Word Similarity for Document Grouping using Soft Computing

Masrah Azrifah Azmi Murad

Department of Information Systems
Universiti Putra Malaysia
43400 UPM Serdang, MALAYSIA

Trevor Martin

Department of Engineering Mathematics
University of Bristol
BS8 1TR, UK

Summary
The technology world has provided a more efficient and quicker way of accessing information through the web and databases in organizations that implement information systems in order to achieve a competitive edge. The simplest way of filtering information is to extract keywords in measuring the documents relevance. Nonetheless, getting to the right document is often a problem. Synonymy i.e., two words with the same meaning, for example, *taxi* and *cab* is a major problem in information searching. This work uses the soft computing techniques in the area of information retrieval and they encompass both fuzzy set theory and probability theory. We propose an algorithm for computing asymmetric word similarities (AWS) to overcome the synonymy problem. The algorithm is computed using mass assignment based on fuzzy sets of words. A key feature of our algorithm is that it is incremental, i.e. words (and documents) can be added or subtracted without extensive re-computation. AWS produced similarity measures of consistently 10% higher than *tf.idf* algorithm and performed successful document groupings.

Key words:
fuzzy set, soft computing, asymmetric word similarity, information retrieval

1. Introduction

Information retrieval studies the representation, organization, storage, retrieval and distribution of information [15]. In this context, information may consist of data items such as numbers and names, or any written texts such as e-mails, memos and reports. The goal of an information retrieval system is to satisfy user information needs by retrieving all possible relevant documents, at the same time retrieving as few as possible of the irrelevant documents. A successful information retrieval system enables the user to determine quickly and accurately whether the contents of the documents are satisfactory. Information retrieval plays a significant role in web searching, which allows people access to various sources of information from anywhere and at anytime. Only relevant documents should be returned based on a user's query. In order to better represent the documents, those with similar topics or contents are grouped together. Nonetheless, finding the right document is often a problem. For instance, if we search for a word *orange* the system will return a list of documents concerned with *color*, *fruit*, *Orange County*, or the *mobile phone operator*. In addition, many documents contain ambiguous words, for example, *bank* could be a *river bank* or a *financial institution*, *jaguar* could be a *car* or an *animal* and *domino* could be a *pizza company* or a *game.* User may face with synonymy problems in which two words that could express the same meaning, for example, *taxi* is similar to *cab* and *notebook* is similar to *laptop*.

Soft computing combines various new techniques in artificial intelligence that resembles the human ability in approximate reasoning. The idea of soft computing [20] was introduced by Zadeh in 1994 and has been influenced by his previous work on fuzzy sets [17], analysis of complex systems and decision processes [18], and theory of possibility [19]. The aim is to cope with problems that deals with imprecision and uncertainty. The principle components of soft computing are fuzzy logic, neural network and probabilistic reasoning, with the latter subsuming Bayesian network, genetic algorithm and chaos theory. In this work, the soft computing techniques used are fuzzy set and probability theories in developing the knowledge based systems methods and modeling.

We aim to use the fuzzy sets incorporating mass assignment in finding 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 increased document similarity measures and performed successful document groupings. The remainder of the paper is organized as follows: In section 2 we discuss briefly on vector space model using *tf.idf* weighting scheme, and use it to benchmark against our algorithm, section 3 explains the methodology used, section 4 discusses in detail on the novel algorithm, section 5

Manuscript received August 5, 2007
Manuscript revised August 20, 2007

discusses the results, and finally section 6 concludes the paper.

2. Vector Space Model

There have been several different kinds of document grouping in the literature, among them a vector space model [3] using *tf.idf* [16] technique. The *tf.idf* allows searching using natural language. This method proved to be better than Boolean model in grouping similar documents together by using a keyword matching. Documents and queries are represented as vectors in term space. The algorithm for calculating weight is given by

$$w_{ik} = tf_{ik} * log\ (N/df_k) \quad (1)$$

where tf_{ik} is the frequency of k^{th} term in document i, df_k is the number of documents in which a word occurs and N is the total number of documents in the collection. Cosine measure [3] is used to measure the angle between two vectors, i.e., a document d_j and a user query, q. Two vectors are considered identical if the angle is $\phi = 0$. The degree of similarity of the document d_j with regard to the query q is given by the cosine of the angle between these two vectors, i.e.

$$sim(d_j, q) = \frac{\vec{d_j} \bullet \vec{q}}{\left|\vec{d_j}\right| \times \left|\vec{q}\right|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w^2_{i,j}} \times \sqrt{\sum_{j=1}^{t} w^2_{i,q}}} \quad (2)$$

Many document retrieval systems use the vector model with *tf.idf* technique because it is simple and fast. However, there are a few problems of using the *tf.idf*. For example, *tf.idf* cannot reflect similarity of words and only counts the number of overlapping words and it ignores synonymy and syntactic information. *tf.idf* uses keywords to find documents that contain words that match the query, which could lead to many irrelevant documents returned, as the search will find words in any location and in any document. There could also be some relevant documents missing, as they use different words to express the same interests. Every insertion and deletion on the document collection will force the *idf* [3] value to change and will need to be updated.

In this paper, we demonstrate how finding an asymmetric similarity relation of words performs better document similarity measure, an approach that differs from conventional implementation of similarity measures. Our experiments show that the use of fuzzy sets in computing word similarity can produce higher similarity values between documents in the corpus.

3. Fuzzy Sets and Mass Assignment Theory

Similarity between words is not a crisp relation as there are degrees of similarity, and any representation of word similarity must recognize this. We choose mass assignment theory to represent the uncertainty inherent in this application, as it enables us to use statistical information where available, whilst retaining the intuitive understandability of fuzzy sets.

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 [11]. Unlike a classical set, a fuzzy set does not have a clearly defined boundary by having elements with only a partial degree of membership [5]. For example, consider a height of a person with labels such as *short*, *average*, and *tall*. 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 *short*, *average*, and *tall* in this context.

A mass assignment theory was proposed by Baldwin in 1991 as a general theory for evidential reasoning under uncertainty [4; 5]. This theory is used to provide a formal framework for manipulating both probabilistic and fuzzy uncertainties [5]. Consider the following example taken from [4], 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

$$small = 1 / 1 + 2 / 0.8 + 3 / 0.2 \quad (3)$$

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

$$MA_{small} = \{1\} : 0.2, \{1, 2\} : 0.6, \{1, 2, 3\} : 0.2 \quad (4)$$

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)* [8] 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

LPD_{small} = 1 : 0.2+0.3+0.0667=0.5667,
2 : 0.3+0.0667=0.3667,
3 : 0.0667 (5)

3.1 Semantic Unification

Semantic Unification is a concept in Fril [6] 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

$$medium = 2 / 0.2 + 3 / 1 + 4 / 1 + 5 / 0.2 \quad (6)$$

and the mass assignment would be

$$MA_{medium} = \{3, 4\} : 0.8, \{2, 3, 4, 5\} : 0.2 \quad (7)$$

Thus, the least prejudiced distribution is

$$LPD_{medium} = 2 : 0.05, 3 : 0.45, 4 : 0.45, 5 : 0.05 \quad (8)$$

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

$$about\_3 = 2 / 0.4 + 3 / 1 + 4 / 0.4 \quad (9)$$

with mass assignment as

$$MA_{about\_3} = \{3\} : 0.6, \{2, 3, 4\} : 0.4 \quad (10)$$

We use the point semantic unification algorithm [7] to determine the conditional probability. Thus,

Pr (dice is **about_3** | ice 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.

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

	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

Pr(dice is **about_3**|dice is **medium**) = 0.65

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

$(x)_{f1}$ 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

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

$$= \sum_{x\in X} \mu_{f1}(x)\times LPD_{f2}(x) \quad (11)$$

4. Similarity Measurement

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

Information is stored in documents. Documents may be kept in the World Wide Web or any storage device, for example, disk. These documents contain information which must be retrieved by users in order to develop a useful and meaningful knowledge. Nevertheless, not all

content in the documents is useful in search processing. For example, image or symbols in documents may have less meaning compared to text. Therefore, they need to be removed so that only meaningful data are processed. In this section we outline procedures involved in document preprocessing. Throughout this section, a keyword may also be referred as an index term or simply a term.

The first process is by removing stopwords. Stopwords are common words in English that are meaningless when used as a search terms. These words occur too frequently in a database and are usually ignored by the system when searching is done. Stopwords may be eliminated using a list of stopwords. If a word in the database matches a word in the stoplist, then the word will not be included as part of the query processing. Common stopwords include *the*, *and*, *or*, and *from*. One advantage of using stopwords is that it could reduce the size of the inverted index.

The second process is to stem a word. Morphological variants of words usually have similar meanings. If these words are conflated into a single term, the performance of information retrieval can be improved. This may be done using the process of stemming in such a way that words are stemmed into root form by removing their affixes, i.e. prefixes and suffixes. Variant words such as RELATE, RELATED, RELATING, RELATION and RELATIONS are stemmed by removing various suffixes like -ED, -ING, -ION and -IONS. This leaves the single term as RELATE. In our work, we applied the process of Porter stemming [14] in the programs.

4.2 Similarity Algorithm

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 [12; 13]

> 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 [9]. 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 [1], we compute the frequency distributions of words to fuzzy sets. In Phase II [1], 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

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

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 $W_n(x) = \{w_x, w_{x+1}, ..., w_{x+n}\}$, let $p(x)$ be a word that precedes word x such that

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

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

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

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.

Table 2: Example of Sentences in Document D_j

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.

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*

and

suc(x) be the set of words that succeeds word *x*

while

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
   $m_{cw}[i] = m_{cw}[i-1] - (f_{cw}[i-1] - f_{cw}[i]) * i / T$

Fig. 1 Algorithm for Converting Frequencies to Memberships

The complexity of the above algorithm is the sorting step; nevertheless, the remaining steps are linear in the size of the array. Using the example 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.

Fig. 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 $O(n_i \times n_j)$ operations per semantic unification, where n_j is the cardinality of the fuzzy context set that requires $O(v^2)$ semantic unification and *v* 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 recomputation of the matrix. Figure 3 shows the creation of AWS 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.

Fig. 3 Algorithm for Creating AWS

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. Results and Discussions

To evaluate our method, we compare our system with *tf.idf*. We use Reuters text collection (formerly available from www.research.att.com/~lewis/reuters21578.html) in testing our system, in which 4535 documents are used as a training set and about 4521 documents are used as a test set. Several short queries were used to ensure that our system produces expected results all through out the testing, and these are reported elsewhere. However, in this

paper we will only present the preliminary results [2], using Query 1: *"bomb attack in Madrid"* and Query 2: *"digital library"*. Based on Query 1, the training set has thirty-eight relevant documents retrieved while test set has fifty-four. Based on Query 2, training set has twenty-two documents and test set has thirty documents. Table 3 and 4 show some selection of asymmetric similarity between words taken from an example Query 1.

Table 3: Training – The Fuzzy Approach as a Measure of Contextual Word Similarity

Training			
Words		**Similarity Measures**	
word i	*word j*	$Sim(w_i, w_j)$	$Sim(w_j, w_i)$
injured	would	0.148	0.857
said	quote	0.488	0.024
grain	currency	0.2	0.5
surplus	unwanted	1	1
attack	raid	0.223	0.018
output	produce	0.059	0.111
target	estimate	0.333	0.333
price	policies	0.111	0.042
quantity	bag	0.057	0.2
aid	money	0.5	0.2
arrest	custody	1	0.333

Table 4: Test – The Fuzzy Approach as a Measure of Contextual Word Similarity

Test			
Words		**Similarity Measures**	
word i	*word j*	$Sim(w_i, w_j)$	$Sim(w_j, w_i)$
police	shepherd	0.303	0.091
said	confirm	0.109	0.013
attack	threaten	0.667	0.030
military	pentagon	0.385	0.045
iranian	palestinian	0.173	0.023
spain	country	0.027	0.115
reach	get	0.417	0.167
produce	output	0.182	0.059
increase	prolong	1	0.125
output	amount	0.333	0.091
slump	fell	0.333	0.5

Most word pairs are plausible although some, for example, *grain* and *currency* may be dissimilar. We obtain the value of 1 when two fuzzy sets for both word *i* and word *j* unify, meaning both words occur in similar contexts in the sentences. We emphasized that words are considered to be similar because they frequently share contexts, although some words are quite distant in meaning. Both *tf.idf* and AWS produce the same sets of documents in this case. Figures 4 and 5 [2] show the document-document similarity measures using AWS that is consistently about 10% higher than using *tf.idf*, taken over both document sets, with much less computation required for the AWS calculation. Although the figure doesn't show much significance, it does make a difference when grouping of documents are being done.

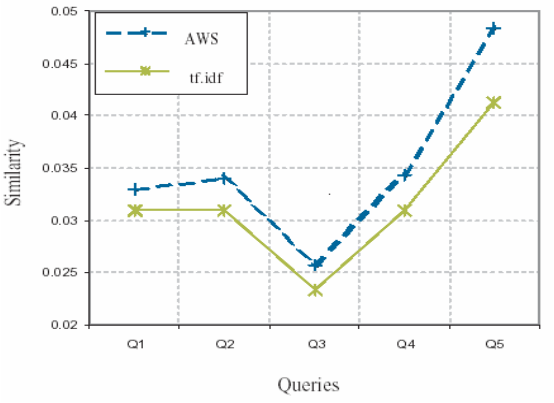

Fig. 4 Average document-document similarity measures using training set

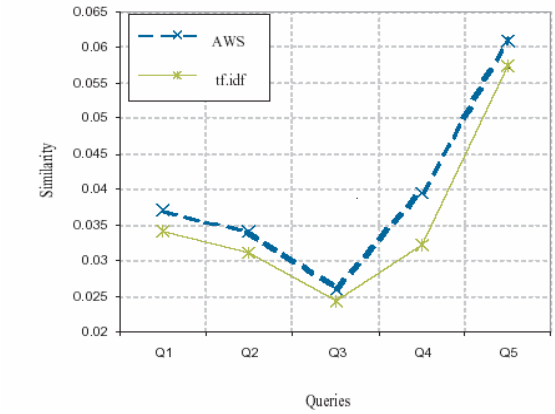

Fig. 5 Average document-document similarity measures using test set

We use the document-document similarity measures to form document groupings based on their similarity values. The grouping is necessary so that similar documents are placed in the same group or cluster. This would help the user to navigate easily through similar documents. The similarity measure is defined as

$$\sum_{i \in doc\,1} \sum_{j \in doc\,2} f_i Sim(w_i, w_j) f_j$$

where *f* is the relative frequency of a word in a document and $Sim(w_i, w_j)$ is the similarity of words outlined in the previous section.

In this section, we present the document groupings using two short queries, i.e. *'apple'* and *'cocoa plantation'*. We group documents using K-means clustering algorithm [10]

with the number of clusters, K=3. From the results shown in Table 5 and 6, AWS managed to produce groups or clusters that contain only similar subjects in them. For example, documents with subject on *'computer'* are grouped together while documents with subject on *'apple'* are placed in a different group. Unlike AWS, one of the group or cluster produced by *tf.idf* contains documents with different subjects.

Table 5: Document Groupings on 'apple'; *double lines indicates clusters*

AWS		***tf.idf***	
Doc	General Subject	Doc	General Subject
D1	Apple computer	D6	Apple fruit
D3	Apple computer	D7	Apple fruit
D4	Apple computer	D1	Apple computer
D5	Apple computer	D5	Apple computer
D2	Fresh apple	D2	Fresh apple
D6	Apple fruit	D4	Apple computer
D7	Apple fruit	D3	Apple computer

Table 6: Document Groupings on 'cocoa plantation'; *double lines indicates clusters*

AWS		***tf.idf***	
Doc	General Subject	Doc	General Subject
D3	Cocoa	D5	Cocoa
D4	Cocoa	D6	Cocoa
D2	Rubber plantation	D1	Coffee plantation
D7	Rubber plantation	D7	Rubber plantation
D1	Coffee plantation	D3	Cocoa
D5	Cocoa	D4	Cocoa
D6	Cocoa	D2	Rubber plantation

6. Conclusions and Future Work

This paper presented a detailed algorithm in computing the asymmetric similarity between words using fuzzy context sets for use in document groupings. The first phase computed the frequencies of n-tuples of context words in a set of documents. The frequencies are converted to fuzzy sets, which represented the family of distributions. The second phase used the frequencies and fuzzy sets to compute the conditional probabilities of two fuzzy sets. The key feature of this algorithm is that it is incremental, such that words and documents can be added or subtracted without extensive recomputation. The effectiveness of AWS has been tested using the Reuters collections. The method is compared against *tf.idf*, and showed that AWS increased the document similarity measures and performed successful document groupings with only relevant subjects appearing in the same group.

The future work on AWS should involve in using the combination of word similarity and semantics of words. The lexical semantics determine if two words have similar meaning or belong to the same lexical category. We hope to investigate the comparative timings and more details of the AWS complexity compared to its competitors.

References

[1] M.A. Azmi-Murad. *Fuzzy Text Mining for Intelligent Information Retrieval.* PhD Thesis. 2005.

[2] M.A. Azmi-Murad and T.P. Martin. Using Fuzzy Sets in Contextual Word Similarity. In *Intelligent Data Engineering and Automated Learning (IDEAL), LNCS 3177*, Springer, pps. 517-522. 2004.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press. 1999.

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

[5] 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.

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

[7] J.F. Baldwin, J. Lawry, and T.P. Martin. Efficient Algorithms for Semantic Unification. In *Proceeding IPMU*, Granada, Spain, pp. 527-532, 1996.

[8] 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.

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

[10] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice Hall, 1988.

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

[12] D. Lin. Extracting Collocations from Text Corpora. *Workshop on Computational Terminology*, Montreal, Canada, 1998.

[13] D. Lin. Automatic Retrieval and Clustering of Similar Words. In *Proceedings COLING/ACL-98*, Montreal, Canada, pp. 768-774, 1998.

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

[15] G. Salton. *Automatic Information Organization and Retrieval*. McGraw-Hill, New York, 1968.

[16] G. Salton and C. Buckley. Term Weighting Approaches in Automatic Retrieval. *Information Processing and Management*, 24(5): pp. 513-523, 1988.

[17] L. Zadeh. Fuzzy Sets. *Information and Control*, 8:338-353, 1965.

[18] L. Zadeh. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Transactions on Systems, Man and Cybernetics*, 3(1):28-44, 1973.

[19] L. Zadeh. Fuzzy Sets as a Basis for a Theory of Possibility. *Fuzzy Sets and Systems*, 1:3-28, 1978.
[20] L. Zadeh. Fuzzy Logic, Neural Networks and Soft Computing. *Communications of the ACM*, 37(3):77-84, 1994.

Masrah received her PhD in Artificial Intelligence from the University of Bristol, United Kingdom in 2005. Since that she has been actively involved in the research area of text mining and information retrieval. She currently heads two research projects funded by the Ministry of Technology and Innvotion (MOSTI) and Research University Grant Scheme, MALAYSIA. She is also a Senior Associate with the Malaysian Industrial-Government Group for High Technology, MALAYSIA.