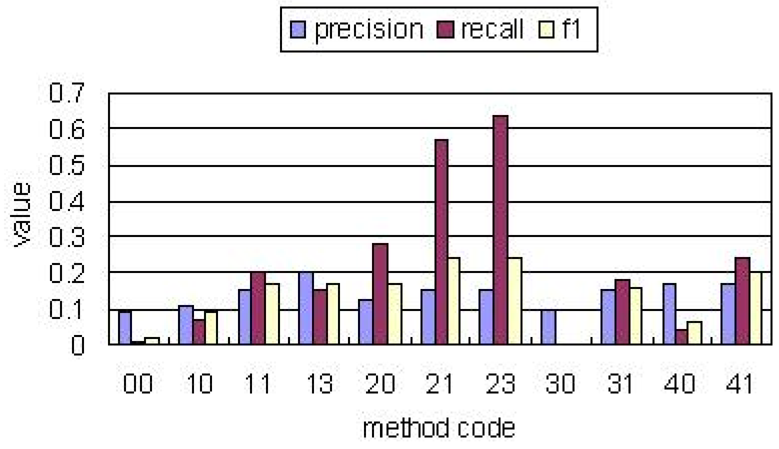

Figure 3: The micro average performance of methods

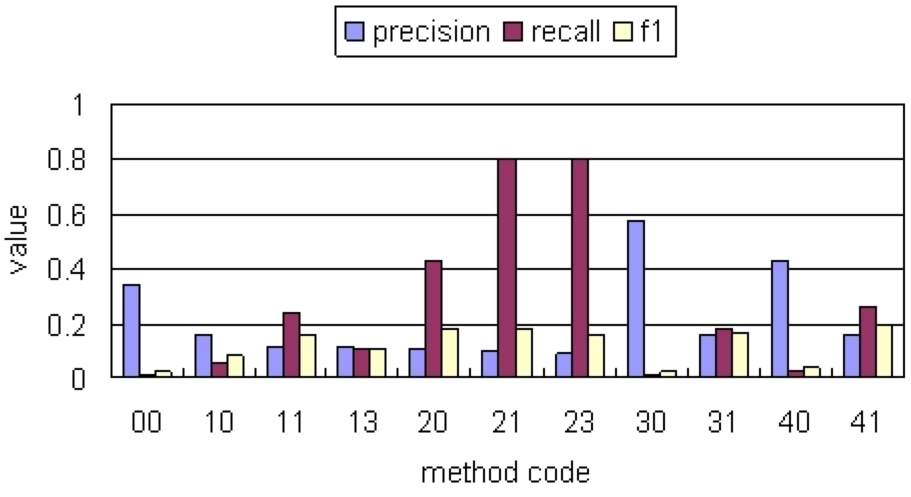

Figure 4: The macro average performance of methods

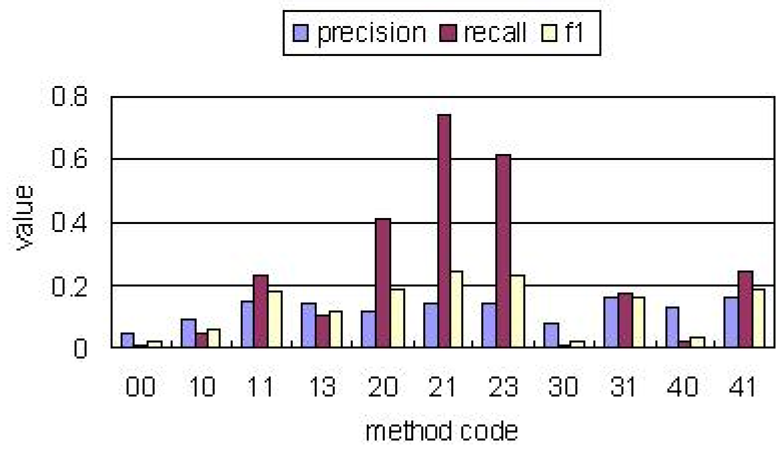

Figure 5: The weighted average performance of methods

about 0.01. Method 31, which employs the same algorithm as method 30, except synonyms are used in place of words, works much better with the average f_1 score about 0.15, again at a price of more than 27 times greater running time. Comparable results are produced by methods 40 and 41. Obviously, synonyms contribute to finding similarity of texts, but as the search space expands so does processing time increase.

Methods 21 and 23 perform slightly better than methods 11, 20, 31, and 41, though they run a little longer than those methods. Method 21 uses WordNet twice, but its performance is not much better than methods 23 and the other four (11,20,31, and 41). This suggests that the double use of WordNet is not worthwhile.

Method 13, which employs noun phrases without WordNet, is much better than the other methods without WordNet (i.e. methods 00, 10, 30, and 40), but worse than methods with WordNet. It suggests that noun phrases contribute to texts' similarity more than single words. Method 23 uses WordNet to find synonyms and hypernyms of noun phrases so that it gains a significant increase in recall score (from about 0.11 to more than 0.60). But precision does not increase, in fact it decreases a little. As a result, the final f_1 score of method 23 is better than the method 13, but not superior to method 21.

The methods with WordNet that compare synonyms between texts will find more similar texts than word strings do. That increases the recall score. In particular, methods 20, 21 and 23 use both synonyms and hypernyms and have the best recall scores, but precision is not improved.

6 Experiments with keyword method for benchmarking

The same task as that described above has been addressed using a simpler keyword method in order to provide benchmarks for the main experiments. 30 of the 50 topics were used, numbers 351 to 380 inclusive. The method is again based on the concept of matching meanings, rather than matching actual words, and WordNet is used to provide synsets (synonyms and adjectives with similar meanings, such as “global” and “international”). The algorithm is designed to reduce problems that come from words having multiple meanings: by enforcing a requirement that more than one keyword must be found in more than one synset there is an attempt to focus on the appropriate context. See the discussion in the conclusion.

Trial1: Results with all synonyms and "synsa" from WordNet

Topic	size	%correct	true +ve	false -ve	false +ve	precision	recall	F
351	743	98.3	23	5	8	0.74	0.82	0.78
352	1747	80.7	204	45	293	0.41	0.82	0.55
353	621	90.2	32	39	22	0.59	0.45	0.51
354	848	83	6	74	70	0.08	0.08	0.08
355	537	69.3	1	1	164	0.01	0.5	0.02
356	1052	91.3	9	14	77	0.1	0.39	0.16
357	538	75.3	43	42	91	0.32	0.51	0.39
358	452	99.6	0	2	0	0	0	0
359	931	53.6	17	22	410	0.04	0.44	0.07
360	597	74	4	10	145	0.03	0.29	0.05
361	852	85	1	1	127	0.01	0.5	0.02
362	391	83.9	3	2	61	0.05	0.6	0.09
363	871	34.1	6	1	573	0.01	0.86	0.02
364	580	85.2	3	0	86	0.03	1	0.06
365	578	99	11	0	6	0.65	1	0.79
366	855	96.5	3	18	12	0.2	0.14	0.16
367	592	45.4	16	26	297	0.05	0.38	0.09
368	580	98.8	1	4	3	0.25	0.2	0.22
369	601	99.7	0	1	1	0	0	0
370	607	67.1	36	21	179	0.17	0.63	0.27
371	808	58.5	2	0	335	0.01	1	0.02
372	535	77.2	4	11	111	0.03	0.27	0.05
373	722	92.9	2	12	39	0.05	0.14	0.07
374	506	78.5	78	1	108	0.42	0.99	0.59
375	577	65.3	10	10	190	0.05	0.5	0.09
376	1300	46.3	30	0	698	0.04	1	0.08
377	569	63.3	5	10	199	0.02	0.33	0.04
378	1485	71.7	74	22	398	0.16	0.77	0.26
379	685	99.9	0	0	1	*0	*0	*0
380	582	92.3	1	0	45	0.02	1	0.04

Averages		precision	recall	f-score
	micro-	0.16	0.54	0.25
	macro-	0.12	0.61	0.2

Trial2 Results with 6 synonyms only

Topic	%correct	true +ve	false -ve	false +ve	precision	recall	F
351	98.3	23	5	8	0.74	0.82	0.78
352	80.8	198	51	285	0.41	0.8	0.54
353	89	21	50	18	0.54	0.3	0.39
354	84.8	4	76	53	0.07	0.05	0.06
355	90.7	1	1	49	0.02	0.5	0.04
356	96	6	17	25	0.19	0.26	0.22
357	83.3	34	51	39	0.47	0.4	0.43
358	99.6	0	2	0	0	0	0
359	56.5	17	22	383	0.04	0.44	0.07
360	74	4	10	145	0.03	0.29	0.05
361	86	1	1	118	0.01	0.5	0.02
362	91.6	2	3	30	0.06	0.4	0.1
363	82.5	0	7	145	0	0	0
364	99.3	2	1	3	0.4	0.67	0.5
365	99	10	1	5	0.67	0.91	0.77
366	96.8	3	18	9	0.25	0.14	0.18
367	91.6	5	37	13	0.28	0.12	0.17
368	98.8	1	4	3	0.25	0.2	0.22
369	99.7	0	1	1	0	0	0
370	75.9	36	21	125	0.22	0.63	0.33
371	61.1	2	0	314	0.01	1	0.02
372	89	3	12	47	0.06	0.2	0.09
373	92.9	2	12	39	0.05	0.14	0.07
374	81	77	2	94	0.45	0.97	0.61
375	86	9	11	70	0.11	0.45	0.18
376	77.5	29	1	291	0.09	0.97	0.16
377	69.8	5	10	162	0.03	0.33	0.06
378	82.1	45	51	215	0.17	0.47	0.25
379	99.9	0	0	1	*0	*0	*0
380	95	1	0	29	0.03	1	0.06

precision	recall	f-score
0.19	0.45	0.27
0.17	0.53	0.26

Comparisons of micro-averages

CL-Trial1 Results with all synonyms and all "synsa" from WordNet
CL-Trial2 Results with 6 synonyms only
JP method described in text

	CL-Trial1	CL-Trial2
precision	0.16	0.19
recall	0.54	0.45
F score	0.25	0.27

JPmethod:	11	20	21	23	31	41
(from	0.15	0.12	0.15	0.15	0.15	0.17
Table 6)	0.2	0.28	0.57	0.64	0.18	0.24
	0.17	0.17	0.24	0.24	0.16	0.2

Comparison of macro-averages

	CL-Trial1	CL-Trial2
precision	0.12	0.17
recall	0.61	0.53
F score	0.2	0.26

JPmethod:	11	20	21	23	31	41
(from	0.12	0.11	0.1	0.09	0.16	0.16
Table 6)	0.24	0.43	0.8	0.8	0.18	0.26
	0.16	0.18	0.18	0.16	0.17	0.2

The process used is as follows.

- Two keywords are extracted from the topic description, by taking the first 2 content words from the title (and description if title is one word long).
- Keywords are looked up in WordNet and a synset is produced for each of them.
- The documents to be searched are analysed. If any member of *both* synsets occurs in a text, then that text is considered relevant.

Results of Trial 1 are shown in Figure 6. The most obvious problem was that precision was low: many irrelevant documents were retrieved. The experiment was repeated, Trial 2, limiting the number of words in a synset. In this case precision improved, and so did the f-score even though recall declined slightly.

Trial 1 took 3m 54secs to process all 30 topics, on a laptop with 1.73GHz processor and 0.98GB RAM. Programming is in C++. Code is available from 2nd author.

This approach could be carried further quite easily by increasing the number of keywords that are taken as seeds. However, extracting keywords might not be so easy in general as it is with this data, and needs further development.

7 Discussion

A summary of some of our results is given in Table 7. The results of the simple keyword approach compare favourably with the more sophisticated main methods, but more work is needed to develop and refine the best of these latter approaches, which have the potential to be more effective. For example, a topic about the Channel Tunnel would produce keywords “channel” and “tunnel”, when they are obviously more meaningful as a phrase. Similarly, the topic subject “remote sensing” should be taken as a phrase; by taking single words the problem of low precision is exacerbated.

Overall, our results so far from these initial experiments do not seem as good as the best reported from the TREC competitions; see for example, reports on previous competitions at the TREC website [11], especially the Filtering Track.

7.1 Limitations of Wordnet and other ontologies

WordNet has many limitations, as is well known, but is arguably the best general ontology available currently. Its limitations, which only become apparent through manual inspection, fall into two categories: those that can be corrected and those that cannot. Considering examples from the data we have been processing we find in the first category some errors of omission - for instance the seed word “anorexia” does not produce the adjective “anorexic”. There are also errors of commission: for instance a synonym of “Antarctica” is given as “continent” as well as an instance of a continent. There is also a bias against recent terminology, since WordNet was originally based on texts written many decades back. For instance “chip” has 9 senses, of which the meaning “semiconductor device” ranks 7 in order of frequency, below the meaning “cow dung”, and other senses. “GM” only occurs as an abbreviation for “gram”, not for “genetically modified”.

However, these limitations can be put right. The far more profound problem, which applies to other ontologies as well as WordNet, is that many words have more then one meaning. The least problematic are words like “bank” which have distinct, unrelated meanings. In an information retrieval task files relating to the wrong meaning can be filtered out by getting more context, as in the keyword method (Section 6). However, there is a fundamental problem with the many words whose meanings shade into one another. Examples include “space” which has a core sense of a void, but could mean a gap between words, a parking space for a car, or the outer reaches of the universe. “Risk” can mean danger but also has a sense related to probability. “Commercial” has the sense of pertaining to commerce, but also can mean an advertisement. In WordNet “fertilization” has the senses “creation” and “enrichment”.

By taking a phrase rather than a single word, as in “Channel Tunnel” or “remote sensing” this problem can often be reduced, but in other cases variations of meaning produce a multitude of unwanted matches. This is the primary cause of the low precision scores, the large number of false positive texts that are retrieved.

In the experiments based on the keyword approach this problem was addressed in a crude way by reducing the number of keywords in a synset, so that the less frequently used are discarded; but though precision will probably improve, recall will probably decline (see Figure 6).

8 Conclusions

In the main experiments we have tested and compared 5 similarity models: Ferret, string matching model, meaning matching model, semantic sequence kin model, and common semantic sequence model, with different feature selection methods, using a corpus of articles from the Financial Times corpus. This work provides a base line for further experiments. We have also conducted experiments with the simpler keyword method, looking for semantic similarities rather than actual word matching.

We conclude that:

- Synonyms, and other synset terms, are important for finding similar texts.

  In similar texts we find that there are many phrases with semantic content that match. We find numerous synonyms, hypernyms and adjectives with related meanings in similar texts. Without them very few similar texts can be found based on matching word strings.

  Hypernyms are not as important as synonyms. They can lead to an increase in recall value, but have no effect on precision. As a result, the methods with both synonyms and hypernyms are not superior to those with synonyms only. We also found that noun phrases contribute to effective identification of similar texts more than single words, though they are not better than synonyms.

  Synonyms are used twice in method 21. It is a little better than methods 11 and 41, but not superior to method 23, which employs synonyms only once, and using WordNet twice increases running time.

- Precision needs to be significantly increased.

  In all the experiments conducted precision has been weak. In some cases this could be due to common words that do not contribute to identifying meaning swamping the more useful words and phrases with noise. The stopword list could be modified by including many terms like “issues”, “factor”, “discussion” that are too general to contribute useful information. However, many words that are generated by WordNet may be irrelevant in one context, but appropriate in another, as discussed above. It is not possible to filter them out of a general ontology - hence the development of domain specific ontologies. Using the keyword approach we tried to reduce the problem by limiting the number of nouns and adjectives taken from the set produced by WordNet. This did improve precision marginally, at the cost of reducing recall.

In order to improve performance we need to develop more accurate ways of characterising the context of a text, so that words and phrases produced by an ontology can be filtered and only relevant ones retained.

Acknowledgements

Dr. JunPeng Bao carried out this work at the University of Hertfordshire, UK, sponsored by the Royal Society as a Visiting International Fellow.

References

[1] J. P. Bao, C. M. Lyon, P. C. R. Lane, W. Ji, and J. A. Malcolm. Copy detection in Chinese documents using the Ferret: A report on experiments *Technical Report 456*, Science and Technology Research Institute, University of Hertfordshire, 2006

[2] J. P. Bao, C. M. Lyon, P. C. R. Lane, W. Ji, and J. A. Malcolm. Copy Detection in Chinese Documents using Ferret. In *Language Resources and Evaluation* In press.

[3] J. P. Bao, J. Y. Shen, X. D. Liu, H. Y. Liu, and X. D. Zhang. Finding plagiarism based on common semantic sequence model. In *Proceedings of the 5th International Conference on Advances in Web-Age Information Management*, volume 3129, pages 640–645. Lecture Notes in Computer Science, 2004.

[4] J. P. Bao, J. Y. Shen, X. D. Liu, H. Y. Liu, and X. D. Zhang. Semantic sequence kin: A method of document copy detection. In *Proceedings of the Advances in Knowledge Discovery and Data Mining*, volume 3056, pages 529–538. Lecture Notes in Computer Science, 2004.

[5] P. C. R. Lane, C. M. Lyon, and J. A. Malcolm. Demonstration of the Ferret plagiarism detector. In *Proceedings of the 2nd International Plagiarism Conference*, 2006.

[6] C. M. Lyon, R. Barrett, and J. A. Malcolm. A theoretical basis to the automated detection of copying between texts, and its practical implementation in the Ferret plagiarism and collusion detector. In *JISC(UK) Conference on Plagiarism: Prevention, Practice and Policies Conference*, 2004.

[7] C. M. Lyon, J. A. Malcolm, and R. G. Dickerson. Detecting short passages of similar text in large document collections. In *Proceedings of Conference on Empirical Methods in Natural Language Processing.* SIGDAT, Special Interest Group of the ACL, 2001.

[8] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing.* Cambridge, MA: The MIT Press, 2001.

[9] WordNet http://wordnet.princeton.edu/

[10] Stanford Lexicalized Parser http://nlp.stanford.edu/downloads/lex-parser.shtml

[11] TREC http://trec.nist.gov

Table 7: The best results of different methods on small data set. These experiments were run on a machine with 2.09GHz processor, 1024MB RAM

Method	topic 351				topic 352			
code	p	r	f_1	time	p	r	f_1	time
00	1.0000	0.1250	0.2222	522	0.0000	0.0000	0.0000	524
10	1.0000	0.1250	0.2222	156	0.0000	0.0000	0.0000	156
11	0.6000	0.3750	0.4615	36570	0.6667	0.2500	0.3636	36733
13	0.7500	0.7500	0.7500	613	0.5000	0.1250	0.2000	613
20	0.6250	0.6250	0.6250	36703	0.3500	0.8750	0.5000	35706
21	0.5000	0.6250	0.5556	36160	0.3684	0.8750	0.5185	37160
23	0.3684	0.8750	0.5185	37338	0.4706	1.0000	0.6400	37137
30	0.0000	0.0000	0.0000	162	0.0000	0.0000	0.0000	161
31	0.0000	0.0000	0.0000	37000	0.0000	0.0000	0.0000	35811
40	1.0000	0.1250	0.2222	163	0.0000	0.0000	0.0000	160
41	1.0000	0.2500	0.4000	36465	0.3333	0.1250	0.1818	36977

*The time is in milliseconds, that is the whole time from reading 32 texts till the similarities are measured over them. The optimal threshold varies in the range [0.01,0.5] depending on the method and topic.

A Appendix A

A.1 Results of the preliminary Experiment 1, on small data set

Two topics (351 and 352) are taken, but only 32 texts in all. 8 are relevant to the first topic, 8 to the second, 16 not relevant to either. The threshold varies from 0.01 to 0.5 by step 0.01.

A.2 Detailed results from main Experiment 2

A.3 Timings

Table 8: False positive and true positive values of methods 1* and 2*

Topic	$T^+ + F^-$	10		11		13		20		21		23	
ID		F^+	T^+	F^+	T^+	F^+	T^+	F^+	T^+	F^+	T^+	F^+	T^+
351	28	19	4	169	10	92	20	58	8	151	14	80	13
352	249	0	0	432	36	114	10	1018	217	1224	235	981	219
353	54	0	0	19	10	20	21	157	15	489	49	293	41
354	80	0	0	56	2	0	0	6	2	636	67	555	64
355	2	0	0	0	0	0	0	141	1	264	2	382	2
356	23	0	0	0	0	0	1	344	8	0	1	0	1
357	85	29	12	100	33	40	20	283	64	243	69	191	58
358	2	0	0	0	0	0	0	0	0	160	1	436	2
359	39	245	21	334	25	65	1	409	28	693	37	682	36
360	14	15	4	0	1	79	3	1	1	115	5	149	6
361	2	0	0	0	0	0	0	0	0	0	0	786	2
362	5	0	0	63	1	0	1	0	0	46	3	1	1
363	7	0	0	12	2	71	3	1	1	23	1	36	1
364	3	0	0	0	0	0	1	236	2	418	3	402	3
365	11	10	4	1	3	1	1	11	4	8	2	37	5
366	21	0	0	54	2	0	0	442	7	291	10	362	13
367	42	0	0	32	2	13	1	127	3	524	35	419	34
368	5	0	0	0	0	0	0	52	1	10	1	21	1
369	1	0	0	0	0	0	0	0	0	0	0	582	1
370	57	0	0	105	16	4	3	0	0	278	40	48	19
371	2	0	0	64	1	4	1	0	0	8	1	25	1
372	15	21	4	48	8	0	0	8	3	160	8	13	2
373	14	0	1	88	5	40	2	0	1	8	1	259	12
374	79	2	3	108	33	1	1	174	43	243	58	253	65
375	20	0	0	73	7	0	0	0	0	13	5	125	9
376	30	0	0	151	2	163	21	720	19	693	24	735	24
377	15	0	1	62	4	0	0	35	4	504	15	207	8
378	96	0	0	153	9	113	22	0	0	1195	88	1380	95
379	0	0	0	0	0	0	0	0	0	0	0	0	0
380	1	0	0	0	0	0	0	0	0	215	1	381	1
381	7	0	0	0	0	0	0	0	0	11	1	32	1
382	6	7	3	2	1	1	1	9	3	0	1	5	2
383	88	0	0	36	4	14	1	0	0	426	68	549	80
384	4	9	1	1	1	15	3	1	2	0	1	0	1
385	36	0	0	1	4	19	2	0	0	185	26	283	31
386	7	39	1	145	3	69	1	22	1	557	7	608	7
387	13	0	0	67	3	59	3	0	0	279	11	386	13
388	17	0	0	66	4	13	3	0	0	160	12	161	12
389	171	0	0	170	8	52	1	421	145	646	169	679	171
390	71	81	5	133	13	76	7	357	53	466	65	313	44
391	188	63	31	258	108	169	24	401	126	634	163	815	184
392	66	0	0	11	5	0	0	0	0	568	64	591	65
393	9	0	0	2	1	0	1	5	1	1	1	0	1
394	5	0	0	47	2	167	1	592	5	297	3	384	3
395	110	0	0	144	50	31	4	0	0	564	102	628	110
396	12	3	4	5	3	1	5	25	5	127	9	26	2
397	8	0	0	0	0	0	0	28	1	10	1	46	5
398	22	0	0	0	0	33	10	207	13	5	2	18	3
399	13	0	0	107	3	7	1	139	5	34	3	208	11
400	50	14	4	49	29	10	5	109	27	121	32	263	46

Table 9: False positive and true positive values of methods 00, 3* and 4*

Topic	$T^+ + F^-$	00		30		31		40		41	
ID		F^+	T^+	F^+	T^+	F^+	T^+	F^+	T^+	F^+	T^+
351	28	5	3	0	0	31	3	4	3	53	7
352	249	0	0	0	0	272	18	0	0	332	21
353	54	0	0	0	0	48	7	0	0	58	9
354	80	3	1	0	0	31	2	0	0	37	5
355	2	0	0	0	0	2	1	0	0	26	1
356	23	4	1	0	0	3	4	0	0	17	5
357	85	4	1	0	3	43	12	2	6	74	26
358	2	0	0	0	0	0	0	0	0	0	0
359	39	0	0	0	0	75	7	0	0	183	9
360	14	0	0	0	0	28	5	0	0	35	4
361	2	0	0	0	0	0	0	0	0	0	0
362	5	0	0	0	0	0	0	0	0	0	1
363	7	0	0	0	0	90	4	0	0	33	1
364	3	0	0	0	0	0	0	0	0	0	0
365	11	0	0	0	1	11	3	0	4	12	6
366	21	0	0	0	0	0	0	0	0	4	2
367	42	0	0	0	0	0	1	0	0	14	4
368	5	0	0	0	0	0	0	0	0	0	0
369	1	0	0	0	0	0	0	0	0	0	0
370	57	0	0	0	0	44	10	0	0	59	12
371	2	0	0	0	0	3	1	0	0	4	1
372	15	0	0	0	0	16	3	0	1	9	7
373	14	0	0	0	0	63	3	1	1	120	6
374	79	0	0	2	1	24	10	2	4	81	38
375	20	0	1	0	0	9	3	0	0	8	5
376	30	0	2	0	0	11	11	8	1	11	10
377	15	0	0	0	0	0	0	0	0	0	0
378	96	0	0	0	0	183	31	0	0	254	48
379	0	0	0	0	0	0	0	0	0	0	0
380	1	0	0	0	0	0	0	0	0	0	0
381	7	0	0	0	0	23	1	0	0	65	1
382	6	0	0	0	0	101	1	5	3	50	3
383	88	0	0	0	0	75	13	0	0	171	26
384	4	0	1	0	0	37	3	13	1	13	2
385	36	0	0	0	0	12	7	0	0	10	5
386	7	0	0	0	0	0	0	0	0	0	0
387	13	0	0	0	0	80	8	0	0	98	9
388	17	0	0	0	0	5	3	0	0	12	4
389	171	0	0	2	1	104	4	12	1	141	6
390	71	0	0	0	1	41	6	0	1	55	7
391	188	9	3	4	3	172	116	12	14	277	144
392	66	0	0	0	1	25	2	0	1	52	4
393	9	0	0	0	0	8	3	0	0	3	2
394	5	0	0	0	0	21	1	0	0	35	1
395	110	0	0	0	0	44	8	0	0	65	10
396	12	0	0	0	0	29	7	0	3	4	7
397	8	0	0	0	0	0	0	0	0	0	0
398	22	0	0	0	0	19	1	0	0	28	5
399	13	0	0	0	0	20	2	0	0	28	3
400	50	0	0	0	0	20	11	0	0	59	21

Table 10: Running time of methods on their optimal threshold in milliseconds. These experiments were run on machines with 2.4GHz processor, 1024MB RAM

Topic	Method code										
ID	00	10	11	13	20	21	23	30	31	40	41
351	11519	1992	45155	13358	36349	50839	50615	1698	45221	1963	45609
352	27274	4358	63137	35472	40157	80811	79714	4457	65170	4497	65834
353	8258	1212	42654	9080	36841	47087	47238	1154	42517	1146	42524
354	12149	1800	47619	14213	37201	51842	51801	1535	47134	1533	47108
355	9470	1582	44963	11178	36589	52189	50538	1533	45601	1590	45420
356	15524	2786	54005	18670	38663	61694	61410	2697	53489	2716	53374
357	8040	1310	42312	10113	35868	49337	46975	1413	43574	1443	42952
358	7253	1028	41726	8151	35610	44608	45200	1010	42976	1010	43332
359	20084	3369	55945	20508	40336	70521	66309	3444	55611	3374	56055
360	10525	1509	45616	11367	36138	51940	49490	1415	44156	1469	44120
361	14683	2325	51458	17175	39413	60677	59553	2621	52136	2538	52313
362	6515	893	41847	6604	37224	45106	44792	806	41999	984	42209
363	13107	1933	49313	14974	38157	55391	54249	2148	47286	2127	48145
364	11406	1702	46071	11314	37437	53021	50595	1670	46599	1778	46902
365	8778	1348	42978	10515	36714	49731	48532	1300	43356	1303	43559
366	14038	2162	50528	15455	38092	59880	58345	2244	50932	2182	50824
367	10641	1494	46080	11122	37171	50562	50185	1446	44923	1600	44798
368	8476	1328	43815	10350	36310	49726	48033	1300	44196	1556	43653
369	10189	1368	43719	10489	36101	49597	48039	1307	43615	1418	43724
370	10684	1525	43687	11470	36463	50127	49279	1441	44244	1441	44130
371	13811	2448	50277	16562	38528	57896	58048	2324	51185	2399	50942
372	8755	1464	44031	10741	36952	50102	50026	1422	44642	1398	44403
373	12864	2214	49208	14870	38064	57854	58370	2105	50199	2113	50328
374	7971	1262	43294	10052	37003	48745	47203	1283	43825	1279	41861
375	8605	1601	42045	10403	35685	48764	49104	1484	43582	1385	43296
376	13594	2192	49735	17737	37762	58691	56766	2070	50584	2311	50895
377	11069	1683	46758	11975	37551	52109	51190	1785	47454	1802	47253
378	23541	3660	60707	27193	40577	74650	70131	3561	61258	3611	60579
379	11256	1538	45606	12700	37444	52246	49095	1580	44616	1762	44972
380	8572	1321	43006	10187	36527	49166	47933	1314	43049	1318	43028
381	13182	1767	47147	12874	37108	53395	52376	1717	48013	2061	47885
382	8437	1320	44081	10294	37796	50649	50030	1271	44358	1270	43974
383	11046	1769	46124	12630	37275	51593	51705	1555	45105	1546	45299
384	7741	1467	44291	9325	37937	50043	49558	1214	44676	1192	44777
385	11546	1562	45698	11651	37662	49627	49793	1824	43885	1611	45705
386	11540	1584	45506	12346	37306	52344	50800	1586	46501	1576	46412
387	6328	932	40248	7087	36285	44120	44166	908	40219	906	41204
388	10701	1459	43904	10554	37549	51245	50191	1313	45724	1314	45504
389	13142	1950	49107	15286	38554	56572	56239	1929	49220	2138	49607
390	11198	1672	46384	12667	37202	52125	51336	1540	46397	1543	46435
391	13988	2221	50817	18067	38923	58528	58830	2191	50659	2540	50563
392	11814	1743	46861	13524	37800	53652	54278	1676	47551	1680	47294
393	12395	1885	49193	13739	38682	55469	54123	1835	49659	1902	49651
394	15101	2502	51672	16237	39193	60504	58681	2773	52421	2759	52223
395	18155	2961	53879	17791	39427	62565	63051	2656	54068	2839	53105
396	11078	1529	44662	11286	35590	48879	48591	1502	44214	1502	45216
397	10679	1729	45430	12685	36946	51906	51657	1589	45752	1590	45974
398	11770	1598	46558	12579	36897	53058	52628	1637	46851	1637	47006
399	8187	1362	43446	10307	37241	49947	49790	1288	44362	1291	44152
400	7682	1154	43293	9085	37780	47303	47633	1101	44151	1105	44114

B Appendix B TxtSim Command Line Guide

B.1 Introduction

This file introduces the command line of the TxtSimConsole, which is a text console of the TxtSim system that aims to find texts similar to a given text in a file collection.

B.2 Requirements

1. JRE(Java Runtime Environment) 1.5.0. The TxtSimConsole is implemented by sun J2SE SDK 5.0, so a right version java environment must have been installed in your machine.
2. WordNet2.0. Because some methods in the TxtSim are based on WordNet.
3. A xml document (jwnl_properties.xml), which is used by JWNL, should specify the configuration of WordNet.

B.3 Command Line

java [java_settings] TxtSimConsole [options] <-m similarity_method_codes -d corpus_dir>|<-C similarity_result_file>

B.4 java_settings

-cp class_path

To set the java class path. The JWNL libs files (jwnl.jar and commons-logging.jar) must be set correctly in order to call WordNet.

-Xmx***m

To set the maximum memory used by the system, such as -Xmx400m.

B.5 TxtSimConsole Options

-a arguments_of_methods

To set the arguments of each method. The arguments between different methods should be delimited by a semicolon(";") without any white space. The arguments of a method should be delimited by a comma(",") without any white space. For example, "20,4,5;3;100,3" is a valid arguments string for 3 methods.

-b benchmark_file

To load the benchmark file. The benchmark contains all positive file names in a txt file, and each line a file name.

-c cluster_method_code

To cluster the texts into groups based on the texts' similarity. Now, there is only one text clustering method in the TxtSim system, the cluster method code is "0".

-C similarity_result_file

To compare the similarity result file with a benchmark file and measures the errors. The benchmark file must be indicated by the option [-b benchmark_file].

-d corpus_dir

> To indicate the directory that contains all texts in a corpus, in which a text is a file encoded in UTF-8.

-l stopwords_file

> To load the stopwords file. If the stop words need be removed before measuring text similarity, then a stopwords file in xml format should be set in this option.

-m similarity_method_codes

> To indicate the code(s) of the method(s) that measures the similarity between texts. A method code consists of 2 digits. The First digit, which is valid from 0 to 4, denotes how to calculate similarity between file fingerprints. The second digit, which is valid from 0 to 3, denotes how to extract fingerprints from a text.

-o output_file

> To indicate the output file name with its whole path. Two formats, i.e. xml and txt, are valid for the output file. It is decided by the extention of the output file name. See the sample files at the end of the file.

-r

> To sort the text similarity output in descending order. The result is not sorted in default.

-s seed_text

> To indicate the seed text, i.e. the query text, file name with its whole path. The file should be encoded in UTF-8.

-t threshold_step:threshold_last

> To set the step and the greatest threshold when the errors are measured in a variety of thresholds. For example, 0.01:0.2 means that the errors are measured with the threshold varying from 0.01 to 0.2 stepped by 0.01. The system default threshold step is 0.1, and the default greatest threshold is 0.9.

-w weights_of_methods

> To set the weight vector of the similarity methods, which is used in the combination function. The weight values between methods should be delimited by a semicolon(";") without any white space, such as "1;2;3" . The TxtSim system will normalize the weights automatically. The default weight of each method is 1.

-x jwnl_property_file

> To set the JWNL property file. If a method uses WordNet, then this option must be set in a right file. Because the TxtSim system calls WordNet based on JWNL.

The following is the meaning of each available method code.

00: Ferret method, namely a kind of 3-grams method.

> The method parameters:
>
> p0(>= 1), p1(>= 1).
>
> p0 is the size of the tuple. The default value is 3.
>
> p1 is the window between 2 tuples. The default value is 1.

10: A string matching method based on semantic sequences without WordNet.

The method parameters:

p0 (>= 1), p1(>= 1), p2(>= 1), p3(>= 1),p4(>= 1), p5({0,1})

p0 is the max number of the semantic strings in a file. The default value is 20.

p1 is the max word distance of the repeated word. The default value is 100.

p2 is the max span of the continued word. The default value is 5.

p3 is the min size of a semantic sequence in word. The default value is 2.

p4 is the min number of the shared words between two similar semantic sequences. The default value is 2.

p5 indicates whether or not normalize a semantic sequence, i.e. does a word appear in a semantic sequence only once? The default value is 0 (false).

11: A string matching method based on semantic sequences with WordNet.

The method parameters:

p0 (>= 1), p1(>= 1), p2(>= 1), p3(>= 1), p4(>= 1), p5({0,1}), p6({1,2,3,4})

p0 is the max number of the semantic strings in a file. The default value is 20.

p1 is the max word distance of the repeated word. The default value is 100.

p2 is the max span of the continued word. The default value is 5.

p3 is the min size of a semantic sequence in word. The default value is 2.

p4 is the min number of the shared words between two similar semantic sequences. The default value is 2.

p5 indicates whether or not normalize a semantic sequence, i.e. does a word appear in a semantic sequence only once? The default value is 0 (false).

p6 indicates the valid POS(s) of each word. The default value is 4.

12: A string matching method based on semantic sequences with WordNet.

The method parameters are the same with the method 11.

13: A string matching method based on noun phrases.

The method parameters:

p0 (>= 1), p1(>= 1), p2(>= 1), p3(>= 1)

p0 is the max number of the semantic strings in a file. The default value is 20.

p1 is the multi-word frequency threshold, namely, if the string repetition count is greater than this threshold, then the string is frequent. The default value is 1.

p2 the single word frequency threshold, namely, if the string repetition count is greater than this threshold, then the string is frequent. The default value is 3.

p3 is the max grams of a noun phrase. The default value is 4.

20: A meaning matching method based on semantic sequences.

The method parameters:

p0({1,2,3,4}), p1 (>= 1), p2 (>= 1), p3 (>= 1), p4(>= 1), p5(>= 1), p6(>= 1), p7(>= 1), p8({0,1}) p0 indicates the valid POS(s) of each word used in the similarity model. The default value is 1.

p1 is the depth to find a word's meaning in its hypernym hierarchy. The default value is 2.

p2 is the max grams of a semantic string. The default value is 4.

p3 is the max number of the semantic strings in a file. The default value is 20.

p4 is the max word distance of the repeated word. The default value is 100.

p5 is the max span of the continued word. The default value is 5.

p6 is the min size of a semantic sequence in word. The default value is 2.

p7 is the min number of the shared words between two similar semantic sequences. The default value is 2.

p8 indicates whether or not normalize a semantic sequence, i.e. does a word appear in a semantic sequence only once? The default value is 0 (false).

21: A meaning matching method based on semantic sequences.

The method parameters:

p0({1,2,3,4}), p1 (>= 1), p2 (>= 1), p3 (>= 1), p4(>= 1), p5(>= 1), p6(>= 1), p7(>= 1), p8({0,1}), p9({1,2,3,4})

p0 indicates the valid POS(s) of each word used in the similarity model. The default value is 1.

p1 is the depth to find a word's meaning in its hypernym hierarchy. The default value is 2.

p2 is the max grams of a semantic string. The default value is 4.

p3 is the max number of the semantic strings in a file. The default value is 20.

p4 is the max word distance of the repeated word. The default value is 100.

p5 is the max span of the continued word. The default value is 5.

p6 is the min size of a semantic sequence in word. The default value is 2.

p7 is the min number of the shared words between two similar semantic sequences. The default value is 2.

p8 indicates whether or not normalize a semantic sequence, i.e. does a word appear in a semantic sequence only once? The default value is 0 (false).

p9 indicates the valid POS(s) of each word used in the semantic sequences model. The default value is 4.

22: A meaning matching method based on semantic sequences.

The method parameters are the same with the method 21.

23: A meaning matching method based on noun phrases.

The method parameters:

p0({1,2,3,4}), p1 (>= 1), p2 (>= 1), p3 (>= 1), p4(>= 1), p5(>= 1), p6(>= 1)

p0 indicates the valid POS(s) of each word used in the similarity model. The default value is 1.

p1 is the depth to find a word's meaning in its hypernym hierarchy. The default value is 2.

p2 is the max grams of a semantic string. The default value is 4.

p3 is the max number of the semantic strings in a file. The default value is 20.

p4 is the multi-word frequency threshold, namely, if the string repetition count is greater than this threshold, then the string is frequent. The default value is 1.

p5 the single word frequency threshold, namely, if the string repetition count is greater than this threshold, then the string is frequent. The default value is 3.

p6 is the max grams of a noun phrase. The default value is 4.

30: Semantic Sequence Kin method (SSK) without WordNet.

The method parameters:

p0 (>= 1), p1(>= 1), p2(>= 1), p3(>= 1), p4({0,1})

p0 is the max word distance of the repeated word. The default value is 100.

p1 is the max span of the continued word. The default value is 5.

p2 is the min size of a semantic sequence in word. The default value is 2.

p3 is the min number of the shared words between two similar semantic sequences. The default value is 2.

p4 indicates whether or not normalize a semantic sequence, i.e. does a word appear in a semantic sequence only once? The default value is 1 (true).

31: Semantic Sequence Kin method (SSK) with WordNet.

The method parameters:

p0 (>= 1), p1(>= 1), p2(>= 1), p3(>= 1), p4({0,1}), p5({1,2,3,4})

p0 is the max word distance of the repeated word. The default value is 100.

p1 is the max span of the continued word. The default value is 5.

p2 is the min size of a semantic sequence in word. The default value is 2.

p3 is the min number of the shared words between two similar semantic sequences. The default value is 2.

p4 indicates whether or not normalize a semantic sequence, i.e. does a word appear in a semantic sequence only once? The default value is 1 (true).

p5 indicates the valid POS(s) of each word. The default value is 4.

32: Semantic Sequence Kin method (SSK) with WordNet.

The method parameters are the same with the method 31.

40: Common Semantic Sequence Model (CSSM) without WordNet.

The method parameters are the same with the method 30.

41: Common Semantic Sequence Model (CSSM) with WordNet.

The method parameters are the same with the method 31.

42: Common Semantic Sequence Model (CSSM) with WordNet.

The method parameters are the same with the method 31.

The part of speech (POS) codes:

- 1: noun;
- 2: noun,adjective;
- 3: noun,adjective,verb;
- 4: noun,adjective,verb,adverb.

The system can run several methods serially, and then combine the results by a linear function with the given weights. The method codes should be delimited by a semicolon(";") without any white space, such as "00;32;21".

B.6 NOTE

The similarity methods based on Noun Phrases (i.e. 13 and 23) need *.stp as input text instead of *.txt no matter the corpus or the seed document. So the TxtSimConsole always finds the noun phrase corpus texts in a special directory, i.e. "path_np" where "path" is the directory containing those *.txt files.

For example, when the method is 00 and the corpus directory is "351_pos", then the TxtSimConsole loads the *.txt files in the directory "351_pos". While the method is 13, the TxtSimConsole will load the *.stp files in the directory "351_pos_np" and the seed document must be a *.stp file too.

However, for the TxtSim command line options "-d corpus_dir" you must NOT add the suffix "_np" to a directory because the TxtSimConsole will add it for methods based on Noun Phrases. for the TxtSim command line options "-s seed_text", if the seed text is not ended with ".stp" then the TxtSimConsole will append the suffix ".stp" to the text file name for methods based on Noun Phrases. Please make sure the *.stp seed file can be found at the path. If the seed text is ended with ".stp" then it is all right for methods based on Noun Phrases.

B.7 Examples

A command line example:

```
java -Xmx400m -cp TxtSim.jar:jwnl.jar:commons-logging.jar TxtSimConsole -r -l stopwords.xml
-o ../data/train351.m31.xml -m 31 -x jwnl_properties.xml -d ../data/train/351 -s ../data/train/351/FT944-
9743.txt
```

The above line means that the TxtSim system measures similarity between the seed file (../data/train/351/FT944-9743.txt) and the files in the corpus (../data/train/351), which are the texts about the topic 351 in the Financial Times corpus, by means of SSK and considering the words in the same synonym with the default parameters. The stop words are removed in the procedure. At last, the text similarities are sorted in descending order as saved to a xml file (../data/train351.m31.xml).

The following is the top 50 lines of the output file train351.m31.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<similarities unit="file" method="SSK(SameMeaning_SS)(p=100,5,2,2,1)"
runningTime="682ms"
corpus="351" note="null"
xmlns="http://www.xjtu.edu.cn/CS/TextMining"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.xjtu.edu.cn/CS/TextMining
```

```
similarities.xsd”
xsi:noNamespaceSchemaLocation=”http://www.xjtu.edu.cn/CS/TextMining
similarities.xsd”>
<similar v1=”FT944-9743” v2=”FT911-3135” value=”1.0000”/>
<similar v1=”FT944-9743” v2=”FT942-3947” value=”1.0000”/>
<similar v1=”FT944-9743” v2=”FT944-9743” value=”1.0000”/>
<similar v1=”FT944-9743” v2=”FT933-1721” value=”0.8165”/>
<similar v1=”FT944-9743” v2=”FT921-16414” value=”0.8165”/>
<similar v1=”FT944-9743” v2=”FT932-4704” value=”0.7071”/>
<similar v1=”FT944-9743” v2=”FT944-16200” value=”0.6325”/>
<similar v1=”FT944-9743” v2=”FT934-14977” value=”0.5000”/>
<similar v1=”FT944-9743” v2=”FT943-8525” value=”0.5000”/>
<similar v1=”FT944-9743” v2=”FT934-2581” value=”0.5000”/>
<similar v1=”FT944-9743” v2=”FT942-14018” value=”0.4966”/>
<similar v1=”FT944-9743” v2=”FT943-2295” value=”0.4714”/>
<similar v1=”FT944-9743” v2=”FT944-4279” value=”0.4556”/>
<similar v1=”FT944-9743” v2=”FT943-7346” value=”0.4264”/>
<similar v1=”FT944-9743” v2=”FT943-13652” value=”0.4082”/>
<similar v1=”FT944-9743” v2=”FT933-15222” value=”0.4082”/>
<similar v1=”FT944-9743” v2=”FT921-3445” value=”0.4082”/>
<similar v1=”FT944-9743” v2=”FT943-1902” value=”0.3757”/>
<similar v1=”FT944-9743” v2=”FT934-11440” value=”0.3583”/>
<similar v1=”FT944-9743” v2=”FT944-10008” value=”0.3536”/>
<similar v1=”FT944-9743” v2=”FT943-9482” value=”0.3536”/>
<similar v1=”FT944-9743” v2=”FT934-11446” value=”0.3493”/>
<similar v1=”FT944-9743” v2=”FT923-13483” value=”0.3333”/>
<similar v1=”FT944-9743” v2=”FT923-9547” value=”0.3086”/>
<similar v1=”FT944-9743” v2=”FT921-14276” value=”0.2887”/>
<similar v1=”FT944-9743” v2=”FT922-8324” value=”0.2887”/>
<similar v1=”FT944-9743” v2=”FT933-7165” value=”0.2887”/>
<similar v1=”FT944-9743” v2=”FT932-17180” value=”0.2780”/>
<similar v1=”FT944-9743” v2=”FT931-2870” value=”0.2722”/>
<similar v1=”FT944-9743” v2=”FT921-6525” value=”0.2722”/>
<similar v1=”FT944-9743” v2=”FT921-13194” value=”0.2722”/>
<similar v1=”FT944-9743” v2=”FT924-13356” value=”0.2673”/>
<similar v1=”FT944-9743” v2=”FT924-13373” value=”0.2673”/>
<similar v1=”FT944-9743” v2=”FT944-12” value=”0.2578”/>
<similar v1=”FT944-9743” v2=”FT932-6225” value=”0.2539”/>
<similar v1=”FT944-9743” v2=”FT932-13583” value=”0.2520”/>
<similar v1=”FT944-9743” v2=”FT931-16617” value=”0.2497”/>
<similar v1=”FT944-9743” v2=”FT931-13614” value=”0.2462”/>
<similar v1=”FT944-9743” v2=”FT922-8327” value=”0.2447”/>
...
</similarities>
```

The following is the same similarity but output in a txt file train351.m31.txt.

```
/// similarities unit=”file” method=”SSK(SameMeaning_SS)(p=100,5,2,2,1)”
/// runningTime=”675ms”
/// corpus=”351”
/// note=”null”
FT944-9743 FT911-3135 1.0000
FT944-9743 FT942-3947 1.0000
FT944-9743FT944-9743 1.0000
FT944-9743 FT933-1721 0.8165
FT944-9743 FT921-16414 0.8165
FT944-9743 FT932-4704 0.7071
FT944-9743 FT944-16200 0.6325
FT944-9743 FT934-14977 0.5000
FT944-9743 FT943-8525 0.5000
FT944-9743 FT934-2581 0.5000
FT944-9743 FT942-14018 0.4966
FT944-9743 FT943-2295 0.4714
FT944-9743 FT944-4279 0.4556
FT944-9743 FT943-7346 0.4264
FT944-9743 FT943-13652 0.4082
FT944-9743 FT933-15222 0.4082
FT944-9743 FT921-3445 0.4082
FT944-9743 FT943-1902 0.3757
FT944-9743 FT934-11440 0.3583
FT944-9743 FT944-10008 0.3536
FT944-9743 FT943-9482 0.3536
FT944-9743 FT934-11446 0.3493
FT944-9743 FT923-13483 0.3333
FT944-9743 FT923-9547 0.3086
FT944-9743 FT921-14276 0.2887
FT944-9743 FT922-8324 0.2887
FT944-9743 FT933-7165 0.2887
FT944-9743 FT932-17180 0.2780
FT944-9743 FT931-2870 0.2722
FT944-9743 FT921-6525 0.2722
FT944-9743 FT921-13194 0.2722
FT944-9743 FT924-13356 0.2673
FT944-9743 FT924-13373 0.2673
FT944-9743 FT944-12 0.2578
FT944-9743 FT932-6225 0.2539
FT944-9743 FT932-13583 0.2520
FT944-9743 FT931-16617 0.2497
FT944-9743 FT931-13614 0.2462
FT944-9743 FT922-8327 0.2447
FT944-9743 FT942-16608 0.2357
FT944-9743 FT931-66 0.2250
FT944-9743 FT942-9158 0.2236
FT944-9743 FT933-4757 0.2236
FT944-9743 FT911-1338 0.2236
FT944-9743 FT942-9164 0.2236
FT944-9743 FT943-11921 0.2236
...
```