On key words and key patterns in Shakespeare's plays

Luboš Popelínský and Jiří Materna

Knowledge Discovery Lab and Natural Language Processing Lab
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00, Brno, Czech Republic
{popel,xmaterna}@fi.muni.cz

Abstract. This paper describes experiments in identifications of keyness in texts using relational learning. 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. We focused to Hamlet, Prince of Denmark and followed the work by Mike Scott [7]. The aim of this paper is to describe an alternative way of automatic identifying keyness and to show that this method can find more comprehensive keyness representation.

Key words: keyness, keyword, machine learning, relational learning

1 Introduction

Keyness of a term (word, multiword expression, a concept from an ontology) for a text document is usually computed as the degree with which the term frequency differs in the document and in the reference corpus. For example, the term 'flood' is a keyword in news reports on natural disasters but surely not for description of floods in Central Europe in 2001. Similarly the word '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. Various measures like information gain or statistical log-likelihood test are used [5, 8] As a result we obtain a list (or a ranked list) of terms that are key for the particular document.

Key words are important for many tasks in semantic text analysis and information retrieval. Automatic finding of key words can also help in text annotation. Comparison of key words set to a document by a human with those that have been found automatically can result in a new insight into the document.

In this contribution we propose a novel way to automatically extract keyness of a document that employs relational learning [3]. Relational learning (or inductive logic programming) aims at learning first order predicate logic formula from examples. The use of this learning method allows to express not only keyness of a keyword but also relations between keywords and between keywords and morphological, syntactical or domain knowledge.

Informally, keyness is a cloud of words or terms (e.g. groups of words with similar meaning) that we call keywords in the rest of this text, their attributes (e.g. morphological, syntactical, or obtained from an ontology) and relations between them that characterizes a given document.

Frequent patterns (also called large item sets) have been originally defined as propositional formulas that are true for at least a given fraction of items in a database [1]. The example is a set of baskets of consumers in a supermarket. In this case, the frequent patterns brings an information about products that appear frequently together in those baskets

A frequent pattern in predicate logic is a logical conjunction of elementary formulas (atoms) that is frequent for a given data. E.g.

```
"there is a word Italian" AND
"somewhere after there is a word player" AND
"player is followed by a noun"
```

is an example of a pattern that consists of three atoms, `"there is a word Italian"`, `"somewhere after there is a word player"` and `"player is followed by a noun"`. 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 *minimal support*. Another example of a frequent pattern is

```
"there is a word B" AND
"somewhere on the right there is a word C" AND
"C starts with a capital letter" AND
"C has tag 'NNP' " AND
"somewhere on the right there is a word D" AND
"D has a tag 'CC'
```

It should be noticed that words `B,C,D` are not necessarily keywords. The pattern only says that key information concerns three words and describes their relative position in a sentence together with their lexical and morphological features.

It is important to say that such patterns (that work with word order) is difficult or impossible to learn with other learning algorithms, like neural networks or Naive Bayes learner, without non-trivial human assistance.

The structure of this paper is following. In Section 2 we review the most famous approaches to keyness. Then we aim our effort much more precisely to clarify Scott's approach. In Section 3 we introduce a novel method of keyness extraction based on relational learning. The texts used in the experiments are described in Section 4. Section 5 brings results of experiments. We conclude with concluding remarks in Section 6.

2 Key words and keyness

2.1 Approaches to keyness

One of the most famous approaches to keyness is Kintsch's and van Dijk's propositional analysis [9]. 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 [9]. 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.

2.2 Mike 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. 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 reference corpus. The reference 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 reference 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 col-locational 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.

Mike Scott defined 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 a reference corpus. In Hamlet Scott 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

Several keywords are unsurprising. For example DEATH is supposed to be a keyword in plays about death. However, among positive keywords there are words like DO, LORD or IT. Moreover, these tokens are negative keywords in the other plays.

In this work we aim at explaining (of course, only partial) why these words - namely DO and LORD - are positive keys in Hamlet and insignificant for keyness and aboutness in the other Shakespeare's plays.

3 Relational frequent patterns and keyness

3.1 RAP and first-order frequent patterns

Relational learning (or inductive logic programming) [3] aims at learning first-order predicate formulas from data. It allows to discover more complex knowledge than commonly used learning algorithms. Moreover, relational learning systems can learn from data of complex structure directly, with no need of pre-processing by humans. Knowledge about a domain (again expressed in first-order logic) can be easily exploited what is another advantage.

Here we use RAP [2], a system for learning first-order maximal frequent patterns. A frequent pattern is maximal if none of its extension is frequent. As consequence, each sub-pattern of a maximal frequent pattern is also frequent.

RAP learns patterns in general-to-specific (top-down) manner. It starts with a pattern of a length 1, e.g. `"there is a word W"`. If this pattern is frequent, RAP is looking for an extension (actually specialization), e.g. `"there is a word W" AND "after B there is C"`, etc. RAP exploits the fact that many frequent patterns are redundant, i.e. cover the same instances, and therefore are not important for the user. RAP mines "interesting" maximal first-order frequent patterns (i.e. patterns that are different as possible) at first by utilizing different search strategies and several pruning methods. By relaxing the pruning criteria the user can obtain all frequent patterns.

The task of mining frequent patterns is in general of exponential complexity with respect to the number of predicates and to their arity. However, RAP can use heuristic search strategy which enables to find very quickly the patterns that are the most significant.

3.2 Formal description and background knowledge predicates

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, 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 [3]. A pattern may 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. `isPoS(Sentence,Word, 'PRP$')` is true if in a `Sentence` there is a `Word` which is a possessive pronoun. `hasVerb(Sentence, Subject, Verb, Object)` returns for a given sentence a triple (subject, verb, object). Background knowledge can also contain domain dependent predicates that express semantics of a word. An example is information about synsets or hypo/hyperonymic relations obtained from domain dependent ontology.

3.3 Method

The method consists of two steps

1. For given minimal support and a maximum length of pattern
   Learn all maximal frequent patterns
2. Given a measure of interestingness
   Select all interesting patterns

In the first step, RAP is used for finding of maximal frequent patterns for the given bias. From the set of patterns only such patterns are selected that follows a criterion of interestingness. To define a level of 'interestingness' in general is difficult. Here following Scott we say that a frequent pattern is interesting if it is infrequent for the reference corpus.

4 Data

We performed experiments with a collection of Shakespeare's plays, From both Hamlet corpus and all Shakespeare's plays corpus, 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. For each sentence we also kept information about the character and the position in the play.

All sentences were morphologically and syntactically tagged by Memory-Based Shallow Parser (MBSP) [10]. MBSP splits each sentence into chunks –name phrases, verb phrases or prepositional phrases. Moreover it can recognize borders of the subject and the object part in the sentence.Memory-based part-of-speech tagger that is a part of MBSP returns for each word its morphological category.

5 Experiments and results

The goal here was to find hidden knowledge about the keywords found by Mike Scott and maybe for dependencies between them. We focused on words DO and LORD.

5.1 Frequent patterns for the keyword `DO`

An important issue in relational learning is background knowledge definition, that is, issue of what predicates can a frequent pattern consist of. We started with the 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(S, W, T)` – in the sentence S, the word W has a token T
`pers(S, P)` – the sentence S is pronounced by P

We set maximum pattern length to 7 but there were no patterns longer than 6 found. It is, however, strongly dependent on minimal support value. The minimal support was set to 10 in order to

eliminate uninteresting patterns. RAP found 24 maximal frequent patterns

```
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)
```

This set of patterns can be split into four groups. In the first group there are patterns that concern Hamlet's speeches. We can see that e.g. the words `I`,`and`,`in`,`it`,`me`,`my`,`not` occur frequently with the word `DO`. The pattern in the second group shows that Polonius is frequently using the word `you`.

It is needed to say that the patterns in the first three groups can be found also with propositional learning algorithms but it is not the case of the temporal patterns in the fourth group. For example the last pattern says

```
key(A), hasWord(A,B), hasWord(A,C), before(A,B,C),
        isWord(A,B,'Do'), isWord(A,C,not)
```

says that DO appeared frequently together with NOT. As there are a lot of negations in Hamlet speeches it can be one of the reasons for the frequent occurrence DO. Another noticeable thing is that DO is pronounced more often by Hamlet than other persons. However, it can be caused by the dominance of Hamlet's utterances.

5.2 Hamlet: key part-of-speech tags

We also looked for morphological information - part-of-speech tags - that are frequent. We wanted to find part-of-speech tags that are key for Hamlet's speeches and are not key for the whole play. After finding frequent patterns, again with temporal operators, we analyze them using Fisher Exact Test. A summary of results together with the probability obtained with Fisher Exact Test is in the table below.

	Hamlet	Others	probability	
WRB	15	7	0.03	Wh-adverbs (HOW, WHY, WHEN)
WP	12	9	0.04	Wh-pronouns (WHO, WHAT, THAT)
IN-DT	16	11	0.01	somewhere after IN there is DT
NN-IN	15	11	0.02	somewhere after NN there is IN
DT-NN	22	15	0.02	somewhere after DT there is NN

IN preposition or subordinating conjunction
DT determiner
NN noun, singular or mass

Table 1. Key POS tags in Hamlet

First two lines of the table display knowledge that is not surprising. It is Hamlet who being often in doubts is asking questions starting with `WHO`, `WHY` or `WHEN`. Next three lines needs further exploration.

5.3 Key patterns for LORD

In this part we focused on a different task than in the previous experiments, to the use of the word `LORD` by main characters in Hamlet. We also used a different method.

LORD is a negative keyword in the reference corpus but is a positive keyword in Hamlet. In Hamlet there is together 3487 sentences and 219 out of them contains the word `LORD`. Hamlet's speeches contains 1105 out of 3487 sentences, i.e. about 32% but the word `LORD` appears only in 12 out of 291 sentences - 4.1%. We generated all

	Total (%)	LORD
Hamlet	31.7	4.1
Ophelia	4.4	10.7
Polonius	9.4	10.7
Horatio	7.8	15.5

Table 2. Frequency of sentences

patterns that were not longer than 10 literals and cover at least 3 examples. We observed that LORD frequently occurs together with MY. The possessive pronoun MY was missing only in 9 sentences out of 413, four times in Hamlet's speech.

We also explored Ophelia's speeches. The co-occurrence of'MY LORD' is again frequent. An example of a pattern

```
key(A),
pers(A,ophelia), hasW(A,B,my), isWisPoS(A,C,D,PRP$),
hasW(A,E,lord), isWisPoS(A,F,G,PRP), hasW(A,H,','),
isWisPoS(A,I,J,NN)
```

is saying that there is frequently a personal pronoun (PRP), a comma, and a noun (singular or mas) together with 'MY LORD'..

6 Conclusion

We introduced a novel way of finding key patterns in text based on relational learning. This method represents keyness as a set of frequent patterns in first-order logic. This method can find more complex key patterns directly without human intervention. We analyzed key patterns in Hamlet for several keywords found by Mike Scott, namely DO and LORD.

In future we want to analyze dependency between key patterns. It frequently happens that two patterns cover most of examples (sentences) together. We also want to explore how various distance measures match with semantics of the covered text.

References

1. Rakesh Agrawal, Tomasz Imielinski, and Arun N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of SIGMOD*, pages 207–216. ACM, 1993.
2. J. Blaťák and L. Popelínský. Mining first-order maximal frequent patterns. *Neural Network World*, 5:381–390, 2004.
3. J. Cussens and S. Džeroski. *Learning Language in Logic*. Springer-Verlag, 2000.
4. Michael Hoey. *Patterns of Lexis in Text*. Oxford University Press, 1991.
5. Christopher Tribble Mike Scott. *Textual Patterns: Key words and corpus analysis in language education*. Philadelphia: John Benjamins, 2006.
6. Mike Scott. *Lexical analysis software for the PC*. Oxford University Press, 1996.
7. Mike Scott. Key words and key sections: Exploring shakespeare. *TALC, Paris*, 2006.
8. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.
9. Kintsch W.A. and Van Dijk T.A. Toward a model of text comprehension and production. *Psychological Review*, 1978.
10. Antal van den Bosch Walter Daelemans. *Memory-based language processing*. Cambridge University Press, 2005.