Keyness in Shakespeare's Plays

Jiří Materna

Natural Language Processing Lab
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00, Brno, Czech Republic
xmaterna@fi.muni.cz, http://nlp.fi.muni.cz

Abstract. This paper describes a novel method of identifying keyness in texts using relational learning. The relational learning is based on Inductive Logic Programing (ILP) in first-order logic. Using this method we can extract first-order logic formulas satisfied in a set of positive examples and not satisfied in a set of negative examples.
We tested this method on a collection of Shakespeare's plays to identify keyness (or aboutness) of particular plays. The research was especially related to Hamlet, Prince of Denmark which was already investigated by Mike Scott [1]. He used his own tool WordSmith, based on wordlists generating. Aim of this paper is to describe another way of automatic identifying keyness and to show that this method can find more comprehensive keyness representation.

Key words: keyword, keyness, Shakespeare, ILP

1 Introduction

In our life we often use the term 'key' to identify something important. The term is so widely used that keyness seems to be generally intuitively obvious. Here, though, we must think about the term more carefully. We should distinguish between language, mind, culture and text keyness [2]. For example, the term 'thee' can be a keyword in recent texts because of its archaic nature but it can hardly be considering a keyword in Shakespeare's language. In this paper, we are interested only in the text keyness, that is, the other aspects must be excluded. In order to eliminate language, mind or culture influence, we use a referential corpus composed of texts of the same type as an investigated text. In our case we use as a referential corpus set of all Shakespeare's plays.

First, we will describe classical methods of identifying aboutness in texts (Kintsch's and van Dijk's propositional analysis, Hoey's method) and then we will aim our effort much more precisely to clarify Scott's approach and a method of keyness extraction in first-order logic.

1.1 Keyness and aboutness

In the past, text linguists worked on related issues without using term keyness. One of the most famous approaches is Kintsch's and van Dijk's propositional

Petr Sojka (Ed.): RASLAN 2007,, pp. 1–6, 2007.
© Masaryk University, Brno 2007

analysis [3]. The method starts by splitting a text into its propositional components. For example, the sentence

```
Three big green elephants were crossing the Main Street yesterday.
```

may have following propositions:

```
The elephants are three
The elephants are big
The elephants are green
The elephants were crossing the street
The street is called Main Street
It happened yesterday
```

It does not matter how each proposition is expressed. We are not dealing with words or clauses, we are handling concepts. It would be possible to replace each proposition with an abstract symbol or paraphrase in another language. Kintsch and van Dijk then proceed to study which of the propositions get referred to most in the entire set. That means, the method identifies the propositions which are most important in the sense that they get linked to most of all in the text [3]. This approach is called *macropropositions*. These macroproposition seem to be, more than the others, what the text is really about.

Another author who has tackled issue of aboutness in texts was Hoey [4]. His method is similar to the Kintsch's and Dijk's one. The difference is that it does not take propositions but whole sentences. Like Kintsh and van Dijk, Hoey seeks out the elements which are most linked. A link for Hoey is based on the plain text before him. What he counts as a link is a repetition of some kind. It need not be only a verbatim repetition but for example grammatical variants like the same lemma, synonym, hyponym or meronym.

2 Keywords and Scott's keyness analysis

The method of identifying keyness used by Scott is based on keywords. Keyword is defined like a word form that is frequent in an investigated text. Repetition here is a simple verbatim repetition, so we don't consider terms 'car' and 'cars' to be the same token.

Simple verbatim repetition alone is not, however, a good indicator of what is important and what a text is about. It is obvious that the most frequent terms will be determiners like 'the' or 'of', verbal auxiliaries and words usually occurring in general texts. These terms can hardly be good indicators of aboutness [5]. What we are looking for are terms like 'Romeo', 'Juliet', 'love', 'death', 'poison' etc. in example of Rome and Juliet.

To eliminate unwanted frequent terms, we often use a referential corpus. The referential corpus should be a set of general texts in the same language and style as an investigated text. We simply compute frequent terms for both investigated and referential corpus and exclude terms frequented in both ones.

To do this, Scott uses his own text processing tool called WordSmith [6]. This tool is based on wordlist computing. Wordlist is a list of text tokens paired with their frequency in corpus. The most useful way of arranging this list is to sort it by frequency, so you can easily filter the infrequent items. The threshold frequency which determines what terms will be considered to be a keyword is usually established experimentally.

In his work, Scott defines keyness as a set of related keywords. He noticed that keywords can be globally spread or locally concentrated in the text, so he was interested in collocational neighbors of each keyword in the text. If there are other keywords nearby, in terms of keyness, they are qualified to be a key together. The important issue is, of course, a span. In his experiments, Scott uses narrow span (1 to 5 tokens) and wide span (11 to 25 tokens). It was demonstrated that wide span rather tends to identify genre keyness, whereas the narrow span is more suitable for text keyness investigation.

3 Keyness in terms of relational learning

In contrast to previous method, relational learning express keyness not only by a set of keywords but it can represent the aboutness or concept by relations between words, their attributes or positions, and even between document segments like sentences, paragraphs or phrases. For us, the key is then a relational pattern which is frequent throughout the document.

Formally, following [7], keyness K is a set of logic formulas in first-order predicate logic with modal operators that are frequent for the document. We call such formulas frequent patterns [8]. A frequent pattern is a conjunction of predicates from a given set of predicates called background knowledge. This set must contain a keyword predicate `keyword/2`. `keyword(D, KW)` predicate holds if `KW` is a keyword for the document `D`.

Background knowledge then consists of relations that are domain independent (e.g. describing relative position of words like `before/3`, `after/3`, `follows/3`, `precedes/3`, their modal variants (e.g. `always_after/3`, `always_before/3`) or that, describing morphological and syntactical categories. E.g. `hasVerb(Sentence, Subject, Verb, Object)` returns for a given sentence a triple (subject, verb, object). Background knowledge can also contain domain dependant predicates that express semantics of a word. An example is information about synsets or hypo/hyperonymic relations obtained from domain dependent ontology.

Each frequent pattern is characterized by a level of significance. A level of significance is given by a number of instances, typically a set of words and their attributes that are covered by this formula. This level of significance is called support.

An example of a frequent pattern is below.

```
word(S, B), after(S, B, C), begCap(S, C), hasTag(S, C, 'NNP'),
after(S, C, D), hasTag(S, D, 'CC')
```

It says that "in the sentence A, there is a word B, somewhere on the right there is a word C which starts with a capital letter and has tag 'NNP' and somewhere right from the word C there is a word D with tag 'CC'".

4 An experiment

We tested this method on a collection of Shakespeare's plays. In our interest was especially Hamlet, Prince of Denmark. Our main goal was try to find some additional information about keywords found by Scott.

He defines two kinds of keywords – positive and negative. Positive keywords are the ones that are outstandingly frequent in the text and negative keywords are outstandingly infrequent comparing to referential corpus.

In Hamlet, he found following positive keywords:

```
SENSE, VERY, DEATH, LORD, DO, IT, MOST, LET, WOO'T,
PHRASE, THE, T, COULD, E'EN, WHY, OF, A, OR, THIS,
WHERETO, HOW
```

Negative keywords for all Shakespeare's plays are listed below:

```
A, AND, DOTH, FATHER, FOR, FROM,GOOD, HE, HER, HIM,
HIS, I, I'LL, IN, IT, KING, LORD, LOVE, MASTER, ME,
MOST, MY, OF, OUR, SHE, SIR, THE, THEE, THEIR, THERE,
THY, THOU, TIS, WE, WHAT, WHY, YOU, YOUR, DO
```

You can see that some keywords are unsurprising. For example 'death' is supposed to be a keyword in play about death. But what is surprising, among positive keywords there are words like 'do' or 'it'. These tokens are even negative keywords in the other plays. There is no idea why these words should be a positive key in Hamlet and relatively insignificant in the other Shakespeare's plays. Scott's tool does not provide any feature to solve this issue, so we tried to answer it using our method.

In order to ease our task we only selected four words to be analyzed:

```
DO, IT, LORD, MOST
```

From both Hamlet corpus (HC) and all Shakespeare's plays corpus (AC), we only selected the sentences containing one of the Scott's keyword (for each keyword separately). Each created document was then splitted into sentences. It implies that we did not take into account any relations that concern two or more sentences. These sentences were then tagged by Memory-Based Shallow Parser [9] on morphological and syntactical level. Information about characters and position in the play for each sentence was added too.

When the data were prepared, we employed relational learning system RAP [10]. RAP is an ILP system for mining maximal frequent patterns in first-order logic written in Prolog. It uses generate-test paradigm for finding

first-order clauses which cover at least N examples. The value N is so called minimal support and supposes to be given by user.

The most important issue in relational learning is background knowledge definition, that is, issue of what predicates can a frequent pattern consist of. Firstly, we tried to use following predicates:

`key(S)` – in the document, there is a sentence S
`hasWord(S, W)` – somewhere in the sentence S, there is a word W
`before(S, A, B)` – in the sentence S, there is a word A before B
`isWord(W, T)` – the word W has a token T
`pers(S, P)` – the sentence S is pronounced by P

Even though we set maximal pattern length to 7 there were no longer patterns than 6 found. It is, however, strongly dependent on minimal support value. We set it to 10 in order to eliminate uninteresting patterns. All maximal patterns found for keyword 'DO' you can see below. A trivial pattern with 'do' token was omitted.

```
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,I)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,and)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,in)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,it)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,me)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,my)
key(A), pers(A,hamlet), hasWord(A,B), isWord(A,B,not)
key(A), pers(A,polonius), hasWord(A,B), isWord(A,B,you)
key(A), hasWord(A,B), isWord(A,B,your)
key(A), hasWord(A,B), isWord(A,B,what)
key(A), hasWord(A,B), isWord(A,B,we)
key(A), hasWord(A,B), isWord(A,B,think)
key(A), hasWord(A,B), isWord(A,B,they)
key(A), hasWord(A,B), isWord(A,B,them)
key(A), hasWord(A,B), isWord(A,B,lord)
key(A), hasWord(A,B), isWord(A,B,know)
key(A), hasWord(A,B), isWord(A,B,is)
key(A), hasWord(A,B), isWord(A,B,his)
key(A), hasWord(A,B), isWord(A,B,but)
key(A), hasWord(A,B), isWord(A,B,as)
key(A), hasWord(A,B), hasWord(A,C), before(A,B,C),
        isWord(A,B,Do), isWord(A,C,you)
key(A), hasWord(A,B), hasWord(A,C), before(A,B,C),
        isWord(A,B,And), isWord(A,C,do)
key(A), hasWord(A,B), hasWord(A,C), before(A,B,C),
        isWord(A,B,Do), isWord(A,C,not)
```

Among frequent patterns you can find some interesting ones. For example the last one. It says that there are a lot of negations in Hamlet. That can be one

of the reasons of frequent occurring 'do' in Hamlet. Another noticeable thing is that 'do' is pronounced more often by Hamlet than other persons. But, it can be caused by preponderance of Hamlet's utterances. You can also notice that there are several other keywords among frequent patterns. It is denoted by co-occurrence these keywords with 'do'. In a similar way you can analyze the other keywords.

5 Conclusion

We described classical methods of text keyness representation and introduced a novel way based on relational learning. This method represents keyness as a set of frequent patterns in first-order logic. Hoping that this method can express keyness better then previous methods, we tried to analyze keywords in Hamlet found by Scott. It was shown on an example of 'do' keyword that relational learning can find more comprehensive information, however, found patterns was not still comprehensive enough.

In our future work we want to extend domain knowledge by other predicates and try to find some other useful information. We will also be interested in so called clouds of patterns. By a special measure we will try to identify near frequent patterns. This clouds may give us a good indication of pattern significance.

References

1. Scott, M.: Key words and key sections: Exploring shakespeare. TALC, Paris (2006)
2. Scott, M., Tribble, C.: Textual Patterns: Key words and corpus analysis in language education. Philadelphia: John Benjamins (2006)
3. Kintsch, W., Van Dijk, T.: Toward a model of text comprehension and production. Psychological Review (1978)
4. Hoey, M.: Patterns of Lexis in Text. Oxford University Press (1991)
5. Phillips, M.: Lexical Structure of Text: Discourse Analysis Monograph 12. University of Birmingham Printing Section (1989)
6. Scott, M.: Lexical analysis software for the PC. Oxford University Press (1996)
7. Popelínský, L.: Keyness and relational learning. In: Keyness in Text, University of Siena (2007)
8. Cussens, J., Džerovski, S.: Learning language in logic. Lecture notes in computer science. Lecture notes in artificial intelligence. Berlin : Springer (2000)
9. Daelemans, W., van den Bosch, A.: Memory-based language processing. Cambridge University Press (2005)
10. Blaťák, J., Popelínský, L., Nepil, M.: RAP: Framework for mining frequent datalog patterns. Proceedings of the first KDID workshop at ECML/PKDD 2002 (2002)