Similarity-Based Estimation for Document Summarization using Fuzzy Sets

Masrah Azrifah Azmi Murad* masrah@fsktm.upm.edu.my
Department of Information Systems
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia, 43400, Serdang, MALAYSIA

Trevor Martin trevor.martin@bris.ac.uk
Department of Engineering Mathematics
University of Bristol
BS8 1TR, UK

Abstract

Information is increasing every day and thousands of documents are produced and made available in the Internet. The amount of information available in documents exceeds our capacity to read them. We need access to the right information without having to go through the whole document. Therefore, documents need to be compressed and produce an overview so that these documents can be utilized effectively. Thus, we propose a similarity model with topic similarity using fuzzy sets and probability theories to extract the most representative sentences. Sentences with high weights are extracted to form a summary. On average, our model (known as MySum) produces summaries that are 60% similar to the manually created summaries, while *tf.isf* algorithm produces summaries that are 30% similar. Two human summarizers, named P1 and P2, produce summaries that are 70% similar to each other using similar sets of documents obtained from TREC.

Keywords: fuzzy sets, mass assignment, asymmetric word similarity, topic similarity, summarization

1. INTRODUCTION

Information is increasing every day and thousands of documents are produced and made available in the Internet. The amount of information available in documents exceeds our capacity to read them. We need access to the right information without having to go through the whole document. Therefore, documents need to be compressed and produce an overview so that these documents can be utilized effectively. To generate a summary, we need to identify the most important information in the document avoiding the irrelevant or less important ones, [1] discussed three phases involved in summarizing text automatically: 1) selection of more salient information, 2) aggregation of information, and 3) generalization of specific information to a more general concept.

Recently, researchers have tried to extract a summary using various techniques such as word frequencies [2; 3; 4] and clustering [5]. The first automated sentence extraction system [6] uses term frequencies to weight sentences, which are then extracted to form an abstract. Since then,

* Corresponding author

many approaches have been explored in automatically extracting sentences from a document. Most existing text summarization systems use an extract approach. This approach is known to be safe because important information is taken or copied from the source text. An abstract approach produces a summary at least some of whose material is not available in the source text, but at the same time retaining the content originality. A summary can be generic meaning the content is broad and not addressed to any specific audience. Nonetheless, it could be tailored or user-focused in which the content is addressed to a group with specific interests. Further, the content of a summary can be indicative or informative. Indicative content provides an indication of the main topics. Therefore, it helps the user to decide whether to proceed with reading the source text or otherwise, while informative content represents the original document.

The objective of this work is to produce a similarity model with topic similarity using the theories of probability and fuzzy sets incorporating mass assignment to find the similarity between two words. We compute frequencies of triples of words exist in the document collection and convert these frequencies to fuzzy sets. Probability of two words is then computed using the semantic unification of two fuzzy sets. Our results show that using asymmetric word similarity with topic similarity able to extract the most relevant sentences and produce summaries that are almost similar to manually created summaries. The remainder of the paper is organized as follows: In section 2 we discuss briefly on common algorithm, *tf.isf*, and use it to benchmark against our algorithm, section 3 explains the methodology used, section 4 discusses in detail on the word similarity algorithm, section 5 discusses the similarity model in extracting sentences, section 6 discusses the results, and finally section 7 concludes the paper.

2. *tf.isf*

This section outlines related work done in summarization particularly extracting sentences from a document. We described a method known as *tf.isf* (term frequency x inverse sentence frequency) [4]. This method is used later in comparing against MySum (our proposed model) and a manually created summary. *tf.isf* is an adaptation of the conventional *tf.idf* [7]. Sentences are extracted using average sentence similarities and those with high weights (above a certain threshold) are extracted to form a summary. The computation of *tf.isf* is similar to the computation of *tf.idf*.

$$w_{ik} = tfi_k \; x \; idf_{ik} \tag{1}$$

The only difference is the notion of *document* that is being replaced by *sentence*. Each sentence is represented as a vector of *tf.isf* weights. Sentences with high values of *tf.isf* are selected to produce a summary of the source text. Hence, the *tf.isf* measure of a word *w* in a sentence *s*, is computed using the following

$$tf.isf(w,s) = tf\,(w,s) \times isf(w) \tag{2}$$

where *tf(w, s)* is the number of times the word *w* occurs in sentence *s*. *isf(w)* is the inverse sentence frequency of word *w* in sentence *s* given by

$$isf(w) = log\; S \;/\; sf(w) \tag{3}$$

where *sf(w)* is the number of sentences in which the word *w* occurs and *S* is the total number of sentences in the document. For each sentence *s*, the average *tf.isf* of the sentence is computed by calculating the average of the *tf.isf(w, s)* weight over all of the words *w* in the sentence, as shown in the following formula

$$\sum_{i=1}^{W(s)} tf.isf\,(i,s) \Big/ W(s) \tag{4}$$

where *W(s)* is the number of words in the sentence *s*. Sentences with the largest values of average *tf.isf* are selected as the most relevant sentences and will be produced as a summary. Using *tf.isf* is simple and fast. Further, *tf.isf* only relies on the frequency of words in documents, therefore, it's possible to use *tf.isf* in summarizing texts other than English. However, *tf.isf* may not be a good algorithm in extracting sentences. For example, *tf.isf* cannot reflect similarity of words and only count the number of overlapping words. The algorithm does not consider any synonymy and syntactic information. In addition, there could be some relevant or important sentences missing, as they use different words to express the same interests.

3. FUZZY SETS AND MASS ASSIGNMENT

This section outlines the theory of fuzzy sets and mass assignment that are used extensively in our work. A fuzzy set is an extension to a classical set theory, which has a problem of defining the border of the set and non-set [8]. Unlike a classical set, a fuzzy set does not have a clearly defined boundary by having elements with only a partial degree of membership [9]. For example, consider a weight of a person with labels such as *thin, average,* and *fat*. These labels are considered fuzzy because not everyone will agree with the same subset of the value domain as satisfying a given label. Nevertheless, if everyone agrees, we could write precise definitions of *thin, average*, and *fat* in this context.

A mass assignment theory was proposed by Baldwin in 1991 as a general theory for evidential reasoning under uncertainty [9; 10]. This theory is used to provide a formal framework for manipulating both probabilistic and fuzzy uncertainties [9]. Consider the following example taken from [10], suppose we have a set of people labeled 1 to 10 who are asked to accept or reject a dice value of *x* as *small*. Suppose everyone accepts 1 as *small*, 80% accept 2 as *small* and 20% accept 3 as *small*. Therefore, the fuzzy set for *small* is defined as

$$small = 1 \,/\, 1 + 2 \,/\, 0.8 + 3 \,/\, 0.2 \tag{5}$$

where the membership value for a given element is the proportion of people who accept this element as satisfying the fuzzy set. The probability mass on the sets is calculated by subtracting one membership from the next, giving MA_{small} as

$$MA_{small} = \{1\} : 0.2, \{1, 2\} : 0.6, \{1, 2, 3\} : 0.2 \tag{6}$$

The mass assignments above correspond to families of distribution. In order to get a single distribution, the masses are distributed evenly between elements in a set. This distribution is known as *least prejudiced distribution (LPD)* [11] since it is unbiased towards any of the elements. Thus, in the example above, the mass of 0.6 is distributed equally among 1 and 2 and the mass 0.2 is distributed equally among 1, 2 and 3. Therefore, the *least prejudiced distribution* for *small* is

$$\begin{aligned} LPD_{small} = \; & 1 : 0.2+0.3+0.0667=0.5667, \\ & 2 : 0.3+0.0667=0.3667, \\ & 3 : 0.0667 \end{aligned} \tag{7}$$

3.1 Semantic Unification

Semantic Unification is a concept in Fril [12] proposed by Baldwin in 1992 that is used to unify vague terms by finding a support for the conditional probability of the match. Unification is possible if two terms have the same meaning, however, if they only have similar meaning, then the match will not be perfect and can be supported with a support pair. A mass assignment with the least prejudiced distribution is used to determine the unification of two fuzzy sets. For example, suppose the fuzzy set for *medium* in the voting model is

$$medium = 2 / 0.2 + 3 / 1 + 4 / 1 + 5 / 0.2 \quad (8)$$

and the mass assignment would be

$$MA_{medium} = \{3, 4\} : 0.8, \{2, 3, 4, 5\} : 0.2 \quad (9)$$

Thus, the least prejudiced distribution is

$$LPD_{medium} = 2 : 0.05, 3 : 0.45, 4 : 0.45, 5 : 0.05 \quad (10)$$

Suppose we want to determine the *Pr (about_3 | medium),* and the fuzzy set is

$$about\_3 = 2 / 0.4 + 3 / 1 + 4 / 0.4 \quad (11)$$

with mass assignment as

$$MA_{about\_3} = \{3\} : 0.6, \{2, 3, 4\} : 0.4 \quad (12)$$

We use the point semantic unification algorithm [11] to determine the conditional probability. Thus,

Pr (dice is **about_3** | dice is **medium**)
= 0.6Pr (dice is 3 | dice is **medium**) + 0.4Pr (dice is {2, 3, 4} | dice is **medium**)
= 0.6 (0.45) + 0.4(0.05 + 0.45 + 0.45)
= 0.65

The point semantic unification can be calculated using the following tableau.

	0.8 : {3,4}	**0.2 : {2,3,4,5}**
0.6 : {3}	1/2 x 0.8 x 0.6	1/4 x 0.2 x 0.6
0.4 : {2,3,4}	0.8 x 0.4	3/4 x 0.2 x 0.4

TABLE 1: Tabular Form of the Pr (about_3 | medium).

The entries in the cells are the supports from the individual terms of the mass assignments. Each entry has an associated probability. Thus, the *Pr (about_3 | medium)* is 0.65. The computation of the probability above can be shown using the following formula. Consider two fuzzy sets *f1* and *f2* defined on a discrete universe *X*. Let

$\mu_{f1}(x)$ be the membership of element *x* in the fuzzy set *f1*.
$MA_{f1}(S)$ be the mass associated with set *S*.
$LPD_{f1}(x)$ be the probability associated with element *x* in the *LPD*.

(and similarly for *f2*). Therefore

$$Pr(f1 \mid f2) = \sum_{\substack{S1 \subseteq X, \\ S2 \subseteq X, \\ S1 \cap S2 \neq \phi}} \frac{MA_{f1}(S1) \times MA_{f2}(S2)}{|S2|}$$

$$= \sum_{x \in X} \mu_{f1}(x) \times LPD_{f2}(x) \quad (13)$$

4. ASYMMETRIC WORD SIMILARITY

In this section, we propose a novel algorithm in computing word similarities asymmetrically using mass assignment based on fuzzy sets of words. We concentrate on how sentences use a word, and not on their meaning. Words in documents are considered to be similar if they appear in similar contexts. Therefore, these similar words do not have to be synonyms or belong to the same lexical category. Further, this algorithm is incremental such that any addition or subtraction of words (and documents) will only require minor re-computation.

4.1 Document Preprocessing and Similarity Algorithm

Before the measurement of the similarity algorithm is implemented, documents need to go through preprocessing stage so that only meaningful keywords are obtained from those documents. The first step is to remove common words, for example, *a, the, or,* and *all* using a list of stop words. If a word in a document matches a word in the list, then the word will not be included as part of the query processing. The second step is to stem a word to become a root word, for example, *subtraction* becomes *subtract.* In this work, we applied the process of Porter stemmer [13] to every word in the document.

The underlying objective of our method is the automatic computation of similar words. The method is based on the observation that it is frequently possible to guess the meaning of an unknown word from its context. The method assumes that similar words appear in similar contexts and therefore, these words do not have to be synonyms or belong to the same lexical category. A key feature of the algorithm is that it is incremental, i.e. words and documents can be added or subtracted without extensive re-computation. Our method is based on finding the frequencies of n-tuples of context words in a set of documents where frequencies are converted to fuzzy sets, which represent a family of distributions, and find their conditional probabilities. Consider the following example, taken from [14]

A bottle of *tezgüno* is on the table.
Everyone likes *tezgüno.*
Tezgüno makes you drunk.
We make *tezgüno* out of corn.

From the sentences above, we could infer that *tezgüno* may be a kind of an alcoholic beverage. This is because other alcoholic beverages, for example, *beer* tends to occur in the same contexts as *tezgüno*. The idea that words occurring in documents in similar contexts tend to have similar meanings is based on a principle known as the Distributional Hypothesis [15]. We use this idea to produce a set of related words, which can be used as the basis for taxonomy, or to cluster documents. In this experiment, we use Fril to compute asymmetric similarities such that the similarity between <w1> and <w2> is not necessarily the same as between <w2> and <w1> expressed as

$$ws(<w1>,<w2>) \neq ws(<w2>,<w1>)$$

This is because to compute similarity between two fuzzy sets, i.e. ws(<w1>,<w2>), we multiply the memberships of fuzzy sets of <w1> with the corresponding frequencies in frequency distributions of <w2>. In order to calculate ws(<w2>,<w1>), we multiply the memberships of fuzzy sets of <w2> with the corresponding frequencies in frequency distributions of <w1>. In most cases, the values for two fuzzy sets are different; therefore, the similarity measures will be different. In the next phases, we present the algorithms used in finding the similarity between words. AWS consists of two phases. In Phase I [16], we compute the frequency distributions of words to fuzzy sets. In Phase II [16], we find the conditional probabilities of the fuzzy sets using the semantic unification algorithm and show the creation of AWS matrix.

Phase I – Computation of frequency distributions to fuzzy sets

Each document is described by a set of all words called vocabulary. We run a pre-processing procedure by removing inappropriate words and stemming words. Removing inappropriate words allow us to save space for storing document contents and at the same time reduce the time taken during the search process. We define a document D_j that is represented by a set of an ordered sequence of n_j words as the following

$$D_j = \{w_0, w_1, w_2, \ldots, w_{nj}\}$$

with w being the sub-sequence of document D_j. The ordering of words in the document is preserved. We calculate the frequency distributions of every word available in the document. For any sub-sequence $W_n(x) = \{w_x, w_{x+1}, \ldots, w_{x+n}\}$, let $p(x)$ be a word that precedes word x such that

$$p(x) = \{w_{x-k}, w_{x-k+1}, \ldots, w_{x-1}\}$$

and $s(x)$ be a word that succeeds word x such that

$$s(x) = \{w_{x+l+1}, w_{x+l+2} \ldots, w_{x+l+k}\}$$

where k and l are a given block of k words preceded and succeeded by blocks of l words, and n is the total number of words in the document. We give a value of 1 to k and l as we need to consider the start and end of the document. Consider a document D_j containing sentences as the following.

The quick brown fox jumps over the lazy dog. The quick brown cat jumps onto the active dog. The slow brown fox jumps onto the quick brown cat. The quick brown cat leaps over the quick brown fox.

TABLE 2: Example of Sentences in Document D_j

From the sentences, we obtain

W(1)=quick, p(1)=the, s(1)=brown
W(2)=brown, p(2)=quick, s(2)=fox

using

p(x) = W(x-1)
s(x) = W(x+1)

The computation of frequency distributions of words in the document will be built up incrementally. Hence, for each word x, we incrementally build up a set <context-of-x> containing pairs of words that surround x, with a corresponding frequency. Let

pre(x) be the set of words that precedes word x
suc(x) be the set of words that succeeds word x
$N = \{pre(x), x, suc(x)\}$ being the total number of times the sequence of *{pre(x), x, suc(x)}* occurs in document D_j

Thus, the frequency of each <context-of-x> is given by the following

$$f_{cw} = \{pre(x), x, suc(x)\} / N$$

Once we computed the frequency distributions of each word, we convert the frequencies to memberships as shown in the following algorithm.

Input:

f_{cw} :	**array of frequency counts.**
T :	**total frequency count for this word =** $\sum_{P,S} f_{cw}(P,S)$ **where P and S are precedence and successor respectively.**

Output:

m_{cw}:	**array of memberships**
1.	**Sort frequency counts into decreasing order, $f_{cw}[0]$ ... $f_{cw}[n-1]$ such that $f_{cwi} \geq f_{cwj}$ iff i > j**
2.	**Set the membership corresponding to maximum count, $m_{cw}[0] = 1$**
3.	**for i=1 ... n-1, i.e., for each remaining frequency count $m_{cw}[i] = m_{cw}[i-1] - (f_{cw}[i-1] - f_{cw}[i]) * i / T$**

FIGURE 1: Algorithm for Converting Frequencies to Memberships

The complexity of the above algorithm lies in its sorting step, nevertheless, the remaining steps are linear in the size of the array. Using the example of sentences in Table 2, we obtain the frequencies for word *brown* with *N=6*

quick - brown - cat occurs three times
quick - brown - fox occurs two times
slow - brown - fox occurs once

We use mass assignment theory to convert these frequencies to fuzzy sets (as described in Figure 1), and obtain the fuzzy set for word *brown* as

(quick, cat):1, (quick, fox):0.833, (slow, fox):0.5

In the next phase, we use the fuzzy sets to compute the probability of any two words.

Phase II – Computation of Word Probabilities

To compute a point semantic unification for two frequency distributions f_{cw1} and f_{cw2}, we calculate membership for f_{cw1} and multiply by the frequency for the corresponding element in f_{cw2}.

Input:

m_{cw1} :	**array of memberships.**
f_{cw2} :	**array of frequency counts.**
T_{cw2} :	**total frequency counts for w2 =** $\sum_{P,S} f_{cw2}(P,S)$ **where P and S are precedence and successor respectively.**

Output:

Semantic Unification Value - Pr (w1|w2) , Pr (w2|w1)

1.	**Convert f_{cw1} to m_{cw1} using steps in Algorithm I.**
2.	**Calculate the sum of m_{cw1} multiply by f_{cw2} for the common elements giving the point semantic unification for two frequency distributions.**
3.	**To compute the asymmetric probability, simply reverse the calculation in steps 1 and 2.**

FIGURE 2: Point Semantic Unification Algorithm

Hence, for any two words <w1> and <w2>, the value

Pr (<context-of-w1>|<context-of-w2>)

measures the degree to which <w1> could replace <w2>, and is calculated by semantic unification of the two fuzzy sets characterizing their contexts. For example, suppose there is sentences in the document that give the fuzzy context set of *grey* as

(quick, cat):1, (slow, fox):0.75

We calculate the asymmetric word similarity of the two fuzzy sets of *brown* and *grey* using point semantic unification algorithm, giving the conditional probabilities as

Pr (brown | grey) = 0.8125
Pr (grey | brown) = 0.625

By semantic unification of the fuzzy context sets of each pair, we obtain an asymmetric word similarity matrix. For any word, we can extract a fuzzy set of similar words from a row of the matrix. We also note that there are important efficiency considerations in making this a totally incremental process, i.e. words (and documents) can be added or subtracted without having to recalculate the whole matrix of values as opposed to a straightforward implementation that requires $O(n_a \times n_b)$ operations per semantic unification, where n_b is the cardinality of the fuzzy context set that requires $O(n^2)$ semantic unification and *n* is the size of the vocabulary. Therefore, any addition of a new word or a new document using a straightforward implementation would require the whole re-computation of the matrix. Figure 3 below shows the creation of AWS matrix with elements described in the algorithm as having non-zero values.

1.	**Store each word with a list of its context pairs with number of times each context pair has been observed.**
2.	**Calculation of the corresponding memberships and elements are not done until needed. Otherwise,** **if a word *W* is read, then mark elements *Pr(W|w_i)* and *Pr(w_i|W)* as needing recalculation.**
3.	**If a new context, *P-W-S* is read,** **search for other words w_j which have the same context *P-w_j-S*.** **mark the elements *Pr(W|w_j)* and *Pr(w_j|W)* as needing calculation.**

FIGURE 3: Algorithm for Creating AWS Matrix

This process creates an asymmetric word similarity matrix *Sim,* whose rows and columns are labeled by all the words encountered in the document collection. Each cell *Sim(w_i, w_j)* holds a value between 0 and 1, indicating to which extent a word *i* is contextually similar to word *j*. For any word we can extract a fuzzy set of similar words from a row of the matrix. Many of the elements are zero. As would be expected, this process gives both sense and nonsense. Related words appear in the same context (as with *brown* and *grey* in the illustration above), however, unrelated words may also appear, for example, the phrase *{slow fat fox}* would lead to a non-zero similarity between *fat* and *brown.*

5. THE SIMILARITY MODEL

Recall the AWS algorithm we have described in Section 4 above

$$Sim(w_i, w_j) \tag{14}$$

We now introduce the sentence similarity measures *sim*(S_i, S_j) to find the similarities between sentences available in a document using AWS. Hence

$$\sum_{i \in sentence1} \sum_{j \in sentence2} f_i Sim(w_i, w_j) f_j \tag{15}$$

where *f* is the relative frequency of a word in a sentence and *Sim*(w_i,w_j) is the similarity matrix developed in Section 4. We also compute the asymmetric sentence similarity, which would produce a different similarity measure. We introduce a topic similarity measure *sim*(S_i, *t*) for the purpose of increasing the importance measure of a sentence *Si* to the topic *t*. We compute a weight for topic similarity using the frequency of overlapping words in the sentence as well as the topic. Identical words will have a value of 1, with 0 for non-identical words. Hence, the formula is defined as

$$\sum_{i \in sentence} \sum_{j \in topic} f_i ws(w_i, w_j) f_j \tag{16}$$

where *f* is the relative frequency of a word in a sentence and topic respectively and *ws*(w_i,w_j) is the similarity of overlapping words. We named the two similarity measures above (as in Eq. 15 and 16) as the two score functions and these score functions will be used in extracting sentences from the document.

Sentence Extraction

The two score functions, i.e. sentence similarity and topic similarity measures are used to compute the weight for each sentence. We measure the importance of a sentence S_i as an average similarity *AvgSim(i)*. The weight of a sentence is defined by summing similarity measure of sentence S_i with other sentences in the document divided by *N* the total number of sentences. Thus, the *AvgSim(i)* is defined as

$$\frac{\sum sim(S_i, S_j)}{N} \tag{17}$$

where *sim*(S_i, S_j) is the pairwise asymmetric sentence similarity. Next, we add the weight of average sentence similarity and topic similarity to produce the final score of a sentence, given in the following formula

$$MySum = AvgSim(i) + sim(S_i, t) \tag{18}$$

Once the score for each sentence has been computed, the sentences are ranked in descending order. Sentences with high values will be selected to produce a summary. Then the sentences are arranged according to their chronological order in the original article to form a summary.

6. RESULTS

In order to evaluate the effectiveness of our method, we compare the summaries produced by our system against the manually created summaries produced in the DUC 2002 [17]. In addition, we also compare the performance of other system, *tf.isf* against the manually created summaries. Our final comparison is between MySum and *tf.isf* against the manually created summaries. Each document of DUC 2002 produces two versions of manually created summaries written by two different human readers. In this experiment, we produce a hypothetical test by making a comparison of summaries produced by two different human summarizers. This is to show that in reality it is very unlikely for two different systems or humans to produce an identical summary from a document.

In DUC 2002, Task 1 is a single-document summarization in which the goal is to extract 100 word summaries from each document in the corpus. We use the summaries produced in Task 1 in our comparison stage. The comparison is made using individual matching, i.e. each sentence in a summary produced by MySum or *tf.isf* is compared against each sentence of the manually created summary. In this case, a sentence generated by MySum or *tf.isf* and a sentence from manually created summary are considered similar if the similarity is equal to the proportion of identical words. If all sentences produced by MySum or *tf.isf* are the same as the sentences in the manually created summary, the similarity measure is equal to 1. However, if only a proportion of sentences are equal to the sentences in the manually created summary, the similarity measure would be the number of sentences generated by MySum or *tf.isf* that is similar to the number of sentences in the manual summary divided by total number of sentences in the manually created summary. In this paper, we presented only a few of our results, while the remaining is reported elsewhere. Figures 4 to 6 show the comparison of similarity produced by MySum and *tf.isf* against human summarizers, P1 and P2. In the figures, the comparisons of two summaries produced by P1 and P2 are used as a hypothetical test.

On average, MySum produces summaries that are 60% similar to the manually created summaries, while *tf.isf* produces summaries that are 30% similar. It is worth pointing out that the human summarizers, P1 and P2 produce summaries that are 70% similar to each other. Overall, MySum produces a fairly good result and none of the documents generated by MySum produce a zero similarity comparison against the manually created summaries. Our method shows that it could generate a summary from a document as close to what a human summarizer could produce.

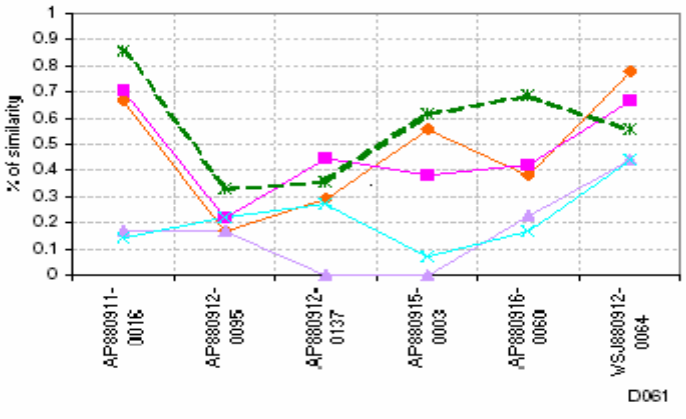


FIGURE 4: Result on Summarization using Document Set D061

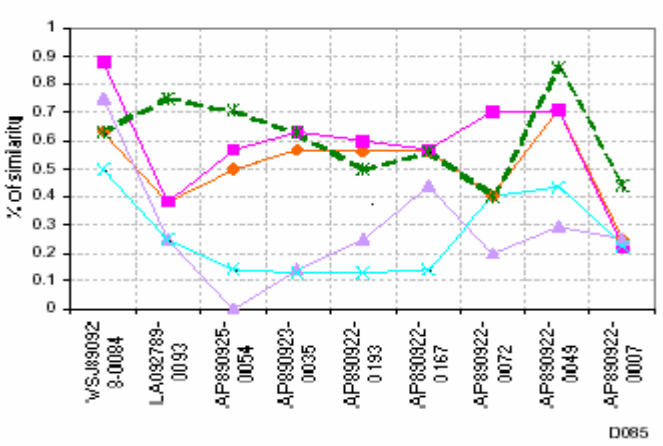


FIGURE 5: Result on Summarization using Document Set D085

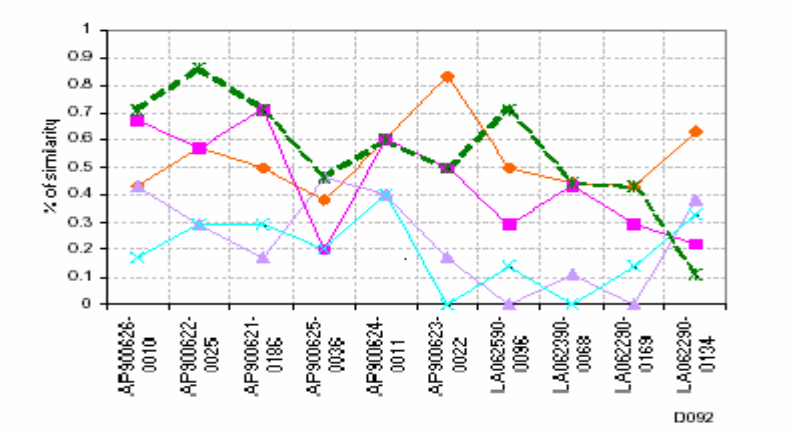
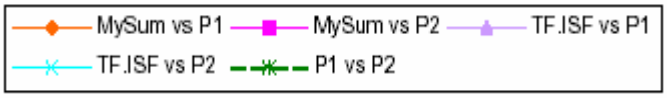


FIGURE 6: Result on Summarization using Document Set D092



7. CONCLUSION AND FUTURE WORK

This paper presented a detailed algorithm in computing the asymmetric similarity between words using fuzzy context sets and topic similarity in extracting the most relevant sentences. The asymmetric word similarity measure words that appear in similar context in the sentences, while topic similarity compute the frequency of overlapping words appear in the sentence and topic. Experiments show that using the combination of both the word similarity and topic similarity able to extract the most important sentences from a document that is fairly close to the manually created summaries. Although MySum did not produce an exact summary to the one created by human, on average, MySum is able to give a representable extractive summary. The difference between MySum and human summarizers in producing summary is only 10 percent. On the other hand, MySum outperforms *tf.isf* when compared against the manually created summaries. In future, we hope to test MySum for multi-document summarization. This work can also be extended in looking at abstract summarization or how to combine similar sentences together as how a human summarizer would do.

8. REFERENCES

1. K. Sparck Jones. *"Automatic Summarizing: Factors and Directions"*. In I. Mani and M.T. Maybury, Editors, Advances in Automatic Text Summarization, Cambridge, MA: The MIT Press, pp 1-12, 1999

2. S.H. Lo, H. Meng, and W. Lam. *"Automatic Bilingual Text Document Summarization"*. In Proceedings of the Sixth World Multiconference on Systematic, Cybernetics and Informatics. Orlando, Florida, USA, 2002

3. S. Yohei ''*Sentence Extraction by tf/idf and Position Weighting from Newspaper Articles (TSC-8)*'' NTCIR Workshop 3 Meeting TSC, pp 55-59, 2002

4. J. Larocca Neto, A.D. Santos, C.A.A. Kaestner, and A.A. Freitas. "*Document Clustering and Text Summarization*". In Proceedings of the 4th Int. Conf. Practical Applications of Knowledge Discovery and Data Mining (PADD-2000), London: The Practical Application Company, pp 41---55, 2000b

5. M. Amini and P. Gallinari. *"The Use of Unlabeled Data to Improve Supervised Learning for Unsupervised for Text Summarization".* In SIGIR, Tampere, Finland, 2002

6. H. Luhn *"The Automatic Creation of Literature Abstracts"*. IBM Journal of Research and Development, 2(92):159 - 165, 1958

7. G. Salton and C. Buckley. "*Term-weighting Approaches in Automatic Text Retrieval".* Information Processing and Management 24, pp 513-523, 1988. Reprinted in: Sparck Jones K. and Willet P. (eds). Readings in Information Retrieval, Morgan Kaufmann, pp 323-328, 1997

8. G.J. Klir and B. Yuan. "*Fuzzy Sets and Fuzzy Logic - Theory and Applications".* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1995

9. J.F. Baldwin. *"Fuzzy and Probabilistic Uncertainties".* In Encyclopedia of AI, 2nd ed., S.C. Shapiro, Editor 1992, Wiley, New York, pp. 528-537, 1992

10. J.F. Baldwin. *"Combining Evidences for Evidential Reasoning".* International Journal of Intelligent Systems, 6(6), pp. 569-616, 1991a

11. J.F. Baldwin, J. Lawry, and T.P. Martin. *"A Mass Assignment Theory of the Probability of Fuzzy Events"*. Fuzzy Sets and Systems, (83), pp. 353-367, 1996

12. J.F. Baldwin, T.P. Martin and B.W. Pilsworth. "*Fril - Fuzzy and Evidential Reasoning in Artificial Intelligence".* Research Studies Press Ltd, England, 1995

13. M.F. Porter. *"An Algorithm for Suffix Stripping".* Program, 14(3):130-137, 1980

14. D. Lin. *"Extracting Collocations from Text Corpora"*. Workshop on Computational Terminology, Montreal, Canada, 1998

15. Z. Harris. *"Distributional Structure".* In: Katz, J. J. (ed.) The Philosophy of Linguistics. New York: Oxford University Press, pp. 26-47, 1985

16. M.A. Azmi-Murad. "*Fuzzy Text Mining for Intelligent Information Retrieval".* PhD Thesis, University of Bristol, April 2005

17. DUC. *"Document Understanding Conferences"*. http://duc.nist.gov, 2002