Document Clustering Using TextRank and Wikipedia Based Semantic Analyser

AUTHORS
Yemin Sajid, ID: 144412
Md. Ariful Islam, ID: 144416

A Thesis submitted to the Academic Faculty in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Islamic University of Technology

Bangladesh

Document Clustering Using TextRank and Wikipedia Based Semantic Analyser

Islamic University of Technology

Prepared by
Yemin Sajid - 144412
Md. Ariful Islam - 144416

Supervised by
Dr. Abu Raihan Mostafa Kamal
Professor
Department of Computer Science and Engineering
Islamic University of Technology

Declaration of Authorship

This is to certify that the work presented in this thesis is the outcome of the analysis and investigations carried out by Yemin Sajid and Md. Ariful Islam in the Academic Year 2017-2018, under the supervision of Dr. Abu Raihan Mostofa Kamal, Professor at the Department of Computer Science and Engineering (CSE), Islamic University of Technology (IUT), Gazipur, Bangladesh. It is also declared that neither this thesis nor any part of this thesis has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Authors

_____ _____

Yemin Sajid
ID: 144412

Md. Ariful Islam
ID: 144416

_____

Dr. Abu Raihan Mostofa Kamal
Professor
Department of Computer Science and Engineering
Islamic University of Technology
Board Bazar, Gazipur-1704, Bangladesh

Introduction

Document Clustering is an important tool for many Information Retrieval (IR) tasks. Document clustering is particularly useful in many applications such as automatic categorization of documents, grouping search engine results, building a taxonomy of documents, and others. Generally speaking, text document clustering methods attempt to segregate the documents into groups where each group represents some topic that is different than those topics represented by the other groups. The huge increase in amount of information present on web poses new challenges in clustering regarding to underlying data model and nature of clustering algorithm. Any clustering technique relies on concepts such as a data representation model, a similarity measure, a cluster model, a clustering algorithm.

The contribution of this report is as follows. First, we state the problem statement, and address the problem we want to solve. Second, we mention the research challenges we will face to solve what is stated in the problem statement. Also related researches are mentioned, where we cover some of the related researches in this area. Third, we present an approach to document clustering along with the result analysis. And also some future directions that can be taken to improve the current approach of organizing the articles.

Problem Statement

Articles like “New Iphone Announcement” and “Google’s Newest Phone” are similar. On the other hand “X has been elected as the new Mayor” and “Election 2018” are similar. We can naturally find this connection that clusters these 4 articles into 2 distinguishable groups. But
with the enormous success of the Information Society and the World Wide Web, the amount of textual electronic information available has significantly increased when given an enormous amount of articles, this become almost impossible to do it manually.

Medium, a social content sharing platform has over 60 million monthly readers. Everyday, thousands of people turn to Medium to publish their ideas and perspectives. And the number of article posted is increasing exponentially. In 2015, number of medium posts published was 1.9 million. Where, in 2016, it reached 7.5 million. That make a 140,000 number of stories written weekly in 2016.

This illustrates the enormous success of the textual electronic article in the World Wide Web. But if these articles are not properly channeled and clustered, it is almost impossible to consume the information. But at the same time, it is impossible to do this manually. Hence, comes the need for automated solution.

Research Challenge

Generally speaking, to organize the documents in separate groups it involves keyphrase extraction, semantic relatedness between keyphrases and clustering algorithm to cluster the in a group.

To break it down, an article can be composed on any topic. So we need a way to understand each article of what is written about. Keyword/keyphrase plays an important role here to understand the topic of the article. Basically a keyword/keyphrase of a document to words/phrases that seem to have higher importance in that particular document. As a result, keyword/keyphrase potentially hold the title and the topic of the whole document. So once we understand the keyword or key phrase of an article, we can tell what the article is written about.

Once we get the key phrases of all the corpus, we now have to find out that which articles are related to which of them. For that we need to find the semantic relatedness between the keyphrases hence finding the relatedness between the articles.

After calculating the semantic relatedness between the keywords/key phrases, we then have to cluster the semantically related articles together. Thus we will have different clusters of topic, separated from each other. Each cluster will contain only those articles, those who are semantically related. That means, the articles that shares similar topic will be grouped together and will be separated from other topic group. Thus organizing the massive collection of articles based on topics and making it easier for the consumer to access these articles.

So to wrap it up, the following are the research challenges that we have -

1. *Keyword/Keyphrase extraction* from the article.
2. Finding *Semantic Relatedness* between the Keyphrases.
3. *Clustering* the articles based on the semantic relatedness.

Related Research

As stated earlier, in order to cluster the articles, we had to go through Keyword/Keyphrase extraction, Semantic Relatedness and also Clustering algorithm. This section will cover some of the major relative researches in these respected fields. So followings are the summary of some of the major relative researches that was reviewed in the field of Keyword/Keyphrase extraction, Semantic Relatedness and Document Clustering algorithm.

For related research done in Keyword/Keyphrase extraction we have summarized these following research -

1. Automatic Keyphrase Extraction: A Survey of the State of the Art.
2. TextRank: Bringing Order into Texts.

And for research in the field of Semantic Relatedness, we have summarized -

3. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis.

And finally for document clustering, we have summarized -

4. Text Document Clustering on the basis of Inter passage approach by using K-means.

Summary of these mentioned researches are as follows -

Automatic Keyphrase Extraction: A Survey of the State of the Art

Kazi Saidul Hasan and **Vincent Ng**
(2014)

Introduction

Automatic Keyphrase extraction concerns the automatic selection of **Important** and **Topical** phrases from the body of a document.

Application:

- Natural Language Processing
- Information Retrieval

Text summarization, Text categorization, Opinion mining, Document Indexing

Factors of Corpora

1. **Length :** Increasing length means more candidate keyphrases.
2. **Structural consistency:** Lack of structural consistency renders structural information less useful.
3. **Topic change:** Conversation, chats tend to change the topic.
4. **Topic correlation:** Uncorrelated topics (in conversation and chats) increase the difficulty of keyphrase extraction.

Keyphrase Extraction Approaches

Two step process:

1. **Selecting Candidate Words and Phrases:** Heuristics like - using a stop word list to remove stop words, certain part of speech tags like noun adjectives verbs, allowing n-grams that appear in Wikipedia, extracting from that those satisfy pre-defined lexico-syntax
2. **Selecting Correct Keyword from the Candidates:** using Supervised and Unsupervised approaches.

Supervised Approach

It focuses on two issues:

1. **Task Reformulation:** Early approach was binary classification. The goal is to train a classifier on documents annotated with keyphrases to determine whether a candidate phrase is a keyphrase. Algorithm: Naive Bayes, Decision trees, Bagging, Boosting, Maximum entropy, multi-layer perceptron, SVM.

   Weakness: cannot determine which keyphrase is more suitable.

   Solution: Ranking approach by making a ranker learn to rank two candidate keyphrase.

2. **Feature Design:** Divided in two categories:

A. Within-Collection Features: Within the training document. Further divided into three types.

I. Statistical features: tf*idf - frequency of appearance in the document and inverse document. Distance - number of words preceding and total words ratio. Supervised Keyphraseness - occurrence in the training set. Phrase length and spread.
II. Structural features: where the candidate is located.
III. Syntactic features: Syntactic patterns of a keyphrase PoS tags, suffix sequence. Not very useful.

B. External Resource_based Features: Outside the training document - lexical knowledge bases (wikipedia), web.

Wikipedia based keyphraseness is computed as a canditated's document frequency * number of wikipedia link appearance and total appearance.
Another feature is appearance in the query log of search engine.

Another feature is semantically relatedness. Coherence features.

Unsupervised Approach

Four groups:

1. **Graph-Based Ranking:** A candidate is important if it is related to a large number of candidates and candidates that are important.

Basic idea is - to build a graph from the input document and rank its nodes according to their importance. Each node is a candidate and edges are relatedness. Edges are weighted based on the syntactic and/or semantic relevance. Top ranked candidates are selected. TextRank is well know in this area.

Drawback: Does not guarantee that all the main topics will be represented.

2. **Topic Based Clustering:** Grouping the keyphrases in topics. Potential keyphrase should be comprehensive in a sense that they should cover all the main topics in a document. Here are three approaches of this.

I. KeyCluster: Clusters semantically similar candidates using wikipedia and co-occurrence-based statistics. Each cluster is a topic. Candidates close to the centroid of each are potential keyphrases to cover the document.

   Performs better than TextRank.

   Drawback: gives equal importance to all the cluster i.e. topics.

II. Topical PageRank: Runs TextRank multiple times; once for each of topics. Final score of a candidate - sum of its scores for each of the topics, weighted by the probability of that topic in the document.

   Performs better that tf*idf and TextRank.

III. CommunityCluster: This cluster gives more weight to more important topics. But unlike TPR it extract all candidate keyphrases from an important topic assuming that even if the focus is little, it is still under that important topic.

3. **Simultaneous Learning:** Simultaneous summarization and keyphrase extraction as they can potentially benefit from each other. Because,

I. An important sentence is connected to other important sentences.
II. An important word is linked to other important words.

Based on these two assumption S-S, bipartite S-W and W-W graph is constructed.

The weight of an edge in -

I. S-S graph corresponds to their content similarity.
II. S-W graph denotes the word's importance in the sentence it appears.
III. W-W graph co-occurrence or knowledge-based similarity between two words.

Top-scored words are used to form keyphrase.

Drawback: keywords may not cover all important topics. TopicClustering can be used on W-W graph to solve the problem.

4. **Language Modeling:**

Scored based on **Phraseness** and **Informativeness**. The values are estimated using Language Models trained on **foreground** (set of document from which keyphrases are to be extracted) and **background** (general knowledge about the world, web) corpus.

Unigram LM and n-gram LM constructed respectively.

Phraseness, defined using the foreground LM, is calculated as the loss of information incurred as a result of assuming a unigram LM instead of a n-gram LM.

Informativeness is computed as the loss that results because of the assumption that the candidate is sampled from the background LM rather than the foreground LM.

The loss values are computed using KullbackLeibler divergence.

LMA uses language models instead of heuristics. Language model is trained on the background corpus to determine how unique a candidate keyphrase is to the domain represented by the foreground corpus.

Evaluation:

Evaluation Metrics

Designing evaluation metrics for keyphrase extraction is by no means an easy task. To score the output of a keyphrase extraction system, the typical approach, which is also adopted by the SemEval-2010 shared task on keyphrase extraction, is

1. to create a mapping between the key-phrases in the gold standard and those in the system output using exact match, and then
2. Score the output using evaluation metrics such as precision (P), recall (R), and F-score (F).

Evaluation Techniques :

- Exact match
- Human evaluation
- Predictions

The State of the Art

Dataset	Approach and System [Supervised?]	Score		
		P	R	F
Abstracts (*Inspec*)	Topic clustering (Liu et al., 2009b) [×]	35.0	66.0	45.7
Blogs	Topic community detection (Grineva et al., 2009) [×]	35.1	61.5	44.7
News (DUC -2001)	Graph-based ranking for extended neighborhood (Wan and Xiao, 2008b) [×]	28.8	35.4	31.7
Papers (SemEval -2010)	Statistical, semantic, and distributional features (Lopez and Romary, 2010) [✓]	27.2	27.8	27.5

Analysis

With the goal of providing directions for future work, we identify the errors commonly made by state-of-the-art key-phrase extractors below.

Error Analysis

- Over-generation errors
- Infrequency errors
- Redundancy errors
- Evaluation errors

Recommendations

We recommend that background knowledge be extracted from external lexical databases (e.g., YAGO2 (Suchanek et al., 2007), Freebase (Bollacker et al., 2008), BabelNet (Navigli and Ponzetto, 2012)) to address the four types of errors discussed above.

Conclusion and Future Directions

1. Incorporating background knowledge.
2. Handling long documents.
3. Improving evaluation schemes.

TextRank: Bringing Order into Texts

Rada Mihalcea and **Paul Tarau**
(2004)

Introduction

A graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information.

Applying a similar line of thinking to lexical or semantic graphs extracted from natural language documents, results in a graph-based ranking model that can be applied to a variety of natural language processing applications, where knowledge drawn from an entire text is used in making local ranking/selection decisions. Such text-oriented ranking methods can be applied to tasks ranging from automated extraction of keyphrases, to extractive summarization and word sense disambiguation.

We introduce the TextRank graph-based ranking model for graphs extracted from natural language texts. We investigate and evaluate the application of TextRank to two language processing tasks consisting of unsupervised keyword and sentence extraction, and show that the results obtained with TextRank are competitive with state-of-the-art systems developed in these areas.

The TextRank Mode

The basic idea implemented by a graph-based ranking model is that of "voting" or "recommendation". When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting the vote determines how important the vote itself is, and this information is also taken into account by the ranking model.

Formally, let $G = (V, E)$ be a directed graph with the set of vertices V and set of edges E, where E is a subset of $V \times V$. For a given vertex V_i, let $In(V_i)$ be the set of vertices that point to it (predecessors), and let $Out(V_i)$ be the set of vertices that vertex V_i points to (successors). The score of a vertex V_i is defined as follow-

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

Starting from arbitrary values assigned to each node in the graph, the computation iterates until convergence below a given threshold is achieved. After running the algorithm, a score is associated with each vertex, which represents the "importance" of the vertex within the graph. Notice that the final values obtained after TextRank runs to completion are not affected by the choice of the initial value, only the number of iterations to convergence may be different.

Undirected Graphs

For loosely connected graphs, with the number of edges proportional with the number of vertices, undirected graphs tend to have more gradual convergence curves.Figure 1 plots the convergence curves for a randomly generated graph with 250 vertices and 250 edges, for a convergence threshold of 0.0001. As the connectivity of the graph increases (i.e. larger number of edges), convergence is usually achieved after fewer iterations, and the convergence curves for directed and undirected graphs practically overlap.

Weighted Graphs

It is unusual for a page to include multiple or partial links to another page, so the graph is unweighted. Buti n our model the graphs are build from natural language texts, and may include multiple or partial links between the units (vertices) that are extracted from text. It may be therefore useful to indicate and incorporate into the model the "strength" of the connection between two vertices V_i and V_j as a weight W_{ij} added to the corresponding edge that connects the two vertices. So, we introduce a new formula for graph-based ranking that takes into account edge weight-

$$WS(V_i) = (1-d) + d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j)$$

Figure 1 plots the convergence curves for the same sample graph that was used, with random weights in the interval 0–10 added to the edges. While the final vertex scores (and therefore rankings) differ significantly as compared to their unweighted alternatives, the number of iterations to convergence and the shape of the convergence curves is almost identical for weighted and unweighted graph.

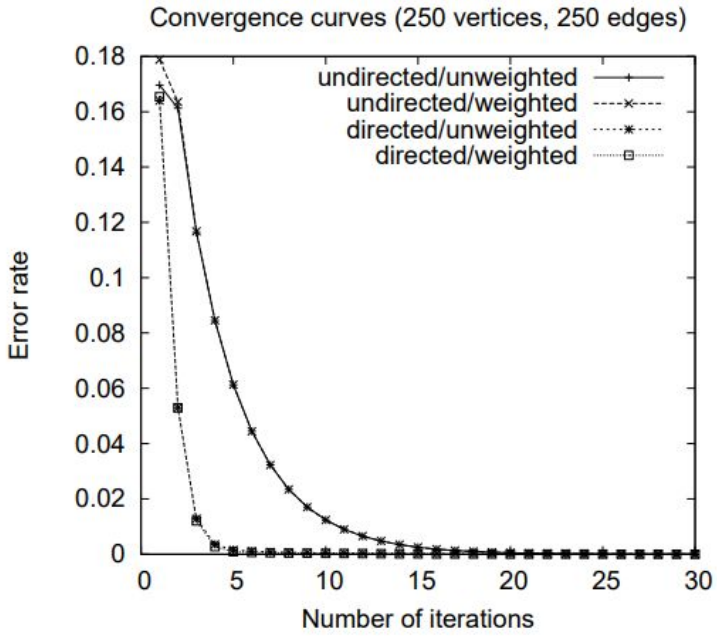

Figure: Convergence curves for graph-based ranking: directed/undirected, weighted/unweighted graph, 250 vertices, 250 edges.

Text as a Graph

To enable the application of graph-based ranking algorithms to natural language texts, we have to build a graph that represents the text, and interconnects words or other text entities with meaningful relation. Regardless of the type and characteristics of the elements added to the graph, the application of graph-based ranking algorithms to natural language texts consists of the following main step-

1. Identify text units that best define the task at hand, and add them as vertices in the graph.
2. Identify relations that connect such text units, and use these relations to draw edges between vertices in the graph. Edges can be directed or undirected, weighted or unweighted.
3. Iterate the graph-based ranking algorithm until convergence.
4. Sort vertices based on their final score. Use the values attached to each vertex for ranking/selection decisions.

TextRank for Keyword Extraction

The TextRank keyword extraction algorithm is fully unsupervised, and proceeds as follows-

1. First the text is tokenized, and annotated with part of speech tags – a preprocessing step required to enable the application of syntactic filters.To avoid excessive growth of the graph size we

consider only single words as candidates for addition to the graph, with multi-word keywords being eventually reconstructed in the post-processing phase.

2. All lexical units that pass the syntactic filter are added to the graph, and an edge is added between those lexical units that co-occur within a window of N word.
3. After the graph is constructed (undirected unweighted graph), the score associated with each vertex is set to an initial value of 1, and the ranking algorithm described in previous section, is run on the graph for several iterations until it converges – usually for 20-30 iterations, at a threshold of 0.0001.
4. Once a final score is obtained for each vertex in the graph, vertices are sorted in reversed order of their score, and the top T vertices in the ranking are retained for post-processing.

For the data used in our experiments, which consists of relatively short abstracts, T is set to a third of the number of vertices in the graph. Figure shows a sample graph built for an abstract from our test collection.

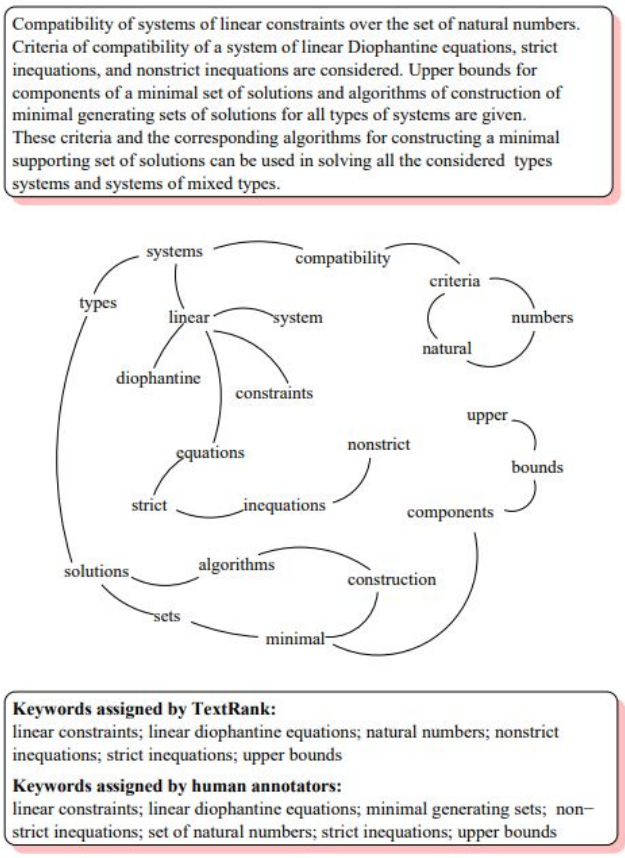

Figure: Sample graph build for keyphrase extraction from an Inspec abstract

Evaluation

The data set used in the experiments is a collection of 500 abstracts from the Inspec database, and the corresponding manually assigned keyword. The same test data set as used in the keyword extraction

experiments reported in (Hulth, 2003). In her experiments, Hulth is using a total of 2000 abstracts, divided into 1000 for training, 500 for development, and 500 for test. Since our approach is completely unsupervised, no training/development data is required, and we are only using the test document for evaluation purposes.

Method	Assigned		Correct		Precision	Recall	F-measure
	Total	Mean	Total	Mean			
TextRank							
Undirected, Co-occ.window=2	6,784	13.7	2,116	4.2	**31.2**	43.1	**36.2**
Undirected, Co-occ.window=3	6,715	13.4	1,897	3.8	28.2	38.6	32.6
Undirected, Co-occ.window=5	6,558	13.1	1,851	3.7	28.2	37.7	32.2
Undirected, Co-occ.window=10	6,570	13.1	1,846	3.7	28.1	37.6	32.2
Directed, forward, Co-occ.window=2	6,662	13.3	2,081	4.1	31.2	42.3	35.9
Directed, backward, Co-occ.window=2	6,636	13.3	2,082	4.1	31.2	42.3	35.9
Hulth (2003)							
Ngram with tag	7,815	15.6	1,973	3.9	25.2	**51.7**	33.9
NP-chunks with tag	4,788	9.6	1,421	2.8	29.7	37.2	33.0
Pattern with tag	7,012	14.0	1,523	3.1	21.7	39.9	28.1

Table: Results for automatic keyword extraction using TextRank or supervised learning (Hulth, 2003)

The results are evaluated using precision, recall, and F-measure. Overall, TextRank system leads to an F-measure higher than any of the previously proposed systems. Notice that TextRank is completely unsupervised, and unlike other supervised systems, it relies exclusively on information drawn from the text itself, which makes it easily portable to other text collections, domains, and languages.

Why TextRank Works

Intuitively, TextRank works well because it does not only rely on the local context of a text unit (vertex), but rather it takes into account information recursively drawn from the entire text (graph). Through the graphs it builds on texts, TextRank identifies connections between various entities in a text, and implements the concept of recommendation. A text unit recommends other related text units, and the strength of the recommendation is recursively computed based on the importance of the units making the recommendation. For instance, in the keyphrase extraction application, co-occurring words recommend each other as important, and it is the common context that enables the identification of connections between words in text.

Through its iterative mechanism, TextRank goes beyond simple graph connectivity, and it is able to score text units based also on the "importance" of other text units they link to. The text units selected by TextRank for a given application are the ones most recommended by related text units in the text, with preference given to the recommendations made most influential ones, i.e. the ones that are in turn highly recommended by other related units.

The underlying hypothesis is that in a cohesive text fragment, related text units tend to form a "Web" of connections that approximates the model humans build about a given context in the process of discourse understanding.

Conclusion

In this paper, we introduced TextRank – a graph-based ranking model for text processing, and show how it can be successfully used for natural language application and which performs competitively with that of previously proposed state-of-the-art algorithms. An important aspect of TextRank is that it does not require deep linguistic knowledge, nor domain or language specific annotated corpora, which makes it highly portable to other domains, genres, or language.

Review

Here, we can see that-

1. TextRank achieves better better result though being completely unsupervised.
2. It relies exclusively on information drawn from the text itself.
3. So it does not require deep linguistic knowledge, nor domain or language specific annotated corpora.
4. As a result, it easily portable to other text collections, domains, and languages.

Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis

Evgeniy Gabrilovich and **Shaul Markovitch**
(2007)

Introduction

It has long been recognized that in order to process natural language, computers require access to vast amounts of common-sense and domain-specific world knowledge. However, prior work on semantic relatedness was based on purely statistical techniques that did not make use of background knowledge or on lexical resources that incorporate very limited knowledge about the world.

This paper proposes a novel method called, Explicit Semantic Analysis (ESA) for fine-grained semantic representation of unrestricted natural language texts. This method represents meaning in a high-dimensional space of natural concepts derived from Wikipedia.

Explicit Semantic Analysis

We represents texts as a weighted mixture of a predetermined set of natural concepts, which are defined by humans themselves and can be easily explained. To achieve this aim, they have used concepts defined by Wikipedia articles. The advantages can be listed as-

1. It uses of vast amounts of highly organized human knowledge encoded in Wikipedia.
2. Wikipedia undergoes constant development so its breadth and depth steadily increase over time.

They have used machine learning techniques to build a semantic interpreter that maps fragments of natural language text into a weighted sequence of *Wikipedia concepts* ordered by their relevance to the input.This way, input texts are represented as weighted vectors of concepts, called *interpretation vectors*. Computing semantic relatedness of texts then amounts to comparing their vectors in the space defined by the concepts, for example, using the cosine metric.

To speed up semantic interpretation, they have built an inverted
index, which maps each word into a list of concepts in which it appears. They also used the inverted index to discard insignificant associations between words and concepts by removing those concepts whose weights for a given word are too low.

Let T = {wi} be input text, and let vi be its TFIDF vector, where vi is the weight of word wi. Let kj be an inverted index entry for word wi, where kj quantifies the strength of association of word wi with Wikipedia concept cj, {cj ∈ c1, . . . , cN} (where N is the total number of Wikipedia (concepts). Then, the semantic interpretation vector V for text T is a vector of length N, in which the weight of each concept cj is defined as wi∈T vi · kj. Entries of this vector reflect the relevance of the corresponding concepts to text T. To compute semantic relatedness of a pair of text fragments we compare their vectors using the cosine metric.

Figure 1 illustrates the process of Wikipedia-based semantic interpretation.

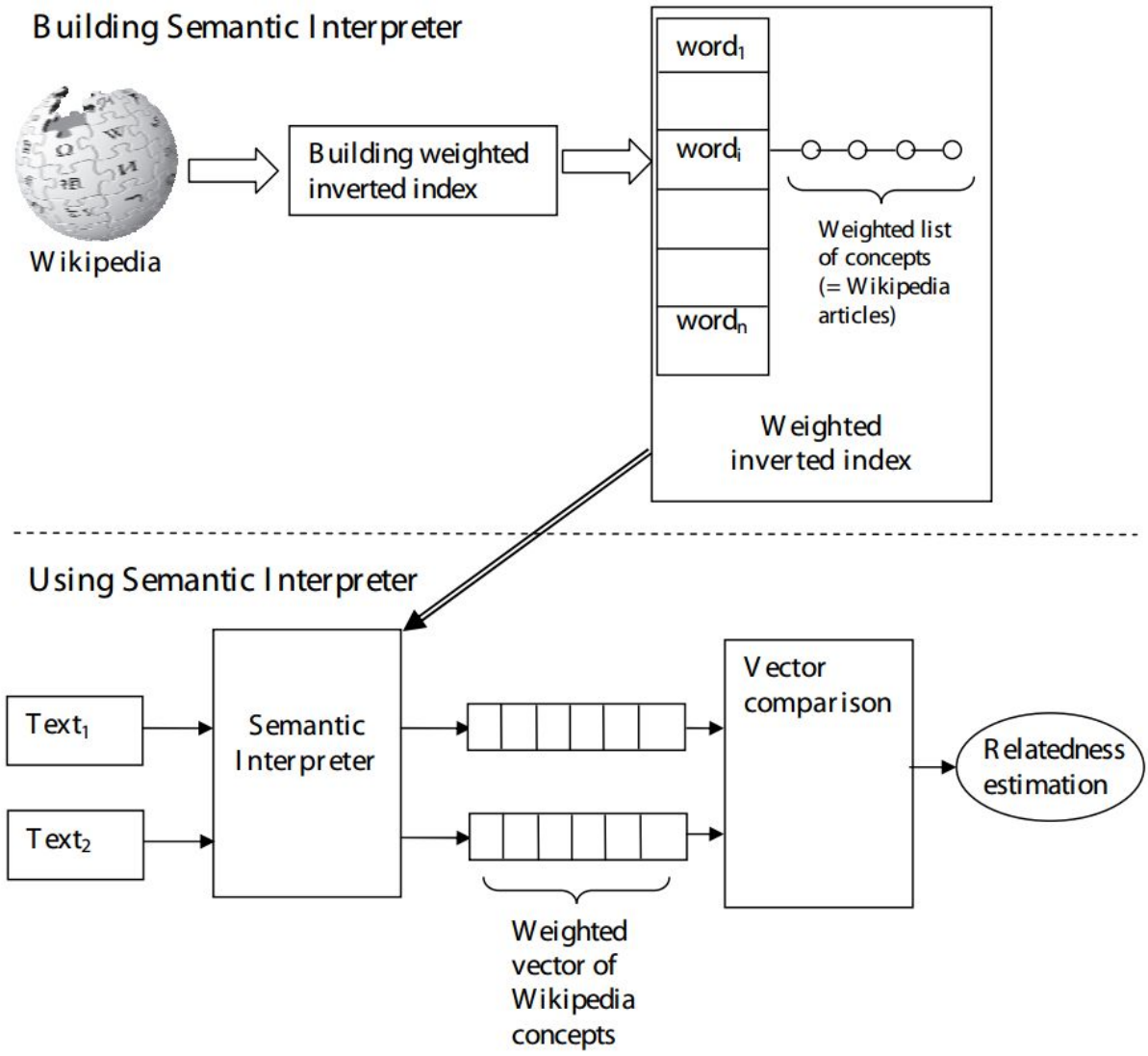

Figure 1: Semantic interpreter

Evaluation and Result

In this work, we use two such datasets, which are to the best of our knowledge the largest publicly available collections of their kind. To assess word relatedness, we use the WordSimilarity-353 collection. And for document similarity, we used a collection of 50 documents from the Australian Broadcasting Corporation's news mail service.

The following table shows the results of applying our methodology to estimating relatedness of individual words.

Algorithm	Correlation with humans
WordNet [Jarmasz, 2003]	0.33–0.35
Roget's Thesaurus [Jarmasz, 2003]	0.55
LSA [Finkelstein *et al.*, 2002]	0.56
WikiRelate! [Strube and Ponzetto, 2006]	0.19 – 0.48
ESA-Wikipedia	0.75
ESA-ODP	0.65

Table: Computing word relatedness

Algorithm	Correlation with humans
Bag of words [Lee *et al.*, 2005]	0.1–0.5
LSA [Lee *et al.*, 2005]	0.60
ESA-Wikipedia	0.72
ESA-ODP	0.69

Table: Computing text relatedness

Related Work

Prior work in the field pursued three main directions: comparing text fragments as bags of words in vector space, using lexical resources, and using Latent Semantic Analysis (LSA).

1. The former technique is the simplest, but performs sub-optimally when the texts to be compared share few words, for instance, when the texts use synonyms to convey similar messages. This technique is also trivially inappropriate for comparing individual words.

2. Lexical databases such as WordNet or Roget's Thesaurus encode relations between words such as synonymy, hypernymy. The obvious drawback of this approach is that creation of lexical resources requires lexicographic expertise as well as a lot of time and effort, and consequently such resources cover only a small fragment of the language lexicon. Specifically, such resources contain few proper names, neologisms, slang, and domain-specific technical terms. Furthermore, these resources have strong lexical orientation and mainly contain information about individual

words but little world knowledge in general.

3. On the other hand, LSA is a purely statistical technique, which leverages word co occurrence information from a large unlabeled corpus of text. LSA does not rely on any human-organized knowledge; rather, it "learns" its representation by applying Singular Value Decomposition (SVD) to the words-by-documents cooccurrence matrix.

WordNet-based techniques are similar to ESA in that both approaches manipulate a collection of concepts. There are, however, several important differences. These are,

1. WordNet-based methods are inherently limited to individual words, and their adaptation for comparing longer texts requires an extra level of sophistication. In contrast, our method treats both words and texts in essentially the same way.

2. Second, considering words in context allows our approach to perform word sense disambiguation (see Table 3). Using WordNet cannot achieve disambiguation, since information about synsets is limited to a few words (gloss); in both ODP and Wikipedia concept are associated with huge amounts of text.

3. Finally, even for individual words, ESA provides much more sophisticated mapping of words to concepts, through the analysis of the large bodies of texts associated with concepts.

Conclusion

We use Wikipedia and the ODP, the largest knowledge repositories of their kind, which contain hundreds of thousands of human-defined concepts and provide a cornucopia of information about each concept. Our approach is called Explicit Semantic Analysis, since it uses concepts explicitly defined and described by humans.

Compared to LSA, which only uses statistical co occurrence information, our methodology explicitly uses the knowledge collected and organized by humans. Compared to lexical resources such as WordNet, our methodology leverages knowledge bases that are orders of magnitude larger and more comprehensive.

Review

Explicit Semantic Analysis uses concepts explicitly defined and described by humans

1. ESA performs better than trivial algorithms.
2. Using Wikipedia is superior than using WordNet due to its massive collection highly organized human knowledge.

Text Document Clustering on the basis of Inter passage approach by using K-means

Rupesh Kumar Mishra, **Kanika Saini Kanika Saini** and **Sakshi Bagri**
(2015)

Introduction

Document clustering usually deals with clustering of documents that revolve around a single topic. To achieve more efficient clustering results, it is important to consider the fact that a document may deal with more than one topic. This research work proposes a new inter-passage based clustering technique which will cluster the segment of the documents on the basis of similarities. The input will be the collection of documents consisting of multi topic segments taken from web. SentiWordNet has been used to calculate the segment score of the segments within the documents. Based upon the segment score segment based clustering is performed on the intra-document level. Once done with intra-document segment based clustering then k-means approach is applied to the entire collection of documents to perform inter-document clustering in which the similar segments of various documents will be clustered under a single cluster.

The approach in brief-

1. Here we assume that each document consists of multiple topic segments.
2. After preprocessing, using tfidf and SentiWordNet score, important keywords regarding each segment are identified.
3. Then next, the overall segment score is computed which is the representative score of that segment and by using this segment score, K-means is applied on the segments for inter document clustering.

Algorithm For Clustering of Text Document

In this section, the algorithm for clustering of text documents is discussed in detail.

A. *Data set used*
They have assumed that the document consists of multi topic segments. We have used a total of

365 documents for clustering of documents from 20 Newsgroups.

B. *Initial Preprocessing and Cleaning*
Stop words removal and stemming is performed on the data set to obtain the key features or key terms.

C. *Identifying the keywords of each segment*
To identify the keywords, they used the overall score of words computed using the tfidf score and sentiment polarity score.

$$tf(t) = (Number\ of\ times\ term\ t\ appears\ in\ a\ document)/(Total\ number\ of\ terms\ in\ the\ document)$$
$$idf(t) = ln\ (Total\ number\ of\ documents\ /\ Number\ of\ documents\ with\ term\ t\ in\ it)$$
$$tf - idf = tf * idf$$
$$Keyword\ score(W_i) = tfidf(W_i) \times Polarity\ Score\ (W_i)$$

D. Restructuring the content of segment according to the keywords identified
After identifying the keywords of each and every segment, we identify the words whose score is nearby to the keywords using the following formula. Threshold is used to determine the relatedness.

$$SentenceScore\ (S_i) = [keywordScore\ (W_i) + Score(W_i)]/n$$

E. Calculating the segment score
After successful restructuring of the contents of the segment, we'll calculate the overall segment score. This score is calculated by averaging the score of all the words in the segment.The segment score of the segment is the representative score of the keyword.

$$SegmentScore\ (S) = (Score\ of\ all\ words\ in\ the\ segment)/(Total\ number\ of\ words\ in\ the\ segment)$$

In case, if in a document there are segments which are similar to each other, then these segments will be combined together and a new segment score will be assigned to the segment which will be the average of the score of the segments.

F. Apply k-means on these segments
K-means clustering is applied on the segments by using their representative segment score. The following steps are carried out-

a. Choose k segments randomly as initial seeds.
b. Repeat the following steps until clusters become stable that is there is no change in the cluster means score.
    i. According to the score of the initial seeds selected, assign each segment to the clusters having scores nearby to it.

ii. Update the cluster means score.

Result

Results have been summarized in table 1 and 2 from 365 newsgroup dataset. Table 1 summarizes the score of the keywords in each and every segment. Keywords are identified on the basis of tfidf and SentiWordNet score. Table 2 determines the cluster mean score and the number of cluster formed during inter document clustering.

Document Id	Segment Id	Representative Keyword of Segment	Representative Score of Segment
Doc1	Seg1	radar	.213
	Seg2	speed	.126
	Segt3	jammers	.450
Doc2	Segt1	stealth	.451
	Seg2	cars	.671
	Segt3	faster	.549
	Seg4	wanted	.761
	Seg5	contact	.579
Doc 3	Segt1	email	.341
	Seg2	programmer	.780
	Segt3	analyst	.356
	Seg4	information	.254
	Seg5	technology	.789
	Seg6	zero	.265
Doc 4	Segt1	db	.245
	Seg2	spelling	.753
	Segt3	decibel	.369
	Seg4	current	.173
Doc5	Segt1	revolutionized	.759
	Seg2	electronics	.359
	Segt3	equipment	.264
Doc6	Segt1	wanted	.244
	Seg2	items	.962
	Segt3	specializes	.259
	Seg4	modular	.367
	Seg5	stuff	.148
Doc7	Segt1	projects	.178
	Seg2	anyone	.844
	Segt3	ftp	.357
	Seg4	electronics	.790
	Seg5	plans	.128
	Segt6	response	.454
Doc8	Segt1	ladder	.543
	Seg2	movies	.262
	Segt3	curves	.573
	Seg4	wire	.577
Doc9	Segt1	base	.255
	Seg2	top	.783
	Segt3	noise	.267
	Seg4	dram	.569
	Seg5	controller	.321
Doc10	Segt1	project	.244
	Seg2	handle	.759
	Segt3	data	.146
	Seg4	manufacturer	.450
	Seg5	implement	.461

Table: Representative Keyword and score of segments

Cluster	Document Id	Segment Id	Cluster Mean Score
Cluster_1	Doc1	Seg1	.911
	Doc5	Seg3	
	Doc2	Seg3	
	Doc6	Seg5	
	Doc4	Seg	
	Doc6	Seg4	
	Doc9	Seg5	
	Doc3	Seg1	
	Doc3	Seg3	
	Doc3	Seg5	
	Doc7	Seg1	
	Doc7	Seg2	
	Doc5	Seg2	
	Doc7	Seg3	
	Doc4	Seg1	
Cluster_2	Doc10	Seg3	.423
	Doc10	Seg4	
	Doc8	Seg4	
	Doc8	Seg3	
	Doc9	Seg5	
	Doc4	Seg2	
	Doc8	Seg2	
	Doc8	Seg1	
	Doc3	Seg2	
	Doc1	Seg3	
	Doc3	Seg4	
	Doc3	Seg6	
	Doc2	Seg1	
	Doc2	Seg2	
	Doc5	Seg1	
	Doc9	Seg4	
Cluster_3	Doc6	Seg1	.639
	Doc9	Seg3	
	Doc9	Seg2	
	Doc6	Seg2	
	Doc6	Seg3	
	Doc2	Seg5	
	Doc1	Seg2	
	Doc7	Seg6	
	Doc4	Seg3	
	Doc4	Seg4	
	Doc9	Seg1	
	Doc7	Seg5	
	Doc7	Seg4	
	Doc2	Seg4	
	Doc10	Seg1	

Table: Cluster mean score when K=3

Conclusion

The approach which they have employed for clustering of text documents provides promising result and takes into account the tfi*df as well as SentiWordNet score. This approach helps in considering the word count as well as the SentiWordNet score of that word in the segment which can be useful for systematic organization of documents with the availability of large amount of documents.

This technique can be applied to web data as huge amount of web data available as there is a need for proper organization of web data. We can also extend this work by using more efficient and effective clustering approach other than K-means for clustering of documents.

Review

Here they have used tfidf and SenttWordNet score for measuring the relatedness and also K-means algorithm for clustering of the documents. From the previous papers that we reviewed, it states that although the technique used in ESA using Wikipedia and WordNet based are same, there are some key differences which makes ESA using Wikipedia superior. Those are -

1. First, WordNet-based methods are inherently limited to individual words, and their adaptation for comparing longer texts requires an extra level of sophistication. In contrast, our method treats both words and texts in essentially the same way.
2. Second, considering words in context allows our approach to perform word sense disambiguation. Using WordNet cannot achieve disambiguation, since information about synsets is limited to a few words (gloss); in both ODP and Wikipedia concept are associated with huge amounts of text.
3. Finally, even for individual words, ESA provides much more sophisticated mapping of words to concepts, through the analysis of the large bodies of texts associated with concepts.

Proposal

For document clustering, most of the work takes the leverage of Bagging approach for the Keyword/Keyphrase extraction and WordNet techniques for semantic relatedness measure. However the mentioned research articles that we mentioned established that unsupervised approach for Keyword/Keyphrase extraction performs much better than that of supervised approach.

And for semantic relatedness, Wikipedia-based ESA performs significantly better than that of WordNet. Wikipedia-based ESA uses of vast amounts of highly organized human knowledge encoded in Wikipedia. Wikipedia undergoes constant development so its breadth and depth steadily increase over time.

So, in this research, we propose a document clustering method for article clustering that uses TextRank for the important Keyword/Keyphrase extraction and then those keywords are used to calculate the semantic relatedness in between the articles. That way we will find out which articles are related to which. And then finally, with this use of article-article similarity matrix, we will cluster the similar articles in a single group. Thus forming different clusters for different topic of the articles and organizes the articles in the better and efficient way.

And because we are using TextRank for keyword extraction and Wikipedia-based ESA for semantic relatedness, this procedure should produce promising outcome because of the higher performance.

Details

To organize the documents in separate groups it involves keyphrase extraction, semantic relatedness between keyphrases and clustering algorithm to cluster the in a group. In this section, we propose a method to cluster the article based on their similarity measures. Here, we describe the algorithm in detail.

1. *Dataset Used*
   We've collected dataset from the online news website. Among them are- Time, The Guardian(UK, AUS), NYDailyNews, Cricbuzz

2. *Extracting the Keywords*
   To extract the Keywords, we have used TextRank, a Graph-based Keyword extraction technique. This algorithm extracts the keyword based on its importance in the document.

   Basically it builds a graph from the input document and rank its nodes according to their importance. Each node is a candidate and edges are relatedness. Edges are weighted based on the syntactic and/or semantic relevance. This way, every possible Keyword is scored.

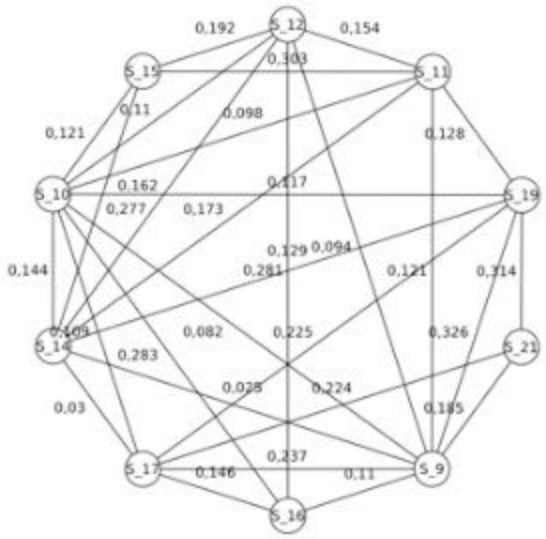

Figure: Graph model using TextRank.

3. *Candidate Keyword Selection*
   Generally, TextRank pick the top one third of the keyword based on their score. But we have used one fifth of the size of the words in a text.
4. *Calculating the Semantic Relatedness*
   After extracting K_i Keywords for each of the article, semantic relatedness between every article is measured. For two document, the score is generated by aggregating the relatedness score between the selected candidate keywords of the two article and then it is normalized to 0 to 1. The formula is -

$$sim(D_a, D_b) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} esa(K_{a_i}, K_{b_j}) * (w_i + w_j)}{\sum_{i=1}^{n} \sum_{j=1}^{m} (w_i + w_j)}$$

The following illustrates the process of article relatedness scoring

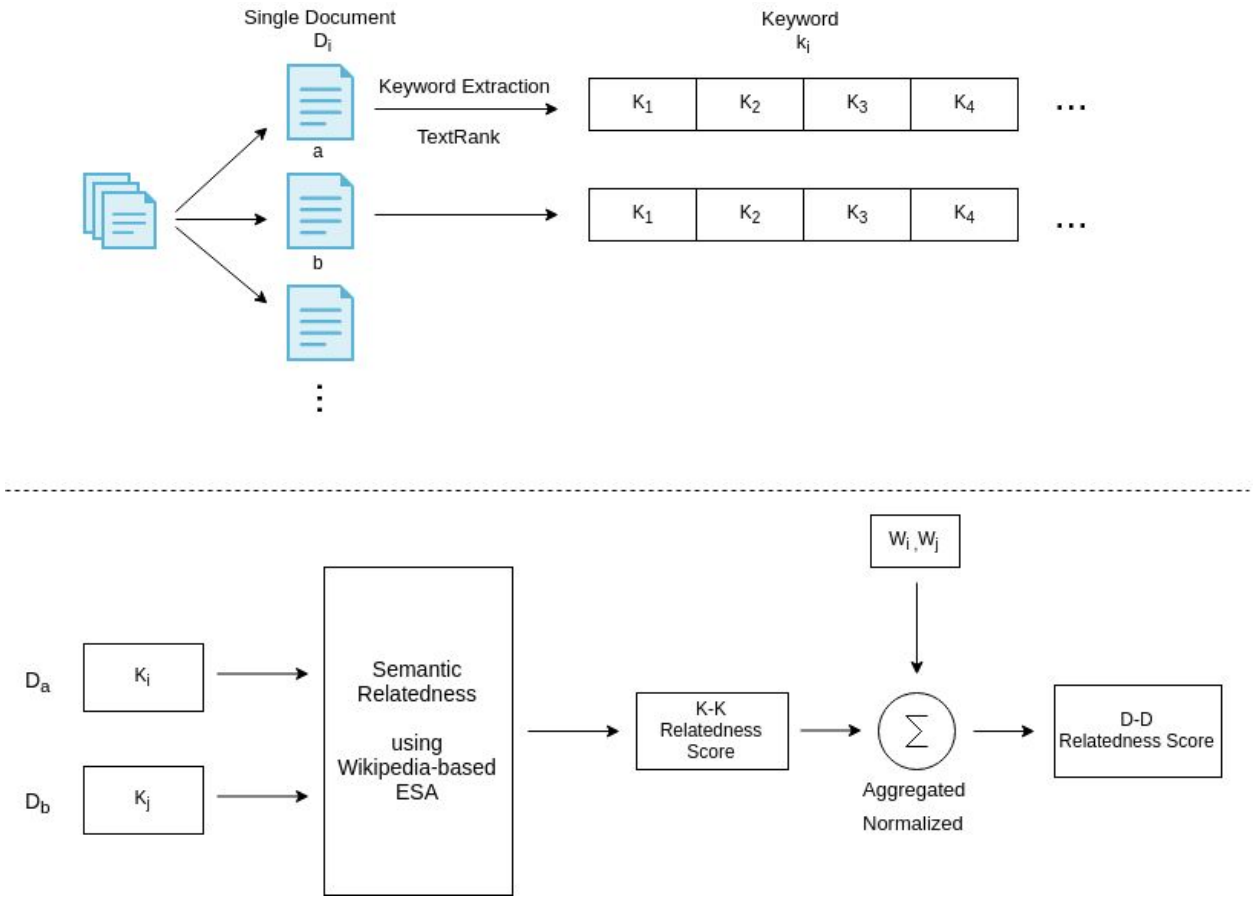

Figure: Finding article relatedness using TextRank and Wikipedia-based ESA

5. *Similarity Matrix*
   From the previous step, a similarity Matrix of $N \times N$ is generated where, N is the number of articles. Here, (i,j) is the relatedness score between news article i and news article j.

6. *Thresholding*
   In this step, we use threshold to define the relatedness between the two article. Any two article are related if they have relatedness score higher than a threshold value, U. If the relatedness score is above the threshold value, we have set it to 1. And if it is below, it is set to 0. So it can be represented as-

$$M(i,j) = \begin{cases} 1 & M(i,j) \geq U \\ 0 & M(i,j) < U \end{cases}$$

Here we have experimented with threshold value of 0.001-0.006 and through result analysis, we have selected 0.006 as our threshold value. Note that this threshold value can be domain specific.

7. *Clustering*
   After thresholding the articles are clustered from the similarity matrix. To make effective clustering we can use FACADE (Fast and Automatic Clustering Approach to Data Engineering) algorithm. It is a K-NN based algorithm and has a complexity of $O(nlog(n))$.

Experiment

- Firstly we've collected articles of two topics sports and crime.
- Then we've made an ideal relation matrix of 30x30 of these documents.
- Then we've extracted keyphrases from each topics and stored in a .csv formatted file for later steps.

```
import csv
import sys
import codecs

from textrank import extract_key_phrases

"""
Keyword Extraction using Textrank algorithm

"""

# version compatibility shim!
if sys.hexversion < 0x3000000:
# Python 2.x
   def opencsv(f): return open(f, mode="rb")
else:
# Python 3.x
    def opencsv(f): return open(f, newline="")

keywords = open('keywords.csv', 'w+')

for i in range(61, 91):
    article_f = codecs.open(
      'articles_reuters/all_articles/' + str(i), encoding='utf-8')
    article = article_f.read()

    article = article.replace(r'[^A-Za-z0-9\s\.]', '')
    keys = extract_key_phrases(article)
    keys = str(list(keys)).replace(
      '-', ' ').replace("u'", "'").replace(', ', '-')

    keywords.write('{},{}\n'.format(i, str(keys)))

    print i
```

- Then we've used the previous 'keywords.csv' file and calculated the semantic relatedness of each document

```
import csv
import sys
import json
import math
import itertools

from Wiki_ESA.cunning_linguistics import SemanticAnalyser

semantic_analyser = SemanticAnalyser()

RESULT_FILE_NAME = 'result.csv'


# version compatibility shim!
if sys.hexversion < 0x3000000:
   # Python 2.x
   def opencsv(f): return open(f, mode="rb")
else:
   # Python 3.x
   def opencsv(f): return open(f, newline="")


result_file = open(RESULT_FILE_NAME, 'w+')


"""
Semantic Analysis using Wikipedia Explicit Semantic Analysis
"""
# adding header
for i in range(1, 30):
   result_file.write(','+str(i))

result_file.write('\n')


inf = opencsv("keywords.csv")

articles = list(csv.reader(inf))
```

```
i = 1
for article1 in articles:

   result_file.write(str(i))
   for j in range(i): # padding
       result_file.write(',')


   for article2 in articles:
       # rel(a,a) not necessary and rel(a,b) = rel(b,a)
       if article1[0] >= article2[0]:
           continue
       # cleaning data to feed into json decoder
       json_data = '{ "data" : %s }' % article1[1].replace(
           '-', ',').replace("'", '"')
       article1_keys = json.loads(json_data)

       json_data = '{ "data" : %s }' % article2[1].replace(
           '-', ',').replace("'", '"')
       article2_keys = json.loads(json_data)

       total_cosine_similarity = 0
       key_count = 0

       key1_w = len(article1_keys['data'])
       key2_w = len(article2_keys['data'])


       for key in article1_keys['data']:
           key2_w = len(article2_keys['data'])
           for key2 in article2_keys['data']:
               try:
                   cs = semantic_analyser.cosine_similarity(key, key2)

                   if math.isnan(cs):
                       continue
                   total_cosine_similarity += cs * (key1_w + key2_w)

                   key_count += key1_w + key2_w
               except RuntimeWarning as e:
                   print key, key2
```

```
                key2_w -= 1

            key1_w -= 1

        is_related = 1 if float(
            total_cosine_similarity / key_count) > THRESHOLD else 0

        print article1[0], article2[0],total_cosine_similarity / key_count

        result_file.write(',{}'.format(total_cosine_similarity / key_count))


    i = i+1
    result_file.write('\n')

result_file.close()
```

- Then we've analysed the result file to check with our predefined relation matrix.
- We calculate the effectiveness by taking the difference between of these two matrices and normalising the number of ones in the whole matrix by 100%.

Result

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0.0019	0.0027	0.0064	0.0295	0.0045	0.0043	0.004	0.0027	0.0038	0.0031	0.008	0.0043	0.0032	0.0234	0.0019	0.0026	0.0017	0.0018	0.0051	0.0014	0.0006	0.0011	0.0031	0.0012	0.0007	0.0007	0.0027	0.0018	0.0015	
2		0.0023	0.0051	0.0017	0.0026	0.004	0.003	0.0024	0.0023	0.0024	0.002	0.002	0.0021	0.0033	0.0047	0.0018	0.0057	0.0043	0.0033	0.002	0.0013	0.0018	0.0023	0.0024	0.0014	0.002	0.0042	0.0026	0.003	
3			0.0019	0.003	0.0021	0.0035	0.0022	0.0021	0.002	0.0017	0.0017	0.0161	0.0033	0.0029	0.0022	0.0022	0.0015	0.0018	0.0018	0.0012	0.0011	0.0012	0.0025	0.0018	0.0009	0.0012	0.0023	0.002	0.0017	
4				0.0021	0.0026	0.0043	0.0032	0.0028	0.0028	0.0029	0.0042	0.0022	0.0036	0.0043	0.0038	0.0017	0.0021	0.0026	0.0041	0.0019	0.0018	0.0125	0.0043	0.0017	0.0022	0.0045	0.0057	0.0031	0.0046		
5					0.0185	0.0438	0.0038	0.0103	0.0093	0.0023	0.0046	0.0027	0.0159	0.0209	0.0021	0.013	0.0024	0.0027	0.0049	0.0014	0.0013	0.0006	0.0019	0.0029	0.0014	0.0014	0.0034	0.0031	0.0023			
6						0.0388	0.0276	0.0092	0.011	0.0058	0.0084	0.0025	0.0029	0.0163	0.0031	0.0019	0.0035	0.0035	0.0046	0.0046	0.0016	0.004	0.004	0.0126	0.0023	0.002	0.0038	0.0024	0.0084	
7							0.0695	0.0266	0.0396	0.0165	0.0041	0.004	0.0118	0.0448	0.005	0.0118	0.0047	0.0059	0.0054	0.0114	0.0029	0.0025	0.0069	0.0247	0.003	0.0028	0.0068	0.0049	0.0171	
8								0.0234	0.0344	0.0148	0.0095	0.0033	0.0028	0.0369	0.0048	0.0019	0.0035	0.005	0.0049	0.0112	0.0025	0.0028	0.0135	0.0256	0.0028	0.0023	0.0055	0.0036	0.0151	
9									0.0141	0.0067	0.0032	0.0032	0.0036	0.0163	0.0045	0.0021	0.0043	0.0038	0.0029	0.0053	0.0026	0.002	0.0036	0.0089	0.0021	0.0019	0.0041	0.0052	0.0075	
10										0.0072	0.0055	0.0026	0.009	0.0217	0.0039	0.0061	0.0029	0.0041	0.0033	0.0059	0.0022	0.0017	0.0048	0.0109	0.0025	0.0023	0.0056	0.0031	0.0083	
11											0.0067	0.0047	0.0078	0.0092	0.0024	0.0034	0.0024	0.003	0.0026	0.0036	0.0019	0.0018	0.0031	0.0061	0.0023	0.0021	0.0054	0.0025	0.0064	
12												0.0091	0.0081	0.0069	0.0028	0.0009	0.0021	0.0019	0.0031	0.0016	0.0047	0.0011	0.005	0.002	0.0013	0.001	0.0029	0.0018	0.0027	
13													0.0092	0.004	0.0023	0.0027	0.0022	0.002	0.0018	0.0012	0.0019	0.0008	0.0015	0.0027	0.0008	0.0008	0.0032	0.0025	0.0026	
14														0.009	0.0024	0.0107	0.0026	0.0022	0.0024	0.0023	0.0017	0.0039	0.0049	0.0022	0.0024	0.0039	0.0028	0.0026	0.0048	
15															0.0039	0.0087	0.0035	0.004	0.0042	0.007	0.003	0.0023	0.0066	0.0137	0.0023	0.002	0.0048	0.0035	0.0093	
16																0.0015	0.0044	0.007	0.0064	0.0029	0.0029	0.0033	0.0073	0.0028	0.0019	0.0079	0.0092	0.0067	0.0061	
17																	0.0032	0.0018	0.0017	0.0011	0.0004	0.0013	0.0014	0.0007	0.0006	0.0007	0.002	0.0024	0.0039	
18																		0.0041	0.0071	0.0021	0.013	0.0019	0.0055	0.0037	0.0037	0.0014	0.0059	0.0052	0.0074	
19																			0	0.0054	0.0039	0.003	0.011	0.0052	0.0078	0.0088	0.0025	0.0082	0.0085	0.0076	
20																				0.0037	0.0218	0.0049	0.0123	0.0036	0.0032	0.0081	0.0071	0.0056	0.0074	
21																					0.0228	0.0034	0.0046	0.0155	0.0096	0.0184	0.006	0.0028	0.0049	
22																						0.0043	0.0041	0.0113	0.002	0.0127	0.0088	0.0019	0.0026	
23																							0.0074	0.0155	0.0243	0.0035	0.0033	0.0019	0.0077	
24																								0.0091	0.0034	0.0119	0.0075	0.0062	0.0084	
25																									0.0159	0.0167	0.0044	0.0035	0.0119	
26																										0	0.0019	0.0032	0.0024	0.0144	
27																											0.0095	0.0071	0.008	
28																												0.0078	0.0069	
29																													0.007	
30																														

Generated output (after thresholding with 0.006)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1				1	1							1			1															
2																														
3													1																	
4																							1							
5						1	1		1	1				1	1		1													
6							1	1	1	1		1			1										1					1
7								1	1	1	1			1	1		1				1			1	1			1		1
8									1	1	1	1			1						1			1	1					1
9										1	1				1										1					1
10											1			1	1		1								1					1
11												1		1	1										1					
12													1	1	1															
13														1																
14															1		1													
15																	1				1			1	1					1
16																			1	1				1			1	1	1	1
17																														
18																				1		1								1
19																							1		1	1		1	1	1
20																						1		1			1	1		1
21																						1			1	1	1	1		
22																									1		1	1		
23																								1	1	1				1
24																									1		1	1	1	1
25																										1	1			1
26																														1
27																												1	1	1
28																													1	1
29																														1
30																														

Desired Output

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1		1	1	1	1	1	1	1	1	1	1	1	1	1	1															
2			1	1	1	1	1	1	1	1	1	1	1	1	1															
3				1	1	1	1	1	1	1	1	1	1	1	1															
4					1	1	1	1	1	1	1	1	1	1	1															
5						1	1	1	1	1	1	1	1	1	1															
6							1	1	1	1	1	1	1	1	1															
7								1	1	1	1	1	1	1	1															
8									1	1	1	1	1	1	1															
9										1	1	1	1	1	1															
10											1	1	1	1	1															
11												1	1	1	1															
12													1	1	1															
13														1	1															
14															1															
15																														
16																	1	1	1	1	1	1	1	1	1	1	1	1	1	1
17																		1	1	1	1	1	1	1	1	1	1	1	1	1
18																			1	1	1	1	1	1	1	1	1	1	1	1
19																				1	1	1	1	1	1	1	1	1	1	1
20																					1	1	1	1	1	1	1	1	1	1
21																						1	1	1	1	1	1	1	1	1
22																							1	1	1	1	1	1	1	1
23																								1	1	1	1	1	1	1
24																									1	1	1	1	1	1
25																										1	1	1	1	1
26																											1	1	1	1
27																												1	1	1
28																													1	1
29																														1

Result Analysis

We’ve got best result by thresholding by 0.006.
144 errors out of 450 relations.

The error rate is 32.44 %

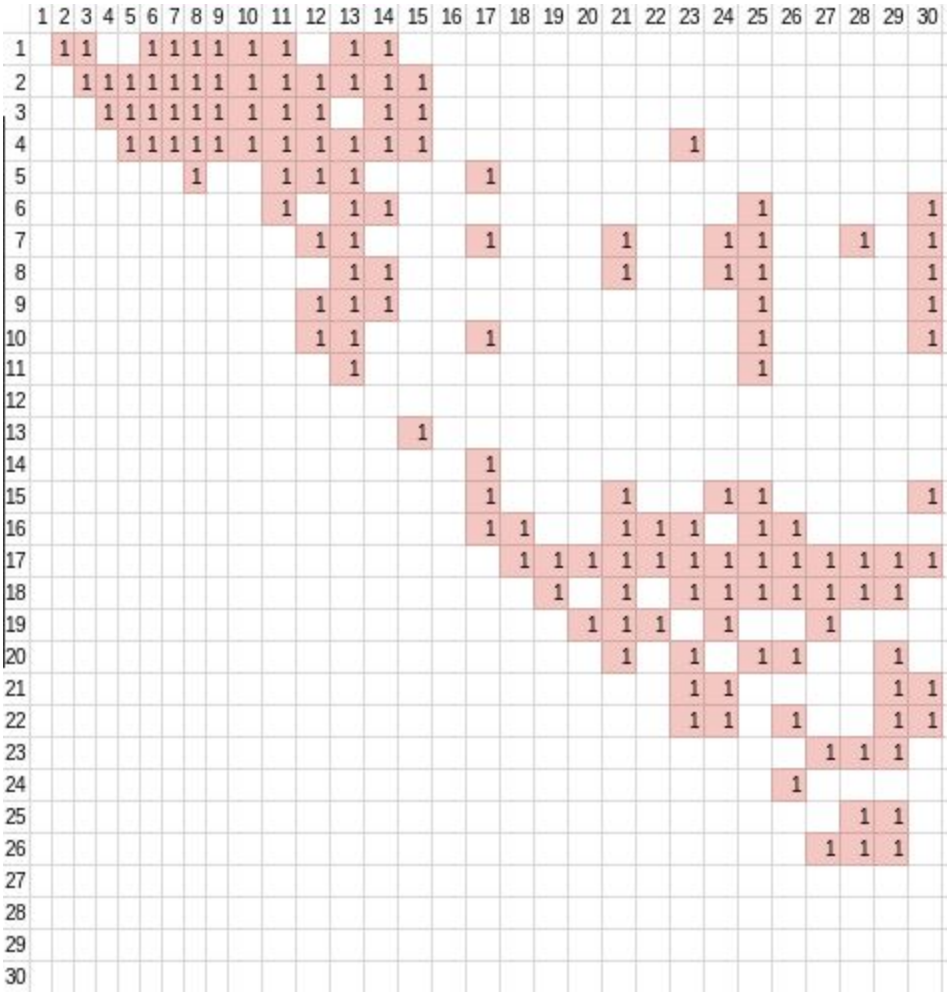

Figure : Indication of errors

Conclusion

We can easily notice huge scopes of improvements in the current approach. The result of this approach depends heavily on the TextRank and ESA algorithm. We can use more updated Wikipedia data dump, pre process the documents for better results.

This approach can give better results if optimises for a specific domain.

Another important factor is the article itself. Some article can be under certain topic and the topic related key-terms are not mentioned. Although it can be easily clustered with manual domain specific knowledge, it is challenging for automated system for clustering.

Also the inter-relations between the keywords can be used to build a hierarchical organisation of the documents in future.

References

Cui, Weiwei, et al. “TextFlow: Towards Better Understanding of Evolving Topics in Text.” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, 2011, pp. 2412–2421., doi:10.1109/tvcg.2011.239.

Hasan, Kazi Saidul, and Vincent Ng. “Automatic Keyphrase Extraction: A Survey of the State of the Art.” *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, doi:10.3115/v1/p14-1119.

Huang, Anna, et al. “Clustering Documents Using a Wikipedia-Based Concept Representation.” *Advances in Knowledge Discovery and Data Mining Lecture Notes in Computer Science*, 2009, pp. 628–636., doi:10.1007/978-3-642-01307-2_62.

Kang, Yin, et al. “An Integrated Method for Hierarchy Construction of Domain-Specific Terms.” *2014 IEEE/ACIS 13th International Conference on Computer and Information Science (ICIS)*, 2014, doi:10.1109/icis.2014.6912181.

Kaur, Bhavneet, and Sushma Jain. “Keyword Extraction Using Machine Learning Approaches.” *2017 3rd International Conference on Advances in Computing,Communication & Automation (ICACCA) (Fall)*, 2017, doi:10.1109/icaccaf.2017.8344699.

Kulathunga, Chalitha, and D.d. Karunaratne. “An Ontology-Based and Domain Specific Clustering Methodology for Financial Documents.” *2017 Seventeenth International Conference on Advances in ICT for Emerging Regions (ICTer)*, 2017, doi:10.1109/icter.2017.8257786.

Merrouni, Zakariae Alami, et al. “Automatic Keyphrase Extraction: An Overview of the State of the Art.” *2016 4th IEEE International Colloquium on Information Science and Technology (CiSt)*, 2016, doi:10.1109/cist.2016.7805062.

Puustjarvi, J., and L. Puustjarvi. “Using Semantic Web Technologies in Visualizing Medicinal

Vocabularies." *2008 IEEE 8th International Conference on Computer and Information Technology Workshops*, 2008, doi:10.1109/cit.2008.workshops.19.

Rose, Stuart, et al. "Automatic Keyword Extraction from Individual Documents." *Text Mining*, Apr. 2010, pp. 1–20., doi:10.1002/9780470689646.ch1.

Shah, Neepa, and Sunita Mahajan. "Semantic Based Document Clustering: A Detailed Review." *International Journal of Computer Applications*, vol. 52, no. 5, 2012, pp. 42–52., doi:10.5120/8202-1598.

Song, Yangqiu, et al. "Automatic Taxonomy Construction from Keywords via Scalable Bayesian Rose Trees." *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, Jan. 2015, pp. 1861–1874., doi:10.1109/tkde.2015.2397432.

Zhao, Dexin, et al. "Keyword Extraction for Social Media Short Text." *2017 14th Web Information Systems and Applications Conference (WISA)*, 2017, doi:10.1109/wisa.2017.12.

Gabrilovich, Evgeniy, and Shaul Markovitch. "Computing semantic relatedness using wikipedia-based explicit semantic analysis." In *IJcAI*, vol. 7, pp. 1606-1611. 2007.