AMSI VACATION RESEARCH SCHOLARSHIPS 2018-2019

Single Document Key Phrase Extraction and Clustering

Guo Feng Anders Yeo

Supervised by Dr. Vural Aksakalli

RMIT University

Vacation Research Scholarships are funded jointly by the Department of Education and Training and the Australian Mathematical Sciences Institute.

AMSI Australian Mathematical Sciences Institute

Australian Government
Department of Education and Training

1. Abstract

Key phrase identification has many applications, this method has been created specifically to identify key phrases for ad words. The Frequency Key Phrase (FKP) algorithm has been created to specifically be coupled with a clustering method to create ad groups. The key phrase identification and clustering work on a single document meaning a document repository is not required and the technique is not domain specific. No performance metrics have been set for the extraction or clustering. The FKP algorithm has 2 parameters, phrase length and extra thresholding and works well when phrase length is over estimated and extra threshold can be kept at zero or increased for long documents. The clustering has more parameters which require a little parameterization to get the algorithm to work optimally. An addition multiple key phrase clustering method which is still in an experimental phase.

2. Introduction and Motivation

This purpose of this report is to investigate methodologies for identification and clustering of advertising key phrases in Internet advertising, specifically on Google search advertisement networks (Google earns one billion dollars every three days from advertising alone). An efficient approach for an advertiser on Google is to bid on a set of key phrases and the performance of the campaign is dependent on the string of words used as key phrases as well as their grouping into appropriate ad groups. By intelligently selecting key phrases linked to a specific advertisement campaign and then clustering them into similar ad groups, an advertiser can improve overall user experience as well as achieve a better return on its advertisement spending. In this report, we present an algorithm to identify a group of efficient key phrases from any document and also implement a layered architecture using clustering metrics for any advertisement campaign. An additional method for multiple key phrase clustering which was created to place a single key phrase into multiple different clusters.

This report is structured as follows, section 3 will cover the key phrase extraction algorithm, the justifications and reasoning for each step in the process. Section 4 will cover the clustering metrics. Section 5 explains the experimental multiple key phrase clustering. Section 6 will discuss the overall performance and recommended parameters for each presented algorithm.

3. Key Phrase Extraction

The key phrase extraction algorithm uses term frequency to determine key phrases. The aim of this algorithm is to be a fast, single document key phrase extraction method. This algorithm shall be called, the Frequency Key Phrase algorithm (FKP), because it utilises term frequency to determine key words. The inputs of this algorithm are: the document, the maximum phrase length and the additional threshold. The output is the key words and key phrases. This section will explain each step of the FKP algorithm in execution order.

3.1 Converting document to lowercase and breaking into sentences

The purpose of changing everything to lowercase is to reduce duplicate candidate entries later on, as words at the beginning of sentences are capitalized. Whilst testing the algorithm, better results came from splitting based on punctuation that often finishes an idea or statement, thus the exclamation mark and question mark are also used to finish sentences in this context. The sentence structure is used when determining the weights.

3.2 Convert stop words into asterisks and stem all other words

Replacing stop words with asterisks ensures that the position of the remaining words in their corresponding sentences are still maintained. This is relevant when calculating the weights. Stemming is that act of shortening a word to just get their root form. The porter stemming algorithm appeared to be the best stemming algorithm to use. The purpose of stemming is to group together the same object or idea that appear in plural form or

difference tense. An example is being able to recognize that swim, swims and swimming have the same root word, whilst still being able to differentiate between words with different roots such as such as swimmers and swimmingly.

3.3 LZ78 compression

The LZ78 compression algorithm enables the creation of longer phrases within context, ensuring that not too many nonsensical phrases are created. The algorithm works by first checking single words and adding them to a library if that word is not already in the library. If a word is already in the library and has a neighboring word that is not separated by an asterisk, then it creates a library entry containing the original word and the next word. The maximum length of a phrase is user specified. This continues, adding longer phrases (if possible) into the library. For example, if the word “water” was already within the library and the LZ78 compression algorithm was now up to the string, “water bottle”. The algorithm would now store the string “water bottle” because “water” was already a library entry, then proceed to store the word “bottle”. This enables the creation of a candidate list with single words and phrases.

3.4 Removing only number entries

FPK removes any entry with only numbers such as “100”, but leaves entries such as “$100”. The algorithm also leaves numbers if they are part of a phrase such as “3 dogs”. The algorithm is also robust enough to handle floats such at “3.1415” and large amounts separated by a comma such as, “250,000”. The purpose of removing only number entries is due to numbers on their own having no meaning.

3.5 Calculating weights

The frequency weights are how many times a key phrase or key word appears in the entire document. Influence weights are determined by the position of the candidate term within sentences. Finally, the phrases are given an additional multiplier for the length of each phrase.

The frequency weight is an integer counter for each time the candidate term appears. The frequency weight also ends up ensuring that nonsensical candidate phrases produced during the LZ78 compression do not have high weighting.

The influence weight checks that the phrase appears at least once at the first half of a sentence or the final quarter, if it does not, halve the weight. This ensures that the phrase is a topic of at least one sentence. The length multiplier is an integer multiplier for the length of the phrase, it is capped at 3, such that a candidate phrase of length 4 will still have an additional multiplier of 3. The final weights are calculated as follows:

$$FinalWeight = (FrequencyWeight) * (Influence\ Weight) * (Phrase\ Length)$$

It is worth noting that when FKP is run on documents that have no punctuation such as Youtube transcripts, influence weight is ignored.

3.6 Selecting candidates with weights over threshold

The threshold for cutting the weights off for candidate terms is determined by:

$$\frac{\sqrt{N}}{10} + 1 + User\_input$$

Where N is the total number of words in the document. This equation was worked out by observation and generally yields good results. An additional user inputted weight can be added to include more or less words. Generally, an extra threshold is needed for longer documents.

3.7 Independence of Candidate words

If a word is present as a single word and within a phrase after the threshold cut off point, it should be determined whether or not the word is dependent on the other words in the phrase. FKP revises the frequency and influence weights for the single word and phrase to see if the new weights are still above the threshold. If it is not then the key phrases, key word or at rare occasions both are removed. The revised weights are done later to lower the time complexity of the algorithm, such that not every shared word has its weight revised. An example of words that are shared, would be the example of the key phrase

"water bottle" and the key word "water". The algorithm would recalculate the weights for the instances of the word "water" that do not appear in the phrase "water bottle". When revising the weights, the new weight for "water" would only be counted when it was counted within "water bottle" and any other phrases that contained the word "water".

3.8 Reverse Stemming

The method used for reverse stemming is; for phrases longer than a single length word, determine the original word contextually. The purpose of reverse stemming is for clarity in the final output. For single words, a list of candidate original words is collected. The frequency of each of these original words is taken and the candidate with the highest frequency is selected. The frequency also ignores any time the word appears within another longer key phrase. If there are equal frequencies between candidate, the longer word is selected. Should these words be the same length, the word that appeared first is selected. The highest frequency is chosen first, as it will return contextually how the key word was used. Followed by choosing the longer word because it may simply just be the plural or also have the same meaning as the shorter potential choice.

3.9 Removing nonsensical entries

The last check the algorithm does is remove any final phrases that end in only a number. This formatting usually only happens when a list or results are reported and do not make much grammatical sense, such as "emails 5" but would leave the entry if it were "5 emails". Such entries should not be included as a key phrase and only usually appear when results are presented multiple times in the document. Words that are less than 3 letters long and alone are often noise, specific notation or units such as "cm". Since the algorithm does not remove particular symbols, repeated patterns such as "---" may be included in the final keywords are then removed. This is determined if the final entry does not contain any alphanumeric symbols.

4. Key Phrase Clustering Metrics

Two different clustering metrics were investigated that were used together.

4.1 K Nearest Neighbors

The distance metric used here was the distance between two identified key words in the document. The average distance would be taken when comparing k nearest instances of the key phrase A to the nearest instances of word B. The algorithm is also robust enough to ignore instances where a key word is also within a key phrase.

4.2 Surrounding Similarity

Surrounding similarity is a measure of comparison of the string of words present before and after different key phrases. The parameters for the surrounding similarity metric is the size of the surrounding similarity window and whether exact positioning should be used.

The document that surrounding similarity is being run on should be the original document with all words converted to lower case and stop words not replaced by asterisks. Lower case ensures that instances where a word at the start of a sentence that is capitalized is regarded as the same word that is not capitalized.

First a surrounding similarity window size must be set, generally between 3 and 15 words yield good results. For each instance that key phrase A appears, the algorithm compares the number of words equal to the surrounding similarity window size before the key phrase A and compares the words to the surrounding similarity window size of words before each instance of key phrase B. The same is done for the words that are after key phrases A and B.

The similarity scores are then normalized by the number of shared words divided by the length of the surrounding similarity window of key phrase A multiplied by the length of the surrounding similarity window of key phrase B. This has been implemented such that if the

full surrounding similarity window size is unobtainable due to the key phrase being at the start or end of the document, they will not be penalized as heavily. Such that the score is:

$$SimilarityScore(A,B) = 2\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{SharedWordsBefore(A_i,B_j) + SharedWordsAfter(A_i,B_j)}{NonemptyBefore(A_i) + NonemptyAfter(A_i) + NonemptyBefore(B_j) + NonemptyAfter(B_j)}$$

Where n is the number of times word A appears and m is the number of times the word B appears.

An additional measure can be added to surrounding similarity, scores based on the exact positioning within the similarity window. This addition will exaggerate the contextual similarity between two key phrases. Two different methods of implementing this addition were trialed. The first was simply adding a extra similarity score for words that were in the exact same position, where the amount of extra score is user specified. The second was giving a set score if the word was in the exact same position, and if it was not, remove the score at a fixed rate based on the difference of exact scoring.

Another measure, that cannot work along side the exact positioning was combining the before and after surrounding similarity windows. This would end up comparing the words in the surrounding similarity windows before and after with each other. A lower score may be specified for words that are shared between after windows and before windows and vice versa.

4.3 Putting the Metrics together

The K nearest neighbor and surrounding similarity metrics were put together in two different ways.

The first method was to give a weighting for the surrounding similarity metric that would be relative to the K nearest neighbor which is user inputted. The scores for the surrounding similarity are then normalized in terms of the K nearest neighbor scores. Since for the K nearest neighbor metric, lower values were better and the higher values were better for the surrounding similarity, the final scoring metric would be the surrounding similarity score

multiplied by the weight given to the surrounding similarity subtracted from the k nearest neighbor, such that:

$$metric = KNearestNeighbor - SurroundingSimilarity * RelativeWeight$$

The second method does not involve a weight for the surrounding similarity score relative to the K nearest neighbor. It is simply the K nearest neighbor score divided by the surrounding similarity score, such that the metric is:

$$metric = \frac{KNearestNeighbor}{SurroundingSimilarity}$$

Since the surrounding similarity scores are always less than or equal to 1, the lower surrounding similarity scores will make the metric large compared to those with a high surrounding similarity score.

4.4 Shared word bonus

The shared word bonus reduces the score between any key phrases that share any words with other key phrases or contain any key words. This was done by dividing the scores by 100 of key phrases and key words that shared words or contained each other. This ensured that key phrases that share words are grouped together.

5. Multiple Key Phrase Clusters

The clusters were initially created using complete linkage and average linkage. These clustering methods alone do not have the capability of having a single key phrase appearing in multiple clusters. This method produces a way in which a single key phrase can appear in multiple clusters. The method in this section is an extension of clustering, such that it does not need to be used.

5.1 Creating sections using the distribution of Key phrases

The distribution of key phrases is used to split the document into a user specified number of sections. The distribution is determined by aggregating by a user specified amount and

counting the number of words within each aggregate. The location of the candidate splits is first selected as the local minimums of the aggregated distribution. At each candidate splitting point, the number of key phrases is counted in each section before and after the potential split and the difference between the number of key phrases in the section after the split and the number of key phrases in the section before the split. The location of the split is then the split with the lowest difference whose difference is not less than the total number of key phrases divided by the number of sections multiplied by 1.2. This ensures that the sections are the most even and not too small. The number of splits is determined by the number of user specified sections.

5.2 Clustering within sections

The clusters are then created by treating each section as its own document and using the metrics previously presented, create clusters for each section. This will result in a total number of clusters equal to the number of sections multiplied by the number of clusters if the same settings are used for each section.

6. Discussion and Conclusion

6.1 FKP Algorithm

The FKP algorithm was primarily inspired by two other key phrase extraction algorithms, KPE and Genex. The first step of the FKP algorithm is very similar to the first step of KPE with the exception that KPE breaks sentences only on full stops. The improvement that breaking on questions marks and exclamation marks means that two ideas or thoughts are put into a single sentence if they were split by a question mark.

The second to fourth step of the algorithm are the same as KPE. The LZ78 compression is similar to the candidate selection in Genex's extractor. Instead of first taking single words then selecting phrases for the top scoring single words as Genex's extractor does, LZ78 allows single words and corresponding phrases to be taken simultaneously. Since extractor uses frequency as one of the determining factors for the score of each phrase, the results are very similar.

KPE removes numbers within its first preprocessing step which leads to problems when a name contains letters and numbers such as the phrase “LZ78 compression algorithm”. FKP only removes single numbers because numbers out of context are meaningless.

The calculation of weights is also similar to KPE. KPE’s initial weights are determined by the frequency weights and influence weights. The idea of adding a multiplier for key phrases came from Genex, where longer phrases have a multiplier on their score. The threshold was intuitively created through trialing different formulas. Generally for longer documents an addition threshold of 1 should be added for every 4000 words.

The method used for reverse stemming was not included in KPE or Genex. The frequency is first used to ensure that the word is presented in the most contextual way within the document, as in if it was determining the reverse stemming based on tense or plural.

There is a limitation in the final step of FKP when removing key words that are two letters long or less, two letter acronyms such as abbreviating “Intellectual Quotient” to “IQ” which will be removed as a key word. This limitation may be avoided by not removing key words two letters long. Generally and in most cases the acronym will be grouped with another word such as “IQ test” and the final list of key words will be cleaner.

No performance metric has been set, it is graded on how noisy the final outputs are.
The FKP algorithm has been performed on many different kinds of documents. Generally, documents that are more focused tend to have cleaner results. The algorithm performed best on advertisements description and details taken from Udemy, followed closely by research papers then news articles and product reviews. Youtube transcripts have mixed results in terms of noise depending on the style of video. The videos that heavily rely on the accompanying visuals, such as following alongside someone coding, generally are quite noisy and require additional weights to be placed onto the cut off threshold. Whereas videos of a speech or presentation are similar to those of research papers. This again follows that more focused documents present cleaner results. Generally, Youtube transcripts require additional thresholding to get rid of noise but speech mannerisms are still present such as the words “basically” and “yeah”. Since Youtube transcripts have no

punctuation, influence weights could not be utilized which may have resulted in poorer results.

The advertisements on Udemy were often short and focused on the main topics of the course they were selling. The main emphasis of what was being sold is often repeated, ensuring that they appear as key phrases.

Research papers were often quite good at explaining their ideas. Since their ideas were often built around some existing idea or theorem, the ideas would be referenced multiple times allowing FKP to easily pick up on them. News articles often do the same thing, they explain current events in terms of what the main issues are for that article. Product reviews, specifically those relating to technology often talk about the specs of the product and elaborate on what makes them good or bad, enabling the algorithm to easily pick up on the specs and often the repeated positive or negative words used in the review, such as “good” or “poor”.

FKP was run on the entirety (including blurb) of the first Harry Potter Novel. As expected, the key phrases taken were the names of characters, houses and locations, many “Harry, [adjective]”, simple verbs and the odd object from the story such as “hat”, “owl”, “magic” and “broom”.

The algorithm was run on Shakespeare’s Romeo and Juliet, since old English stop words are not included in the list of stop words, the resulting key words are old English stop words and character names. This also means that the only aspect of the algorithm that needs changing when doing a different language is the stop words list and possibly the stemming algorithm.

The adjustment of the weight threshold, specifically for those which FKP runs well on gives different insights into the key phrases, especially when coupled with the clustering algorithms. Decreasing the threshold added a different caliber of words into the potential key phrases, whilst increasing the threshold would pinpoint the more important key phrases.

Overall the results of FKP are good considering that it is a single document key phrase extractor. Since FKP was created with the intent on being run on a single document and not needing to rely on a document repository. Resulting in, FKP is unable to use the TFxIDF (Term frequency, inverse document frequency) attribute that many other key phrase algorithms use. The other issue with dependence on a document repository is your results will either be format specific and domain specific, due to different formats and text styles of different documents and the importance of phrases within the repository compared to those in the single document.

6.2 Clustering Metrics

There is little documentation on metrics for single document key phrase clustering. The K nearest neighbor metric performed well for most documents. When K was set to 1, the algorithm was essentially looking for which key words and phrases were closest to one another in a single instance. The resultant clusters which were quite good, with the exception that there would occasionally be an odd word in the wrong cluster. Since no metric has been set to grade the clusters, it was done by observation, in the sense that do these clusters make sense in terms of the words composing them.

When K is increased above 1, the clusters are better than the clusters when K is set to 1. This is due to the smoothing of taking the average. For example, if a key word, A was to appear at the end of a sentence. Another key word, B appeared at the beginning of the following sentence and A and B should not be clustered together, then when K is set to 1, they would be clustered together.

The maximum that K should be set to for any document, is the minimum frequency of every key phrase and key word. Although increasing K above 1 yields better clusters, increasing K too high, or even this recommended maximum produces poorer clusters. The best resulting clusters were usually produced when K was set to 2 or 3.

Surrounding similarity was always used with K nearest neighbor, but K nearest neighbor was not always used with surrounding similarity. Surrounding similarity was usually a correcting metric, it would ensure that the odd word that appeared in the incorrect cluster would switch to the correct one.

Surrounding similarity exhibits different behaviors when the window size was changed. When the window size was set to 5 or 3, the surrounding similarity was comparing the immediate words around them. This meant that a higher surrounding score was assigned when comparing objects to objects or verbs to verbs etc. When the window size was set to 10 or 15, the surrounding similarity was comparing the nature of sentences before and after the key phrases, resulting in higher scores for when key phrases were used in the same contextual manner, such as in the conclusion of a paragraph or opening of an idea. There is no superior window size in forming clusters. The correctional abilities that both scenarios of window sizes provide is good but different.

The relative weighting of surrounding similarity to K nearest neighbor is difficult to determine. In most cases a relative weighting of one is sufficient. Generally, for larger window sizes a higher weighting is required for the subtraction method. This is due to the score being divided by the number of words considered in each instance of a word appearing. More often than not, no words will be shared or a low number of words will be shared, resulting in the score being very low. So even normalizing the low score with the K nearest neighbor score will result in low scores with little weight on the final score.

The method of putting the two metrics together is also dependent on how they are expected to perform together. The method that involves dividing the K nearest neighbor by similarity score penalizes the final result more than the subtraction method. It depends on how heavily the score should be penalized based on the surrounding similarity windows behavior.

The cross checking between the windows after and the windows before do not seem to have any additional benefit to the clusters. Due to there being no reasonable way to

perceive the benefits of cross checking the before and after windows, it was probably just not a good idea.

Concerning the linkage method of clustering, all documents were first tested with complete linkage. More often than not, complete linkage was able to yield good results in clustering. In the odd occasion when there were not, average linkage was used instead. Simple linkage only seemed to work when the K in K nearest neighbors was set to 1. Though it is a naïve way of grouping simply based on which words were closest together and the final clusters were often not as good as when K was increased and complete linkage was used.

When running the clustering method on a large document (10,000 words or more), it is recommended to run the clustering on a small portion, then find the best weighting and parameters for clustering before running the entire document. This ensures that the surrounding similarity acting as a correction will match the linguistic style of the document for the best outcome.

Overall the metrics worked well together when the parameters for surrounding similarity were fined tuned and the division method of putting the weights together was used. On some occasions surrounding similarity was not required for corrections as the clusters were good enough.

6.3 Multiple key phrase clustering

The multiple key phrase clustering seems to work best when the keywords and key phrases have different cluster potentials. This is due to the purpose of the of the method to place a single key word or key phrase into multiple different clusters. When clustering without the multiple key phrase clustering method works well, the multiple key phrase clustering appears redundant.

The multiple key phrase clustering method has many possible extensions. The method produces many clusters, some are good clusters, some are not. A method of selecting good clusters is required to create an output of final clusters. Possible methods that have

been considered but not tested include using the frequency in which words appear in the same cluster or creating a performance metric to grade which of the clusters are good. This addition may not be needed, the current end result of the multiple key phrase clustering method serves its purpose in making it possible to place different words in different clusters allowing them to be clustered in different context.

Another possible extension to the multiple key phrase clustering would be to automate the parameters for clustering within each section. If the clusters within each section are optimized then the final clustering is likely to be even better.

7. Acknowledgments

I would like to thank Dr. Vural Aksakalli for supervising me.

7.1 Document suppliers for trialing algorithms

D.Williams. “Writing Samples 2012-2014” Personal email (29th November 2018)

V.Pham. “Mentoring Report” Personal email (29th November 2018)

K.Yeo “Extended Essay on Van Gough” Personal email (26th November 2018)

8. Appendix

8.1 FKP algorithm

The user inputs are: the maximum phrase length of key phrases and the additional threshold.

1. Convert document to lowercase and break text into sentences. Sentences are determined by a full stop, exclamation mark or question mark.
2. Remove all stop words and replace them with an asterisk, *. All other words get stemmed by the porter stemming algorithm.
3. Use LZ78 compression algorithm to create a library of candidates up to the length of the user inputted maximum phrase length.
4. Remove any library entries that are only numbers.
5. Calculate the weights based on frequency, position in sentence and length of phrase.
6. Select candidates with weights over the threshold where the threshold is

$$\frac{\sqrt{words\ in\ the\ document}}{10} + 1 + User\_input$$

7. Revise weighting for shared words.
8. Reverse Stemming.
9. Remove any key phrases and words that end with only a number and single words with less than three letters or contain no alphanumeric symbols.

8.2 FKP run of the report

No extra thresholding has been used and the maximum word length is 7

- (words)
- (k, nearest, neighbors)
- (key, word)
- (number)
- (algorithm)
- (surrounding, similarity)
- (document)
- (weights)
- (key, phrase)
- (multiple, key, phrase, clustering)
- (multiple, key, phrase, clustering, method)
- (results)
- (metric)
- (section)
- (single, words)
- (surrounding, similarity, window, size)
- (influence, weights)
- (multiplier)
- (reverse, stemming)
- (specifically)
- (work)
- (performance)
- (adding)
- (single, key, phrase)
- (run)
- (comparing)
- (position)
- (frequency, weights)
- (times)
- (youtube, transcripts)
- (score)
- (frequency)
- (set)
- (surrounding, similarity, score)
- (candidate)
- (stop, words)
- (additional)
- (final)
- (created)
- (fkp, algorithm)
- (length)
- (threshold)
- (key, phrase, extraction, algorithms)
- (removing)
- (share, words)
- (surrounding, similarity, windows)
- (window, sizes)
- (sentences)
- (entries)
- (determined)
- (ensures)
- (good)
- (fkp)
- (split)
- (appears)
- (lz78, compression)
- (surrounding, similarity, metric)
- (advertisement)
- (present)
- (cluster, metric)
- (terms)
- (instances)

- (generally)

8.3 Clustering run of the entire report

1	2	3	4
[metric] [cluster, metric]	[single, words] [share, words] [words] [stop, words]	[multiple, key, phrase, clustering] [multiple, key, phrase, clustering, method] [key, word] [single, key, phrase] [key, phrase, extraction, algorithms]	[times] [frequency, weights] [influence, weights]
5	**6**	**7**	**8**
[surrounding, similarity, window, size] [surrounding, similarity, metric] [surrounding, similarity, score] [length] [entries] [surrounding, similarity, windows] [window, sizes] [k, nearest, neighbors] [surrounding, similarity]	[fkp] [fkp, algorithm] [lz78, compression] [multiplier] [candidate] [split] [position] [removing] [reverse, stemming]	[section] [final] [frequency] [results] [weights] [key, phrase] [work] [performance] [additional] [document] [created] [good] [appears] [ensures] [run] [number] [score] [set]	[terms] [threshold] [specifically] [algorithm] [adding] [advertisement] [present] [youtube, transcripts] [sentences] [comparing] [generally] [determined] [instances]

8.4 Multiple key phrase clustering run of the entire report

Section 1

1	2	3
[fkp] [fkp, algorithm]	[algorithm] [appears] [words] [stop, words] [weights] [sentences] [ensures] [position]	[frequency] [terms] [document] [candidate] [entries] [results] [split]
4	**5**	**6**
[key, phrase] [key, word] [single, key, phrase] [key, phrase, extraction, algorithms] [multiple, key, phrase, clustering] [multiple, key, phrase, clustering, method] [cluster, metric] [performance] [present]	[additional] [length] [section] [threshold] [determined]	[set] [work] [metric] [specifically] [adding] [created] [advertisement]

Section 2

1	2	3
[created]	[adding]	[lz78, compression]
4	**5**	**6**
[words] [single, words]	[length] [entries]	[work] [algorithm] [ensures]

Section 3

1	2	3
[algorithm] [removing] [length] [appears]	[words] [single, words] [share, words] [threshold] [instances]	[lz78, compression] [frequency, weights] [weights] [influence, weights]
4	**5**	**6**
[additional] [document] [present] [generally] [adding] [work] [results] [good] [position] [sentences] [ensures] [terms] [multiplier] [youtube, transcripts] [fkp] [run]	[times] [determined] [reverse, stemming] [key, phrase] [key, word] [frequency] [candidate]	[number] [entries] [final]

Section 4

1	2	3
[stop, words] [share, words] [single, words] [words] [key, word]	[metric] [cluster, metric]	[appears] [k, nearest, neighbors] [comparing] [surrounding, similarity, metric]

		[surrounding, similarity, windows] [surrounding, similarity] [surrounding, similarity, score] [work] [position]
4	**5**	**6**
[specifically] [entries] [run] [length] [instances] [surrounding, similarity, window, size] [window, sizes]	[key, phrase] [multiple, key, phrase, clustering] [single, key, phrase] [times] [section] [candidate] [algorithm] [fkp, algorithm] [key, phrase, extraction, algorithms] [frequency] [frequency, weights] [determined] [reverse, stemming] [terms] [ensures] [sentences] [split] [adding] [created] [multiplier] [score] [lz78, compression] [number]	[good] [advertisement] [results] [present] [youtube, transcripts] [generally] [additional] [document] [threshold] [performance] [weights] [influence, weights] [fkp] [removing] [set] [final]

Section 5

1	2	3
[stop, words] [words] [key, word]	[document] [created] [good]	[terms] [position] [threshold] [adding] [algorithm] [specifically] [fkp]
4	**5**	**6**
[instances] [score] [number] [determined] [generally] [window, sizes] [sentences] [comparing]	[additional] [key, phrase] [multiple, key, phrase, clustering] [multiple, key, phrase, clustering, method] [final] [frequency] [work] [performance] [results]	[metric] [cluster, metric] [surrounding, similarity] [surrounding, similarity, windows] [set] [k, nearest, neighbors] [ensures] [run] [appears] [weights]

9. References

Lim, M.H., Wong, S.F. and Lim, T.M., (2013), "Automatic keyphrase extraction techniques: a review", *Proceedings of the IEEE Symposium on Computers & Informatics, Langkawi, April 7-9* , pp. 196-200.

Turney, Peter. "Learning Algorithms for Keyphrase Extraction." *Information Retrieval* 2.4 (2000): 303-36. Web.

Youssif, Aliaa & Ghalwash, Atef & Amer, Eslam. (2011). KPE: An Automatic Keyphrase Extraction Algorithm.

Iosif, Elias, and Alexandros Potamianos. "Unsupervised Semantic Similarity Computation Between Terms Using Web Documents". *IEEE Transactions On Knowledge And Data Engineering*, vol 22, no. 11, 2010, pp. 1637-1647. *Institute Of Electrical And Electronics Engineers (IEEE)*, doi:10.1109/tkde.2009.193.