# Persistence correlation analysis as an aid in searching for rare but significant textual relationships

Les Hatton
CISM, University of Kingston*

February 28, 2007

**Abstract**

This paper describes an implementation of novel search algorithms developed from the set of methods known as *Chance Discovery*. The algorithms presented here use multiple phrase correlation across entire documents and identify ubiquitous phrase connections using the inverse of the varimax statistic. The concept of co-location is extended here to include close proximity using persistence functions. The object here is to discover connections between significant events which may be rare but of considerable importance. These algorithms are exercised here on a number of texts take from Project Gutenberg. The algorithms will all be available to researchers under the GNU Public Licence (GPL) and download locations cited. An implementation of the algorithm which runs on Windows or Linux is already available and is freely downloadable.

## 1 Overview

Searching large unstructured texts for connections which may lead to fruitful new avenues for research is both challenging and fascinating and the computational capability of even a cheap modern personal computer allows previously unprecedented amounts of information to be searched quickly and efficiently. This presents many opportunities for historical and other bibliographic researches particularly as initiatives such as Project Gutenberg [2] continue to make available an increasingly large number of texts electronically to facilitate such research.

The current work is based on the original work in *Chance Discovery* carried out by [7], [5]. In this sense, it is the discovery *of* chance not the discovery *by* chance. In other words *chance* is used in the sense of an opportunity.

*L.Hatton@kingston.ac.uk, lesh@leshatton.org

## 2 Preparation

**Word selection and stemming** It is first of all assumed that a document is comprised of N sentences, $s_0, s_1, ..., s_{N-1}$. Each sentence consists of a number of words, terminated in some way, such as full stop ".". Prior to any processing, a list of non-significant words of little meaning (the *stopword* list) is prepared in the manner described by [7] and might include words such as "and" or "the". Words are also optionally pre-processed with a stemming function M(). A stemming function simply reduces related words, for example *run*, *running* and *runs*, to a common stem in the manner described by [6].

**Phrase selection** *Phrases* are defined to be sequences of words separated by *stopwords*. Finally, if a phrase consists of several words, each combination of words which is a true subset of the original phrase is optionally included as a phrase candidate. So if a phrase contains the words (abc), then *generated phrases* (ab) and (bc) could also be added.

**Significant term extraction** The document is then analysed and a list of words and phrases in decreasing order of occurrence is produced. There are a number of ways in which such ordering can be carried out.

- Generated phrases in the document are culled in favour of the phrase they are generated from if they occur at the same frequency again in the manner described in [7]
- The phrases could be sorted strictly inversely to the number of words they contain
- Some combination of phrase frequency and their length can be used

The net result of this is that the M most frequently occurring words or phrases are retained by whatever measure is used. These are known as *terms*, $t_0, t_1, ..., t_{M-1}$. All other terms are discarded.

The original work of [7] now extracts concepts it defines as *foundations*, *columns* and *roofs* (keyword extractions) from a document and graphs these based essentially on co-occurrence, i.e. when terms occur together in different locations in the document and how they are linked together. In this paper, this will be treated in a rather different way with co-occurrence being generalised to co-proximity and then treated formally as cross-correlation. Co-proximity is defined as belonging to the same sentence or a nearby sentence in a sense to be defined shortly.

## 3 Algorithm

It is now assumed that the document has been preliminarily analysed into sentences and a list $T = t_0, t_1, ..., t_{M-1}$ of the M most frequently occurring terms produced where a term is a word or a phrase. In the implementation described here, these terms can be used as nodes in a graph in the same sense as [7] or as a non-graphical HTML file with links.

Now let $c_{st}$ be the count of term t in sentence s. Define the cross-correlation between terms i and j as:-

$$X_{ij} = \sum_{s=0}^{N-1} \{c_{si} c_{sj}\} \tag{1}$$

for $\{i, j = 0, .., M-1\}$. The cross-correlation will be used as a basic measure of co-occurrence between two phrases.

**Persistence** It is perfectly possible that significant co-occurrence may occur when two terms are often associated but in nearby sentences so there is an argument that the above should be accompanied by a *persistence function* $p_k$ which spreads the definition of co-occurrence across A adjacent sentences, (k = 0,.., A-1). This gives a modified cross-correlation function:-

$$X'_{ij} = \sum_{s=0}^{N-1} \{c'_{si} c'_{sj}\} \tag{2}$$

where c' is the convolution of c with the persistence function p as defined by

$$c'_{si} = \sum_{j=0}^{A-1} \{c_{s+j,i} p_j\} \tag{3}$$

An example of a persistence function might be p = {0.25,0.5,0.25}, A = 3. Co-occurrence corresponds to a persistence function which is a delta function, i.e. a single value of 1. Co-proximity is now defined to be co-occurrence where the persistence function is not a simple delta function.

Once the cross-correlations have been calculated, a novel method of identifying the *outreach* $R_i$ of term i across a whole document is introduced. This is defined as follows:-

$$R_i = \{\frac{\{\sum_{j=1}^{N} {X'_{ij}}^2\}^2}{\sum_{j=1}^{N} {X'_{ij}}^4}\} \tag{4}$$

The larger this statistic is, the more a particular term $t_i$ correlates with all the other significant terms in $D_{terms}$. (This statistic downgrades data with a few large values at the expense of data with a large number of small values). Terms with the largest value of this can be taken as those underlying a document - they connect with many other terms. On the other hand, terms with a small value of this could be considered candidates for chance events in the sense defined below.

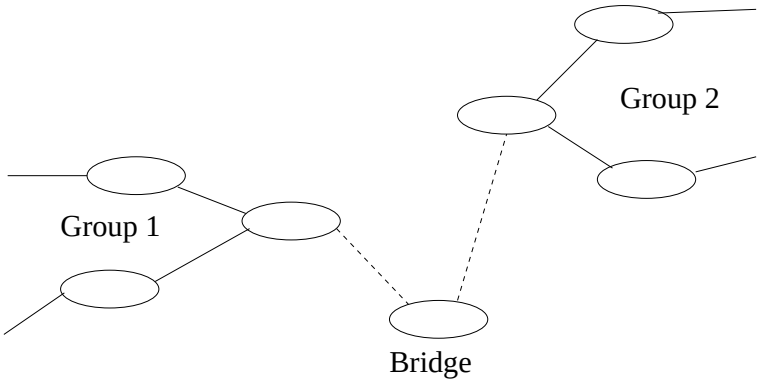


Figure 1: An example of a bridge between two sub-graphs or groups of terms which are more strongly connected. The bridge represents one form potential chance.

### 3.1 Identifying chance events

Chance events, (unlikely but significant events), are identified in several ways:-

1. *Bridge.* Any node which itself is of low outreach but which connects two nodes which are both of higher outreach, not otherwise connected, but themselves connected into separate sub-graphs of generally high outreach are identified as chances because they connect two separate sub-graphs. This is illustrated in Figure 1.

2. *Differential chance or whisper.* A differential chance or *whisper* is found by re-calculating correlations between terms after the strongest correlations have been removed. Define the differential correlation $Y'_{ijkl}$ between two pairs of terms, $\{t_i, t_j\}$ and $\{t_k, t_l\}$ as

$$Y'_{ijkl} = \sum_{s=0}^{N-1} \{(c'_{si} \oplus c'_{sj})(c'_{sk} \oplus c'_{sl})\} \tag{5}$$

for $\{i, j, k, l = 0, .., M-1\}, i \neq j \neq k \neq l$ and $\oplus$ an exclusive or operation which returns zero if a count of zero is present in either of the input terms for a given sentence number or the largest value if both are non-zero. The idea behind this is to pick up connections which might otherwise disappear under the noise. For example, consider the count vectors for four terms in a document with 10 sentences as follows,

$$\begin{aligned} c'_0 &= \{1, 0, 0, 0, 0, 6, 0, 5, 0, 1\} \\ c'_1 &= \{0, 0, 2, 0, 0, 6, 0, 5, 0, 0\} \\ c'_2 &= \{1, 0, 2, 0, 4, 0, 3, 0, 0, 1\} \\ c'_3 &= \{1, 0, 0, 0, 4, 0, 3, 0, 0, 1\} \end{aligned}$$

In other words, term $t_0$ has a count vector $c'_0$ which corresponds to occurrences of once in the first sentence, six times in the sixth sentence, five

times in the eighth sentence and once in the tenth sentence. It can be seen that terms 0 and 1 are strongly correlated and terms 2 and 3 are also strongly correlated but terms 1 and 2 are weakly correlated. Computing the differential correlation reveals this smaller but potentially interesting correlation. This is a somewhat contrived example as for any significant document, counts are usually 1 or 0 in a given sentence position for a given term, but at least this illustrates the principle behind the identification of such *whispers*.

# 4 Implementation

## 4.1 Additional functionality

In its first release (August 2006), the algorithm was enhanced in a number of ways as follows:-

- Significant word list. In a bibliographic search, it is often useful to look for specific words and how they interact in a particular document or set of documents. All words in the significant word list are rated higher than any other phrase no matter how high it naturally occurs.
- Metrication. Phrases can be sorted according to the original [7] method or using alternative metrics including phrase length or some combination of phrase length and occurrence rate.
- Word stemming. This is typically quite expensive and so is optional.

**Presentation** Displaying directed and undirected graphs is itself a significant piece of work, so here the output of the algorithm was rendered in a text format suitable for input to the well-known GraphViz, [1]. An example of the output of the algorithm with a primitive set of stopwords is shown in Listing 1. Red links identify terms with a high value of outreach.

**Listing 1: an example output by searching for all significant phrases containing "ark" and "covenant" on the King James Bible.**

```
digraph G {
     size = "8,8";
     come_unto_jordan_ [style=filled,color=red];
     come_unto_jordan_ -> ark_were_ [label="2"  ];
     covenant_that_ [style=filled,color=red];
     covenant_that_ -> are_written_in_this_book_of_ [label="1"  ];
     covenant_shall_he_ [style=filled,color=red];
     covenant_shall_he_ -> covenant_shall_ [label="1"  ];
     covenant_shall_he_ [style=filled,color=red];
     covenant_shall_he_ -> corrupt_by_ [label="1"  ];
     covenant_shall_ [style=filled,color=red];
     covenant_shall_ -> corrupt_by_ [label="1"  ];
     covenant_shall_ [style=filled,color=red];
     covenant_shall_ -> be_in_your_ [label="1"  ];
     covenant_of_thy_god_ [style=filled,color=red];
```

```
    covenant_of_thy_god_ -> be_lacking_ [label="1"  ];
    covenant_it_ [style=filled,color=red];
    covenant_it_ -> by_keeping_ [label="1"  ];
    covenant_i_ [style=filled,color=red];
    covenant_i_ -> blood_of_thy_covenant_ [label="1"  ];
    covenant_had_ [style=filled,color=blue];
    covenant_had_been_ -> covenant_had_ [label="1"  ];
    covenant_had_been_ [style=filled,color=red];
    covenant_had_been_ -> been_faultless_ [label="1"  ];
    covenant_had_also_ [style=filled,color=red];
    covenant_had_also_ -> covenant_had_ [label="1"  ];
    covenant_had_also_ [style=filled,color=red];
    covenant_had_also_ -> also_ordinances_of_ [label="1"  ];
    covenant_had_also_ [style=filled,color=red];
    covenant_had_also_ -> also_ordinances_ [label="1"  ];
    covenant_had_ [style=filled,color=red];
    covenant_had_ -> been_faultless_ [label="1"  ];
    covenant_had_ [style=filled,color=red];
    covenant_had_ -> also_ordinances_of_ [label="1"  ];
    covenant_had_ [style=filled,color=red];
    covenant_had_ -> also_ordinances_ [label="1"  ];
    came_to_ekron_ -> ark_of_god_came_ [label="1"  ];
    brought_up_from_kirjathjearim_ -> ark_of_god_had_ [label="1"  ];
    be_ye_mindful_ -> always_of_ [label="1"  ];
    be_three_ -> ark_shall_ [label="1"  ];
    be_borne_ -> ark_may_ [label="1"  ];
    ark_of_god_home_ -> ark_home_ [label="1"  ];
    ark_of_god_again_ -> again_to_jerusalem_ [label="1"  ];
    ark_abode_ -> abode_in_kirjathjearim_ [label="1"  ];
    also_ordinances_of_ -> also_ordinances_ [label="1"  ];
}
```

The corresponding diagram output by GraphVis is shown as Figure 2.

An optional method of showing the output in HTML format is used in an accompanying tool which is freely available at [3].

## 4.2 Source code

The algorithm has been implemented in about 2700 lines of C using a 300 line lexical analysis front-end built using the GNU program *flex* and will be available under the GPL. The word-stemming algorithm is an implementation of the algorithm due to [6] and is available at [4].

**Targets** This accompanying software [3], runs without change on Linux and Windows although it was developed under Linux. It is believed that GraphViz is also available on Windows but there is an optional html output in case GraphViz is not available and the software does not require GraphViz to be installed as it currently uses the HTML output option.

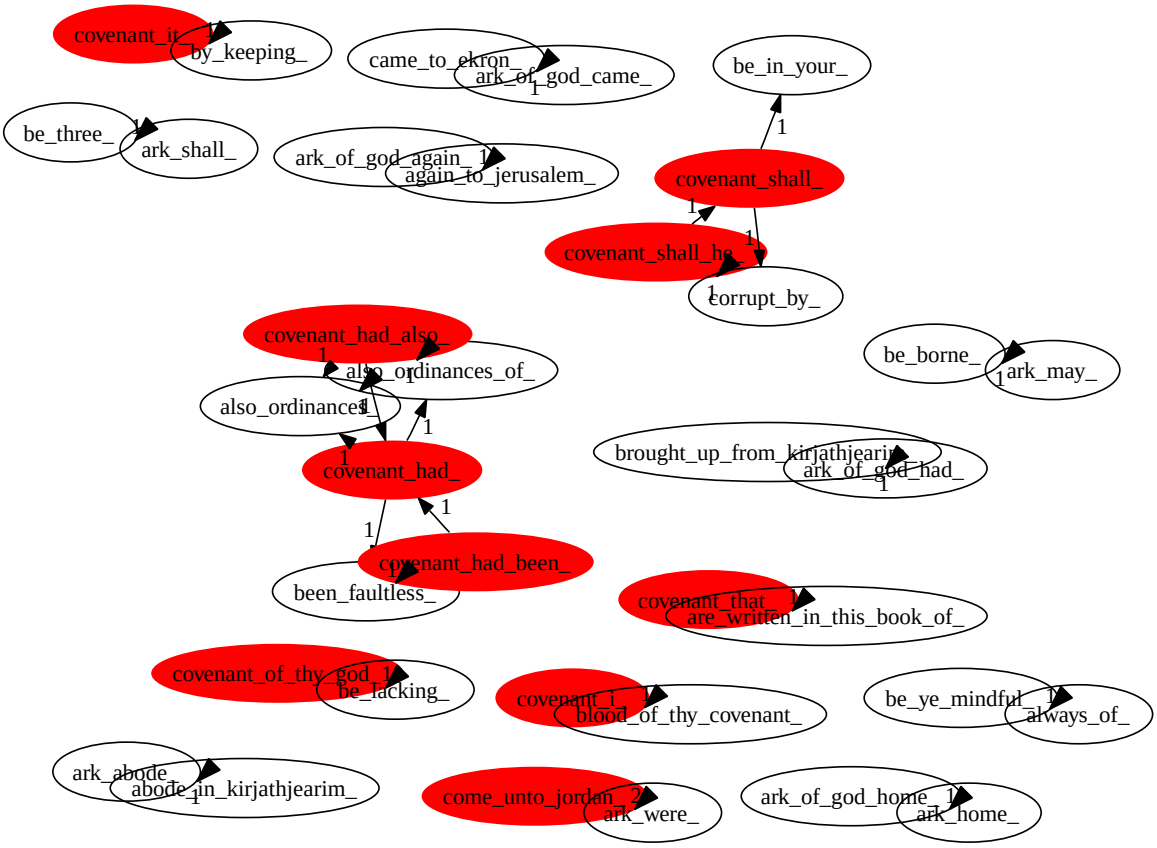


Figure 2: The output from GraphViz of the data in Listing 1.

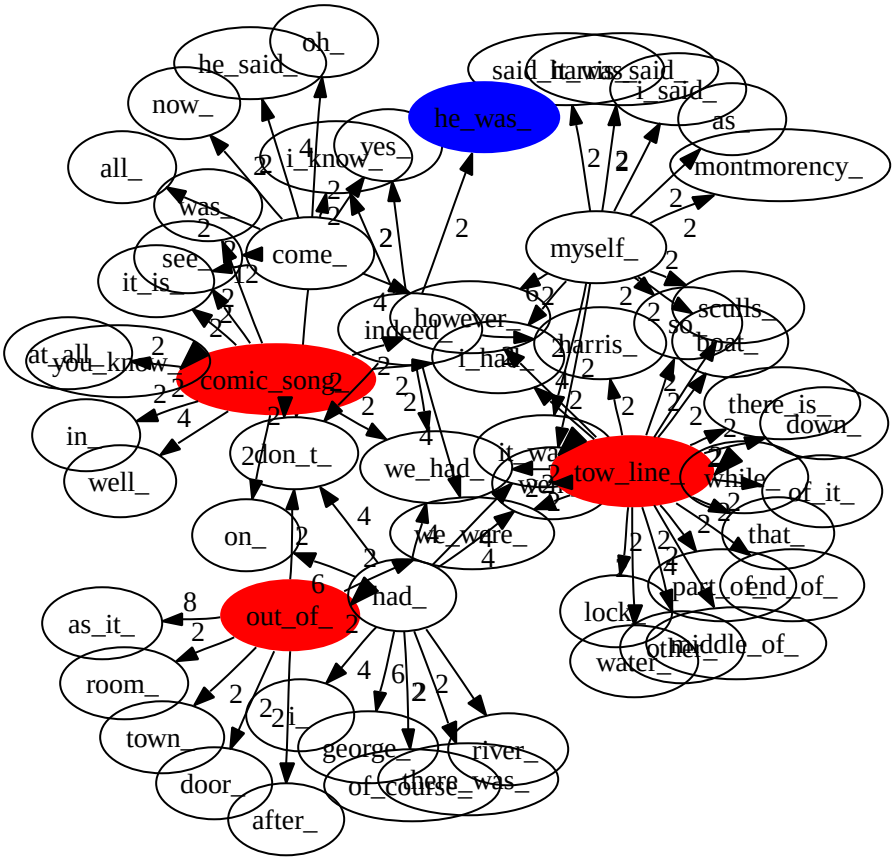


Figure 3: The output from Three Men in a Boat

# 5 Examples and Timing

This section simply details some examples of entire books taken from Project Gutenberg and passed through the algorithm described in this paper.

**Three Men in a Boat** Figure 3 shows the output of Jerome K. Jerome's Three Men In a Boat passed through with default values with word-stemming turned off. For those familiar with this gentle book, the algorithm has picked out some interesting incidents. This takes approximately 3 seconds on a 2GHz Athlon running under Linux and about 12 seconds if word-stemming is included.

Figure 4 shows the output of Jerome K. Jerome's Three Men In a Boat passed through with default values and word-stemming turned on. Word-stemming fundamentally changes the balance of importance of phrases and makes it harder to read the output so it should be considered only as an option which needs careful monitoring.

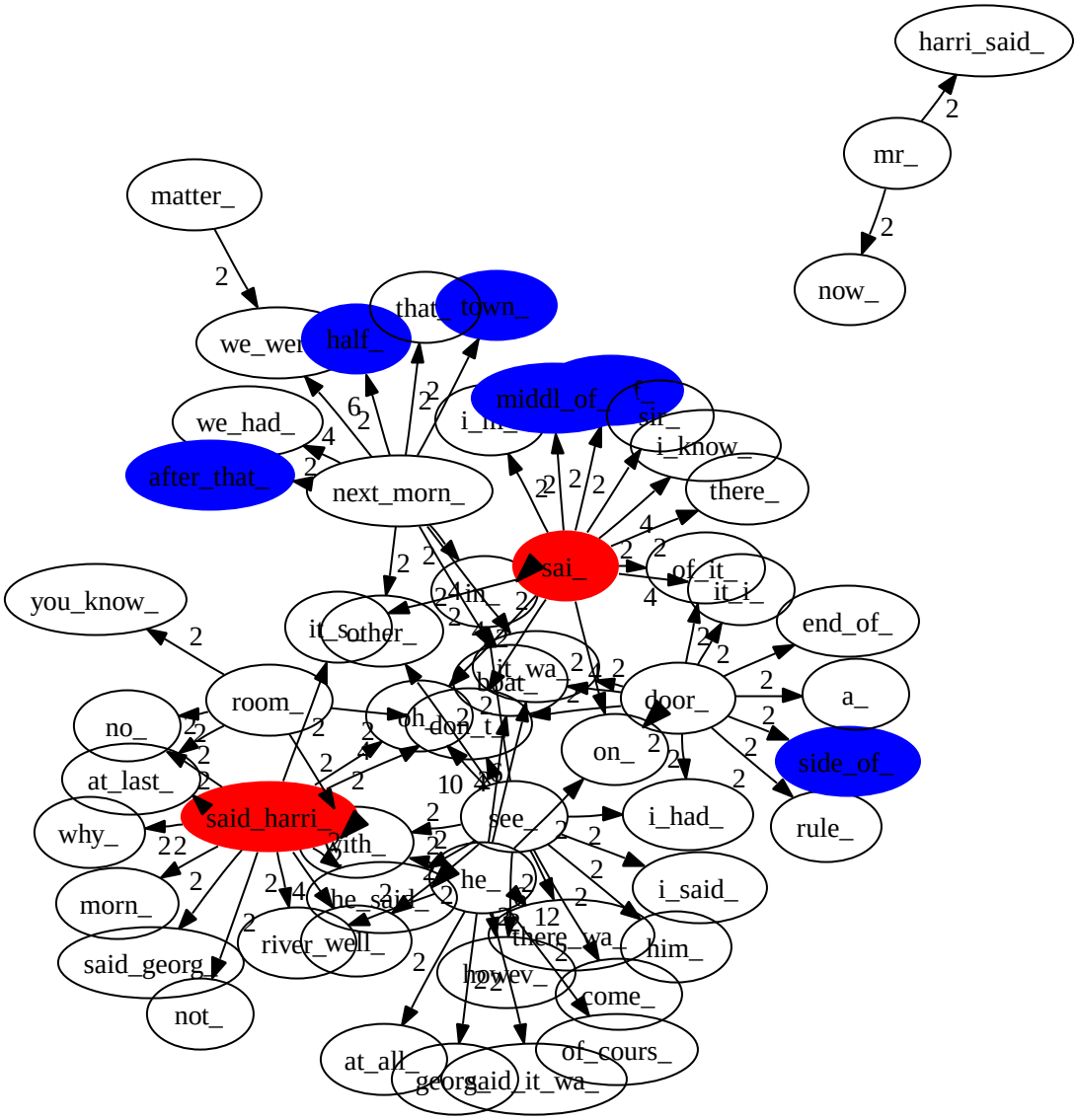


Figure 4: The output from Three Men in a Boat with word-stemming turned on.

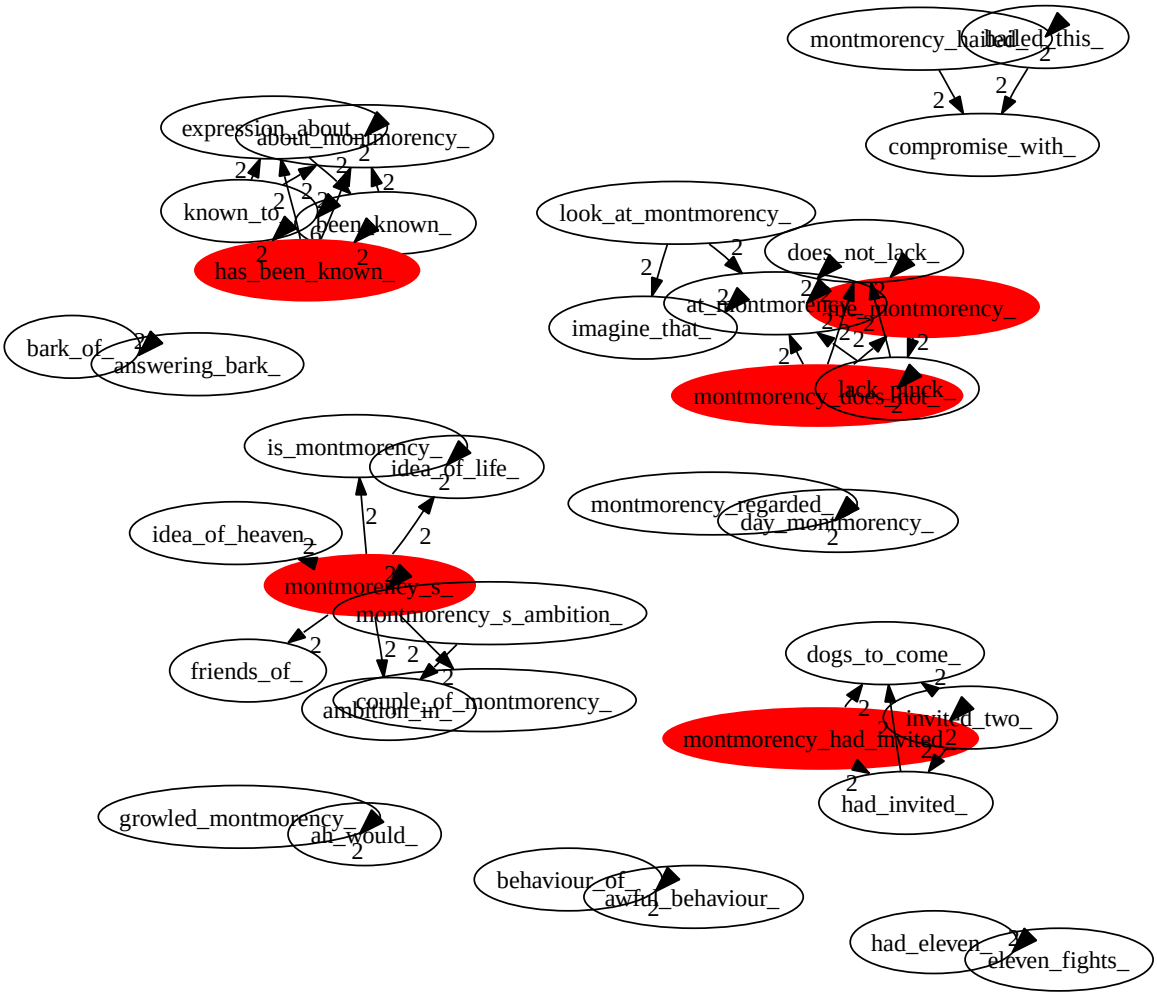


Figure 5: The output from Three Men in a Boat with the word *Montmorency* input as a significant word.

Lastly, for fans of Montmorency, the dog in this book, Figure 5 is the output when the significant word *Montmorency* is used in the analysis.

**The adventures of Sherlock Holmes** Figure 6 shows the output of Arthur Conan Doyle's The Adventures of Sherlock Holmes passed through with default values and word-stemming turned off. Note how the algorithm has picked out the Project Gutenberg header information in a separate section in the top right hand corner. This again takes approximately 3 seconds on a 2GHz Athlon running under Linux. The plot time is negligible.

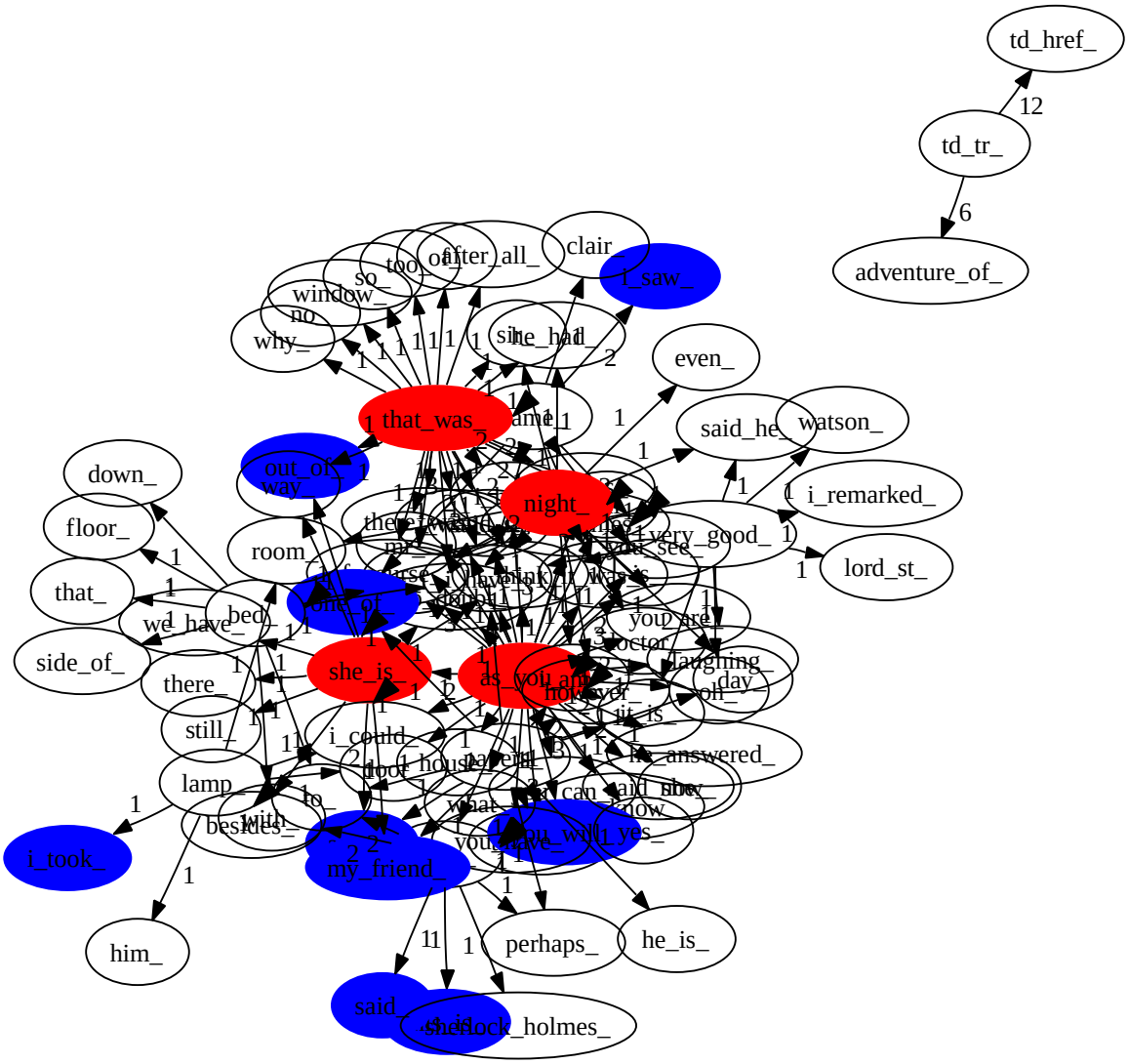


Figure 6: The output from The Adventures of Sherlock Holmes

# 6 Conclusions

An implementation of a set of algorithms to find rare but significant textual relationships in documents has been presented along with a GPL implementation which can be freely downloaded.

Although experimentation continues with setting parameters like the maximum number of nodes, links and sort mechanisms, the algorithm is exceptionally fast and is already providing intriguing insights into the significant parts of documents.

# 7 References

# References


[1] E. Koutsofios E. Gansner and S. North. A collection of programs for drawing directed and undirected graphs, 1991-. http://www.graphviz.org/.

[2] Project Gutenberg. An online collection of ebooks and documents for which copyright has expired in the united states, 1971-. http://www.gutenberg.org/.

[3] L. Hatton. An implementation of a chance discovery algorithm using cross-correlation and varimax weighting, 2006. http://www.leshatton.org/.

[4] L. Hatton. An implementation of a well-known word stemming algorithm, 2006. http://www.leshatton.org/.

[5] Y. Ohsawa. Chance discoveries for making decisions in complex real world. *New Generation Computing*, 20:143–163, 2002.

[6] M.F. Porter. An algorithm for suffix stripping. *Automated Library and Information Systems*, 14(3):130–137, 1980.

[7] N.E. Benson Y. Ohsawa and M. Yachida. Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. *Proc. of advanced digital library conference*, pages 12–18, 1998.