2007

Blog Analysis with Fuzzy TFIDF

Chi-Shu Ho
San Jose State University

Blog Analysis with Fuzzy TFIDF

A Writing Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Chi-Shu Ho

December 2007

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

_____

Dr. H. Chris Tseng

_____

Dr. Teng Moh

_____

Dr. Soon Tee Teoh

APPROVED FOR THE UNIVERSITY

_____

Abstract

These days blogs are becoming increasingly popular because it allows anyone to share their personal diary, opinions, and comments on the World Wide Wed. Many blogs contain valuable information, but it is a difficult task to extract this information from a high number of blog comments. The goal is to analyze a high number of blog comments by clustering all blog comments by their similarity based on keyword relevance into smaller groups. TF-IDF weight has been used in classifying documents by measuring appearance frequency of each keyword in a document, but it is not effective in differentiating semantic similarities between words. By applying fuzzy semantic to TF-IDF, TF-IDF becomes fuzzy TF-IDF and has the ability to rank semantic relevancy. Fuzzy VSM can be effective in exploring hidden relationship between blog comments by adapting fuzzy TF-IDF and fuzzy semantic for extending Vector Space Model to fuzzy VSM. Therefore, fuzzy VSM can cluster a high number of blog comments into small number of groups based on document similarity and semantic relevancy.

Table of Contents:

List of Figures:

List of Tables:

1. Introduction

Everyday people put tremendous variety of information on the World Wide Web. The World Wide Web has become a useful knowledge base for anyone who is seeking for an answer or sharing information. Today one of the most popular and useful tool on the Internet is blog. The definition of blog provided by Wikipedia is,

> "A blog (short for web log) is a user-generated website where entries are made in journal style and displayed in a reverse chronological order."

People can use blogs to publish what is happening in his or her daily life or opinions about anything, such as politics, entertainment, or product reviews. Other individuals can also post their opinions and comments in another person's blog. According to the blog search engine Technorati, approximately 120,000 blogs are created daily. Many of these blogs contain valuable information, but trying to determine in which blogs contain valuable information is a difficult task due to the abundance of blogs on the World Wide Web.

In order to make sense of the valuable information from blog comments, latent semantic analysis can be used. The latent semantic analysis uses a term-document matrix that takes the frequency of occurrences into consideration. This is accomplished through ordinary Term Frequency–Inverse Document Frequency weight (TF-IDF) which is used to measure the importance of a term by the appearance or frequency of the term in the document. However, one limitation of TF-IDF is that it is unable to recognize semantic similarities between words. A new approach is adapting fuzzy semantic for extending TF-IDF to fuzzy TF-IDF. The fuzzy TF-IDF ranks a term's level of importance and semantic relevancy. Therefore, fuzzy TF-IDF is effective in exploring hidden relationships between blog comments. Using fuzzy TF-IDF and Vector Space Model to cluster the blog comments into small number of groups is one key step in getting an overall idea of a blog document, especially when dealing with a high number of comments.

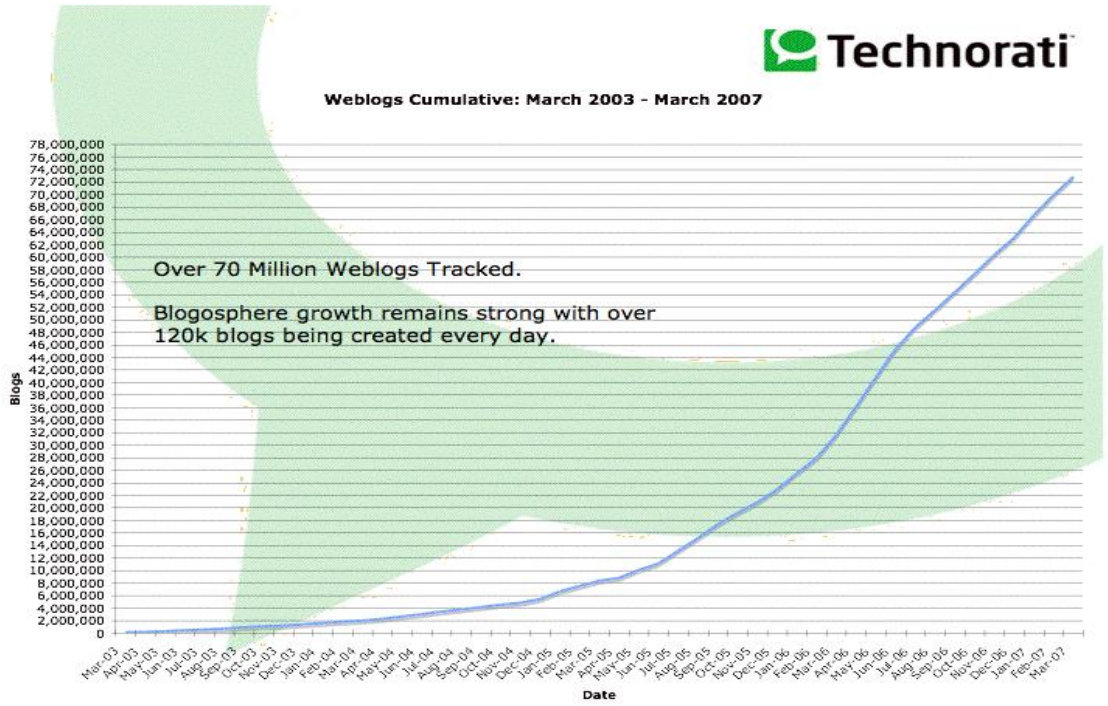

Figure 1.1: Weblog Cumulative: March 2003 – March 2007 from Techbirati

2. Blog Analysis

There are three steps to extract valuable information from a high number of blog comments. The first step is collecting raw blog data from blog sites on the World Wide Web. Second step is to classify blog data by determining the keywords in the blog content. Using keywords to establish relationship between blog data is the key for this project. The technique to analyze blog relationship is latent semantic analysis. The latent semantic analysis uses a term-document matrix that takes the frequency of occurrences into consideration. This is accomplished through TF-IDF weight that is used to measure the importance of a keyword by the appearance or frequency of the keyword in the document. The next section will provide more details on how TF-IDF works. The last step is clustering a high number of blog into smaller groups by using Vector Space Mode with TF-IDF weight to compute blog similarity.

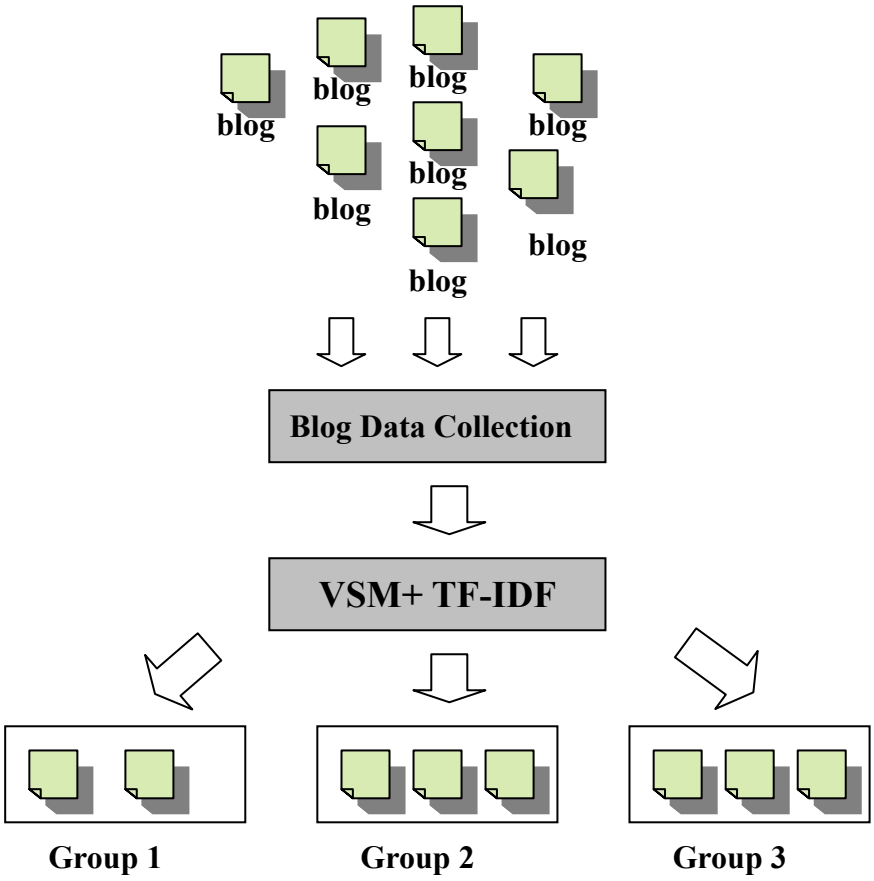

Figure 2.1: Process diagram for blog analysis

3. Term Frequency – Inverse Document Frequency

Term Frequency - Inverse Document (TF-IDF) is used to evaluate the relationship for each word in the collection documents. This algorithm has been widely used in information retrieval and text mining. Words with high TF-IDF weight imply a strong relationship with the document they appear in, suggesting that if that word were to appear in a query, the document could be of interest to the user (Ramos, 2003). Search engines also use variations of the TF-IDF weighting scheme in order to score or rank a document's relevancy.

3.1 TF-IDF Weighting

The term frequency (TF) is a measure of how many times a term appears in the document. Terms appearing many times in a document are most likely to be relevant within the document. (Zhang et al., 2005)

$$\mathrm{tf}_i = \frac{n_i}{\sum_k n_k}$$

n_i = the number of occurrences of the considered term

n_k = the number of occurrences of all terms

The inverse document frequency (IDF) is a measure of the general importance of the term (Zhang et al., 2005). It is calculated by dividing the total number of documents by the number of documents that contain a specific term. High values of IDF mean rare terms, and low values of IDF are common terms.

$$\mathrm{i}df_{i\mathrm{i}} = \log \frac{N}{df_i}$$

N = total number of documents

df_i = the number of documents that contain term *i*

The TF-IDF weighting is term frequencies multiplied by inverse document frequency.

$$\text{TF-IDF} = \mathrm{tf}_{\mathrm{i}} * \log \frac{N}{df_i}$$

The definition of TF-IDF provided by Wikipedia is,

> "A high weight in TF–IDF requires a combination of a high term frequency and a low frequency of documents that contain the term among the whole collection of documents; the weights hence tend to filter out common terms."

TF-IDF counts different words to provide independent evidence of similarity. One of the advantages of using the TF-IDF method is that it does not require large training data sets in order to distinguish between various documents. By representing a document through a vector space model computed via TF-IDF, comparing a document to other documents or queries is simply achieved through the application of a similarity function (Bailey et al., 2001). However, this technique is not effective in handling semantic similarities between words.

3.2 Example

Using sample documents from Table 3.2.1 demonstrate how TF-IDF works and show why TF-IDF cannot differentiate synonymous words. Table 3.2.1 contains four documents with similar content. Document #1 and document #2 can possibly be describing the same house because the words used are very close in meaning. By utilizing TF-IDF algorithm, these four documents can be weighed and examined.

Table 3.2.1: Four documents that contain similar content.

Document	Content
1	This big house has an incredible view.
2	This large house has an excellent view
3	This small house has an awful view
4	This flower is beautiful

Using the TF algorithm above to calculate the term frequency weight, the word *big* only appears in document #1 once out of seven words. Therefore, the term frequency weight of *big* in document #1 is 0.1428. The rest of the term frequency values are derived using the same technique. All the term frequency (TF) weights are displayed in Table 3.2.2.

Table 3.2.2: Term frequency (TF) calculation for TF-IDF experiment #1

Doc	This	big	house	has	an	incredible	view
1	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428
	This	large	house	has	an	excellent	view
2	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428
	This	small	house	has	an	awful	view
3	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428
	This	flower	is	beautiful			
4	0.25	0.25	0.25	0.25			

In the four documents from table 3.2.1, the word *this* appears in all four documents. Using the IDF algorithm the word *this* receives an Inverse document frequency weight of 0 (log 4/4 = 0). The reason for using the logarithm for IDF weight is to avoid a high IDF weight (when large amount of documents divided by small count of term). Table 3.2.3 contains all the Inverse document frequency weights of the four documents.

Table 3.2.3: Inverse document frequency (IDF) calculation for TF-IDF experiment #1

Term	IDF Weight
this	0
big	2.0
large	2.0
small	2.0
flower	2.0
house	0.415
has	0.415
is	2.0
an	0.415
incredible	2.0
excellent	2.0
awful	2.0
beautiful	2.0
view	0.415

Finally, each term's TF value and its corresponding IDF value in each document is multiplied. In Table 3.2.4, all the final values computed from the TF-IDF weight, are displayed below from the four documents. At a glance, document #1, document #2 and document #3 have the same weight but document #4 has a different weight. Based on these weights, the first three documents are grouped separately from document #4. That means document #1, document #2, and document #3 are most relevant. In this example, document #1 and document #2 appear to have very similar meanings. However, TF-IDF is not effective in recognizing synonymous words.

Table 3.2.4: TF-IDF calculation for TF-IDF experiment #1

Doc	This	big	house	has	an	incredible	view
1	0	0.2856	0.059	0.059	0.059	0.2856	0.059
	This	large	house	has	an	excellent	view
2	0	0.2856	0.059	0.059	0.059	0.2856	0.059
	This	small	house	has	an	awful	view
3	0	0.2856	0.059	0.059	0.059	0.2856	0.059
	This	flower	is	beautiful			
4	0	0.50	0.50	0.50			

The previous experiment, TF-IDF cannot differentiate synonymous words, but still able to identify that document #4 is not relevant to the document #1, document #2, and document #3. Next experiment is to test if TF-IDF is able to handle one word in each document. Table 3.2.5 is used for this experiment.

Table 3.2.5: Each document contains only one word.

Document	**Content**
1	big
2	large
3	small
4	beautiful

Using the TF algorithm to calculate the term frequency weight, all four documents only contain one word. Therefore, the term frequency weight for all documents is 1. The term frequency weights are shown below:

Table 3.2.6: Term frequency (TF) calculation for TF-IDF experiment #2

Doc	big
1	1
	large
2	1
	small
3	1
	beautiful
4	1

Because only one word in each document, an Inverse document frequency for each word is 2.0. Table 3.2.7 shows all the Inverse document frequency weights of the four documents.

Table 3.2.7: Inverse document frequency (IDF) calculation for TF-IDF experiment #2

Term	**IDF Weight**
big	2.0
large	2.0
small	2.0
beautiful	2.0

By multiplying each word's TF weight and its corresponding IDF weight in each document, each document has the same TF-IDF weights. In other words, all four documents are relevant to each other. However, only the words *big* and *large* have the closest meaning. The TF-IDF weight is shown below:

Table 3.2.8: TF-IDF calculation for TF-IDF experiment #2

Doc	big
1	2.0
	large
2	2.0
	small
3	2.0
	beautiful
4	2.0

The TF-IDF weight weaknesses have been exposed by these two experiments. The result form Table 3.2.4 and Table 3.2.8 demonstrated that TF-IDF does not have the ability to evaluate synonymous words and a single word in a document.

4. Vector Space Model (VSM)

The Vector Space Model is a widely used model in information retrieval. It represents documents through the terms that the document contains. Normally, terms are single words or keywords. Each term associated weight represents its contribution to the document. In the VSM, each document is represented as a vector with each dimension corresponding to a separate term (Erkan, 2006). The purpose for using VSM is to perform relevancy rankings and clustering of relevant documents.

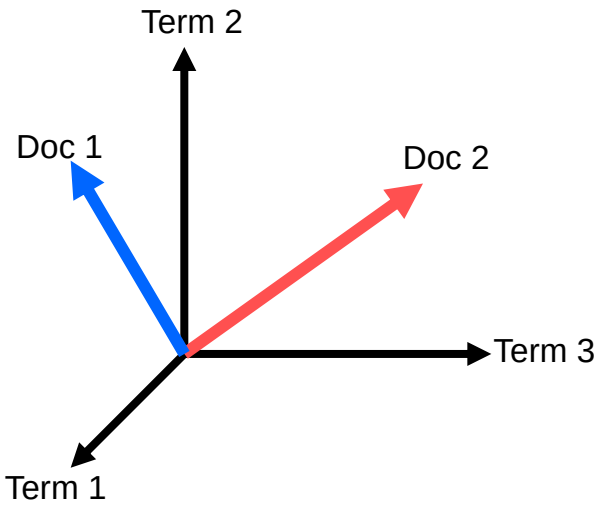

Figure 4.1: Document vector representation

4.1 The Cosine Coefficient

The cosine coefficient is the most commonly used formulae to determine document similarity by which measuring the angle between document vectors. The cosine of the angle between two vectors is an indication of vector similarity and is equal to the dot-product of the vectors normalized by the product of the vector lengths (Schultz, 1999). The cosine similarity between vectors A and B is expressed by:

$$cos(\theta) = \frac{A \bullet B}{|A||B|}$$

The Cosine coefficient provides a similarity score on a scale between 0 and 1. If the similarity score is equal or close to zero that means the documents are very dissimilar. If similarity score is equal or close to 1, that means that the query and the document are very similar. In order to rank documents, measuring the cosine of the angle between a document vector and a query vector is required.

$$\cos_{q_k,d_i} = \frac{\sum_{j=1}^{j=n}\left(W_{q_k,t_j} \times W_{d_i,t_j}\right)}{\sqrt{\sum_{j=1}^{j=n}\left(W_{q_k,t_j}{}^2\right) \times \sum_{j=1}^{j=n}\left(W_{d_i,t_j}{}^2\right)}}$$

W_{q_k} = Query vector

W_{d_i} = Document vector = TF-IDF

4.2 Example

The same documents shown on Table 3.2.1 will demonstrate how VSM works. Using the IDF terms from Table 3.2.3, query vectors for each document are built. The query vectors for each document are shown below in Table 4.2.1.

Table 4.2.1: Query vector for VSM experiment #1

Term	W_{query1}	W_{query2}	W_{query3}	W_{query4}
an	1	1	1	0
awful	0	0	1	0
beautiful	0	0	0	1
big	1	0	0	0
excellent	0	1	0	0
flower	0	0	0	1
has	1	1	1	0
house	1	1	1	0
incredible	1	0	0	0
is	0	0	0	1
large	0	1	0	0
small	0	0	1	0
view	1	1	1	0

Calculate a cosine value representing the similarity between Query #1 and Document #1 vector.

1. The vector for Query #1 looks like:

   W_{query1} = <1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1>

2. The Document #1 vector looks like:

   W_{doc1} = <0.1394, 0, 0, 0.6719, 0, 0, 0.1394, 0.1394, 0.6719, 0, 0, 0, 0.1394>

3. Multiply W_{query1} by W_{doc1}, the vector looks like this:

   $W_{query1} \times W_{doc1}$ = <0.1394, 0, 0, 0.6719, 0, 0, 0.1394, 0.1394, 0.6719, 0, 0, 0,

   0.1394>

4. The sum of all values in the previous step, the result value will be the top half of the cosine formula. It looks like this:

   $$\sum_{j=1}^{j=n}\left(W_{query1,t_j} \times W_{doc1,t_j}\right) = 0.1394 + 0.6719 + 0.1394 + 0.1394 + 0.6719 + 0.1394$$

   $$= 1.901$$

5. The sum of the squared values in the Query #1 vector.

   $$\sum_{j=1}^{j=n}\left(W_{query1,t_j}{}^2\right) = 1\text{^}2 + 1\text{^}2 + 1\text{^}2 + 1\text{^}2 + 1\text{^}2 + 1\text{^}2 = 6$$

6. The sum of the squared values in the Document #1 vector.

   $$\sum_{j=1}^{j=n}\left(W_{doc1,t_j}{}^2\right) = 0.1394\text{^}2 + 0.6719\text{^}2 + 0.1394\text{^}2 + 0.1394\text{^}2 + 0.6719\text{^}2$$

   $$+ 0.1394\text{^}2 = 0.9806$$

7. Multiply the results of step 5 and step 6, and then take the square root. The result at this step will be the bottom half of the formula.

   $$\sqrt{\sum_{j=1}^{j=n}\left(W_{query1,t_j}{}^2\right) \times \sum_{j=1}^{j=n}\left(W_{doc1,t_j}{}^2\right)} = \sqrt{5.8836} = 2.425$$

8. The cosine coefficient is calculated by taking the sum of all values from step 4 and dividing it by the result of step 7. The cosine coefficient in this example will be:

   $\cos_{query1,doc1}$ = 1.901 / 2.425 = 0.7837

Repeating the 8 steps above on the other documents will provide the rest of the similarity score between documents. The complete table of similar scores is shown in Table 4.2.2.

Table 4.2.2: Similarity score for VSM experiment #1. The shaded cells indicate the base document of each row.

	Doc 1	**Doc 2**	**Doc 3**	**Doc 4**
Doc 1	0.7837	0.2635	0.2635	0
Doc 2	0.2635	0.7837	0.2635	0
Doc 3	0.2635	0.2635	0.7837	0
Doc 4	0	0	0	1

The rule for grouping similar documents is determined by using the similarity score of the same document as a base. For example, Document #1 vector compared to Query #1 vector has a score of 0.7837, which should be the highest score within the row because it is compared against the same document. The rule for grouping requires the other documents in the same row to have similarity scores of 80% or higher of the base score. In this example, the base similarity score is 0.7837 multiplied by .80, which equals 0.6270. Consequently, scores must be higher than 0.6270 in order to be grouped together. If there are no documents higher than that score, then base document (document #1 compared to query #1) will be considered its own group.

Applying the grouping rule, there are four different groups. In group one, there is only Document #1 because there are no values greater than or equal 0.6270 (80% similarity of the base document). In the other three groups, the same scenario occurs. For example, the only document that belongs to group two is Document #2. In this experiment, VSM is also unable to recognize synonymous terms.

Next experiment is to test if VSM is capable of computing document similarity of a document which only has one word in it. Documents in Table 3.2.5 are used for this experiment. Using the IDF terms from Table 3.2.7 to build query vectors for each document, the query vectors for each document are shown below in Table 4.2.3.

Table 4.2.3: Query vector for VSM experiment #2

Term	W_{query1}	W_{query2}	W_{query3}	W_{query4}
big	1	0	0	0
large	0	1	0	0
small	0	0	1	0
beautiful	0	0	0	1

Follow the same step to computer similarity score between documents. The similarity score for experiment #2 is shown below.

Table 4.2.4: Similarity score for VSM experiment #2.

	Doc 1	**Doc 2**	**Doc 3**	**Doc 4**
Doc 1	1	0	0	0
Doc 2	0	1	0	0
Doc 3	0	0	1	0
Doc 4	0	0	0	1

Four different groups have been generated after applying the grouping rule. The grouping result from the experiment #2 is identical with the grouping result from the experiment #1. According to the results of the two experiments in this section, VSM also lacks the ability to differentiate synonymous words among documents with only a single word.

5. The Project

5.1 Fuzzy Semantic

Using fuzzy semantic can expand the search of semantically related queries for more relevant results. According to Tseng and Vu, using fuzzy semantic for the search query is much more effective in retrieving web information by extending the search to semantically related fuzzy key phrases. The Fuzzy Semantic search was shown to have an average improvement over the conventional search of up to 50% (Tseng, 2006). Integrating this technique can make TF-IDF more robust and efficient. The fuzzy membership function can be used to relate synonyms in fuzzy set groups. In

Figure 5.1.1, several words are related to the keyword *big*.

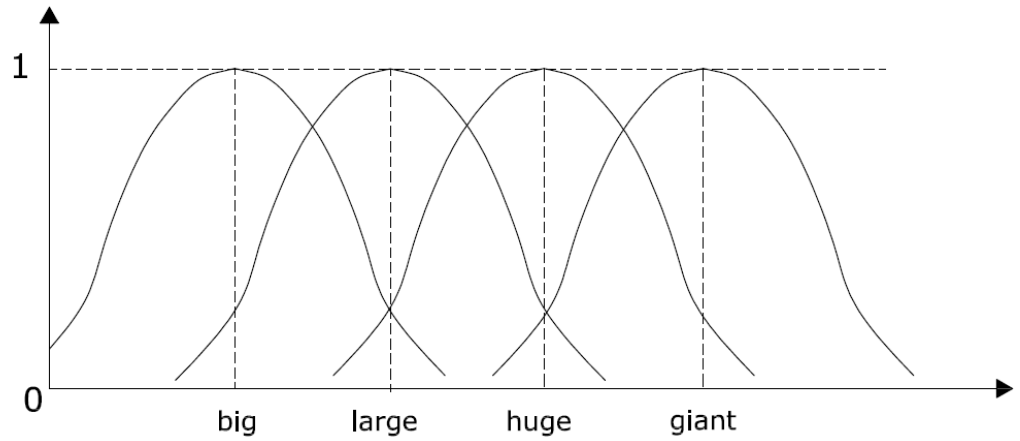

Figure 5.1.1: **Membership function of constraining variable *big***

The word *big* is closer to the word large in accordance with the fuzzy set description above. Thus, the relevance score of these two words will be relatively higher. Calculate the membership function values, $\mu_{FNS}(z)$, of z with regard to the constraining keyword X with the following Gaussian membership function formula: $\mu_{FNS}(z) = e^{\frac{-(z-\bar{z})^2}{\sigma}}$, where $\bar{z}$ is the mean of the Gaussian membership function representing the fuzzy set for the keyword X and σ is a positive constant that determines the shape of the Gaussian membership function.

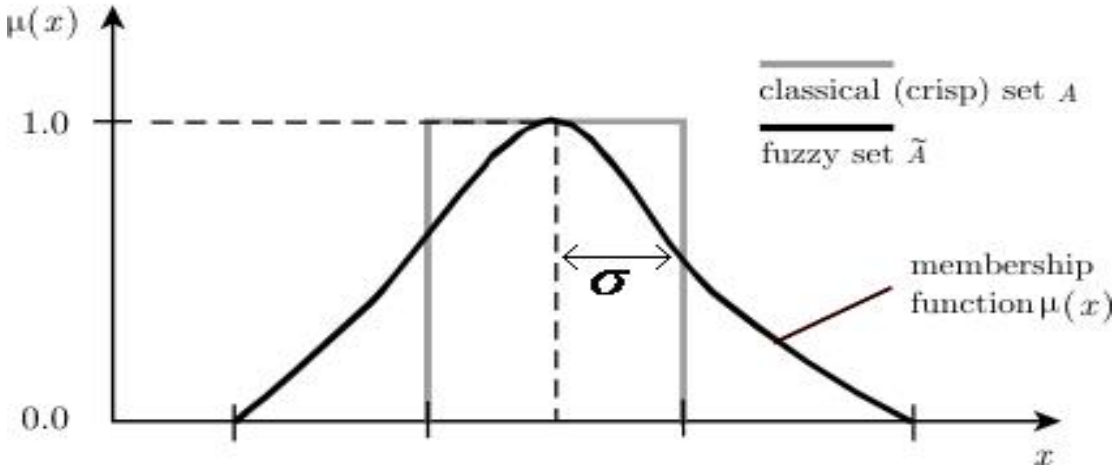

Figure 5.1.2: Gaussian membership function

5.2 Fuzzy TFIDF

By applying fuzzy semantic, TF-IDF can be more effective in handling synonymous words. Thus, TF-IDF becomes fuzzy TF-IDF. Although fuzzy semantic can be applied to either TF or IDF, applying fuzzy semantic to TF is the only way to obtain the results needed. Before explaining the application of fuzzy logic to term frequency (TF), a discussion on why fuzzy logic is not applied to inverse document frequency (IDF) is required. The inverse document frequency (IDF) is a measure of the general importance of the term. It is based on the number of documents that contain the specific term. Even if a term appears multiple times in a document, indicating some importance, IDF still counts that term only once. The score returned by fuzzy membership is based on the relationship between the synonymous words and does not depend on how many times it appears in documents. Therefore, there is no reason to apply fuzzy semantic to IDF. The term frequency (TF) is a measure of how many times a term appears in the document. In either the same document or different documents, users may use different words that are synonymous to describe the same object. Traditional TF-IDF does not understand different words that have the same or close meanings and computes them separately. However, fuzzy TF-IDF is capable of computing synonymous words by applying fuzzy membership function to term frequency in order to evaluate synonymous words. Using mathematics, it can be demonstrated that the fuzzy membership function can be applied to term frequency.

Membership function formula: $\mu_{FNS}(z) = e^{\frac{-(z-\bar{z})^2}{\sigma}}$.

When reducing σ value, the shape becomes sharp. The overlapped area of two words is reduced. Also, the value of $\mu_{FNS}(z)$ goes down. When σ value is close to 0, the words can be considered to be unrelated to other words in the fuzzy set. In other words, no fuzzy semantic applies and the value of $\mu_{FNS}(z)$ is close to 0.

When the σ value increases, the shape becomes flat and the overlapped area of the two words increases. That means that each word in the fuzzy set is closely related.

When the shape is completely flat, the value of $\mu_{FNS}(z)$ will be equal to 1.

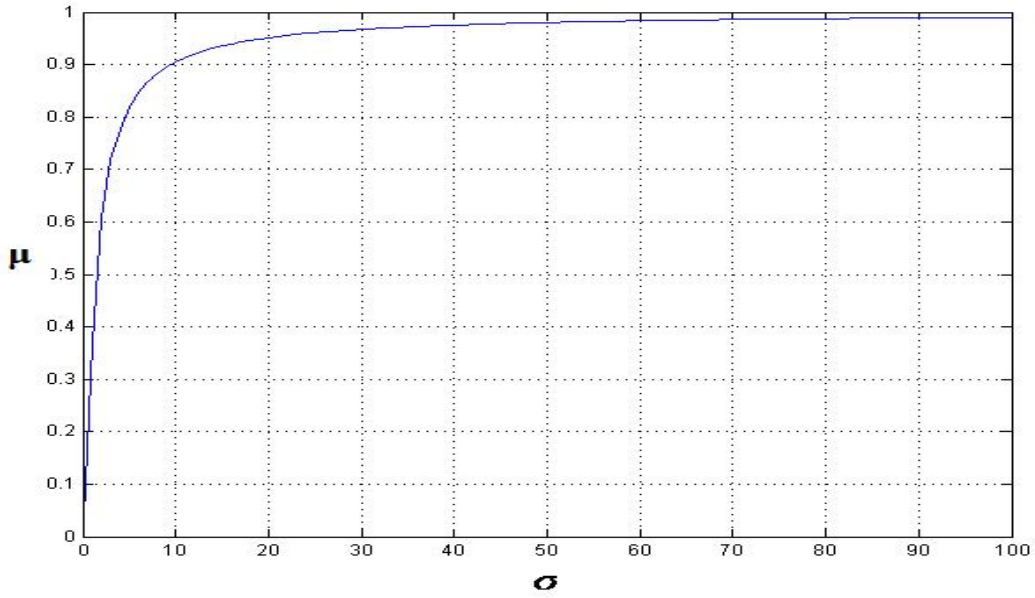

Figure 5.2.1: **Fuzzy membership score versus σ value**

A fuzzy set is required to demonstrate how to apply fuzzy semantic to TF-IDF. Table 5.2.1 is a fuzzy set that contains 3 different semantically related groups. To be consistent, all terms σ value are equal to 10.

Table 5.2.1: Three semantically related groups in a fuzzy set

Group	Term	σ	Term	σ	Term	σ
1	big	10	large	10	huge	10
2	fine	10	good	10	nice	10
3	wonderful	10	excellent	10	incredible	10

Using to the same documents from Table 3.2.1 and by manipulating the value of σ, the Table 5.2.2 shows the TF-IDF value plus the fuzzy membership score with σ =10.

Table 5.2.2: Fuzzy term frequency with σ = 10

	This	big	house	has	an	incredible	view
1	0.1428	1.048	0.1428	0.1428	0.1428	1.048	0.1428
	This	large	house	has	an	excellent	view
2	0.1428	1.048	0.1428	0.1428	0.1428	1.048	0.1428
	This	small	house	has	an	awful	view
3	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428
	This	flower	is	beautiful			
4	0.25	0.25	0.25	0.25			

Table 5.2.3 is shows TF-IDF value plus fuzzy membership score with σ = 0.0001.

Table 5.2.3 Fuzzy term frequency with σ = 0.0001

	This	big	house	has	an	incredible	view
1	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428
	This	large	house	has	an	excellent	view
2	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428
	This	small	house	has	an	awful	view
3	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428	0.1428
	This	flower	is	beautiful			
4	0.25	0.25	0.25	0.25			

From the tables above, when σ = 10, term frequency of *big*, *large*, *incredible*, and *excellent* increase, but the rest of the term values remain the same. When σ = 0.0001, the term frequency values in the Table 5.2.1 are identical with those from Table 3.2.2 which use TF-IDF.

Table 5.2.4: Fuzzy TF-IDF vectors

	This	big	house	has	an	incredible	view
1	0	2.095	0.059	0.059	0.059	2.095	0.059
	This	large	house	has	an	excellent	view
2	0	2.095	0.059	0.059	0.059	2.095	0.059
	This	small	house	has	an	awful	view
3	0	0.2856	0.059	0.059	0.059	0.2856	0.059
	This	flower	is	beautiful			
4	0	0.50	0.50	0.50			

By classifying each document based on fuzzy TF-IDF weight, it can easily be determined that document #1 and document #2 are most related.

The documents in Table 3.2.5 will test if fuzzy TF-IDF is able to identify a single word in each document. By applying fuzzy semantic with $\sigma = 10$, the results of term frequency value are shown below.

Table 5.2.5: Fuzzy term frequency with σ = 10 for a single word in a document

Doc	big
1	1.904
	large
2	1.904
	small
3	1
	beautiful
4	1

The results clearly show that the term frequency value of *big* and *large* increased, and the term frequency values of *small* and *beautiful* remain unchanged. The inverse document frequency values remain the same as before. By multiplying each word's TF weight in Table 5.2.5 and its corresponding IDF weight in Table 3.2.7, the fuzzy TF-IDF weights for each document are shown in Table 5.2.6.

Table 5.2.6: Fuzzy TF-IDF vector for a singe word in a document

Doc	big
1	3.808
	large
2	3.808
	small
3	2.0
	beautiful
4	2.0

The result from Table 5.2.6 showed that fuzzy TF-IDF weight has the ability to compute a single word in a document. Therefore, fuzzy TF-IDF is able to evaluate synonymous words and a single word in a document.

5.3 Fuzzy Vector Space Model

Since fuzzy semantic improves TF-IDF's ability to handle synonymous words, does it mean fuzzy semantic can be applied to a vector space model? The cosine coefficient measuring document similarity is computed using the cosine angle between the query vector and the document vector. If each term weight in document vector equals its TF-IDF weight, is it possible to apply fuzzy semantic to vector space by using fuzzy TF-IDF weight or is it also possible to apply fuzzy semantic to query vector as well?

5.4 Fuzzy Vector Space Model Experiment

By using the fuzzy TF-IDF for document vector to compute document similarity, the similarity score is shown below in Table 5.4.1.

Table 5.4.1: Similarity score for applying fuzzy semantic to document vector

	Doc 1	**Doc 2**	**Doc 3**	**Doc 4**
Doc 1	0.6095	0.0326	0.2299	0
Doc 2	0.0326	0.6095	0.2635	0
Doc 3	0.0326	0.0326	0.7839	0
Doc 4	0	0	0	1

From table 5.4.1, it can be demonstrated that the cosine angle between query #1 and document #2 is very close to zero. In addition, the similarity score between query #1 and document #3 is higher than query #1 and document #3 despite the fact that document #1 and document #3 are very different. The current scores indicate that document #1 and document #2 are very dissimilar. However, it appears document #1 and document # 2 should be very similar.

By using an example, it can be demonstrated why query #1 and document #2 has a low similarity score. The term types in query vector are *an*, *awful*, *beautiful*, *big*, *excellent*, *flow*, *has*, *house*, *incredible*, *is*, *large*, *small*, and *view*. The vectors of query #1 are <1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1>, and the vectors of document #2 are <0.0199, 0, 0, 0, 0.7065, 0, 0.0199, 0.0199, 0, 0, 0.7065, 0, 0.0199>. By multiplying the vectors of query #1 to vectors of document #2, the equation is shown below:

$$W_{\text{query1}} \times W_{\text{doc2}} = <0.0199, 0, 0, 0, 0, 0, 0.0199, 0.0199, 0, 0, 0, 0, 0.0199>$$

The sum of all values:

$$\sum_{j=1}^{j=n} \left(W_{\text{query1},t_j} \times W_{\text{doc2},t_j} \right) = 0.0199 + 0.0199 + 0.0199 + 0.0199 = 0.0796$$

Although the vectors from Document #2 use fuzzy TF-IDF weight, the vectors of Query #1 are not able to identify the relationship between synonymous words. Therefore, the term *large* and *incredible* in Query #1 are equal to 0 and when this value is multiplied with the vectors of Document #2, their vector results become 0. Then the sum of the product of the vectors from Document #2 and Query #1 will have a very small value. Therefore, the cosine coefficient between Document #2 and Query #1 will be:

$$\mathbf{COS}_{\text{query1,doc2}} = 0.0796 / 2.445 = 0.0326$$

Therefore, applying fuzzy semantic to the both query vector and document vector is required.

To apply fuzzy semantic to the query vector, find out which term exists in any fuzzy set group is needed. Using the same fuzzy set groups from Table 5.2.1, terms *big* and *large* are in the fuzzy set group #1, and terms *excellent* and *incredible* are in the group #3. If the vector of these four terms is 0 in any query vector, applying fuzzy

semantic to these terms is necessary. For example, in vector of query #1, the term *big* has vector of 1 and another term *large* has vector of 0, that means the term *large* has no relationship with the term *big*. In other words, these two terms are very dissimilar. Because term *big* and *large* are in the same fuzzy set group (Table 5.2.1), that indicates there is a relationship between these two terms. Since the term *large* has vector of 0, fuzzy semantic can be applied to the term *large* in order to establish the relationship with term *big*. If the term *large* vector is greater than or equal to 1 in vector of query #1, then it is not necessary to apply fuzzy membership because there is no hidden relationship between them. As a result, utilizing the fuzzy membership score between the term *big* and *large* can replace the vector of term *large* in vector of query #1.

By applying fuzzy semantic to all query vectors, the new query vector table is shown below.

Table 5.4.2: Query vectors with fuzzy membership of each document

Term	W_{query1}	W_{query2}	W_{query3}	W_{query4}
an	1	1	1	0
awful	0	0	1	0
beautiful	0	0	0	1
big	1	0.9048	0	0
excellent	0.9048	1	0	0
flower	0	0	0	1
has	1	1	1	0
house	1	1	1	0
incredible	1	0.9048	0	0
is	0	0	0	1
large	0.9048	1	0	0
small	0	0	1	0
view	1	1	1	0

Now the vectors of query #1 are <1, 0, 0, 1, 0.9048, 0, 1, 1, 1, 0, 0.9048, 0, 1>. Multiply the vectors of query #1 to vectors of document #2, the result is shown below:

$W_{query1} \times W_{doc2}$ = <0.0199, 0, 0, 0, 0.6392, 0, 0.0199, 0.0199, 0, 0, 0.6392, 0, 0.0199>

Again, sum of all values above will show:

$$\sum_{j=1}^{j=n}\left(W_{\text{query1},t_j} \times W_{\text{doc2},t_j}\right) = 0.0199 + 0.6392 + 0.0199 + 0.0199 + 0.6392 + 0.0199$$

$$= 1.358$$

The value from the bottom half of bottom half the cosine coefficient formula is shown below:

$$\sqrt{\sum_{j=1}^{j=n}\left(W_{\text{query1},t_j}{}^2\right) \times \sum_{j=1}^{j=n}\left(W_{\text{doc2},t_j}{}^2\right)} =$$

$$\sqrt{(1^2 + 1^2 + 0.9048^2 + 1^2 + 1^2 + 1^2 + 0.9048^2 + 1^2)} \times$$

$$\sqrt{(0.0199^2 + 0.7065^2 + 0.0199^2 + 0.0199^2 + 0.7065^2 + 0.0199^2)}$$

$$= \sqrt{7.637 \times 0.9998} = 2.763$$

Finally the cosine coefficient between Document #2 and Query #1 will be:

$$\text{COS}_{\text{query1,doc2}} = 1.358 / 2.763 = 0.491$$

By applying the fuzzy TF-IDF for document vectors and query vectors, the similarity score is shown below in Table 5.4.3.

Table 5.4.3: Similarity score for applying fuzzy semantic to document vector and query vector

	Doc 1	**Doc 2**	**Doc 3**	**Doc 4**
Doc 1	0.5403	0.4916	0.2037	0
Doc 2	0.4916	0.5402	0.2038	0
Doc 3	0.0326	0.0326	0.7839	0
Doc 4	0	0	0	1

Categorizing documents into a group based on similarity score between documents is needed. According to the grouping rule, the base document in the first row is document #1, and its similarity score is 0.5403. To group any document that is 80% similar or higher to document #1, it is necessary to determine which similarity score in the first row is equal or higher to 0.4322 ($0.5403 \times 80\% = 0.4322$). In the first

row, only the similarity score of document #2 qualify for grouping with document #1. In the second row, only document #1 is higher than the base document (document #2). Therefore, document #1 and document #2 will group together. In the third and fourth rows, no document similarity score is equal or higher than the base document. Thus, base documents in the third and fourth rows will own its group. As a result, these four documents will be categorized into three groups that are shown below.

Table 5.4.4: Group information

Group 1	document #1, document #2
Group 2	document #3
Group 3	document #4

The results Table 5.4.4. shows that fuzzy VSM is able to recognize synonymous terms and cluster similar documents into groups. Next experiment uses documents from Table 3.2.5 to test if fuzzy VSM is capable of clustering documents that only have a single word in it. Applying fuzzy semantic to the query vector from Table 4.2.3, the new query vector table is shown below.

Table 5.4.5: Query vector with fuzzy membership – single word experiment

Term	W_{query1}	W_{query2}	W_{query3}	W_{query4}
big	1	0.9048	0	0
large	0.9048	1	0	0
small	0	0	1	0
beautiful	0	0	0	1

After applying fuzzy semantic, the value of term *large* in the query vector #1 and the value of term big in the query vector #2 increase to 0,9048, and the rest of the term values remain the same. Using the same method to computer the similarity score between query vectors and document vectors, the similarity score for this experiment is shown in Table 5.4.6.

Table 5.4.6: Similarity score between each document - single word experiment

	Doc 1	Doc 2	Doc 3	Doc 4
Doc 1	0.7415	0.6709	0	0
Doc 2	0.6709	0.7415	0	0
Doc 3	0	0	1	0
Doc 4	0	0	0	1

Following the grouping rule for clustering document, the base document score for the first and second rows is 0.5932 (0.7415×80% = 0.5932). Therefore, fuzzy VSM will categorize document #1 and document #2 into the same group. Because there is no document similarity score is equal or higher to the base document's score in the third and forth rows. Thus, base documents in the third and fourth rows will own its group In this experiment, four documents have been categorized into three groups that is shown below.

Table 5.4.7: Group information - single word experiment

Group 1	document #1, document #2
Group 2	document #3
Group 3	document #4

5.5 Using Large Data Set

The previous sections demonstrated how to apply fuzzy semantic to Term Frequency Inverse Document (TF-IDF) and Vector Space Model (VSM). They also show how fuzzy TF-IDF and fuzzy VSM is used to differentiate synonymous words and reveal hidden relationship between documents. Now it is time to test fuzzy TF-IDF and fuzzy VSM on the real blog comments. fuzzy TF-IDF and fuzzy VSM group all comments based on comment's similarity. The blog topic is "I Guess We Can Add pervert" and post on the topix.net. There are 70 comments in the topic thread. Applying a fuzzy semantic set with a different σ value to these 70 comments demonstrate how fuzzy TF-IDF and fuzzy VSM can classify comments into groups. Therefore, the difference between tradition TF-IDF and VSM and fuzzy TF-IDF and fuzzy VSM will be easy to see.

Use the fuzzy set group from Table 5.2.1. When σ value increases from 0 to 20, the group size is affected, as shown in the red line in the graph below.

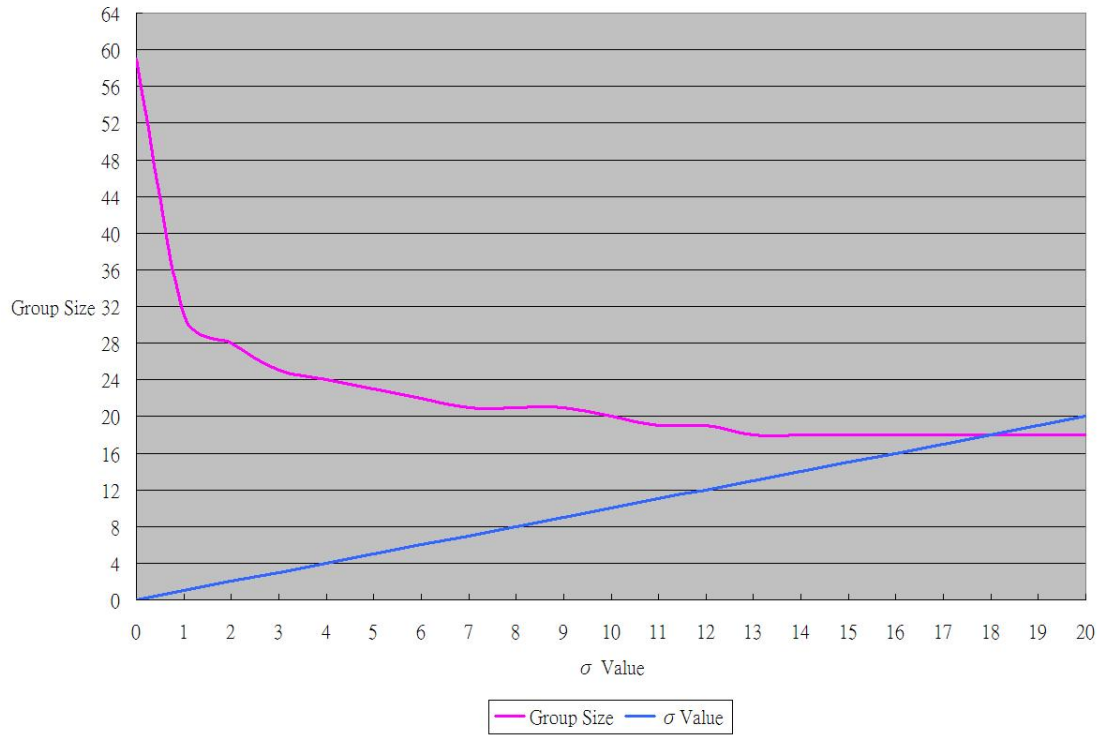

Figure 5.5.1: σ value versus group size with sample fuzzy set group

When σ value is 0, that means there is no fuzzy semantic to applied to TF-IDF and VSM. We classified 70 comments into 59 groups. When σ value starts increasing, the total number of group size begins decreasing. That is because fuzzy semantic has been found the hidden relationship between all comments. After σ value passes 20, the total number of group size will remain steady at 18. In other words, no hidden relationship between all comments has been found. When using fuzzy set group from Table 5.2.1, 70 comments have been distinguished into 18 groups. We used a new fuzzy set that is developed for this blog topic to test if that will affect the total number of groups. There are 18 groups in this new fuzzy set. Groups in the fuzzy set are shown below.

Table 5.5.1: Relevancy group list in the fuzzy set

Group 1	president, presidency, clinton, whitehouse
Group 2	good, better, nice, best, greatest, wonderful, awesome
Group 3	lie, lies, lied, lying, cheated, liar
Group 4	child, children, boys, girls
Group 5	terrorism, terror, terrorists, bombing, killing, torture
Group 6	war, weapons, soldiers, fight, shooting, harm, destroy
Group 7	big, large, largest
Group 8	weak, weakness, poor
Group 9	approved, prove, proves, proven
Group 10	corrupt, corruption, scandals, scum
Group 11	result, answer, resolution
Group 12	government, congress, congressmen, democrat, democrats, republican, republicans, politicians

Figure 5.5.2 shows how group size will be affect by increasing σ value in the new fuzzy set group.

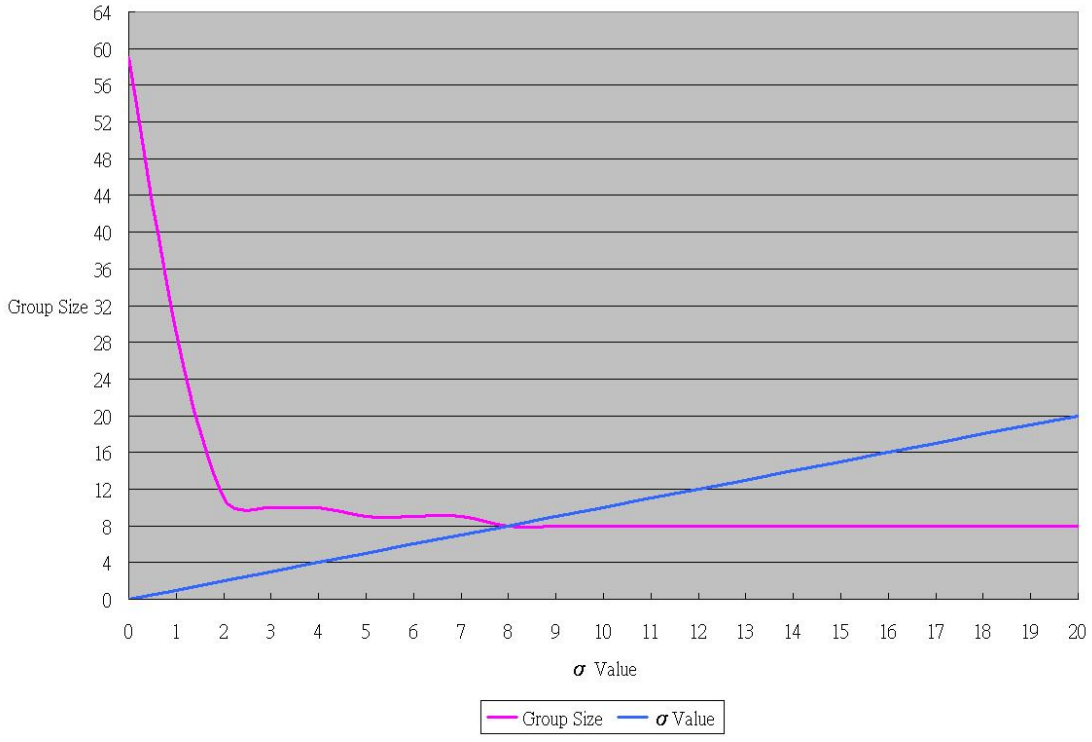

Figure 5.5.2: σ value versus group size when applying new fuzzy set

Because of the new fuzzy set, more hidden relationships between comments have been exposed. In this experiment, all comments have been distinguished into 8 groups when σ value is 8. A σ value of 8 yields 8 groups in this current experiment but in the previous figure 5.5.1, a σ value of 8 yields 22 groups. We can see that the group size has dropped significantly. The next experiment is to test if fewer groups in the fuzzy set can be another factor that affects fuzzy TF-IDF and fuzzy VSM result. Instead of using 12 groups like in the previous experiment, only use six groups are used, group # 1 to group #6 in the fuzzy set from Table 5.5.1. The group size versus σ is shown below.

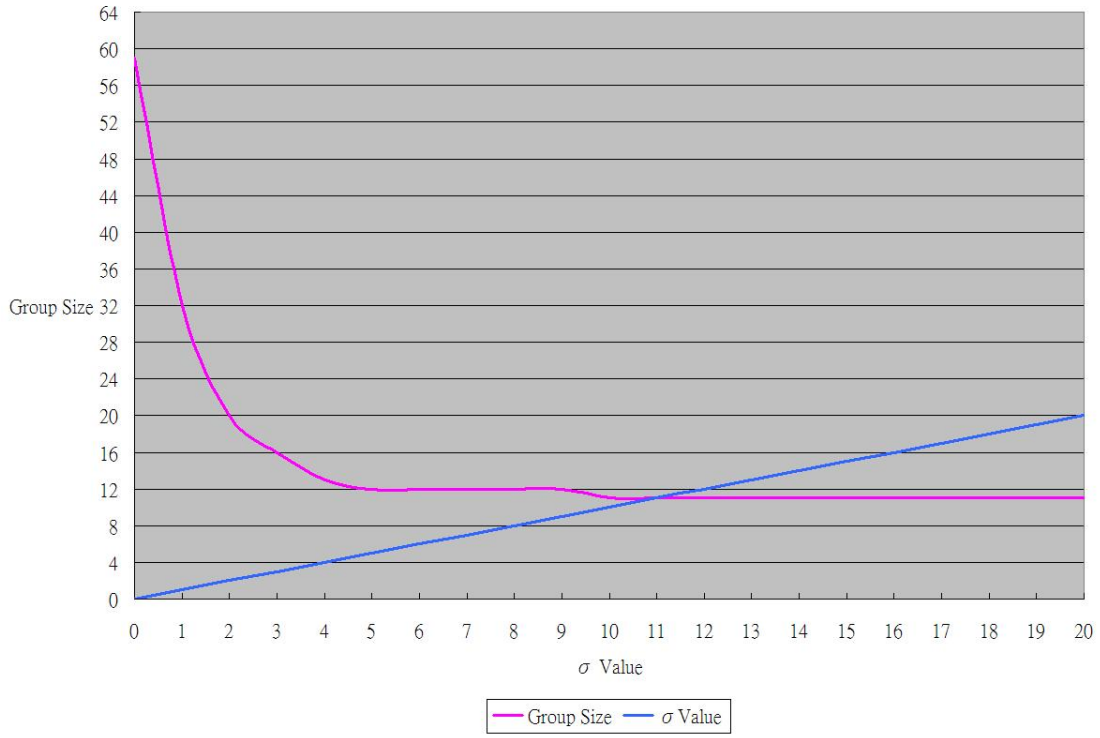

Figure 5.5.3: σ value versus group size with half groups in the fuzzy set

As shown in Figure 5.5.3, the results demonstrate that once the σ value reaches 10, the number of groups decreases to 11. Comparing the results from Figure 5.5.3 with those from Figure 5.5.2, showed only half the number of groups from Table 5.5.1 in the fuzzy set has been used. However, fuzzy VSM can still reveal most hidden relationships between comments. In this experiment, the total number of groups can be changed by using a different fuzzy set. Therefore, both σ value and fuzzy set groups are major factors that affect fuzzy TF-IDF and fuzzy VSM.

5.6 Result Analysis

Based on the results of Figure 5.5.1, Figure 5.5.2, and Figure 5.5.3, there are two major factors that affect the grouping of similar documents: the σ value and semantically related groups in the fuzzy set. The σ value controls a term's shape. When a term's σ value decreases, the overlapped area of two terms decreases. In other words, these two terms are less related to each other and have a low fuzzy membership score. When a term's σ value increases, the overlapped area of the two terms increases. Therefore, these two terms are more related and will have a high fuzzy membership score. High or low fuzzy membership score will have an effect on fuzzy TF-IDF weight and impact the similarity score computed by fuzzy VSM. That is why fuzzy VSM clusters comments into fewer groups when the σ value increases. When deciding a term's σ value the amount of information required should be taken into consideration.

In addition, using different semantically related groups in the fuzzy set will affect grouping result. When analyzing blog comments, a specific fuzzy set should be developed for that blog. Therefore, a specific fuzzy set can cluster similar comments into groups and retrieve the information based on the semantically related keyword groups in the set. The easiest way to develop a fuzzy set is using inverse document frequency (IDF) to collect all terms with high IDF value, and then classifying related terms into semantically related groups. By using different fuzzy sets, VSM is able to cluster blog comments from the different topic threads into relevant groups.

6. Conclusion

This project uses keywords in each document to represent documents and then use these keywords to identify document similarities. Term Frequency - Inverse Document (TF-IDF) has been used to evaluate a keyword's importance in a collection documents, but it lacks the ability to differentiate synonymous words. Without the ability to differentiate synonymous words, TF-IDF losses accuracy. Applying fuzzy semantic to TF-IDF improves TF-IDF ability to recognize synonymous words and to improve its accuracy. Different experiments in the previous section, has demonstrated that fuzzy TF-IDF can be effective in handling semantic similarities between words. By applying fuzzy semantic to Vector Space Model (VSM) it is able to explore hidden relationships between documents and in turn VSM uses that information to cluster similar documents into groups.

Although fuzzy TF-IDF and fuzzy VSM can be effective in differentiating synonymous words, there are still many areas that can be improved such as developing more semantically related groups in a fuzzy set to explore hidden relationship between documents. This in turn makes it possible for searching and comparing algorithm to more efficient by reducing computing time. fuzzy TF-IDF has the ability to recognize similar words. The project's goal is to analyze a high number of blog comments by clustering all blog comments by their similarity based on keyword relevance into smaller groups. After utilizing different methods in several experiments, the results obtained from fuzzy TF-IDF and fuzzy VSM were satisfactory.

7. Reference

Bailey C., El-Beltagy, S.R. and Hall, W. (2001). "Link Augmentation: A ContextBased Approach to Support Adaptive Hypermedia". In Proceedings of the 3rd Workshop on Adaptive Hypertext and Hypermedia, August 14-18, rhus, Denmark, pp 55-62.

Bestgen, Y. (2006). Improving Text Segmentation Using Latent Semantic Analysis: A Reanalysis of Choi, Wiemer-Hastings, and Moore (2001). Computational Linguistics, 32(1), 5-12.

Dumais, S., Furnas, G., Landauer, T., Deerwester, S., Harshman, R. (1988). Using Latent Semantic Analysis to Improve Access to Textual Information. Proceedings of The SIGCHI Conference on Human Factors in Computing Systems, 281-285.

Erkan, G. 2006. Language model-based document clustering using random walks. In Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, 479-486.

Liu, W., Yao, J., Yao, Y., (2005). Constructive Fuzzy Sets with Similarity Semantics. Annual Conference of the North American Fuzzy Information Processing Society, 591-596.

Ramos, J. (2003) Using TF-IDF to Determine Word Relevance in Document Queries. Retrieved April 10, 2007, from Rutgers University, Institute for Computer Science department Web site, http://www.cs.rutgers.edu/~mlittman/courses/ml03/iCML03/papers/ramos.pdf

Schultz J.M., Liberman, M. (1999) Topic Detection and Tracking using idf Weighted Cosine Coefficient. Proceedings of the DARPA Broadcast News Workshop, 189-192.

Sun, Y., Karray, F., Basir, O., Sun, J., Kamel, M. (2005). Fuzzy Methodology for Enhancement of Context Semantic Understanding. IEEE Internatinoal Conference on Fuzzy System, 143-148.

Tseng, C., Yu, T. (2006). A perception-based search with fuzzy semantic, *Fuzzy Logic and The Semantic Web*, Elsevier, Amsterdam, .

Wang, X., Sun, J., Chen, Z., Zhai, C. (2006). Latent Semantic Analysis for Multiple-Type Interrelated Data Objects. Proceedings of The 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 236-243.

Xu, C. (2004). A Keyword-Based Semantic Prefetching Approach in Internet News Services. IEEE Transactions on Knowledge and Data Engineering, 16(5), 601-611.

Zhang, Q., Zhang, L., Dong S., Tan J. (2005) Document indexing in text categorization Proceedings of 2005 International Conference on Volume 6, Issue , 18-21.