**Keyword Extraction Methods**

**1. Introduction:**

Keyword extraction is a text analysis method that automatically extracts the most important words and expressions from an unstructured piece of text. Keywords are important phrases/expressions that accurately represent the underlying message of the document. Keyword extraction uses natural language processing to break down human language to a form that can analyzed by machines. In this summary, we focus on unsupervised learning techniques which is applied on data that is neither classified or labelled.

**2. Statistical Approaches to Keyword Extraction:**

The basic idea of statistical approaches is to find the score of terms present in the document by implementing different types of statistical methods calculated over a document or across several documents. Next, the terms can be organized and the top n terms can be extracted.

**2.1 TF-IDF (Term Frequency-Inverse Document Frequency) :** In (TF-IDF), a score is computed for each word to signify its importance in the piece of text. We first vectorize the document by using a bag-of-words. Then, to represent a piece of text, count the number of times each word appears in the document and is put in the corresponding vector entry.

**2.1.1 Algorithm for TF-IDF:** Terminology: t – term (word), d – document, N – count of the corpus (total document set)**.** Term Frequency of a word in the document is dependent on the word t and the document d. The final value of the normalized tf will be in the range [0,1]. Document Frequency measures the importance of documents in the whole set of the corpus. df is the count of occurrences of term t in the document set N.

Inverse Document Frequency (idf) is the inverse of the document frequency which measures the informativeness of term t. idf will be low for most words such as stop words because they are present in most documents, and N/df will give a low value for that word.

However, when the value of N is large i.e., we have a large corpus, the value of idf(t) explodes. To dampen its effect, we compute the log of the idf. Moreover, since the occurrence of some words may be 0 in the corpus, the df will be 0. Since we can’t divide by 0, in order to smoothen out idf, we add 1 to df(t). Finally, we get the tf-idf (t, d) by multiplying the values of tf(t, d) and idf(t).

**2.1.2 Analysis of the Results of TF-IDF:**  The model is more accurate than a simple word frequency model, since it incorporates the inverse document frequency of a word. The model’s results are more accurate when the number of documents is large since it has a larger collection of words for its set and dictionary. However, TF-IDF takes a lot of space since it uses a dictionary to count words in a piece of text and a set to store all unique words. TF-IDF can be improved by assigning weights to words based on their position so that it can calculate the words which rely on high term frequency evenly.

**2.2 RAKE (Rapid Automatic Keyword Extraction):** RAKE employs a list of stopwords and phrase delimiters to detect the most relevant words in a piece of text. A matrix of word co-occurrences is created, where each row displays the number of times that a given content word co-occurs with every other content word in the candidate phrase. Lastly, the words in the matrix are given a score that can be calculated as the degree of a word in the matrix divided by its word frequency and ordered to get the most common keywords.

**3. Graph Based Approaches:** Graph based ranking algorithms decide the importance of a vertex within a graph, based on global information drawn from the graph. It works on the principal of voting. When a vertex links to another through an edge, it casts a vote for the other. The more the number of votes for a vertex, the more the importance of the vertex. The importance of the vertex casting the vote also determines the weight given to the vote.

Formally, let be a directed graph with set of vertices and set of edges , where E is a subset of . For a given vertex Vi, let be the set of vertices that point to it (predecessors), and let be the set of vertices that vertex Vi points to (successors). The score of a Vi can be calculated as:

where d is a damping factor, usually set as 0.85, which integrates into the model the probability of jumping from a given vertex to another random vertex in the graph. In graph-based approaches, the words present in a piece of text are represented as nodes in a graph and the edges connecting these nodes are decided based on a co-occurrence sliding window that traverses the entire document. Edges are added between all the nodes present in any particular sliding window and the graph formed is unweighted and undirected in nature. Next, we iterate through the graph until convergence and get the most common nodes.

**3.1 Text Rank:** In Text Rank, a piece of text is tokenized and annotated with part of speech tags – a preprocessing step that is required to enable application of syntactic filters. All lexical units (words) that pass the syntactic filters are added to the graph and an edge is added between those lexical units, as nodes, that co-occur within a window of n words, that create an undirected and unweighted graph is constructed. Next, a score is calculated for each vertex and the PageRank algorithm is applied for many iterations until it converges. Lastly, the vertices are sorted in reverse order of their score and the top T vertices are extracted.

**3.1.1 Analysis of the Results of Text Rank:** The text rank algorithm gives results similar to those in the research paper and online implementations for window size = 5 and by applying a syntactic filter to only consider nouns and propositions. Moreover, a larger window decreases the accuracy of the results since all words in the text receive a higher score, while a smaller window fails to capture the importance of side-by-side words in the same context.

Text Rank is efficient for fast and lightweight extraction of keywords. It can be applied on documents, articles, and any piece of text to get the underlying keywords of the piece of text that are representative of the document. It is also completely unsupervised and draws information only from text itself. However, text rank still cannot achieve the same results as that of supervised models, since a limitation is imposed on the number of keywords to be selected, which creates a smaller dataset. Text Rank can be improved by modifying the algorithm to consider the position of the words in a piece of text or by considering a set of documents instead of a single one to extract global information.

**3.2 Expand Rank:** Expand Rank creates a set of similar documents D for a given document to provide more knowledge to improve single document phrase extraction. This allows the model to use global information rather than just focusing on local information.

**3.3 Position Rank:** Position Rank is similar to Text Rank, but it incorporates the position of the word’s occurrence in a long document. Larger weights are assigned to words that are found earlier in a document compared to those that occur in the later part of the document.

**­4. Related Works:**

**4.1. Text Rank: Brining Order into Texts:**

The results and implementation of the Text Rank algorithm in this paper is based on this research paper. The problem presented in the paper is to find an unsupervised method for keyword extraction whose results compare favorably with previously published results using supervised learning model benchmarks. Text Rank relies on the local context of a text unit or vertex and takes into account information recursively drawn from the entire graph. As such, it can identify connections between keywords by the concept of recommendation. Since the algorithm is recursive, the strength of the recommendation is computed based on the importance of the units making the recommendation. This enables the algorithm to realize contextual similarities and the accurate identification of representative keywords in the text.

(<https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf>)

**4.2 Using TF-IDF to Determine Word Relevance in Document Queries:**

The results and implementation of the TF-IDF algorithm in this paper is based on this research paper. The problem presented in the paper is to determine what words in a corpus of documents are more favorable to use in a query. TF-IDF is more favorable than its simple variant TF since TFIDF can find documents that contain relevant information on the query. However, TF-IDF cannot identify the relationship between words. For larger documents, the algorithm requires a lot of space since it recursively calculates the score for each word.

(<https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1424&rep=rep1&type=pdf>)

**5. Conclusion:**

In this summarization, we implemented Text Rank, a graph based ranking model for text processing. It achieves results similar to those obtained by supervised methods, while being lightweight and does not require a deep understanding of linguistic knowledge. We also introduce TF-IDF, as a statistical method for extracting keywords. It is highly effective when applied to a large corpus of documents, since that ensures the algorithm has a large dataset to operate on. Moreover, we analyze some other graph based and statistical methods of keyword extraction and comparing their strengths and weaknesses to see if their results are practically applicable.

**References:**

1. <https://medium.com/mlearning-ai/10-popular-keyword-extraction-algorithms-in-natural-language-processing-8975ada5750c>
2. <https://monkeylearn.com/keyword-extraction/>