The Application of Fuzzy Logic to the Construction of the Ranking Function of Information Retrieval Systems

M.S. Project Report

Neil O. Rouben

Department of Computer Science

University of Massachusetts, Amherst

First Advisor: James Allan

Second Advisor: William Kilmer

January 2005

Abstract

The quality of the ranking function is an important factor that determines the quality of the Information Retrieval (IR) system. Each document is assigned a score by the ranking function and this score indicates the likelihood of relevance. In this paper the approach to construction of the ranking function with the use of fuzzy logic is explored. Factors from tf.idf weighting scheme are used in the construction of Ranking Fuzzy Inference System (R-FIS). R-FIS is demonstrated to be a user friendly model that could be easily extended and modified. The resulting R-FIS consists of the following fuzzy logic rules that represent basic tf.idf principles: if (coord is high) → (relevance is high); if (coord is low) → (relevance is low); and for each of the terms: if (tf is high) and (idf is high) → (relevance is high); if (tf is low) and (idf is low) → (relevance is low). Performance of R-FIS is approximately equal to that of the baseline tf.idf based system Apache Lucene (P10 difference -0.82%; MAP difference -0.61%). System's performance is evaluated on the data from NIST TREC 2004 Robust Retrieval Track.

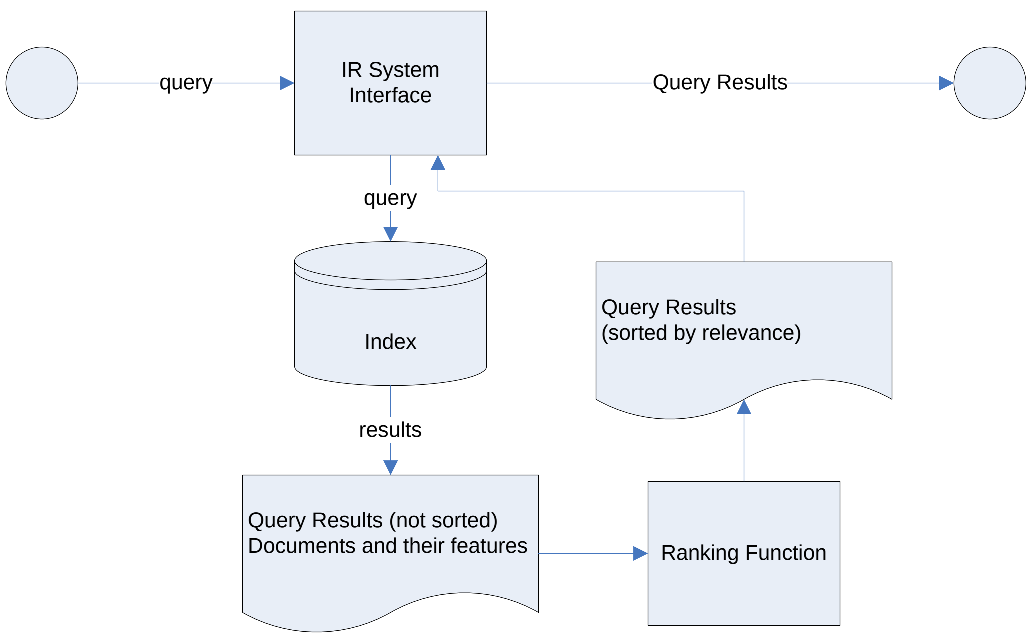
The approach to further increase performance of R-FIS by tuning its parameters with the use of Adaptive Neuro-Fuzzy Inference System (ANFIS) is explored. Tuning parameters of R-FIS with ANFIS resulted in a small improvement of P10 +1.89%. A customized tuning method is likely to achieve a better performance increase.

The approach of automatically deriving initial rules for the R-FIS with the use of ANFIS is explored. Initial rule derivation with the use of ANFIS with subtractive clustering has exhibited potential. In order to be able to better understand and extract knowledge from the resulting structure of the R-FIS more work is required investigating various approaches of analysis.

1 Introduction

1.1 Information Retrieval System

The amount of information accessible electronically has been growing at an astounding rate. An Information Retrieval (IR) system is a tool that allows the efficient searching of vast amounts of information. The ability to return at least passable results for any topic is an important feature of any information retrieval system [14]. The quality of the ranking function is an important factor that determines the quality of the Information Retrieval (IR) system. The ranking function assigns a score to each document. A document with a higher score is more likely to be relevant than a document with a lower score.



1.2 Approaches to Defining Ranking Function of IR system

1.2.1 Boolean Model

The Boolean model is based on set theory and Boolean algebra. Queries are represented through Boolean expressions. The Boolean model considers that index terms are either present or absent, represented by binary weights {0, 1}. The similarity of document d_j to the query q is defined as:

$$sim(d_j, q) = \begin{cases} 1 & if \quad \exists q_{cc} \mid (q_{cc} \in q_{dnf}) \wedge (\forall_{k_i}, g_i(d_j) = g_i(q_{cc})) \\ 0 & otherwise \end{cases}$$

where:

q_{dnf} - query in the disjunctive normal form

q_{cc} – any of the conjunctive components of q_{dnf}

The main advantages of the Boolean model are the clean formalism and simplicity. The main disadvantages are: binary decision criterion without any notion of a grading scale; difficulty of translating the query into Boolean expressions. [2]

1.2.2 Vector Model

In the Vector model, document d_j and query q are represented as t-dimensional vectors, where t is the total number of index terms in the system. The vector model allows partial matching. This is accomplished by assigning non-binary weights to index terms in queries and in documents. These term weights are used to compute the degree of similarity between the document and the query. The vector model evaluates the degree of similarity of the document d_j with regard to the query q as the correlation between the vectors d_j and q. This correlation can be quantified by the cosine of the angle between these two vectors:

$$sim(d_j, q) = \frac{d_j q}{|d_j| \times |q|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2 \times \sum_{i=j}^{t} w_{i,q}^2}}$$

The tf.idf term-weighting scheme uses weights which are given by

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$$

where

f – term frequency (tf) indicates how often a term t occurs in document d; usually tf is normalized $tf_{t,d} = \frac{freq_{t,d}}{\max_l \; freq_{l,d}}$

Inverse document frequency (idf) – indicates how rare occurrence of a term is in document corpus; usually idf is normalized: $idf_t = \log \frac{N}{n_t}$; where N – total number of documents in the systems, and n_t - number of documents in which term t appears.

1.2.3 Fuzzy Set Model

The Fuzzy Set model is a model of information retrieval that uses fuzzy logic as its foundation. Fuzzy Logic is used to define fuzzy relationships between query terms and documents. Each query term defines a fuzzy set and each document has a degree of membership in corresponding set. The Fuzzy Set model performs query expansion based on principles of fuzzy logic. A thesaurus is constructed by defining a term-term correlation matrix. The correlation matrix is used to define a fuzzy set associated to each index term k_i. Document d_j has a degree of membership $\mu_{i,j} = 1 - \prod_{k_l \in d_j} (1 - c_{i,l})$. The procedure to compute the document's relevance given a query is analogous to the procedure used by the Boolean model, except rules of fuzzy logic are used [9]. The fuzzy set model approach is not popular among the information retrieval community and has been discussed mainly in the literature dedicated to fuzzy theory [2].

Recent attempts utilizing fuzzy search were tried at TREC 2001 with the search engine NexTrieve. NexTrieve used a combination of the exact search and fuzzy search. The conference paper that describes NexTrieve [21] unfortunately does not provide details on the theoretical foundation and implementation of the system. It appears that application of the fuzzy logic was to the position, and to the scoring: terms in different parts of the document would get different scores and not all of the words would need to be present in order for the document to get a high score. According to the authors of NexTrieve, one of the biggest drawbacks of the system was that it did not take into account word frequency within a document and document length which has been shown to be a crucial part of the ranking score. Performance of NexTrieve system was substandard with average precision of 0.13; and after some additional modifications (adding word frequency, and document length parameters) were made, it went up to 0.19; which was still substandard [21].

1.2.4 Current Approaches – TREC 2004 Robust Retrieval Track

This section provides a brief overview of the approaches of some of the systems of TREC 2004 Robust Retrieval Track.

The Queens College, CUNY system (PRIC) has achieved the best evaluation results. PIRC used a variation of Query Expansion (QE) by utilizing the web as an external thesaurus to supplement a given topic description [7].

The Fondazione Ugo Bordoni system used Divergence From Randomness (DFR) modular probabilistic framework together with a parameter-free version of Rocchio's Query Expansion [1].

The University of Illinois at Chicago system used a variation of Query Expansion based on phrase recognition and classification. Phrases in the query are identified and classified into 4 types: proper names, dictionary phrases, simple phrases and complex phrases [8].

The University of Glasgow system (Terrier) is based on the Divergence From Randomness framework and utilizes two pre-retrieval performance predictors: a weighting function recommender (WFR) mechanism, and automatic tuning of the term frequency normalization parameters. For the performance predictors, the following features were used: average inverse collection term frequency, and standard deviation of idf. The goal of WFR is to cope with the poorly-performing queries by recommending the optimal weighting functions, including document weighting and term weighting (query expansion) functions, from a set of candidate weighting functions on a per-query basis [10].

The University of Michigan and Virginia Tech system utilized Genetic Programming (GP), Query Expansion and features of Okapi BM25. This approach is somewhat similar to the ANFIS R-FIS approach, since in both cases machine learning techniques are used to create a ranking function [16].

For recent approaches (TREC 2001) that utilize fuzzy logic see 1.2.3.

1.3 Motivation for using Fuzzy Logic in construction of Ranking Function of IR system

The Quality of the ranking function is an important factor that determines the quality of the Information Retrieval (IR) system. Constructing and tuning the ranking function is typically a complicated process. With the use of Fuzzy Logic, it is possible to simplify this process.

1.3.1 Theoretical fit between Fuzzy Logic Model and Information Retrieval Model

The Information Retrieval system retrieves documents based on a given query. Both the documents and in most cases, the queries, are instances of natural language. Natural langue is often vague and uncertain [12]. It is difficult to judge something that is vague and uncertain with deterministic crisp formulas and/or crisp logical rules. Fuzzy logic is based on the theory of fuzzy sets, a theory which relates to classes of objects with unsharp boundaries in which membership is a matter of degree [17]. Documents, queries and their characteristics could easily be viewed as fuzzy granular classes of objects with unsharp boundaries and fuzzy memberships in many concept areas [18].

Fuzzy logic is a logical system, which is an extension of multivalued logic [17]. Use of fuzzy logic provides the benefits of the Boolean method while overcoming its drawbacks. Since the concept of fuzzy logic is quite intuitive, the fuzzy logic model provides a framework that is easy to understand for a common user of IR system. Documents retrieved by a query are evaluated by the rules of the FIS that have precise semantics. Unlike the Boolean model that is based on binary decision criterion {relevant, not relevant}, fuzzy logic expresses relevance as degrees of memberships (e.g., document | query could have a relevance measure with the following degrees of membership: 0.7 highly relevant and 0.5 moderately relevant and 0.1 not relevant). Fuzzy logic is tolerant of imprecise data. [2]

1.3.2 Rules Creation / Expert Knowledge Transfer Process

Since IR deals with natural language, many of the rules that are used to determine relevance of documents come from experts and from experience. Before the rules are converted to formulas, they are often communicated as observations in natural language (e.g., If most of the terms of the query are present in the document, then the document is likely to be relevant; if a term of a query occurs in a document often, that will increase the likelihood of the document being relevant, etc.). Fuzzy logic allows incorporating rules into the system in a natural way. The basic concept underlying FL is that of a linguistic variable, a variable whose values are words rather than numbers. FL may be viewed as a methodology for computing with words rather than numbers. Even though words are inherently less precise than numbers their use is closer to human intuition. Computing with words allows the tolerance for imprecision. [17]

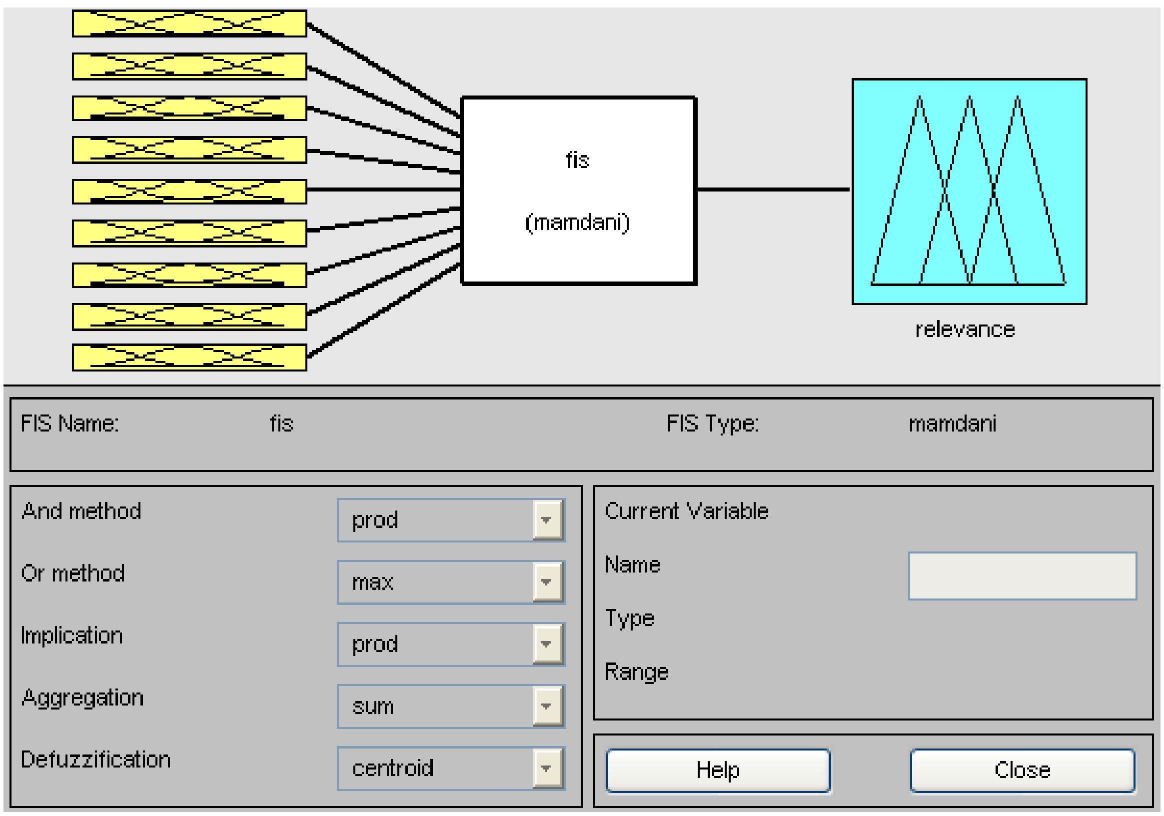
1.3.3 Ease of Training / Tuning

Various graphical user interfaces such as Matlab Fuzzy Logic Toolbox provide a convenient way to view all the components of an FIS; to modify them easily; and to examine and verify the effects of changes. Parameters of FIS could be modified systematically by utilizing various optimization approaches. The fuzzy logic approach is a very flexible one. It is possible to make small improvements without disturbing the integrity of the system. This could be done though changing parameters of the parts of the systems such as rules and membership functions. If more granularity is desired more rules and/or membership functions could be added; on the other hand if more generality is required, rules could be combined and unimportant parts could be discarded.

2 Background Information

2.1 Fuzzy Inference System

2.1.1 Knowledge Based R-FIS Construction



R-FIS is constructed with the use of Matlab (version 7) Fuzzy Logic Toolbox. The first step in construction of an FIS is to define rules. Rules will be derived from somewhat common knowledge about information retrieval and of the tf.idf method. The order of the rules in FL does not affect the output.

- If many of the terms of the query are found in the document (coord), the document is likely to be highly relevant.

- If not many of the terms of the query are found in the document (coord), the document is not likely to be highly relevant.

For each of the query terms the following rules could be constructed:

- If a query term in a document has high tf and idf measures, the document is likely to be highly relevant.

- If a query term in a document does not have high tf and idf measures, the document is not likely to be highly relevant.

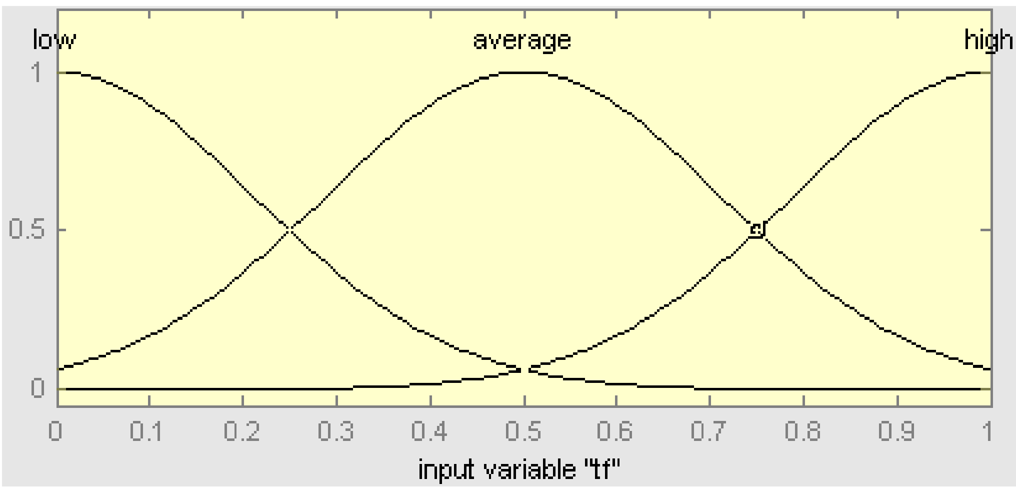


Figure 1

It is necessary to give mathematical meaning to the linguistic variables mentioned in above rules: high relevance, not high relevance, many terms, not many terms, high tf, not high tf, high idf, not high idf. All of the inputs are normalized so their values $\in [0, 1]$. The input and output range would then be $[0, 1]$. It is necessary to define fuzzy sets. All input variables and output variables currently have two fuzzy sets associated with each variable: high, not high. If greater granularity is desired, more fuzzy sets could be defined such as for example: very low, low, average, above average, very high, etc as in Figure 1. A membership function (mf) is a curve that defines how each point in the input space is mapped to a degree of membership of fuzzy set. There are various membership function types as shown in table bellow.

Function Type	Description
dsigmf	Difference of two sigmoid membership functions.
gauss2mf	Two-sided Gaussian curve membership function.
gaussmf	Gaussian curve membership function.
gbellmf	Generalized bell curve membership function.
pimf	Pi-shaped curve membership function.
psigmf	Product of two sigmoidal membership functions.
smf	S-shaped curve membership function.
sigmf	Sigmoid curve membership function.
trapmf	Trapezoidal membership function.
trimf	Triangular membership function.
zmf	Z-shaped curve membership function.

Table 1

Membership functions for the R-FIS in this case are defined as follows:

For each term the following membership functions are defined for the corresponding values:

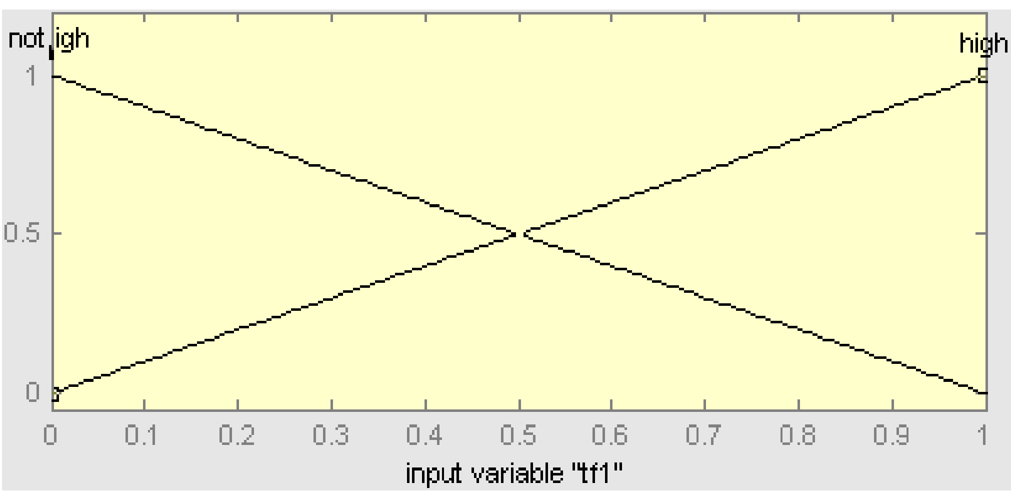


Figure 2 - Membership Functions (tf)

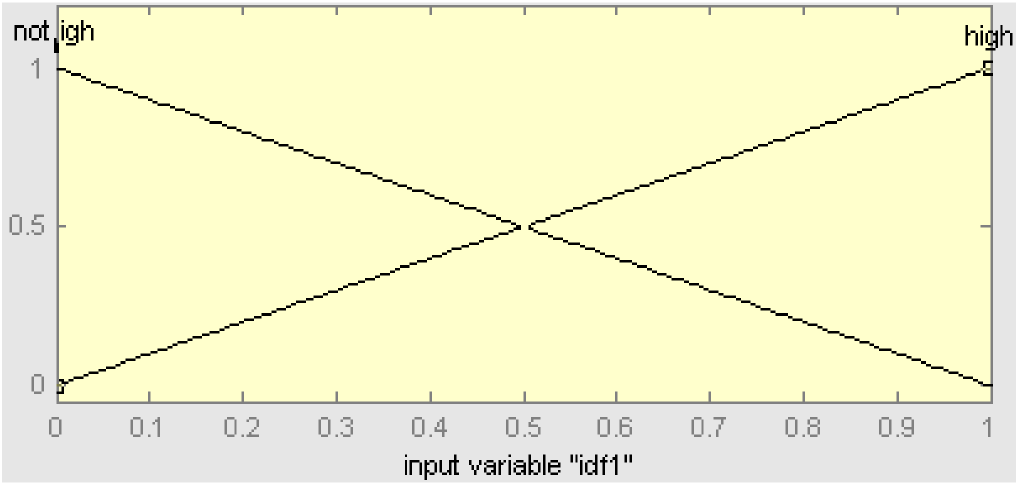


Figure 3 - Membership Functions (idf)

For each document, the following membership functions are defined for the corresponding value:

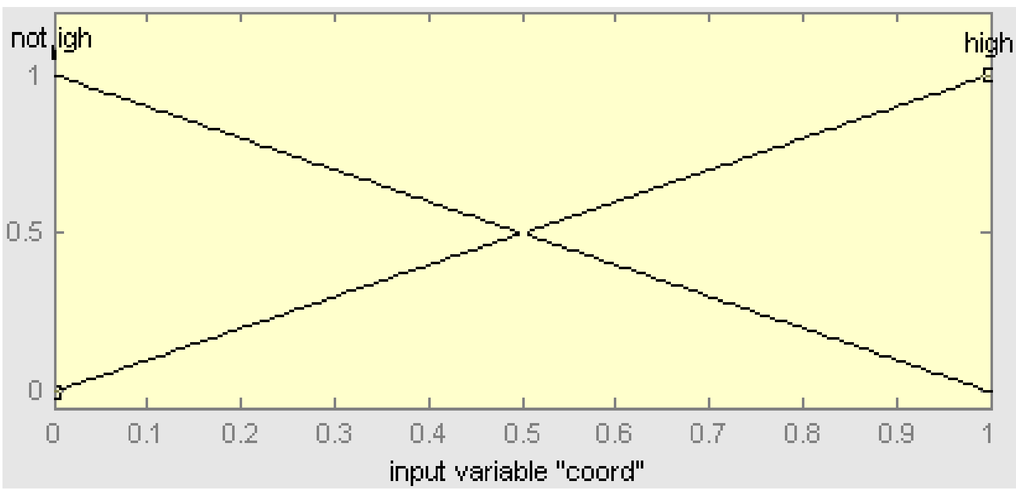


Figure 4 - Membership Functions (coord)

For output-relevance, the membership function is defined as follows:

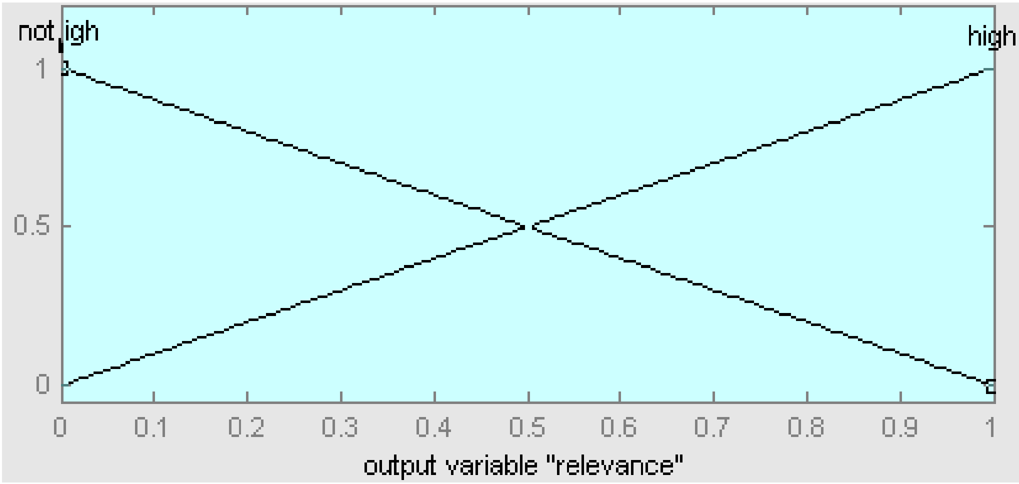


Figure 5 - Membership Functions (relevance)

After membership functions have been defined, it is possible to formally define the rules of the R-FIS.

1. If (tf1 is high) and (idf1 is high) then (relevance is high) (0.33)
2. If (tf2 is high) and (idf2 is high) then (relevance is high) (0.33)
3. If (tf3 is high) and (idf3 is high) then (relevance is high) (0.33)
4. If (tf1 is not high) and (idf1 is not high) then (relevance is not high) (0.33)
5. If (tf2 is not high) and (idf2 is not high) then (relevance is not high) (0.33)
6. If (tf3 is not high) and (idf3 is not high) then (relevance is not high) (0.33)
7. If (tf4 is high) and (idf4 is high) then (relevance is high) (0.33)
8. If (tf4 is not high) and (idf4 is not high) then (relevance is not high) (0.33)
9. If (coord is high) then (relevance is high) (0.05)
10. If (coord is not high) then (relevance is not high) (0.05)

Figure 6 –Rules of R-FIS

Rules that are considered to be more important could be given larger weight as in case of rules 1 – 8 with weight of 0.33; if a rule is not as important, it could be given smaller weight as in case of rules 9-10 with weight of 0.05. Logical operators are defined as follows:

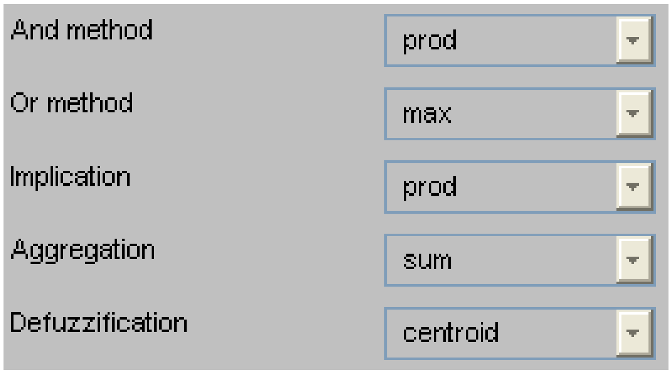


Figure 7 - Logical Operators

This concludes construction of the R-FIS.

2.1.2 Fuzzy Inference Process

The Fuzzy Inference Process is performed automatically, but in order to explain how the system functions, each of the steps will be examined bellow.

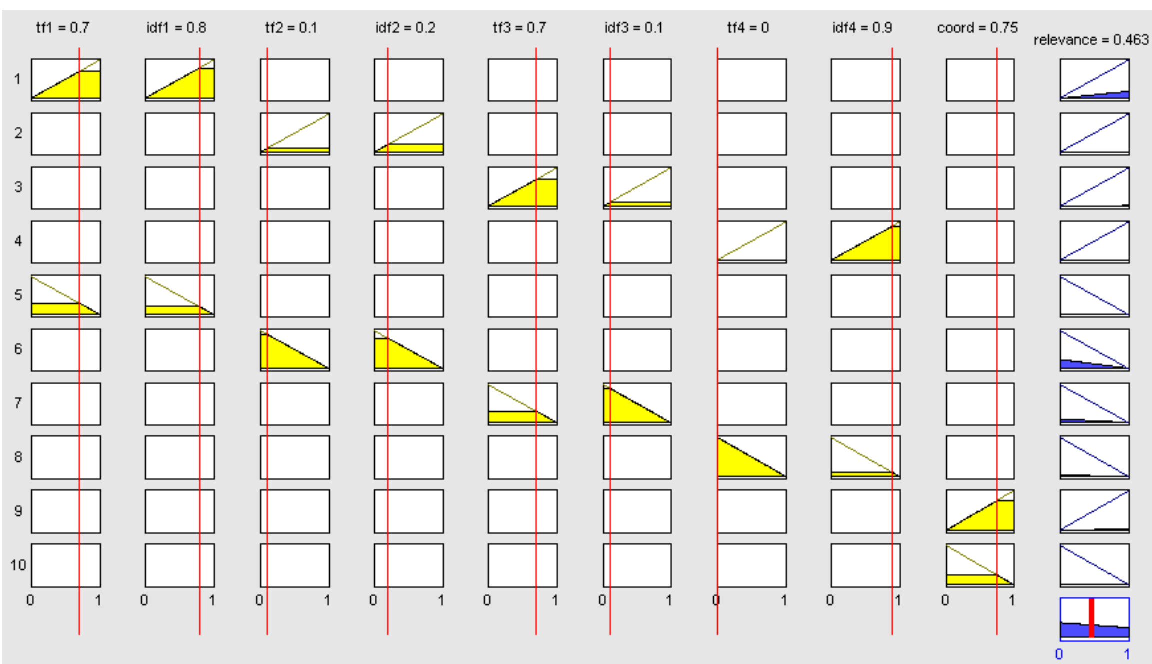


Figure 8 - Fuzzy Inference Process

2.1.2.1 Fuzzify Inputs

The first step is to take the crisp numerical values of the inputs and determine the degree to which they belong to each of the appropriate fuzzy sets via membership functions [4]. For example: tf1 = 0.7, this would translate into 0.7 degree of membership in fuzzy set “high” and 0.3 degree of membership in fuzzy set “not high”. Same procedure would be applied to all of the inputs.

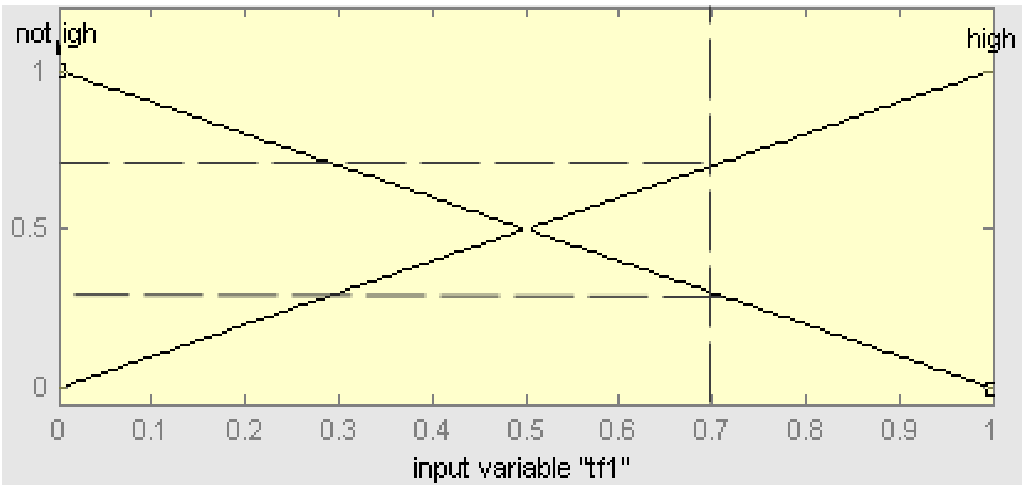


Figure 9 - Fuzzify Inputs

2.1.2.2 Apply Fuzzy Operator

Once the inputs have been fuzzified, the degree to which each part of the antecedent has been satisfied for each rule is known. If the antecedent of a given rule has more than one part, the fuzzy operator is applied to obtain a number that represents the result of the antecedent for that rule [4]. Let’s examine rule 1: if (tf1 is high) and (idf1 is high) then (relevance is high). In this case the input to the fuzzy operator is two membership values from fuzzified input variables: tf1 has a 0.7 degree of membership in fuzzy set “high” and idf1 has a 0.8 degree of membership in fuzzy set “high”. “prod” was selected as a fuzzy operator for “and method” so the result is 0.7 * 0.8 = 0.56. This procedure is applied to every rule.

2.1.2.3 Apply Implication Method

The input for the implication process is a single number given by the antecedent, and the output is a fuzzy set. A consequent of the implication method is a fuzzy set represented by a membership function, which weights appropriately the linguistic characteristics that are attributed to it. The consequent is reshaped using a function associated with the antecedent (a single number) and weight of the rule. Implication is applied to every rule [4]. For example applying the implication method to rule1 this would result in the following:

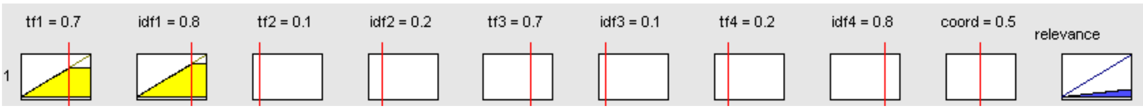


Figure 10 - Implication

2.1.2.4 Aggregate all outputs

Since decisions are based on all of the rules in an FIS, the rules must be combined in order to make a decision. Aggregation is the process by which the fuzzy sets that represent the outputs of each rule are combined into a single fuzzy set. The input of the aggregation process is the list of fuzzy sets that represent the outputs of each rule. The output of the aggregation process is a fuzzy set. [4]

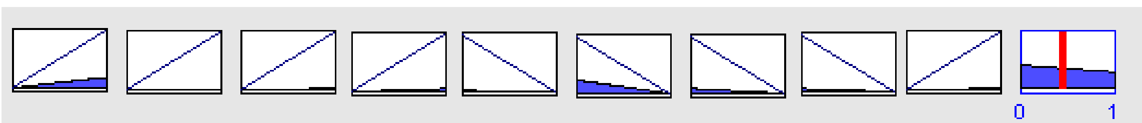


Figure 11 - Aggregation

2.1.2.5 Defuzzify

The input for the defuzzification process is the aggregate output fuzzy set and the output is a single number. Fuzziness helps the rule evaluation during the intermediate steps; however the final desired output for each variable is generally a single number. Fuzzy set must be defuzzified in order to resolve a single output value from the set. There are various methods for defuzzification such as: centroid, bisector, middle of maximum (the average of the maximum value of the output set), largest of maximum, and smallest of maximum. In this case the centroid method is used. The centroid method returns the center of the area under the curve. In this case it is 0.469.

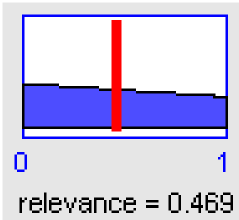


Figure 12 - Defuzzification

2.2 ANFIS-R-FIS Construction

This approach utilizes the ANFIS method to derive the initial rules of the system and to tune its parameters. Traditionally, FIS systems have been constructed by manually mapping input characteristics to input membership functions, input membership function to rules, rules to a set of output characteristics, output characteristics to output membership functions, and the output membership function to a single-valued output or a decision associated with the output. Constructing an FIS this way implies that membership functions are fixed, and somewhat arbitrarily chosen, and rule structure is essentially predetermined by the user's interpretation of the characteristics of the variables in the model. There are modeling situations in which it is not possible to look at the data and discern what the membership functions should look like. Instead of just looking at the data to choose the parameters associated with a given membership function somewhat arbitrarily, these parameters could be automatically chosen by ANFIS so as to tailor the membership functions to the input/output data in order to account for these types of variations in the data values. Neuro-adaptive learning techniques provide a method for the fuzzy modeling procedure to 'learn' information about a data set, in order to compute the membership function parameters that best allows the associated fuzzy inference system to track the given input/output data. FIS is a network-type structure similar to that of a neural network, which maps inputs through input membership functions and associated parameters, and then through output membership functions and associated parameters to outputs, can be used to interpret the input/output map. The adjustment of parameters associated with the membership functions is facilitated by a gradient vector, which provides a measure of how well the fuzzy inference system is modeling the input/output data for a given set of parameters. Once the gradient vector is obtained, any of several optimization routines could be applied in order to adjust the parameters so as to reduce some error measure. ANFIS uses either backpropagation or a combination of least squares estimation and backpropagation (hybrid) method for membership function parameter estimation [4].

ANFIS places certain constraints on a type of FIS system that could be constructed:

- Be first or zeroth order Sugeno-type systems.

- Have a single output, obtained using weighted average defuzzification. All output membership functions must be the same type and either be linear or constant.

- Have no rule sharing. Different rules cannot share the same output membership function, namely the number of output membership functions must be equal to the number of rules.

- Have unity weight for each rule.

2.2.1 Sugeno Fuzzy Inference Method

The Sugeno fuzzy inference method [13] is similar to the Mamdani method in many respects. The main difference between the Mamdani and Sugeno methods is that the Sugeno output membership functions are either linear or constant. A typical rule in a Sugeno model has the form: If Input1 = x and Input2 = y, then Output is z = ax + by + c. For a zero-order Sugeno model, the output level z is a constant (a=b =0). A sugeno model operates as shown in Figure 13.

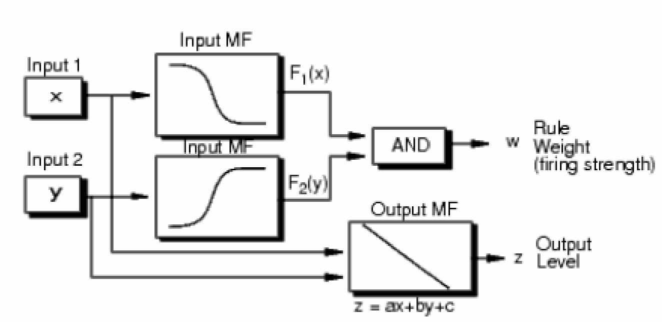


Figure 13 – Sugeno Fuzzy Inference Method [4]

Advantages of the Sugeno Method [4]

- Computationally efficient.
- Works well with linear techniques.
- Works well with optimization and adaptive techniques.
- Guaranteed continuity of the output surface.
- Well-suited to mathematical analysis.
- Models nonlinear systems by interpolating between multiple linear models.

3 Ranking Fuzzy Inference System

3.1.1 Input Variables

The R-FIS could be based on various retrieval models that have defined rules and provide access to underlying features. The Vector model was chosen as the baseline model. The Vector model is a commonly used model. Another reason that this model was chosen as the baseline model is the availability of open source search engines such as Apache Lucene. The use of Lucene allows for a convenient way of extracting of the tf.idf factors for the use in the R-FIS.

The R-FIS input variables are typical variables that are used in tf.idf based systems. It is beneficial to use variables that have been established to be significant in determination of document's relevance. Without them, the results would suffer [3].

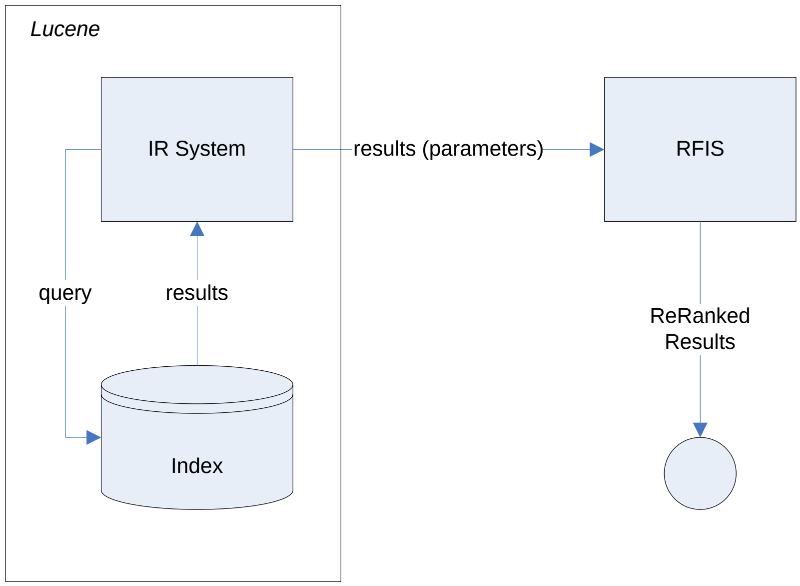
For each term of the query, the following parameters are used: term frequency (tf), inverse document frequency (idf).

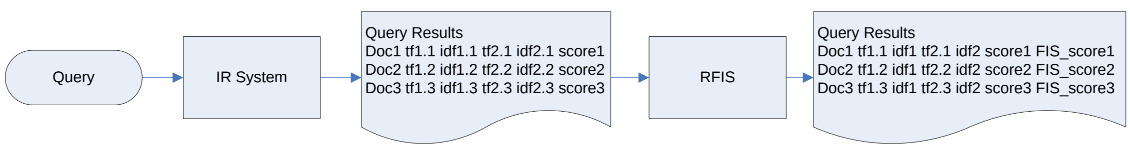
tf – indicates how often a term t occurs in document d; usually tf is normalized:

$$tf_{t,d} = \frac{freq_{t,d}}{\max_l \ freq_{l,d}}$$

idf – indicates how rare the occurrence of a term is in a document corpus; usually idf is normalized: $idf_t = \log \frac{N}{n_t}$; where N – total number of documents in the system, and n_t - number of documents in which term t appears.

For each document a coord value is provided. Coord – score factor based on term overlap with the query: $coord = \frac{overlap}{\max overlap}$; where overlap - the number of query terms matched in the document; max overlap - the total number of terms in the query [6].





3.1.2 Ranking Function

The formula of a typical tf.idf based system is constructed by experts. For example, Lucene's ranking function is as following:

$$score(q,d) = \sum_{t \in q} \text{tf(t in d) * idf(t) * getBoost(t.field in d) * lengthNorm(t.field in d) *}$$

coord(q,d) * queryNorm(q)

Where:

getBoost(t.field in d) – the boost factor for hits on any field of this document

lengthNorm(t.field in d) – the normalization value for a field given the total number of terms contained in a field

queryNorm(q) – the normalization value for a query given the sum of the squared weights of each of the query terms [6].

3.1.2.1 ANFIS-R-FIS

An ANFIS-R-FIS ranking function is constructed automatically with the use of Adaptive Neuro-Fuzzy Inference System that uses a combination of backpropagation and least square estimation in order to tune the parameters of the ranking function. For more detailed description see 2.2.

3.1.2.2 Knowledge Based R-FIS (KB-R-FIS)

A Knowledge Based R-FIS ranking function is constructed by transferring knowledge about tf.idf based systems into fuzzy logic rules and corresponding membership functions. For a more detailed description please see 2.1.

3.1.2.3 KB-R-FIS ANFIS

In this approach, parameters of the model constructed with Knowledge Based R-FIS approach are tuned with ANFIS. For a more detailed description please see 2.1, 2.2.

4 Configuration of Experiments

4.1 System Configuration

Search Engine: Lucene version 1.4.final from Apache Jakarta.

Indexing:
Analyzer: org.apache.lucene.analysis.standard.StandardAnalyzer

All fields are indexed as plain text; there is no special treatment of any fields including <TITLE> field.

4.2 Evaluation

4.2.1 Document Corpus

To evaluate the effectiveness of R-FIS, data from the NIST TREC 2004 Robust Retrieval Track was used. The robust retrieval track explores methods for improving the consistency of retrieval technology by focusing on poorly performing topics. Another option was to evaluate performance on the data from the TREC Ad Hoc Track. Since the Ad Hoc Track has been discontinued, it was decided to use the robust track's data. The document collection for the Robust Track is the set of documents on both TREC Disks 4 and 5, minus the Congressional Record on disk 4. [14]

Source	# Docs	Size (MB)
Financial Times	210,158	564
Federal Register 94	55,630	395
FBIS, disk 5	130,471	470
LA Times	131,896	475
Total Collection:	528,155	1904

Table 2 [15]

The Robust test set contains 250 topics: topics 301-450 (ad hoc topics from TREC 6-8), topics 601-650 (new topics for 2003 robust track), and topics 651-700 (new topics for 2004 robust track) [15].

Topic Set	Number of topics	Mean Relevant per topic	Minimum # Relevant	Maximum # Relevant
Old	200	76.8	3	448
New	49	42.1	3	161
Hard	50	88.3	5	361
Combined	249	69.9	3	448

Table 3 - Relevant document statistics for topic sets [14]

The purpose of this project is to achieve improvements over baseline model. For this reason, odd queries were used for training and even queries were used for evaluation. If the purpose were to achieve good results on Robust Track, an "Old" topic set would have been used for training and evaluation during the development stage. Unfortunately, these two goals are not compatible with each other; since the new topic set may not contain enough queries to verify improvements of the system over baseline model.

Note: Participants in the TREC Robust Track did not have the access to the "New" set of topics. In the ANFIS case, the system was trained on the odd queries and evaluated on the even. However, the R-FIS appears to be consistent across topics sets; systems that performed best on the "Old" topic set, also performed best on the "New" topic set; so it is still likely to be a good indicator of the performance of the system.

4.2.2 Procedure

In order to evaluate the performance of the R-FIS system as an instance of a tf.idf model, it was compared to the performance of the baseline tf.idf system Apache Lucene.

In order to evaluate performance of R-FIS and Lucene as tf.idf based systems in comparison with other approaches and extensions, results are compared with the systems of the participants in TREC 2004 Robust Retrieval Track.

The following programs have been used for evaluation of results:

trec_eval version 7.0beta - evaluates an ad hoc retrieval run, given the results file and a standard set of judged results [14].

robust2004_eval.pl – script to compute TREC 2004 Robust Track evaluation measures: MAP and mean P(10) over different topic sets plus area under the curve and number topics with no relevant for those topic sets for a run using the TREC 2004 robust track topics [14].

5 Analysis and Experiments

5.1 ANFIS-R-FIS

5.1.1 Purpose

Construct an R-FIS utilizing the ANFIS method. Use training data to automatically choose the parameters associated with a given membership function with the use of ANFIS so as to tailor the membership functions to the input/output data in order to account for variations in the data values. Neuro-adaptive learning techniques provide a method for the fuzzy modeling procedure to 'learn' information about a data set, in order to compute the membership function parameters that best allow the associated fuzzy inference system to track the given input/output data.

5.1.2 Training Data & Refinement Set Selection

Training Set – is the set on which the ANFIS is trained. Refinement Set – is the set on which the R-FIS is executed (top documents from the result set produced by the baseline model from the queries of test set). Selection of the refinement set is dependent on the training set. That is why experiments exploring these aspects are combined. Experiments 2_2 – 6_4 explore these aspects. Both training and refinement sets are obtained by selecting top documents for each query produced by Lucene.

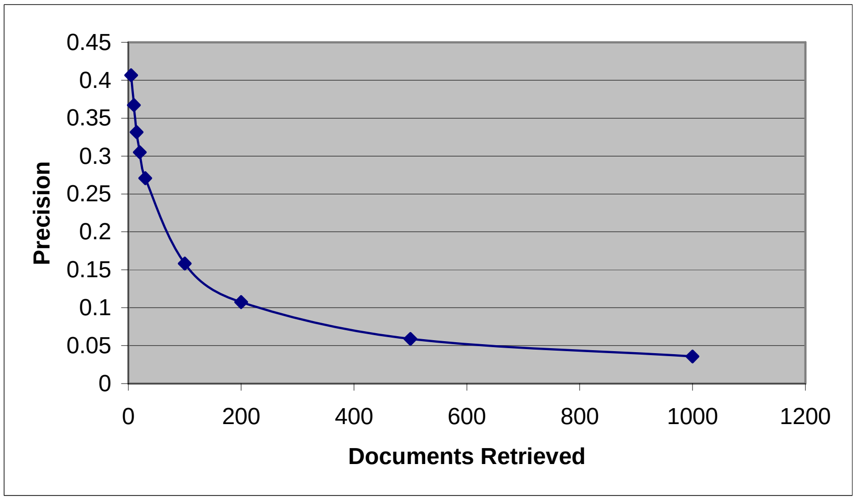


Figure 14 – Lucene's title run

Selecting a good training set is crucial. Figure 14 shows a typical distribution of relevant documents in relation to documents retrieved. ANFIS calculates error as the sum of the squared difference between actual and desired outputs. Training the FIS this way would create an FIS that is optimized for Average Precision; however the measures that need to be optimized are P10 and MAP.

Run Tag	Training Data Size	Refinement Set Size	P10
base_line	*NA*	*NA*	0.3776
2_1	10	100	-0.2722
2_2	10	10	-0.0033
2_3	10	20	-0.1033
3_1	20	1000	-0.2749
3_2	20	10	-0.0033
3_3	20	20	-0.0506
3_4	20	40	-0.1195
4_1	40	1000	-0.3282
4_2	40	10	-0.0033
4_3	40	20	-0.0506
4_4	40	30	-0.0654
4_5	40	40	-0.0857
6_1	100	10	-0.0033
6_2	100	20	-0.0438
6_3	100	40	-0.0817
6_4	100	100	-0.1181

Table 4

Note: Experiments with the 'refinement set size' = 10 could be ignored, since the measure of success is P10 and with the 'refinement set size' = 10 this measure will not change much since top 10 documents will be the same and only their order may change. In order to see the effects of various configurations, measures such as P5 would be needed, but is not easily available.

Training set size does not appear to be an important factor that affects P10. It appears that a bigger training size performs slightly better. This is likely related to the structure of the FIS, or more precisely, the number of parameters that need to be turned; so the larger the set, the better the parameters could be tuned. Effects of the FIS structure on efficiency of the ANFIS are explored in 5.1.3. There are no experiments with "training data size" > 100, since that resulted in a running time of over 12 hours; and is not likely to cause significant improvement since if the training sample size is too large, the system will tend to converge towards marking all of the documents as non relevant since that would minimize error.

It appears that performance of the FIS is better on smaller refinement sets. This is likely caused by the fact that the baseline system from which the refinement set is obtained is better than the FIS; so the FIS is not able to improve the set by refining it. However performance of the FIS is somewhat decent since P10 = 0.2595 for FIS 100/100; so this shows that the FIS can refine the results, but not as good as the baseline system.

5.1.2.1 Hard Topic Set

Run Tag	Training Data Size	Refinement Set Size	Hard P10
base_line	*NA*	*NA*	0.2417
4_4	40	30	0.0166
4_3	40	20	-0.0084
3_3	20	20	-0.0167
6_2	100	20	-0.0167
4_5	40	40	-0.025
6_3	100	40	-0.025
6_4	100	100	-0.075
3_4	20	40	-0.0834
2_3	10	20	-0.1084
2_1	10	100	-0.1167
3_1	20	1000	-0.1667
4_1	40	1000	-0.2097

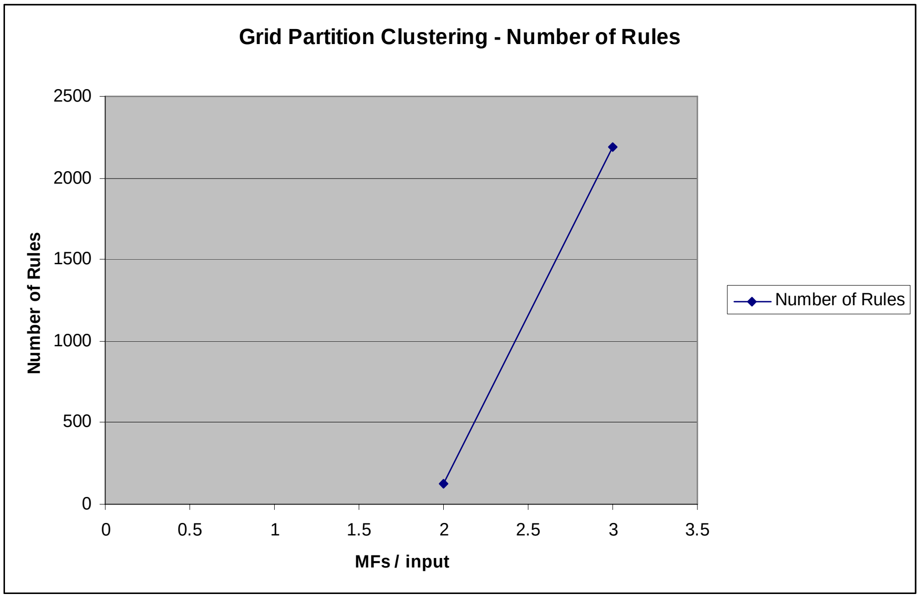
The ANFIS R-FIS has one run (4_4) that is better than the baseline run on the "Hard" topic set and several runs that are slightly worse. In all cases, the training data size is bigger than or equal to the refinement set size. This is somewhat expected, since if the training set size is bigger, the system will be trained on a bigger variety of situations; this way it is unlikely to encounter something unfamiliar during the refinement process. Better performance on hard topics is probably due to the nature of the ANFIS, since it allows for certain parameters to be fine tuned. Unfortunately, it is not easy to examine the FIS system and determine the reason (for the analysis of the FIS see 5.1.4).

5.1.3 FIS Structure Generation

Tag	**Training Set Size**	**Refinement Set Size**	**MFs/Input**	**FIS Generation**	**P10**
baseline	*NA*	*NA*	*NA*	*NA*	*0.3784*
ex7_1	40	20	3	Grid Partition Clustering	-0.0487
ex7_2	40	20	6	Subtractive Clustering	-0.0014
ex7_2_1	20	20	6	Subtractive Clustering	-0.023
ex7_3	40	20	2	Grid Partition Clustering	-0.0487

FIS Generation	**Number of Inputs**	**Number of Rules**	**MFs / Input**	**MFs / Output**
Grid Partition	7	2187	3	2187
Grid Partition	7	128	2	128
Subtractive Clustering	7	6	6	6

5.1.3.1 Grid Partition



Grid Partition (Fuzzy C-Means Clustering) appears to find many clusters. This provides for better granularity but not better performance. Growth of rules generated in relation to MFs/input and to the number of inputs is exponential. In this case, it appears that performance of the systems with fewer MFs/input could be improved to the level of a system with more MFs/input by increasing the number of training epochs; however this result is not expected to hold for other cases as well. Performance growth appears to be linear in relation to MFs/input. This is a bad situation; since a linear improvement in performance of the system could only be achieved through an exponential increase in its complexity. Systems generated by Grid Partition Clustering are very complex; and lose advantages that fuzzy logic provides, such as simplicity and being intuitive. Training of the FIS with so many parameters requires substantial computational resources; epoch for the system with 7 inputs, 3MFs/Input, 2187 rules takes ~3 hours on a 1.8Ghz machine. Requirements for the training data size are proportional to the complexity of the FIS; so in this case, a large training set is required.

5.1.3.2 Subtractive Clustering

Subtractive clustering appears to be a much better suited method for FIS generation in this case. The FIS structure produced by subtractive clustering is much more compact, while the performance is better than that of systems generated with a Grid Partition clustering method. Training of the system with ANFIS is directly proportional to the compactness of the FIS, so in this case training is fast and efficient. There are a number of parameters that are used in subtractive clustering method such as: range of influence, squash factor, accept ratio, reject ratio. Finding optimal parameters could produce an FIS that has even better performance.

5.1.4 FIS Structure Analysis

5.1.4.1 Grid Partitioning

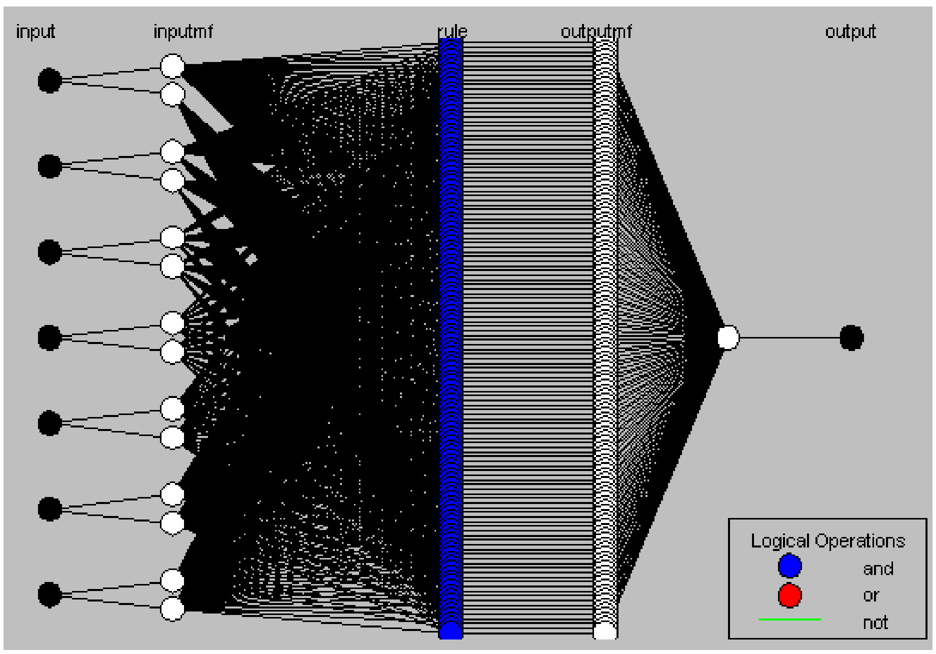


Figure 15 – 2 – 2mf / input

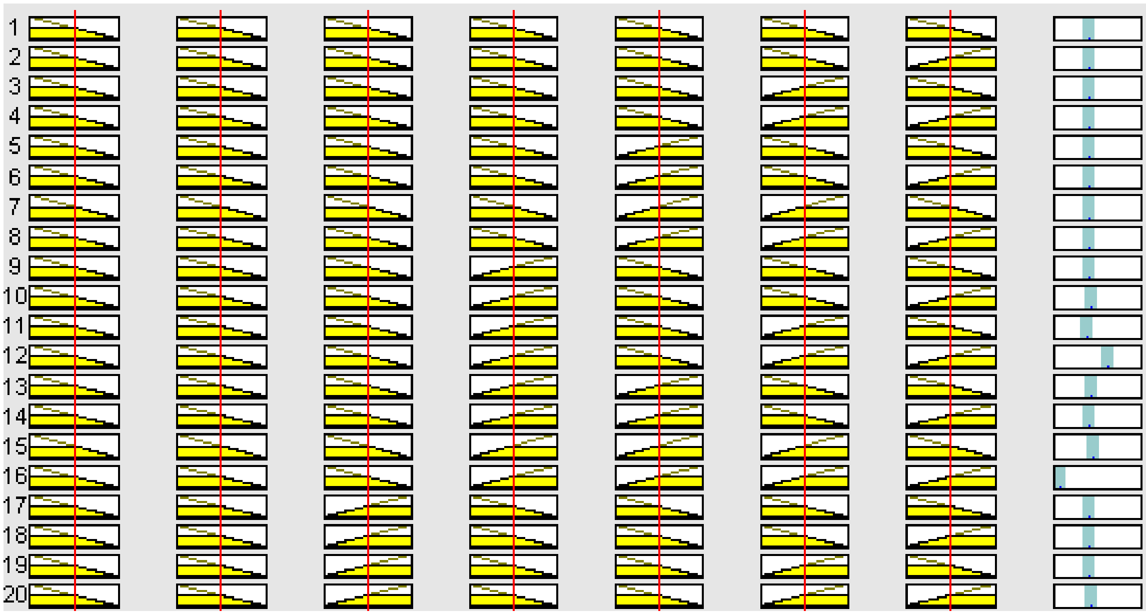


Figure 16 – first 20 out of 2187 rules

As discussed in 5.1.3.1, the structure of the FIS generated with Grid Partitioning is much more complex. It is very difficult to analyze it manually.

5.1.4.2 Subtractive Clustering

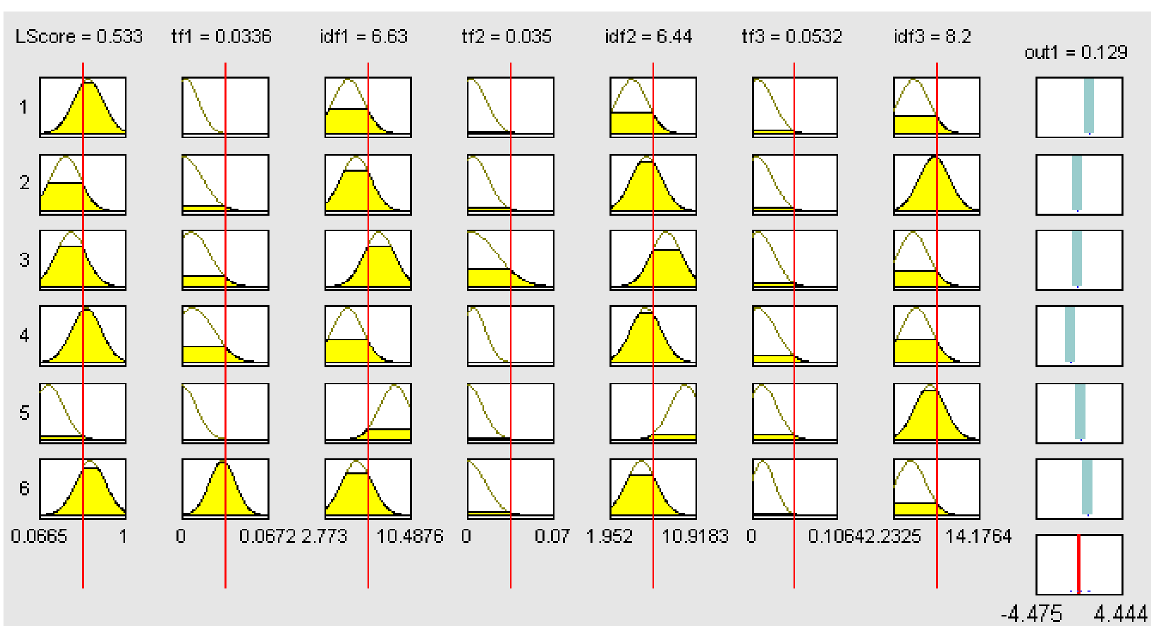


Figure 17 - Rule View

```
[Rules]
1 1 1 1 1 1 1, 1 (1) : 1
2 2 2 2 2 2 2, 2 (1) : 1
3 3 3 3 3 3 3, 3 (1) : 1
4 4 4 4 4 4 4, 4 (1) : 1
5 5 5 5 5 5 5, 5 (1) : 1
6 6 6 6 6 6 6, 6 (1) : 1
```

From examining the structure of the FIS it appears that there is no membership function sharing between the rules.

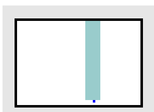


Figure 18 - Output Function

The effect of the rule could be determined by examining output function. If it is situated more on the right side, then this rule will affect ranking positively, and the reverse is true. Analysis of the rules is somewhat complicated by the presence of all of inputs in the rules.

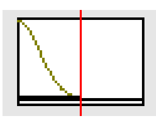


Figure 19 - MF of Input in a Rule

It is possible to analyze the role of an input in a rule by looking at the membership function associated with a given input. For example, the membership function in Figure 19 indicates that the negative influence of the variable decreases as the value of the variable increases.

The rules in Figure 17 are not quite intuitive. One probable reason is that the inputs are not normalized, so the membership functions appear to be somewhat dislocated.

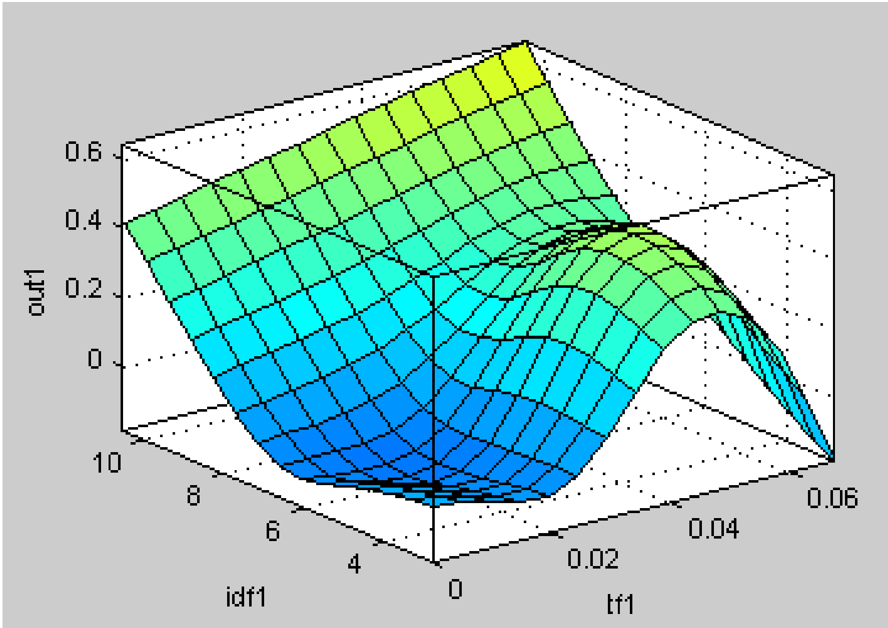


Figure 20 - Rule Surface

The rule surface learned for the most part appears to be on the right track, with the exception of the high tf and low idf values. This could have been caused by an absence of representative training data points in that region.

5.1.5 Extending R-FIS based on variable number of query terms

So far, the ANFIS R-FIS was applied to a fixed number of query terms - 3. Extending ANFIS R-FIS to handle variable number of terms is not straight forward. One of the reasons that extending it manually is difficult is because it is not possible to determine relations between different variables. Complexity of the FIS is linearly dependent on the number of terms in the Subtractive Clustering method; however with the Grid Partition method, the complexity of the FIS is exponential in relation to the number of terms.

One way to extend R-FIS would be by creating an FIS and deciding on a maximum number of terms. The features of the terms that are not present will be 0.

Another way would be to create a separate FIS for each term number instances, and then train them on corresponding training sets. This approach is not quite feasible since it is not easy to obtain sufficient training sets for each instance.

Another approach would be to break each of the queries up in equal number of terms, for example 3, and then add up the results of each of the sub queries. A second FIS could be constructed that takes as input the combined score plus some other cumulative features and outputs a relevance score.

5.1.6 Conclusion

Tag	P10
Baseline	*0.3784*
ex7_2	-0.0014

Table 5

In the beginning of the test runs, a not a very successful branch was explored – FIS Generated with Grid Partition Clustering. It was possible to reach results close in performance to that of baseline system. However, many problems were discovered such as inability to scale, and an overly complex FIS model.

After analysis, it appears that a FIS generated with Subtractive Clustering is a better model with better performance and much more compact representation. Analysis of the FIS rules was possible, but did not quite produce the results that were expected. That was likely caused by the lack of normalized inputs, and limited experimentation with the parameters of subtractive clustering algorithms and application of ANFIS to this model. In order to be able to better understand and extract knowledge from the resulting structure of the R-FIS, various approaches of analysis would need to be explored. Further work in this direction is likely to produce improvements.

One of the problems encountered with the ANFIS approach was the difficulty in training a system that can handle variety of data points. This was likely due to the way ANFIS calculates error (the sum of the squared difference between actual and desired outputs) for training purposes, and the distribution of relevant documents in the retrieval set. To overcome this difficulty, R-FIS was applied to refining the top ranked documents and not the whole answer set.

Extending ANFIS-R-FIS to handle a variable number of parameters appears to be difficult.

5.2 Knowledge Based R-FIS

5.2.1 Purpose

Construct an R-FIS based on the knowledge about the tf.idf method and information retrieval principles. Explore efficient and compact ways of constructing an R-FIS. Explore effects of various parameters of the system on its performance. The desired performance should be comparable to that of the baseline system.

5.2.2 Basic – 2 Rule R-FIS

These experiments are conducted on a very basic 2 Rule R-FIS.

1. If (LScore is high) and (tf1 is high) and (idf1 is rare) and (tf2 is high) and (idf2 is rare) and (tf3 is high) and (idf3 is rare) then (score is high) (1)
2. If (LScore is not high) and (tf1 is not high) and (idf1 is not rare) and (tf2 is not high) and (idf2 is not rare) and (tf3 is not high) and (idf3 is not rare) then (score is not high) (1)

Figure 21 - R-FIS Rules

Tag	**Refinement Set Size**	**Inputs**	**Comments**	**P10**
baseline All	*NA*	*NA*	*NA*	*0.3721*
baseline E	*NA*	*NA*	*NA*	*0.3784*
ex8_1	E 20	LScore, tf, idf		-0.0149
ex8_1_2	E 100	LScore, tf, idf		-0.0608
ex8_1_3	All 20	LScore, tf, idf		-0.0224
ex8_2	E 20	tf, idf		-0.0189
ex8_2_1	E 40	tf, idf		-0.278
ex8_2_2	All 20	tf, idf		-0.0299

ex8_2_3	All 40	tf, idf		-0.0537
ex8_2_4	All 20	tf, idf	adjusted mf range	-0.0286
ex8_3	E 20	tf, idf	FIS same as 8_2_4	-0.0162
ex8_3_1	E 40	tf, idf	FIS same as 8_2_4	-0.0446
ex8_3_2	E 100	tf, idf	FIS same as 8_2_4	-0.0825

All – all queries (odd & even)

E – only even queries

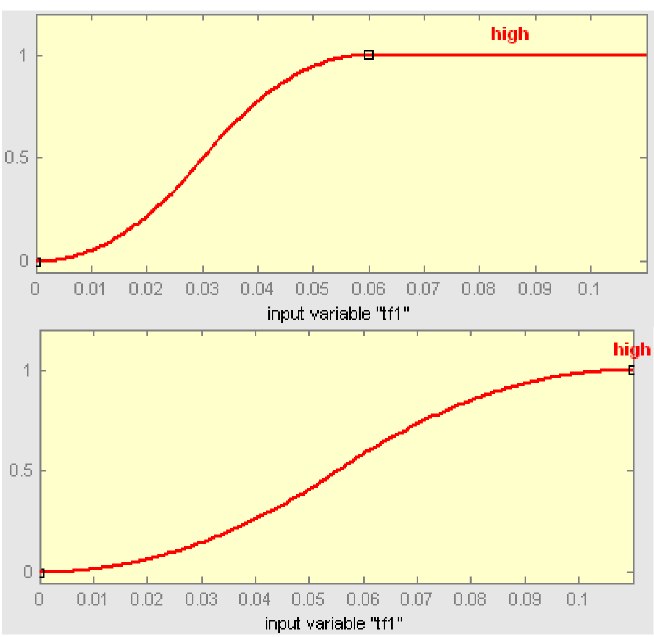
5.2.2.1 Refinement Set Size

An R-FIS system that utilizes smaller refinement set has a better performance. This indicates that the R-FIS is missing certain relevant documents that the other system is not. These results are worse than baseline; but considering the simplicity of the model, are quite good.

5.2.2.2 Inputs

Surprisingly, taking out the relevance score (LScore) of the baseline system Lucene, only slightly decreased its performance. This indicates that the rules of the FIS capture important principles of the underlying model.

5.2.2.3 Membership Function Adjustments



The membership functions were adjusted to their full range. These adjustments produced a small increase in performance. The reason that the increase is small is probably because there are not many data points that are at the extremes of the range.

5.2.3 Aggregation Method

There are a number of different aggregation methods available, such as: max, sum, probor. The nature of the information retrieval dictates that the determination of the ranking should be done based on all of the rules. In this case the sum aggregation method appears to be a much better fit. Sum has also been established to be an appropriate aggregation method by a formula based tf.idf model since terms are combined through the sum operator:

$$score(q,d) = \sum_{t \in q} \ldots\ldots$$

5.2.3.1 Max Aggregation Method

The max aggregation method (ex11_1), as expected, had very poor performance due to the fact that the rule with the biggest output value absolutely dominated the decision without any contribution from any other rules, e.g. if there is one term that has a strong presence and all the rest of the terms are not present, the document would still get a high score.

5.2.3.2 Sum Aggregation Method

The sum aggregation method (ex11_2) has exhibited good efficiency since all the rules were considered in rank determination.

5.2.4 Fuzzy Operator (Rules)

In this case the “and” fuzzy operator is used. The “and” fuzzy operator could be seen as an aggregation applied locally in this case to the terms of the rule. There are different operators for “and” operator such as: min, prod. Prod has a much better theoretical fit, since by using the prod rule, output is determined by all features of the terms and not just the minimum one. Prod has also been established to be an appropriate "and" method by a formula based tf.idf model; since in the formula tf and idf are combined through use of product operator: $score(q,d) = \sum_{t \in q}$ tf(t in d) * idf(t) * ….

5.2.5 Implication

The implication operator determines the shape of the consequent fuzzy set. The prod operator appears to be a better fit in comparison with the "min" operator; since "prod" scales the consequent fuzzy set unlike "min" that truncates the consequent fuzzy set. The prod operator allows the output fuzzy set to retain its shape properties; unlike the min operator that alters the shape of the resulting fuzzy set.

5.2.5.1 Min Implication Method

The min implication method (ex11_3), as expected, had poor performance, since the output fuzzy set became truncated and lost its original shape properties.

5.2.5.2 Prod Implication Method

The prod implication method (ex11_4) had good performance since the output fuzzy set was scaled and retained its original shape properties.

5.2.6 Rules

5.2.6.1 Rule Structure

The simplest rule structure (ex8_3_1) consists of two basic rules: high features $\rightarrow$ high relevance; low features $\rightarrow$ low relevance. This rule is quite effective, but does not provide for a lot of flexibility; and is perhaps over simplistic.

Another rule structure (ex11_6) is to follow the tf.idf model principles. Rules are created by pairing tf and idf together for the corresponding terms. Additional rules could also be added such as coord, or any other rules with features that are useful.

5.2.6.2 One Sided Rule Set (High) vs. Two Sided Rule Set (Low & High)

One sided Rule – ex. if (coord is high) → (relevance is high)

Two sided Rule – ex. if (coord is high) → (relevance is high); if (coord is low) → (relevance is low)

Tag	Rule Types	P10
baseline	*NA*	*0.3721*
ex11_5	One Sided	-0.3204
ex11_6	Two Sided	-0.1224

The two sided rule set exhibits much better performance in comparison to the one sided rule set. The two sided rule set's improved performance is most likely due to the fact that without it, the document was not penalized for the absence of good terms; for example, in a query with 3 terms, if a document-1 with all of three terms present but their tf measure is not high; and another document-2 with just 1 term present but with very high tf measure, document-2 could be ranked higher then document-1 since it did not get penalized for the absence of terms; which intuitively is not right.

There are several ways of creating a two sided rule set. One way is by adding appropriate membership functions to each input and output such as "low" and "high". Another way is by adding partially negated rules to already existing rules; for example if there existed the rule: if (tf is high) and (idf is high) → (relevance is high); the not rule could be added as follows: if (tf is not high) and (idf is not high) → (relevance is not high). The not rule approach is much more compact but it assumes that the opposite side membership function is inversely symmetrical. If that is the case efficiency, of the two approaches is equal, but representation of the not rule approach is much more compact (ex11_10, ex11_17).

However, if the opposing function is not approximately inversely symmetrical, a separate membership function may need to be added.

5.2.6.3 Rule Weights

In order to normalize rule weights, the sum of rule weights was attempted to be kept close to 1. Rule weights for similar rules such as for example for tf.idf rule pairs should be equal. Rule weights could be determined analytically or through experimentation.

5.2.6.3.1 Rule Weight for Coord

Tag	**coord weight**	**P10**
baseline	*NA*	*0.3721*
ex11_8	0.1	-0.0156
ex11_9	0.2	-0.0231
ex11_10	0.05	-0.0082
ex11_11	0.025	-0.0428

From experiments above it appears that the suboptimal weight of coord rule is about 5%. This factor could be different depending on the system and its parameters. The weight of coord is relatively small; it is probably due to the fact that coord rule is in some form present in rules that possess tf terms.

5.2.7 Defuzzification Method

Tag	**Defuzzification**	**P10**
baseline	*NA*	*0.3721*
ex11_10	centroid	-0.0082
ex11_12	bisector	-0.0585
ex11_13	mom	-0.3204
ex11_14	som	-0.3204

Centroid appears to be the most appropriate method of defuzzification for this type of problem and FIS structure. Centroid is defuzzification method that is used most often.

5.2.8 Membership Function Type

Function Type	Description
dsigmf	Difference of two sigmoid membership functions.
gauss2mf	Two-sided Gaussian curve membership function.
gaussmf	Gaussian curve membership function.
gbellmf	Generalized bell curve membership function.
pimf	Pi-shaped curve membership function.
psigmf	Product of two sigmoidal membership functions.
smf	S-shaped curve membership function.
sigmf	Sigmoid curve membership function.
trapmf	Trapezoidal membership function.
trimf	Triangular membership function.
zmf	Z-shaped curve membership function.

Figure 22 - Membershi Function Types

There are various membership types. Gaussian and Sigmoid based functions provide more flexibility; but the number of parameters of functions is bigger, so it makes it harder to tune manually, but it is well suited for the automated optimization approach. Generally Gaussian function better models the underlying variables, but knowledge about variables that are being modeled is necessary in order to define parameters of the function appropriately. Linear functions such as trapmf and trimf appear to do a decent job without a need for further tuning. In certain cases, it seems more beneficial to use linear membership functions such as trapezoidal (trapmf) and triangular (trimf) membership functions. For example, in the case of idf; it appears to be more efficient to use trapmf or trimmf since idf has already been normalized with log function: $idf = \log\left(\frac{N}{n}\right)$.

5.2.8.1 Sigmoid Membership Function Adjustment

Since the sigmoid function has many parameters, it is much better suited for application of the optimization algorithm. Unfortunately, the command line interface of Matlab was producing a core dump, so the only option was manual adjustment.

Tag	Sigm Center	Sigm Left	P10
baseline	*NA*	*NA*	*0.3721*
ex11_16	0.1	0.03	-0.1231
ex11_18	0.38	0.18	-0.0218
ex11_19	0.5	0.1	-0.0143

ex11_20	0.5	0.2	-0.0129
ex11_21	0.5	0.3	-0.0143
ex11_22	0.5	0.05	-0.0129
ex11_23	0.5	0	-0.0129
ex11_24	0.4	0.1	-0.0143
ex11_25	0.4	0.2	-0.0156
ex11_26	0.4	0.05	-0.0122
ex11_27	0.3	0.05	-0.019
ex11_28	0.7	0.05	-0.0129
ex11_29	0.35	0.1	-0.0177

It appears that ex11_26 is the best one; but there are many runs that have performance very near it. It was possible to achieve a good performance with manual adjustment, but automatic adjustment would likely be much more efficient and produce better results. It is also possible to apply statistical analysis in order to determine the parameters of the membership function.

5.2.9 Extending R-FIS based on variable number of terms

All of the experiments up to experiment 12_1 were executed on the fixed number of terms. Experiments 12_1, 14_1 demonstrated a simple procedure to extend R-FIS to handle variable number of terms.

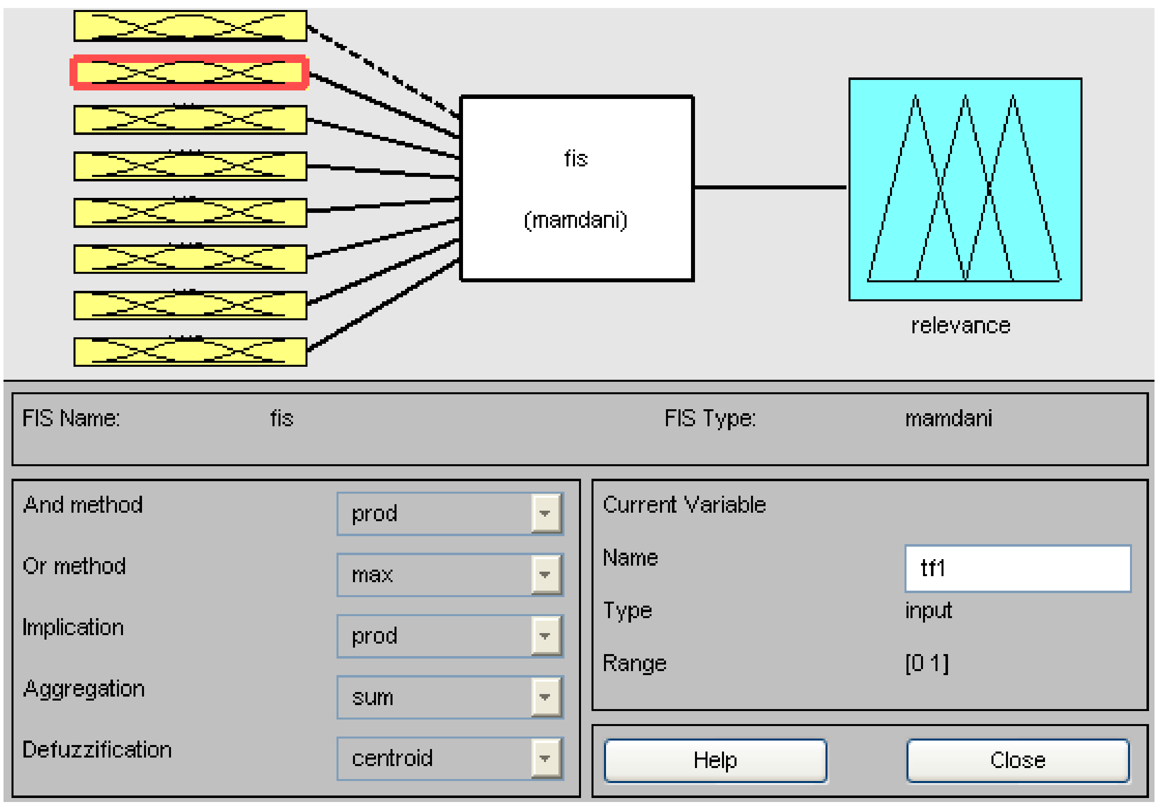
First it is necessary to determine maximum number of terms. This number could be quite high. In experiment 14_1 max_terms = 35.

The R-FIS is automatically constructed for each number of the terms. There are certain properties that are adjusted automatically such as: sum of rule weights = 1; coord weight. In experiment 14_11, coord rule weight was set at a constant level. This proved to hinder the performance of the system, so it could be beneficial to set the weight of the coord rule as a percentage of the weight of the tf.idf rule.

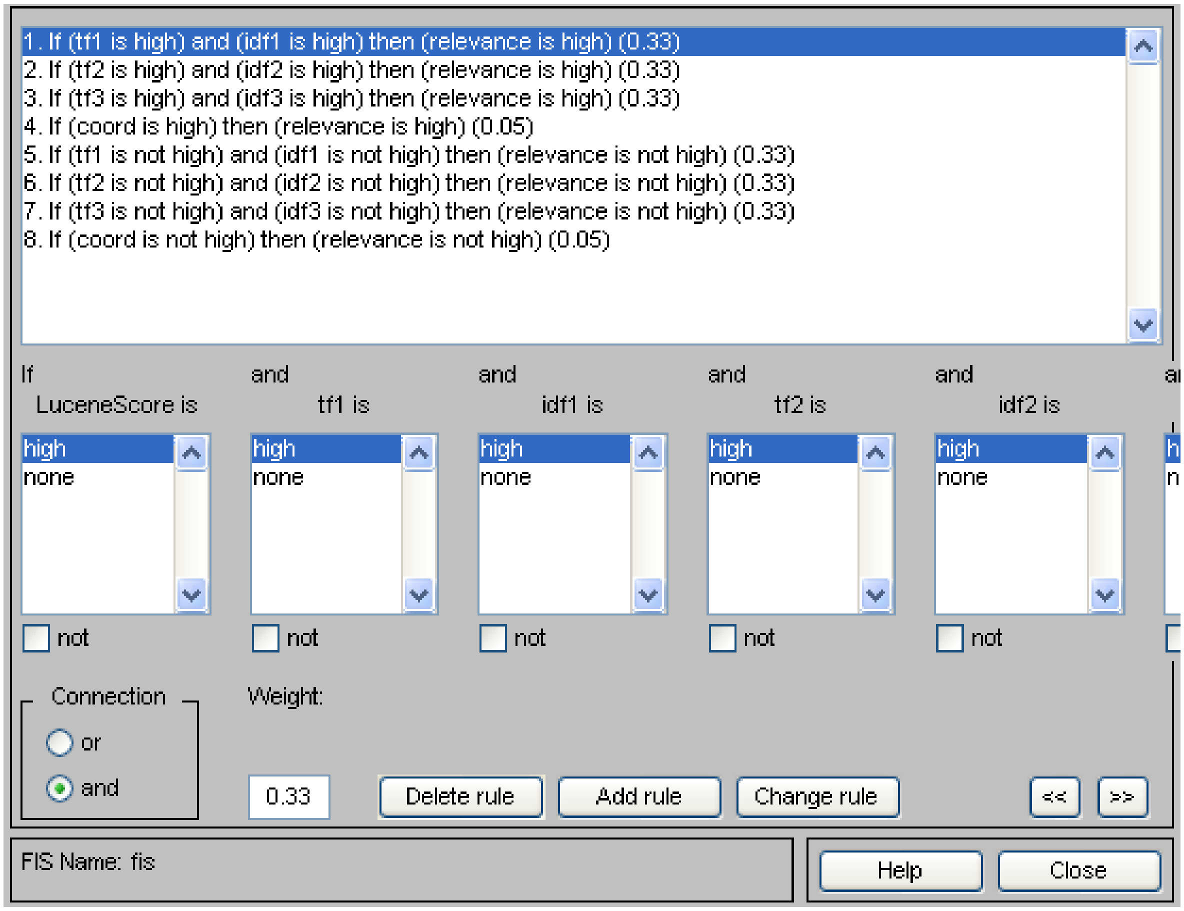
If it is not convenient to generate multiple R-FIS to handle different number of terms, another equivalent approach is possible. Two R-FIS are constructed. The first R-FIS handles up to max_terms; for non existing terms tf and idf are 0. Then the results are passed as inputs to the second R-FIS which automatically adjusts weights of non per-term features based on the number of terms and, finally, outputs the ranking score.

5.2.10 FIS Analysis and Modifications

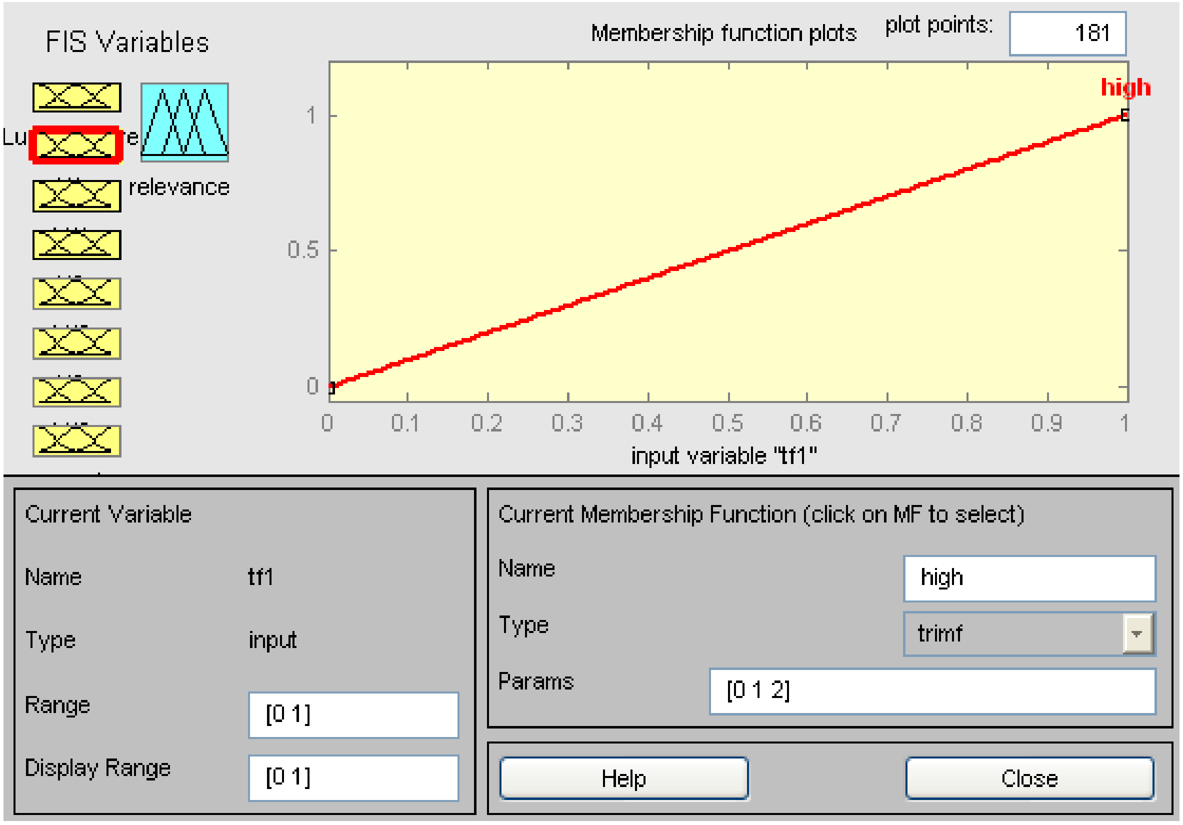
The biggest advantage of Mamdani R-FIS approach is its simplicity and ease of use.



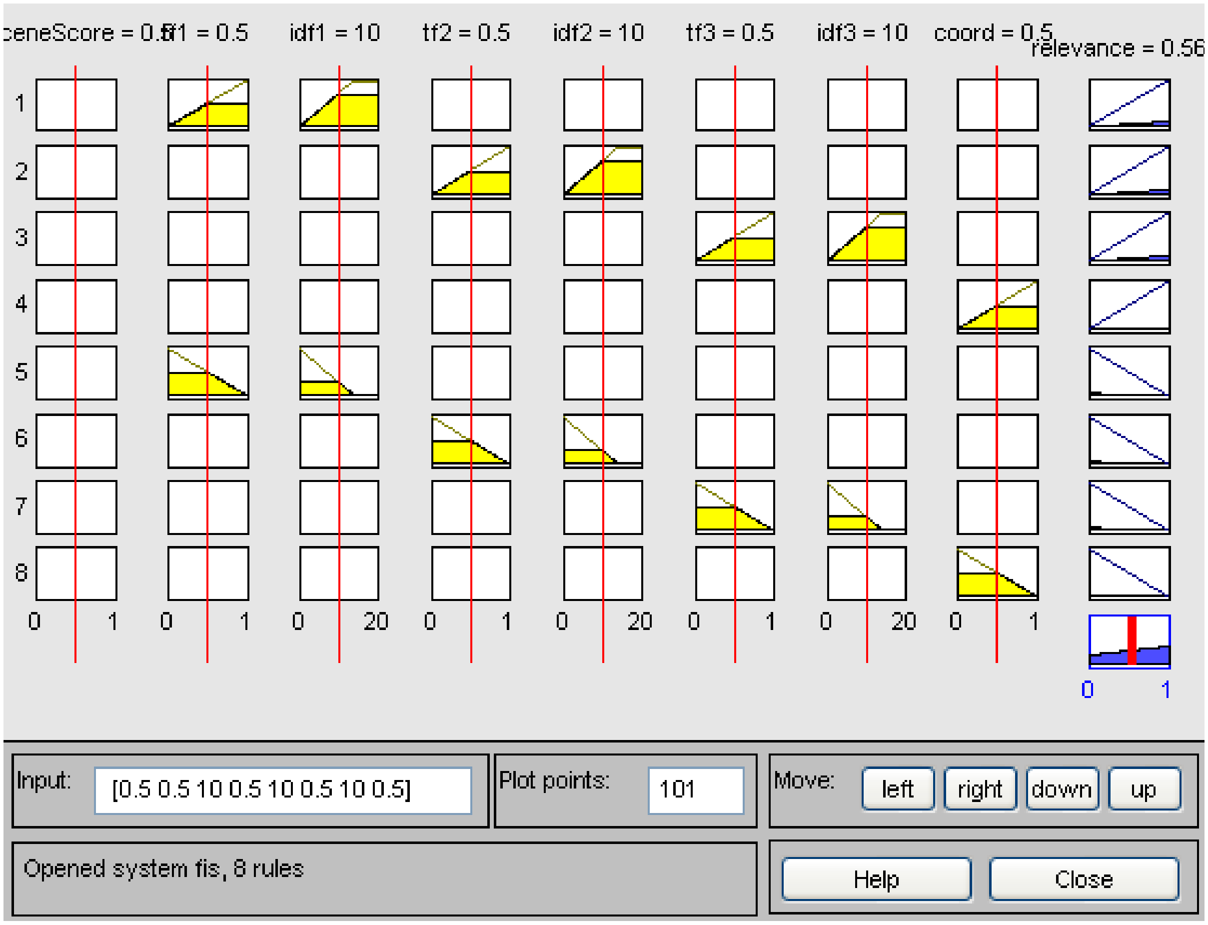
Properties of the FIS can be easily viewed and modified if necessary.



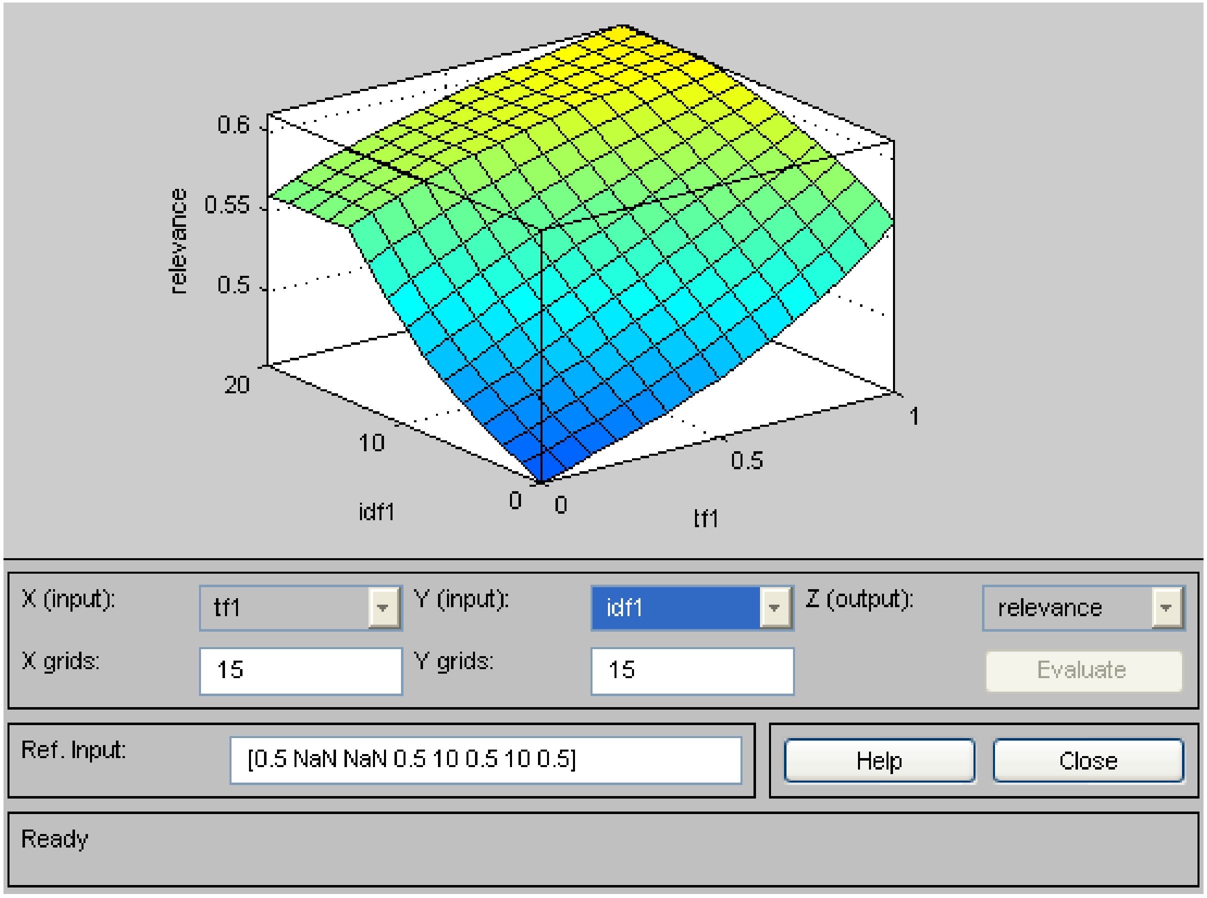
Rules of the system can be viewed and modified very easily.



Membership functions can be easily view and modified.



The way the system rules function can be visually examined.



Interaction between various variables can be examined.

5.2.11 Conclusion

Tag	**P10**
baseline	*0.3721*
ex11_10	-0.0082

Table 6

Construction and use of Knowledge Based R-FIS is intuitive and is relatively simple and its performance is almost equal to (P10 within 0.82%) that of the baseline model. The KB R-FIS model is very easy to extend and modify. Even system with only 2 simple rules had a good performance (ex8_2_2). However, system with more rules had a better performance and provided for more flexibility. Unlike the case of ANFIS R-FIS, there is no need for the refinement set since the KB R-FIS system is able to successfully handle a variety of inputs. The system performed well with only the basic tf.idf features such as: tf, idf, coord. It is likely that this performance could be improved by adding more features and this model provides for a very easy way to do so. For constructing a system with a good performance, a basic understanding of the tf.idf principles and the basic principles of information retrieval theory appears to be sufficient. All of the principles utilized are as follows:

- Each term and its features have an impact on the relevance; they are typically combined through a sum operator.
    - FIS rule aggregation method = sum
- Combining tf and idf features is typically done with a product operator.
    - FIS Logical operator "and" is product.
- High tf idf features contribute to higher relevance; low tf idf features contribute to lower relevance.
    - FIS Rule: if (tf is high) and (idf is high) → (relevance is high)
    - FIS Rule: if (tf is low) and (idf is low) → (relevance is low)
- If many of the terms of the query are present in the document then the document is likely to be more relevant; if few terms are present, then the document is likely to be less relevant.
    - FIS Rule: if (coord is high) → (relevance is high)
    - FIS Rule: if (coord is low) → (relevance is low)

Utilizing only these principles, a system with a good performance was easily constructed. This system was shown to be easily extendable to work on a variable number of query terms. The execution speed of this system is very good – fuzzy logic based systems are widely used in engineering applications where near real time performance is a requirement.

5.3 Tuning Knowledge Based R-FIS with ANFIS

Tag	Training Method	Training Set (Top)	Refinement Set	P10
baseline (title.even.3)	*NA*	*NA*	*NA*	*0.3784*
ex13_01 (mam base)	*NA*	*NA*	*NA*	-0.0322
ex_16_1 (sug base)	*NA*	*NA*	100	-0.0311
ex_16_2	backprop	100	100	0.0175
ex_16_3	hybrid	100	100	-0.0581
ex_17_1 (sug base)	*NA*	*NA*	40	-0.0257
ex_17_2	backprop	100	40	0.0189
ex_17_3	hybrid	100	40	-0.0473
ex_18_1 (sug base)	*NA*	*NA*	20	-0.0095
ex_18_2	backprop	100	20	0.0135
ex_18_3	hybrid	100	20	-0.0298
ex_19_1 (sug base)	*NA*	*NA*	40	-0.0257
ex_19_2	backprop	40	40	0.0135
ex_19_3	hybrid	40	40	-0.0054
ex_20_1 (sug base)	*NA*	*NA*	20	-0.0095
ex_20_2	backprop	40	20	0.0054
ex_20_3	hybrid	40	20	-0.0311
ex_21_1 (sug base)	*NA*	*NA*	20	-0.0095
ex_21_2	backprop	20	20	0
ex_21_3	hybrid	20	20	-0.023

5.3.1 Purpose

To convert Knowledge Based R-FIS to the Sugeno system with the use of the mam2sug function of Fuzzy Logic Toolbox of Matlab. This approach should allow for the tuning of parameters of KB-R-FIS with ANFIS and improved performance.

5.3.2 Construction & Tuning Procedure

This method is the combination of the ANFIS method and Knowledge Based R-FIS method. First, an FIS is constructed by using Knowledge Based R-FIS method and is tested to ensure that it functions as expected. Then it is converted to the Sugeno model with the use of Matlab function mam2sug and the corresponding model is trained with the use of ANFIS.

Due to the restrictions placed by the mam2sug function of Matlab, it is not possible to convert arbitrary Mamdani system to the Sugeno system. The Mamdani system must have certain properties, which can be easily accomplished.

- Rules are not allowed to share an output mf
    - Solution: create a copy of the appropriate output membership function for each of the rules
- Rules can not have negated antecedents
    - Solution: create an inversely symmetrical membership function for the antecedent that is negated in the rule. Replace negated antecedent with the membership in the inversely symmetrical membership function.
- ANFIS replaces all of the weight of the rules with 1
    - Solution: Membership functions have to be scaled appropriately.

Mam2sug transforms a Mamdani FIS structure into a Sugeno FIS structure. The returned Sugeno system has constant output membership functions. These constants are determined by the centroids of the consequent membership functions of the original Mamdani system; the antecedent remains unchanged [4].

5.3.3 Training / Tuning

5.3.3.1 Base Model

For the baseline model used model from experiment13_01, it is one of the best Mamdani R-FIS and it has a normalized idf, which is a desirable property. Tf4 and idf4 were removed since running with 3 terms allows for a bigger testing and training set.

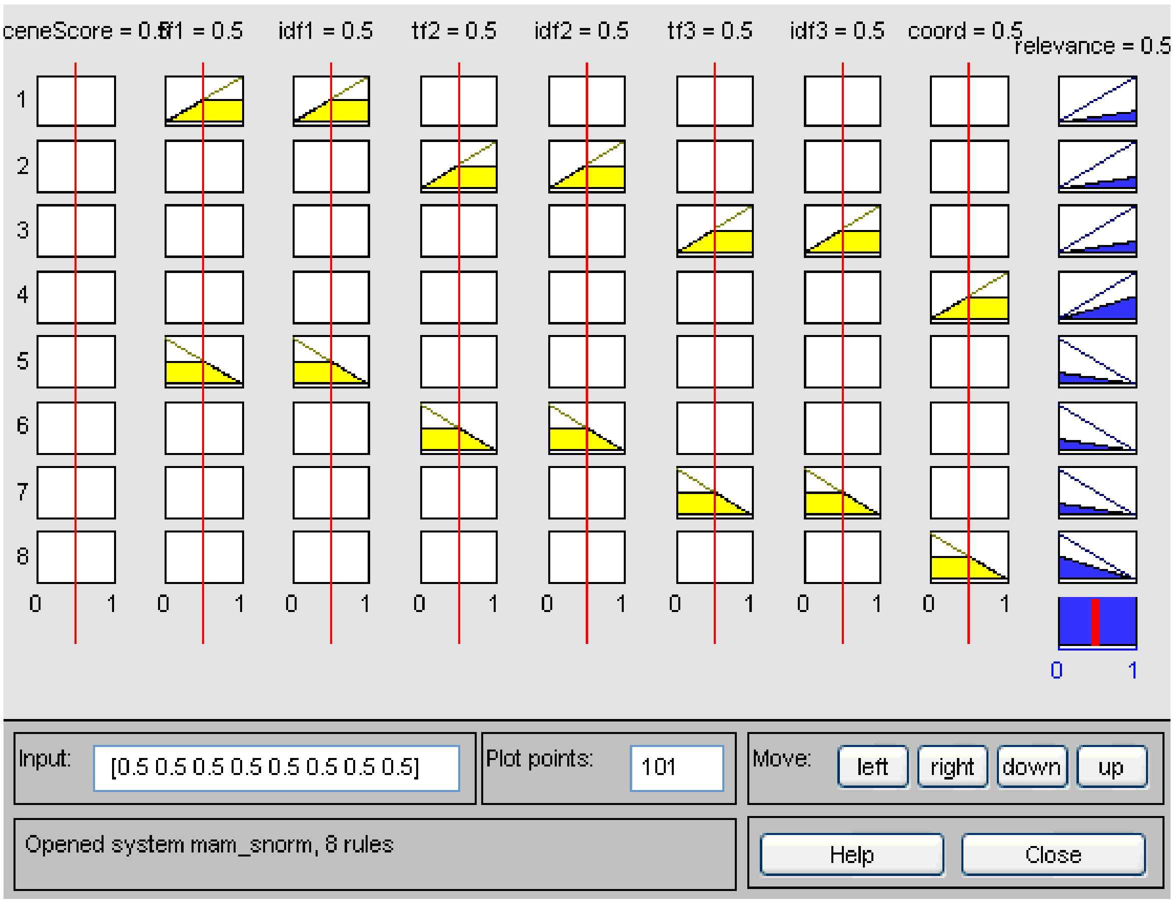


Figure 23 – Mamdani FIS from ex 13_01 (normalized)

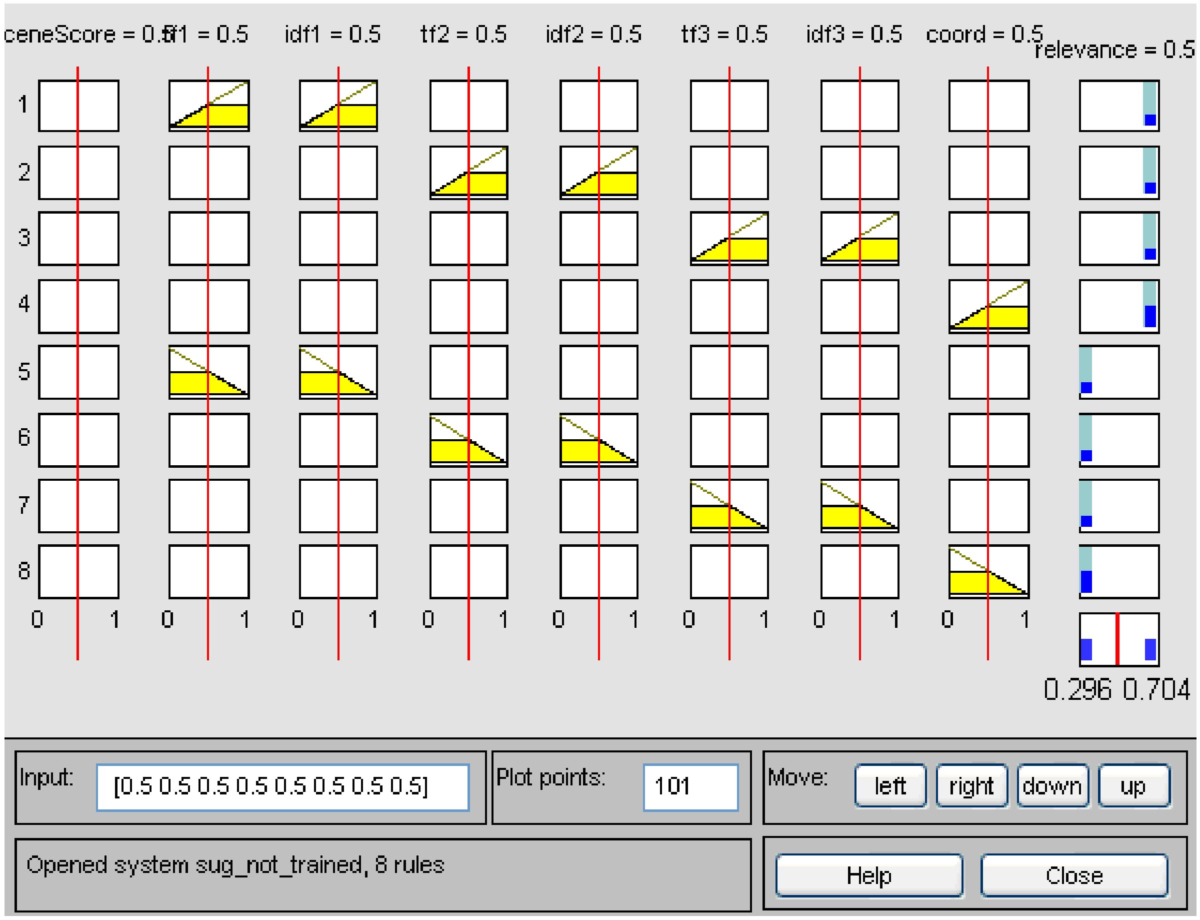


Figure 24 – Sugeno system converted with mam2sug from ex13_01

Figure 24 shows the rules of ex16_1 that was created by mam2sug from ex13_01.

5.3.3.2 Backpropagation Training

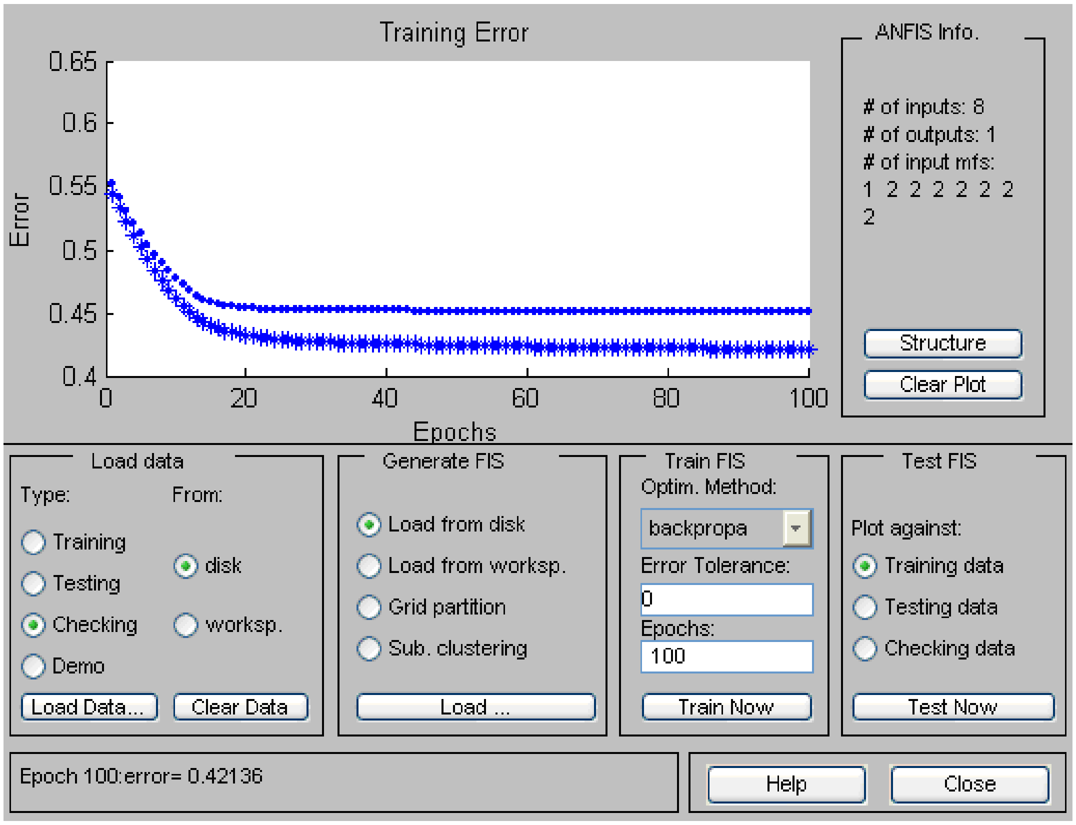


Figure 25 - Backpropagation (epochs; training/checking plot)

Backpropagation training method works better then hybrid and is faster. From the figure above, it can be seen that optimal value is obtained somewhere after 30 epochs. In the beginning of the training, checking and training data point are very close to each other; this indicates that the baseline mamdani model has a good generic set of rules. Somewhere around 20th epoch, training error continued to decline while checking error remained relatively flat; this indicates that overfitting started to occur.

5.3.3.3 Hybrid Training

Hybrid training combines backpropagation and least squares estimation method.

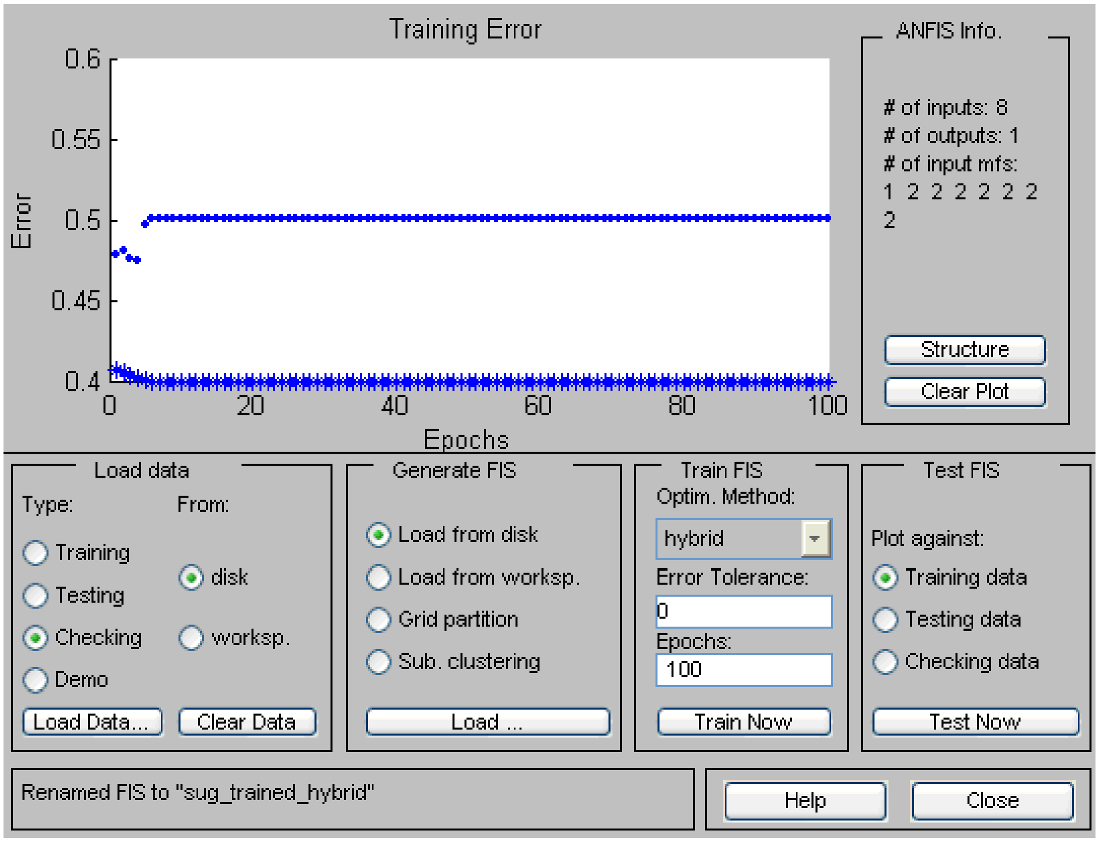


Figure 26 - Hybrid Training

Hybrid training is actually counterproductive. It is probably due to the fact that baseline model was already a good one. It would have been difficult for this method to optimize it further.

5.3.4 Analysis of R-FIS Structure

5.3.4.1 Backpropagation Training

Figure 27 - BP Rules

It is possible to perform an analytical analysis of the rules of R-FIS tuned by ANFIS. A comparison could be done with the baseline rules outlined in Figure 24. Weight for the rules were at 1 in the beginning and tuning has shifted them somewhat significantly. However, the weights are not as important as their proportions in relation to each other. Several surprising 'optimizations' could be noticed: for some reason, positive tf.idf rule for term2 has the biggest positive weight; in the negative case, term2 has the biggest negative weight.

5.3.4.2 Hybrid Training

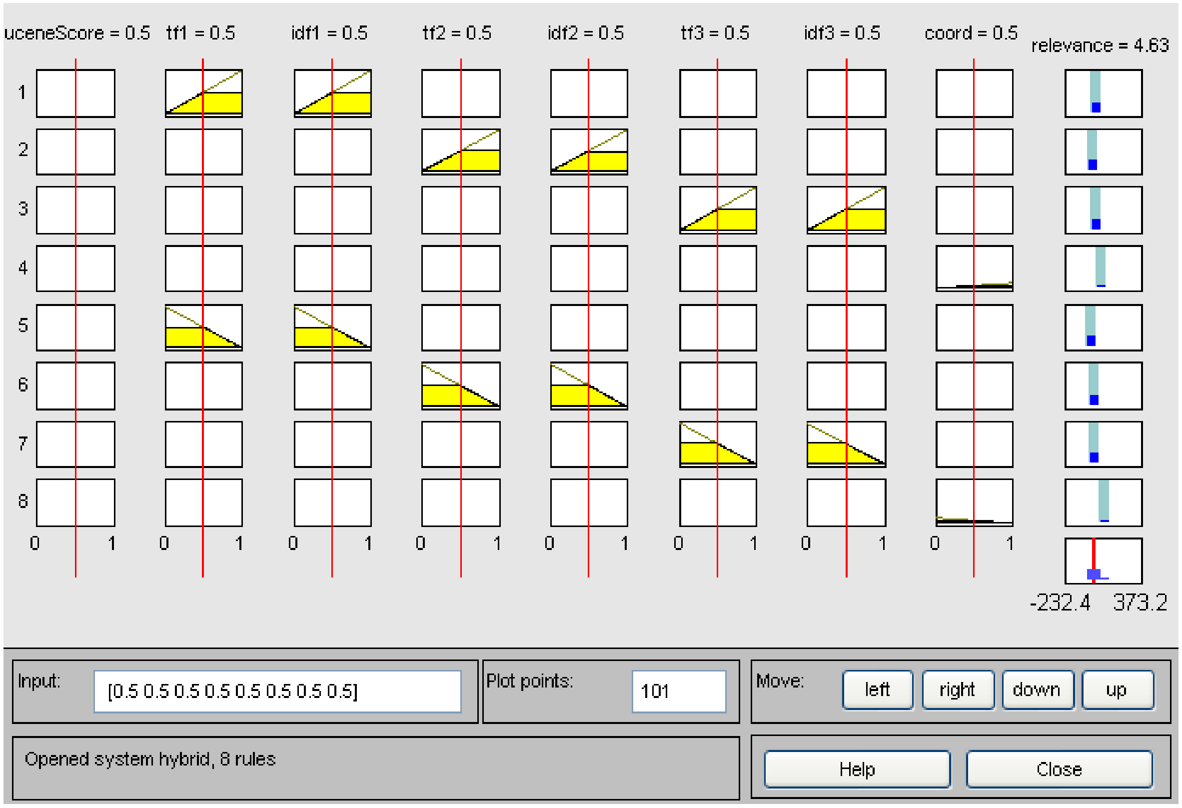


Figure 28 - Hybrid Rules

The performance of the hybrid training method was poor. The resulting system had a worse performance after it was trained than before. The reason for it becomes clear by examining its rules. Positive and negative rules are much closer together. It is not obvious why rules were optimized this way, but from Figure 26, it appears that the system became overfitted.

5.3.5 Conclusion

Tag	**P10**
baseline	*0.3784*
ex_17_2	0.0189

Tuning parameters of R-FIS with ANFIS resulted in a small improvement to P10 of +1.89%. As demonstrated in section 5.3.3.2, initial Knowledge Based R-FIS possesses good generic rules and ANFIS is able to further tune these rules. However, when performing tuning, the user must be cautious not to run too many epochs, since overfitting of the system is likely to occur. Backpropagation has proved to be an efficient and fast method of tuning parameters of R-FIS. On the other hand, using hybrid tuning method after several epochs made the system strongly overfitted, which resulted in a poor performance by comparison with an untrained system.

Analysis of the resulting FIS structure allows for a view of the resulting system rules and to transfer the knowledge back into Knowledge Based R-FIS. If desired, a sugeno system could be kept, but the rules could still be verified. ANFIS calculates error as the sum of the squared difference between the actual and desired outputs. Training an FIS this way would create an FIS that is optimized for Average Precision. It is likely that modifying ANFIS error measurements to P10 would result in a better system. ANFIS has proven to do a good job at tuning the system. However, it could be better if Knowledge Based R-FIS could be tuned directly, but in order to do that, a tuning system would have to be implemented.

5.4 F-TF.IDF Construction – Merging ANFIS / Manual Mamdani / Lucene result sets

5.4.1 Purpose

Explore if the strengths of the systems are mutually exclusive to a certain degree. By merging result sets from different models, it is possible to improve the system and make it more resilient to failure in certain circumstances. However, combining them could potentially weaken strengths that each individual approach had.

5.4.2 Procedure

Result sets from two systems are merged with the following procedure:

Start iterating through the result sets: for each set: take a document from the set and put it into the joined set unless it is already there.

5.4.3 Merging Knowledge Based R-FIS with Lucene

Tag	**Comments**	**MAP**	**P10**
baseline	*Lucene title.even.3*	*0.2038*	*0.3784*
ex_17_2	KB R-FIS + ANFIS R-FIS	0.1426	0.3973
ex_22_1	Combined	0.1424	0.3811

Results of this experiment are not good, as it failed to improve either of the measures.

5.4.4 Merging Knowledge Based R-FIS with Lucene

Tag	**Comments**	**MAP**	**P10**
baseline	*Lucene title.3*	*0.1972*	*0.3721*
ex13_01	KB R-FIS	0.1835	0.3462
ex_22_2	Combined	0.1904	0.3542

Results of this experiment are not good, as it failed to improve either of the measures.

5.4.5 Conclusion

If a system has a particular strength in one area, and weakness in the other and if the goal is to have a system that performs relatively well under most circumstances, combining two systems could prove beneficial. In this case, it appears that there is no obvious benefit of merging result sets of various systems.

6 Comparison of R-FIS and Lucene with TREC Robust participants

6.1 Conditions

6.1.1 Robust Track Participants

Participants in the TREC Robust Track did not have access to “New” set of topics.

6.1.2 Lucene

No modifications of any kind were performed on Lucene. Lucene version 1.4.final from Apache Jakarta was used. The collection was indexed with StandardAnalyzer. All fields were indexed as plain text; there was no special treatment of any fields including the <TITLE> field.

6.1.3 R-FIS

R-FIS used index data from Lucene. The goal of experiments and modification was to explore various theoretical and practical possibilities and to construct a conceptually user friendly system with a performance approximately equal to or better then baseline tf.idf model (Lucene).

All of the experiments were conducted on the title queries with 3 terms. In experiments where ANFIS was used, the system was trained on the odd queries and evaluated on the even. Robust track participants were evaluated on all the queries.

In experiments with Knowledge Based R-FIS, the system was constructed from prior knowledge of tf.idf systems and tested on the whole collection. R-FIS appears to be consistent across topics sets; systems that performed best on the "Old" topic set also performed best on the "New" topic set; so it is still likely to be a good indicator of the performance of the system.

6.2 Comparison

Chinese Academy of Sciences (CAS-NLPR)	Fondazione Ugo Bordoni
Hong Kong Polytechnic University	Hummingbird
IBM Research, Haifa	Indiana University
Johns Hopkins University/APL	Max-Planck Institute for Computer Science
Peking University	Queens College, CUNY
Sabir Research, Inc.	University of Glasgow
University of Illinois at Chicago	Virginia Tech

Table 7 - Robust Track Group Participants [14]

	Old Topic Set			*New Topic Set*			*Hard Topic Set*			*Combined Topic Set*		
Tag	**MAP**	**P10**	**%no**	**MAP**	**P10**	**%no**	**MAP**	**P10**	**%no**	**MAP**	**P10**	**%no**
pircRB04t3	0.32	0.51	5%	0.4	0.55	6%	0.18	0.37	12%	0.33	0.51	5%
fub04Tge	0.3	0.48	13%	0.35	0.48	12%	0.15	0.34	22%	0.31	0.48	12%
uic0401	0.31	0.49	5%	0.33	0.44	6%	0.19	0.38	4%	0.31	0.48	5%
uogRobSWR10	0.3	0.46	16%	0.32	0.45	12%	0.14	0.32	26%	0.3	0.46	15%
vtumtitle	0.28	0.44	20%	0.3	0.43	14%	0.14	0.27	36%	0.28	0.44	19%
humR04t5e1	0.27	0.46	13%	0.3	0.46	12%	0.14	0.33	20%	0.28	0.46	13%
JuruTitSwQE	0.26	0.44	10%	0.27	0.41	10%	0.12	0.28	12%	0.26	0.44	10%
SABIR04BT	0.24	0.42	18%	0.29	0.39	20%	0.12	0.24	32%	0.25	0.41	18%
apl04rsTs	0.24	0.41	13%	0.27	0.39	10%	0.11	0.26	14%	0.25	0.4	12%
polyutp3	0.23	0.42	14%	0.26	0.39	10%	0.08	0.24	24%	0.23	0.41	13%
Lucene title	0.19	0.37	17%	0.23	0.37	10%	0.1	0.25	22%	0.2	0.37	15%
Lucene title (3 terms)	0.18	0.37	14%	0.26	0.39	11%	0.1	0.24	11%	0.2	0.37	12%
Lucene title.even.3	0.2	0.38	14%	0.22	0.38	12%	0.11	0.24	25%	0.2	0.38	13%
ANFIS (ex7_2)	0.11	0.37	12%	0.14	0.41	12%	0.03	0.25	16%	0.12	0.38	12%
KB-R-FIS (ex11_10)	0.17	0.35	14%	0.25	0.4	11%	0.1	0.25	11%	0.19	0.36	12%
Luc+KB-R-FIS (ex_22_1)	0.14	0.38	12%	0.17	0.4	12%	0.05	0.24	25%	0.14	0.38	12%
Luc+ANFIS (ex_22_2)	0.18	0.36	16%	0.22	0.35	12%	0.09	0.24	26%	0.19	0.35	14%

Figure 29 [14]

Figure 29 shows evaluation results for the best title-only run for the top 10 groups as measured by MAP over the combined topic set, plus the best of R-FIS title-only runs and baseline runs of Lucene. Runs of the track participants are ordered by MAP over the combined topic set; runs of the Lucene and R-FIS are at the bottom of the table. Values given are as follows: the mean average precision (MAP), precision at rank 10 averaged over topics (P10), the percentage of topics with no relevant in the top ten retrieved (%no), and the area underneath the MAP(*X*) vs. *X* curve (area) [14].

In the discussion of the R-FIS, the results will primarily focus on the performance of Knowledge Based R-FIS since this approach is the preferred approach due to its theoretical properties and its performance is approximately equal to the performance of other approaches. As discussed in 1.2.4, many of the participants have utilized Query Expansion techniques in order to improve performance of the system. It is likely that absence of Query Expansion in R-FIS and Lucene has prevented them from reaching better results.

Topic Set	Number of topics	Mean Relevant per topic	Minimum # Relevant	Maximum # Relevant
Old	200	76.8	3	448
New	49	42.1	3	161
Hard	50	88.3	5	361
Combined	249	69.9	3	448

Table 8 - Relevant document statistics for topic sets [14]

On the old topic set, the performance of R-FIS was worse than that of the track participants. The baseline model also did not perform well on this topic set. The poor performance of R-FIS is probably due to the limitations of the base tf.idf model, as indicated by the poor performance of the baseline model on this topic set.

On the new topic set, the performance of R-FIS was better than on the old set. MAP was slightly bellow the 10th ranked participant. P10 was better than that of three of the participants. Intuitively, R-FIS would be expected to perform worse on new topic set since “Mean Relevant per topic” is almost half of old topic set, which would indicate harder queries. However, the baseline tf.idf model Lucene also performed better on this topic set. Many of the participating systems performed better on the new topic set in comparison with old topic set. The better performance on this topic set seems to be related to the nature of the queries in the set and the tf.idf weighting scheme being able to perform better in these circumstances. However, the best performing systems have a significant advantage in performance over the baseline approach and R-FIS.

On the hard topic set, the performance of R-FIS decreased, as expected, due to the difficulty of the queries. In comparison with the track participants, R-FIS performed better than the 10th ranked participant. Probable reasons for the improved performance of R-FIS, in comparison with track participants are similar to those of the new topic set, such as: the nature of the queries in the set, and the tf.idf weighting scheme being able to perform better in these circumstances. The best performing systems have a significant advantage in performance over the baseline approach and R-FIS.

On the combined topic set, the performance of the R-FIS was worse than that of the track participants. It demonstrates the need for additional techniques, such as Query Expansion and/or utilizing knowledge and features of other weighting methods, such as Okapi BM25 in order to improve performance of R-FIS.

7 Conclusion

Tag	MAP	P10
baseline [Lucene title (3 terms)]	*0.1972*	*0.3721*
baseline [Lucene title even 3 terms]	*0.2038*	*0.3784*
KBR-FIS 11_10 [3 terms]	-0.0061	-0.0082
ANFIS 7_2 [even]	-0.0836	-0.0014
Luc+MamKT 22_1 [even]	-0.0614	0.0027
Luc+ANFIS 22_2 [even]	-0.0134	-0.0242

Table 9

An easy and intuitive way of constructing a Knowledge Based R-FIS by utilizing the basic principles of the tf.idf model and information retrieval has been demonstrated. The following tf.idf features were used: tf, idf, coord. The following are all of the principles that were utilized:

- Each term and its features have an impact on the relevance; they are typically combined through a sum operator.
    - FIS aggregation method = sum
- Combination of tf and idf features is typically done with a product operator.
    - FIS Logical operator “and” is product.
- High tf idf features contribute to higher relevance; low tf idf features contribute to lower relevance.
    - FIS Rule: if (tf is high) and (idf is high) $\rightarrow$ (relevance is high)
    - FIS Rule: if (tf is low) and (idf is low) $\rightarrow$ (relevance is low)
- If many of the terms of the query are present in the document, then the document is likely to be more relevant; if few terms are present then the document is likely to be less relevant.

- FIS Rule: if (coord is high) $\rightarrow$ (relevance is high)
- FIS Rule: if (coord is low) $\rightarrow$ (relevance is low)

Performance of the resulting system is approximately equal to that of the baseline model (P10 difference -0.82%; MAP difference -0.61%). This model is user friendly, it is easy to modify and extend it. Tuning of R-FIS could be performed by modifying its parameters. The system could be easily examined and verified with the graphical tools of the Matlab Fuzzy Logic toolbox.

ANFIS have proven to be efficient in tuning the Knowledge Based R-FIS. The backpropagation method of ANFIS has proved to be an efficient method for tuning the R-FIS. The hybrid method of R-FIS tuning had exhibited poor performance. For better results, a custom made solution would likely be more efficient.

Applying ANFIS to deriving the initial structure of the FIS has exhibited some potential. Generation of the FIS with the use of Subtractive Clustering allowed for constructing a compact FIS with good performance and with small computational resources required for training the system. Generation of the FIS with the use of Grid Partition Clustering had a number of problems, such as the inability to scale and the resulting FIS was overly complex. ANFIS R-FIS was not able to handle a variety of situations; its use is, therefore, restricted to the refinement of the top of the answer set. Extending the ANFIS R-FIS to handle a variable number of terms was explored and is possible but appears to be difficult.

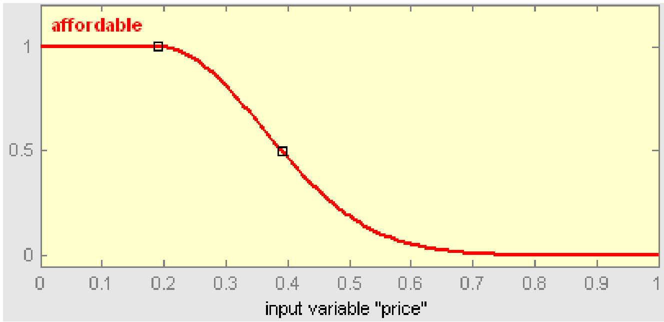
In comparison to current approaches of the TREC 2004 Robust Retrieval Track, both baseline model Lucene and R-FIS did not perform well. After examining the approaches used by the track participants in 1.2.4, the need of incorporating advances in IR became apparent. It is very likely that incorporating techniques such as Query Expansion and/or utilizing knowledge and features of other weighting methods, such as Okapi BM25 into R-FIS, would result in improved performance.

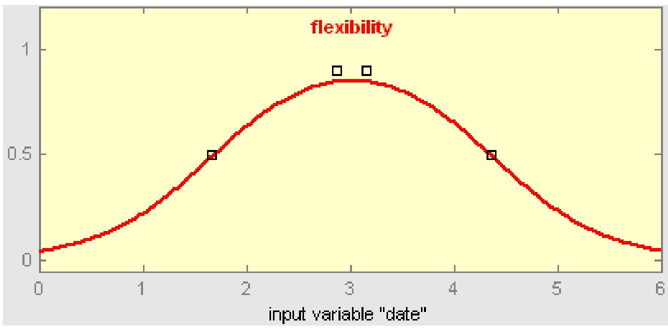
8 Future Research

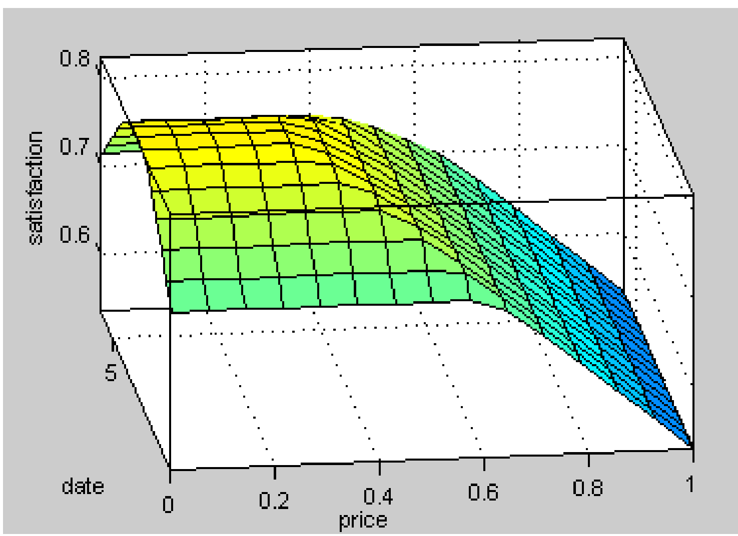
In order to more fully explore the potential of using Fuzzy Logic to determine the ranking function of IR systems, many possibilities ought to be explored. Tuning Knowledge Based R-FIS by applying various automated optimizations and learning methods is likely to improve its performance and contribute to understanding of principles behind it. Custom optimization methods would allow for a system to be optimized for the desired characteristic (ex. P10, MAP, etc.).

It is likely that some of the enhancements that were exhibited by TREC Robust participants (1.2.4) could be incorporated into R-FIS through rules, additional features, and by enhancing the underlying model. Some of the enhancements that could result in improved performance are: Query Expansion; use of Okapi BM25 features or some other variation of Okapi method [11]. The enhancement that is likely to make the biggest positive impact is Query Expansion. Many of the participants of the robust track have successfully used Query Expansion as a component of their systems. R-FIS could also be applied to various retrieval models that have well defined rules and provide access to underlying features.

IR systems are being used in many areas besides search engines. For example, IR systems are also used as advisory systems. Since FL allows for the easy translation of natural language into rules; it could allow users to define their preferences, constructing FIS from specified preferences and then evaluating options based on their preferences against possible choices. For example, if a person is searching for a flight, they can define their preferences in the following way: dates of travel - want to travel on Wed(3), plus minus one day is satisfactory; affordable price very important. Then their satisfaction plot would be as indicated bellow. It would be possible to evaluate various choices with a given FIS.


affordable
1
0.5
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
input variable "price"


flexibility
1
0.5
0
0 1 2 3 4 5 6
input variable "date"


0.8
0.7
0.6
satisfaction
5
date
0 0.2 0.4 0.6 0.8 1
price

9 Appendix

9.1 Lucene Experiments

Tag	Old Topic Set				New Topic Set				Hard Topic Set				Combined Topic Set			
	MAP	P10	%no	area	MAP	P10	%no	area	MAP	P10	%no	area	MAP	P10	%no	area
title	0.1896	0.3665	16.50%	0.0092	0.2316	0.3694	10.20%	0.0179	0.0975	0.25	22.00%	0.0058	0.1979	0.3671	15.30%	0.0102
title.3	0.1763	0.3658	13.50%	0.0088	0.2619	0.3917	11.10%	0.0175	0.0976	0.2423	11.50%	0.0075	0.1972	0.3721	12.90%	0.0097
title.3.even	0.1985	0.3776	13.80%	0.0132	0.223	0.3812	12.50%	0.0166	0.1084	0.2417	25.00%	0.0092	0.2038	0.3784	13.50%	0.013
description	0.1766	0.3555	18.00%	0.0038	0.2186	0.3388	16.30%	0.0066	0.0832	0.242	24.00%	0.0015	0.1849	0.3522	17.70%	0.0041

9.2 Experiments – Absolute Values Table

Tag	Old Topic Set				New Topic Set				Hard Topic Set				Combined Topic Set			
	MAP	P10	%no	area	MAP	P10	%no	area	MAP	P10	%no	area	MAP	P10	%no	area
title	0.1896	0.3665	16.50%	0.0092	0.2316	0.3694	10.20%	0.0179	0.0975	0.25	22.00%	0.0058	0.1979	0.3671	15.30%	0.0102
title.3	0.1763	0.3658	13.50%	0.0088	0.2619	0.3917	11.10%	0.0175	0.0976	0.2423	11.50%	0.0075	0.1972	0.3721	12.90%	0.0097
title.even.3	0.1985	0.3776	13.80%	0.0132	0.223	0.3812	12.50%	0.0166	0.1084	0.2417	25.00%	0.0092	0.2038	0.3784	13.50%	0.013
description	0.1766	0.3555	18.00%	0.0038	0.2186	0.3388	16.30%	0.0066	0.0832	0.242	24.00%	0.0015	0.1849	0.3522	17.70%	0.0041
ex2_1	0.0806	0.1207	46.60%	0.0024	0.0385	0.05	75.00%	0.0032	0.0779	0.125	41.70%	0.002	0.0715	0.1054	52.70%	0.0023
ex2_2	0.0666	0.3724	13.80%	0.0005	0.0917	0.3812	12.50%	0.0025	0.0271	0.225	25.00%	0	0.072	0.3743	13.50%	0.0006
ex2_3	0.0824	0.2759	20.70%	0.0022	0.0907	0.2687	18.80%	0.0028	0.0235	0.1333	41.70%	0.0013	0.0842	0.2743	20.30%	0.0019
ex3	0.0748	0.1103	56.90%	0.0025	0.0405	0.075	62.50%	0.0033	0.0668	0.075	66.70%	0.0034	0.0674	0.1027	58.10%	0.0024
ex3_2	0.0665	0.3724	13.80%	0.0005	0.0905	0.3812	12.50%	0.0012	0.0303	0.225	25.00%	0	0.0717	0.3743	13.50%	0.0005
ex3_3	0.078	0.3379	13.80%	0.0018	0.1014	0.2875	25.00%	0.0014	0.0295	0.225	25.00%	0.0016	0.083	0.327	16.20%	0.0015
ex3_4	0.0809	0.2552	27.60%	0.0033	0.1231	0.2687	25.00%	0.0026	0.0402	0.1583	25.00%	0.0026	0.09	0.2581	27.00%	0.0029

ex4	0.027	0.0475	84.50%	0	0.0311	0.0571	83.70%	0	0.0188	0.032	90.00%	0	0.0278	0.0494	84.30%	0
ex4_2	0.0705	0.3724	13.80%	0.0004	0.089	0.3812	12.50%	0.0032	0.0313	0.225	25.00%	0	0.0745	0.3743	13.50%	0.0005
ex4_3	0.0856	0.331	13.80%	0.002	0.0981	0.3125	18.80%	0.0035	0.0342	0.2333	16.70%	0.0022	0.0883	0.327	14.90%	0.002
ex4_4	0.0912	0.3069	12.10%	0.0028	0.1089	0.3312	18.80%	0.0059	0.0426	0.2583	16.70%	0.0023	0.0951	0.3122	13.50%	0.003
ex4_5	0.0943	0.2931	12.10%	0.0034	0.113	0.2875	18.80%	0.0057	0.0459	0.2167	16.70%	0.0029	0.0983	0.2919	13.50%	0.0034
ex5	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
ex6	0.0771	0.3724	13.80%	0.0005	0.0913	0.3812	12.50%	0.0032	0.0302	0.225	25.00%	0	0.0802	0.3743	13.50%	0.0006
ex6_2	0.0954	0.3448	13.80%	0.0017	0.1109	0.2938	18.80%	0.0042	0.0318	0.225	33.30%	0.0007	0.0987	0.3338	14.90%	0.0015
ex6_3	0.1072	0.3	15.50%	0.0029	0.1264	0.2813	18.80%	0.0066	0.0455	0.2167	33.30%	0.0016	0.1114	0.2959	16.20%	0.0029
ex6_4	0.1088	0.2655	22.40%	0.0041	0.1231	0.2375	18.80%	0.0068	0.0649	0.1667	41.70%	0.0024	0.1119	0.2595	21.60%	0.0039
ex7	0.0854	0.3379	8.60%	0.0022	0.1175	0.3	25.00%	0.001	0.0291	0.2333	16.70%	0.0019	0.0923	0.3297	12.20%	0.0017
ex7_2	0.1147	0.3672	12.10%	0.0018	0.1398	0.4125	12.50%	0.0029	0.0333	0.25	16.70%	0.0024	0.1202	0.377	12.20%	0.0017
ex7_2_1	0.1094	0.3603	10.30%	0.0019	0.1242	0.3375	12.50%	0.002	0.03	0.225	25.00%	0.0012	0.1126	0.3554	10.80%	0.0017
ex7_3	0.0911	0.3379	19.00%	0.0015	0.1136	0.3	12.50%	0.0014	0.0306	0.225	33.30%	0.0008	0.0959	0.3297	17.60%	0.0013
ex8_1	0.1145	0.3621	13.80%	0.0016	0.1262	0.3687	18.80%	0.0031	0.0305	0.2083	33.30%	0.0008	0.1171	0.3635	14.90%	0.0014
ex8_1_2	0.1458	0.3155	20.70%	0.0047	0.1548	0.325	18.80%	0.0072	0.0623	0.175	41.70%	0.003	0.1477	0.3176	20.30%	0.0047
ex8_1_3	0.1028	0.3441	14.40%	0.0007	0.1486	0.3667	13.90%	0.0042	0.038	0.2231	23.10%	0.0004	0.114	0.3497	14.30%	0.0008
ex8_2	0.113	0.3603	13.80%	0.0016	0.1257	0.3562	18.80%	0.0024	0.03	0.2083	33.30%	0.0008	0.1157	0.3595	14.90%	0.0014
ex8_2_1	0.0365	0.097	75.50%	0	0.0495	0.1143	73.50%	0	0.0102	0.046	84.00%	0	0.039	0.1004	75.10%	0
ex8_2_2	0.1013	0.3387	15.30%	0.0007	0.1422	0.3528	13.90%	0.004	0.0376	0.2231	23.10%	0.0005	0.1113	0.3422	15.00%	0.0008
ex8_2_3	0.1139	0.3162	15.30%	0.0021	0.1661	0.325	13.90%	0.0074	0.0464	0.2	19.20%	0.0015	0.1267	0.3184	15.00%	0.0022
ex8_2_4	0.0986	0.3369	14.40%	0.0007	0.1407	0.3639	13.90%	0.004	0.0391	0.2192	23.10%	0.0005	0.1089	0.3435	14.30%	0.0008
ex8_3	0.1108	0.3603	12.10%	0.0016	0.125	0.3687	18.80%	0.0024	0.034	0.2083	33.30%	0.0008	0.1138	0.3622	13.50%	0.0013
ex8_3_1	0.1187	0.331	13.80%	0.0034	0.1374	0.3438	25.00%	0.0079	0.0424	0.1917	33.30%	0.0029	0.1228	0.3338	16.20%	0.0034
ex8_3_2	0.1328	0.3017	20.70%	0.0049	0.1321	0.275	25.00%	0.008	0.0617	0.1583	50.00%	0.0029	0.1327	0.2959	21.60%	0.0048
ex8_4	0.0824	0.1448	44.80%	0.0037	0.0865	0.15	37.50%	0.0068	0.0485	0.075	66.70%	0.0021	0.0833	0.1459	43.20%	0.0037
ex9_1	0.1395	0.3603	12.10%	0.0036	0.1669	0.3812	12.50%	0.0087	0.0454	0.2083	33.30%	0.0029	0.1454	0.3649	12.20%	0.0039
ex9_2	0.1181	0.3759	10.30%	0.0016	0.1342	0.3875	12.50%	0.0047	0.03	0.225	25.00%	0.0009	0.1215	0.3784	10.80%	0.0015
ex10_1	0.076	0.2862	20.70%	0.0015	0.0949	0.2875	18.80%	0.0017	0.021	0.125	50.00%	0.0008	0.0801	0.2865	20.30%	0.0013
ex10_2	0.1219	0.3776	10.30%	0.0022	0.1296	0.3625	18.80%	0.0016	0.0332	0.2167	25.00%	0.0009	0.1235	0.3743	12.20%	0.0018
ex10_3	0.1205	0.3741	13.80%	0.0016	0.133	0.3812	18.80%	0.0026	0.0306	0.225	33.30%	0.0009	0.1232	0.3757	14.90%	0.0015
ex10_4	0.1219	0.3776	10.30%	0.0022	0.1296	0.3625	18.80%	0.0016	0.0332	0.2167	25.00%	0.0009	0.1235	0.3743	12.20%	0.0018

Ex11_10	0.1705	0.3523	13.50%	0.0083	0.2544	0.4	11.10%	0.0193	0.0956	0.2462	11.50%	0.0061	0.1911	0.3639	12.90%	0.0092
ex11_11	0.1569	0.3198	16.20%	0.0066	0.2422	0.3583	13.90%	0.0153	0.0904	0.2077	23.10%	0.0033	0.1778	0.3293	15.60%	0.0073
ex11_12	0.1487	0.2991	18.90%	0.0071	0.217	0.3583	8.30%	0.0223	0.0822	0.1731	30.80%	0.0065	0.1654	0.3136	16.30%	0.0081
ex11_13	0.0468	0.0568	76.60%	0.0028	0.0469	0.0361	75.00%	0.0066	0.0301	0.0385	76.90%	0.0028	0.0468	0.0517	76.20%	0.0032
ex11_14	0.0468	0.0568	76.60%	0.0028	0.0469	0.0361	75.00%	0.0066	0.0301	0.0385	76.90%	0.0028	0.0468	0.0517	76.20%	0.0032
ex11_15	0.1705	0.3523	13.50%	0.0083	0.2544	0.4	11.10%	0.0193	0.0956	0.2462	11.50%	0.0061	0.1911	0.3639	12.90%	0.0092
ex11_16	0.1157	0.245	26.10%	0.0036	0.1882	0.2611	25.00%	0.0091	0.0717	0.1731	34.60%	0.0014	0.1334	0.249	25.90%	0.0043
ex11_17	0.1705	0.3523	13.50%	0.0083	0.2544	0.4	11.10%	0.0193	0.0957	0.2462	11.50%	0.0061	0.1911	0.3639	12.90%	0.0092
ex11_18	0.1708	0.3468	16.20%	0.0096	0.2363	0.3611	11.10%	0.0199	0.0937	0.25	15.40%	0.0094	0.1869	0.3503	15.00%	0.0103
ex11_19	0.1746	0.355	16.20%	0.0097	0.2443	0.3667	11.10%	0.0199	0.097	0.2538	15.40%	0.0094	0.1917	0.3578	15.00%	0.0104
ex11_2	0.0882	0.1541	44.10%	0.0051	0.0774	0.0833	58.30%	0.0053	0.0487	0.0846	50.00%	0.0063	0.0856	0.1367	47.60%	0.005
ex11_20	0.1753	0.3586	15.30%	0.0099	0.241	0.3611	11.10%	0.02	0.0968	0.2654	11.50%	0.0098	0.1914	0.3592	14.30%	0.0105
ex11_21	0.1742	0.3577	15.30%	0.0098	0.2371	0.3583	13.90%	0.0201	0.098	0.2654	11.50%	0.0098	0.1896	0.3578	15.00%	0.0104
ex11_22	0.1749	0.3568	16.20%	0.0097	0.2447	0.3667	11.10%	0.0199	0.0973	0.2538	15.40%	0.0093	0.192	0.3592	15.00%	0.0104
ex11_23	0.1745	0.3559	15.30%	0.0097	0.2452	0.3694	11.10%	0.0199	0.0973	0.25	15.40%	0.0091	0.1918	0.3592	14.30%	0.0103
ex11_24	0.1726	0.3541	14.40%	0.0096	0.2447	0.3694	11.10%	0.0199	0.0956	0.2577	11.50%	0.0093	0.1903	0.3578	13.60%	0.0103
ex11_25	0.1718	0.3559	15.30%	0.0098	0.2367	0.3583	11.10%	0.0199	0.0952	0.2577	11.50%	0.0097	0.1877	0.3565	14.30%	0.0104
ex11_26	0.1728	0.3577	14.40%	0.0095	0.2463	0.3667	11.10%	0.0199	0.0959	0.2577	11.50%	0.009	0.1908	0.3599	13.60%	0.0102
ex11_27	0.1646	0.3486	13.50%	0.0089	0.2441	0.3667	11.10%	0.0192	0.0928	0.2423	11.50%	0.0074	0.1841	0.3531	12.90%	0.0096
ex11_28	0.1765	0.3604	15.30%	0.01	0.239	0.3556	13.90%	0.0201	0.0992	0.2692	11.50%	0.01	0.1918	0.3592	15.00%	0.0106
ex11_29	0.1701	0.3514	14.40%	0.0094	0.2427	0.3639	11.10%	0.0197	0.0951	0.25	11.50%	0.0089	0.1878	0.3544	13.60%	0.0101
ex11_3	0.0517	0.0775	68.50%	0.0022	0.0486	0.0556	75.00%	0.0027	0.0409	0.0885	65.40%	0.0015	0.0509	0.0721	70.10%	0.0022
ex11_4	0.0468	0.0568	76.60%	0.0028	0.0469	0.0361	75.00%	0.0066	0.0301	0.0385	76.90%	0.0028	0.0468	0.0517	76.20%	0.0032
ex11_5	0.0468	0.0568	76.60%	0.0028	0.0469	0.0361	75.00%	0.0066	0.0301	0.0385	76.90%	0.0028	0.0468	0.0517	76.20%	0.0032
ex11_6	0.1205	0.2532	28.80%	0.0036	0.1724	0.2389	27.80%	0.0078	0.0719	0.1538	46.20%	0.0017	0.1332	0.2497	28.60%	0.0041
ex11_7	0.1054	0.2153	32.40%	0.0029	0.1612	0.2194	36.10%	0.0047	0.0632	0.1231	46.20%	0.0013	0.1191	0.2163	33.30%	0.0031
ex11_8	0.1729	0.3505	15.30%	0.0093	0.2381	0.375	11.10%	0.0195	0.0989	0.25	15.40%	0.0085	0.1889	0.3565	14.30%	0.01
ex11_9	0.1721	0.3432	14.40%	0.0093	0.234	0.3667	13.90%	0.0194	0.0987	0.2462	15.40%	0.0081	0.1873	0.349	14.30%	0.0099
ex12_01	0.1821	0.3545	15.50%	0.0088	0.2249	0.3816	10.20%	0.0178	0.0946	0.24	20.00%	0.0051	0.1905	0.3598	14.50%	0.0096
ex12_02	0.1841	0.356	15.50%	0.009	0.2192	0.3714	10.20%	0.0177	0.095	0.244	18.00%	0.0056	0.191	0.359	14.50%	0.0098
ex13_01	0.1758	0.345	16.00%	0.008	0.2152	0.351	12.20%	0.0154	0.0911	0.224	26.00%	0.0039	0.1835	0.3462	15.30%	0.0087
ex13_02	0.1742	0.344	15.50%	0.0079	0.2125	0.3388	14.30%	0.0146	0.0901	0.226	24.00%	0.0039	0.1818	0.343	15.30%	0.0086

ex14_01	0.09	0.1995	32.00%	0.0018	0.1205	0.202	32.70%	0.0029	0.0489	0.134	42.00%	0.0007	0.096	0.2	32.10%	0.0018
ex14_02	0.1333	0.2945	24.50%	0.0034	0.1734	0.2388	26.50%	0.004	0.0678	0.198	36.00%	0.001	0.1412	0.2835	24.90%	0.0034
ex_15_1	0.1528	0.3414	12.10%	0.0066	0.1945	0.4062	12.50%	0.0101	0.0684	0.2417	16.70%	0.0063	0.1618	0.3554	12.20%	0.0065
ex_15_2	0.1593	0.3862	12.10%	0.0071	0.1967	0.4313	12.50%	0.0117	0.0714	0.2417	25.00%	0.0058	0.1674	0.3959	12.20%	0.0071
ex_15_3	0.1407	0.3103	17.20%	0.0062	0.1351	0.275	25.00%	0.007	0.0646	0.2	33.30%	0.0039	0.1395	0.3027	18.90%	0.0058
ex_16_1	0.1412	0.3379	13.80%	0.006	0.1843	0.3812	12.50%	0.0109	0.064	0.1833	25.00%	0.004	0.1505	0.3473	13.50%	0.0058
ex_16_2	0.1601	0.3897	12.10%	0.0072	0.1917	0.4188	12.50%	0.011	0.0727	0.25	25.00%	0.0058	0.1669	0.3959	12.20%	0.0071
ex_16_3	0.1408	0.3259	15.50%	0.0065	0.1524	0.3	25.00%	0.0095	0.0602	0.2	25.00%	0.0039	0.1433	0.3203	17.60%	0.0064
ex_17_1	0.1204	0.3448	12.10%	0.0034	0.1612	0.3812	12.50%	0.0089	0.0419	0.1833	25.00%	0.0023	0.1293	0.3527	12.20%	0.0037
ex_17_2	0.1364	0.3914	12.10%	0.0036	0.1652	0.4188	12.50%	0.0094	0.0498	0.25	25.00%	0.0031	0.1426	0.3973	12.20%	0.0039
ex_17_3	0.1226	0.3414	15.50%	0.0037	0.1313	0.2938	25.00%	0.0096	0.0437	0.2083	25.00%	0.0025	0.1245	0.3311	17.60%	0.0039
ex_18_1	0.1026	0.3638	12.10%	0.0017	0.1309	0.3875	12.50%	0.0056	0.0285	0.1917	25.00%	0.0009	0.1087	0.3689	12.20%	0.0016
ex_18_2	0.1159	0.3862	12.10%	0.0016	0.1304	0.4125	12.50%	0.0047	0.0352	0.2417	25.00%	0.0009	0.119	0.3919	12.20%	0.0016
ex_18_3	0.1018	0.3586	13.80%	0.002	0.1182	0.3125	18.80%	0.0039	0.029	0.2167	25.00%	0.0013	0.1053	0.3486	14.90%	0.0018
ex_19_1	0.1204	0.3448	12.10%	0.0034	0.1612	0.3812	12.50%	0.0089	0.0419	0.1833	25.00%	0.0023	0.1293	0.3527	12.20%	0.0037
ex_19_2	0.1377	0.3862	10.30%	0.0039	0.1653	0.4125	12.50%	0.0093	0.0503	0.25	16.70%	0.0037	0.1437	0.3919	10.80%	0.0042
ex_19_3	0.1376	0.3776	12.10%	0.0038	0.1586	0.3563	18.80%	0.007	0.0498	0.2417	16.70%	0.0034	0.1421	0.373	13.50%	0.004
ex_19_4	0.1191	0.3379	19.00%	0.0037	0.1256	0.2625	25.00%	0.007	0.0443	0.1917	25.00%	0.0026	0.1205	0.3216	20.30%	0.0038
ex_20_1	0.1026	0.3638	12.10%	0.0017	0.1309	0.3875	12.50%	0.0056	0.0285	0.1917	25.00%	0.0009	0.1087	0.3689	12.20%	0.0016
ex_20_2	0.1173	0.3776	10.30%	0.0019	0.1299	0.4063	12.50%	0.0043	0.0358	0.2417	16.70%	0.0015	0.1201	0.3838	10.80%	0.0018
ex_20_3	0.0992	0.3621	12.10%	0.002	0.1156	0.2938	18.80%	0.004	0.0308	0.2167	25.00%	0.0011	0.1027	0.3473	13.50%	0.0018
ex_21_1	0.1026	0.3638	12.10%	0.0017	0.1309	0.3875	12.50%	0.0056	0.0285	0.1917	25.00%	0.0009	0.1087	0.3689	12.20%	0.0016
ex_21_2	0.1143	0.3793	8.60%	0.002	0.1288	0.375	12.50%	0.0047	0.0312	0.2167	8.30%	0.0023	0.1175	0.3784	9.50%	0.0019
ex_21_3	0.1032	0.3569	12.10%	0.0027	0.1202	0.35	18.80%	0.0025	0.0358	0.225	25.00%	0.0014	0.1069	0.3554	13.50%	0.0024
ex_22_1	0.136	0.3759	12.10%	0.0035	0.1656	0.4	12.50%	0.0085	0.0479	0.2417	25.00%	0.003	0.1424	0.3811	12.20%	0.0039
ex_22_2	0.182	0.3555	15.50%	0.009	0.2246	0.349	12.20%	0.018	0.0938	0.238	26.00%	0.0053	0.1904	0.3542	14.90%	0.0098

9.3 FIS Experiments Details

9.3.1 Experiment 1

FIS Input: Lucene Similarity score (LScore)

Training data: Executed odd numbered queries from 04.testset, first 1000 docs for each query ordered by LScore.

Testing data: Executed odd numbered queries from 04.testset, first 1000 docs for each query ordered by LScore.

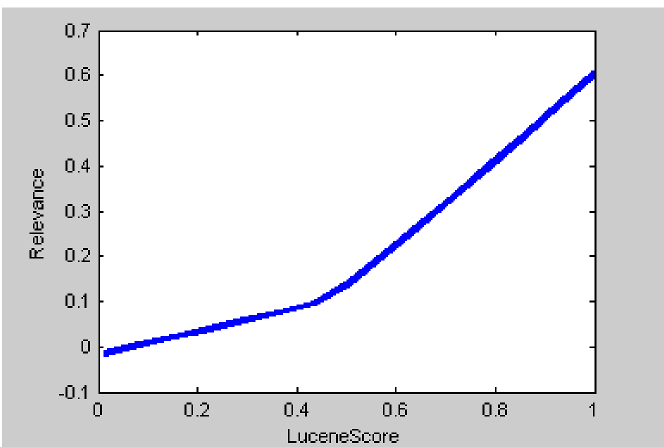


Figure 30

From the rule surface view it is clear that FIS is not changing the order of the documents retrieved by Lucene. This result is somewhat expected since the only input to the system is Lucene's score. An interesting thing to notice would be that around LuceneScore = 0.4 the slope changes; this implicates that the probability of the document being correct is increased after that point, however increase appears to be linear.

9.3.2 Experiment 2.1

Input: Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf (7 inputs).

Training data: odd numbered queries from 04.testset, first 10 docs for each query (in order to maximize P10), queries that have 3 terms.

Testing data: even numbered queries from 04.testset first 1000 docs for each query, queries that have 3 terms.

FIS: Generated with ANFIS, Grid Partition, 3 membership functions per input/output, output mf type – constant.

FIS Info:

Number of nodes: 4426

Number of linear parameters: 2187

Number of nonlinear parameters: 63

Total number of parameters: 2250

Number of training data pairs: 730

Number of checking data pairs: 0

Number of fuzzy rules: 2187

This run has revealed number of challenges that application of ANFIS to IR approach possesses. FIS complexity is exponential in relation to number of terms, inputs, membership functions. Training data size is too small for the FIS of this complexity.

Performed not well in comparison with Lucene. Small training set could have affected performance (number of training data points is smaller than number of modifiable parameters).

9.3.3 Experiment 2.2

Input:	Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data:	odd numbered queries from 04.testset (topics 301-700), first 10 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data:	even numbered queries from 04.testset **first 10 docs for each query**, queries that have 3 terms.

Performance has significantly improved and is getting closer to that of Lucene: P@5 0.4216
ANFIS appears to be much more effective in the top x answer subset refinement; this result is somewhat expected since the ANFIS is trained on the top x so it is likely to perform better on the items that are similar to those on which it has trained and those items are usually located in the top x proximity.

9.3.4 Experiment 2_3

Input:	Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data:	odd numbered queries from 04.testset (topics 301-700), first 20 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data:	even numbered queries from 04.testset first 20 docs for each query, queries that have 3 terms.

9.3.5 Experiment 3_1

Input:	Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data:	odd numbered queries from 04.testset (topics 301-700), first 20 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data:	even numbered queries from 04.testset first 1000 docs for each query, queries that have 3 terms.

Performed slightly worse then experiment 2.

9.3.6 Experiment 3_2

Input:	Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data:	odd numbered queries from 04.testset (topics 301-700), first 20 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data:	even numbered queries from 04.testset first 10 docs for each query, queries that have 3 terms.

9.3.7 Experiment 3_3

Input:	Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data:	odd numbered queries from 04.testset (topics 301-700), first 20 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data:	even numbered queries from 04.testset first 20 docs for each query, queries that have 3 terms.

9.3.8 Experiment 3_4

Input:	Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data:	odd numbered queries from 04.testset (topics 301-700), first 40 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data:	even numbered queries from 04.testset first 20 docs for each query, queries that have 3 terms.

9.3.9 Experiment 4_1

Input:	Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data:	odd numbered queries from 04.testset (topics 301-700), first 40 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data:	even numbered queries from 04.testset first 1000 docs for each query, queries that have 3 terms.

9.3.10 Experiment 4_2

Input:	Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data:	odd numbered queries from 04.testset (topics 301-700), first 40 docs for each query (in order to maximize P10), queries that have 3 terms

Testing data: even numbered queries from 04.testset first 10 docs for each query, queries that have 3 terms.

9.3.11 Experiment 4_3

Input: Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first 40 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data: even numbered queries from 04.testset first 20 docs for each query, queries that have 3 terms.

9.3.12 Experiment 4_4

Input: Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first 40 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data: even numbered queries from 04.testset first 30 docs for each query, queries that have 3 terms.

9.3.13 Experiment 4_5

Input: Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first 40 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data: even numbered queries from 04.testset first 40 docs for each query, queries that have 3 terms.

9.3.14 Experiment 5_1

Input: Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first 1000 docs for each query (in order to maximize P10), queries that have 3 terms
Outcome: Run for 24 hours, but not even 1st epoch was finished, abandoned this test run for now. Pentium 4 1.8 Ghz, 512 MB Ram.

9.3.15 Experiment 6_1

Input: Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first 100 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data: even numbered queries from 04.testset first 10 docs for each query, queries that have 3 terms.

9.3.16 Experiment 6_2

Input: Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first 100 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data: even numbered queries from 04.testset first 20 docs for each query, queries that have 3 terms.

9.3.17 Experiment 6_3

Input: Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first 100 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data: even numbered queries from 04.testset first 40 docs for each query, queries that have 3 terms.

9.3.18 Experiment 6_4

Input: Lucene's score, for each t of q - TF Normalized (tf *Lucene.LengthNorm), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first 100 docs for each query (in order to maximize P10), queries that have 3 terms
Testing data: even numbered queries from 04.testset first 100 docs for each query, queries that have 3 terms.

9.3.19 Experiment 7_1

Input: Lucene's score, for each t of q - **TF Normalized (tf/terms(doc)**), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first **40** docs for each query (in order to maximize P10), queries that have 3 terms
FIS **Grid** Partition

Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.3.20 Experiment 7_2

Input: Lucene's score, for each t of q - TF Normalized (tf/terms(doc)), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first **40 docs** for each query (in order to maximize P10), queries that have 3 terms
FIS ANFIS: **Substructive** clustering **3 mfs**; hybrid training, **3 epochs**
Testing data: even numbered queries from 04.testset first **20 docs** for each query, queries that have 3 terms.

9.3.21 Experiment 7_2_1

Input: Lucene's score, for each t of q - TF Normalized (tf/terms(doc)), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first **20 docs** for each query (in order to maximize P10), queries that have 3 terms
Checking data: even numbered queries from 04.testset (topics 301-700), first 40 docs for each query
FIS ANFIS: Substructive clustering **2 mfs**; hybrid training, **100 epochs**
Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.3.22 Experiment 7_3

Input: Lucene's score, for each t of q - TF Normalized (tf/terms(doc)), idf
Training data: odd numbered queries from 04.testset (topics 301-700), first **40** docs for each query (in order to maximize P10), queries that have 3 terms
2 mfs
Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.4 Knowledge Based R-FIS Experiments

9.4.1 Experiment 8_1

Input: Lucene's score, for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

Comments:
Even though this results are not very different from Lucene; but the system is quite different. It is a very simple FIS with only two rules of the form:

- 1. If (LScore is high) and (tf1 is high) and (idf1 is rare) and (tf2 is high) and (idf2 is rare) and (tf3 is high) and (idf3 is rare) then (score is high) (1)
- 2. If (LScore is not high) and (tf1 is not high) and (idf1 is not rare) and (tf2 is not high) and (idf2 is not rare) and (tf3 is not high) and (idf3 is not rare) then (score is not high) (1)

9.4.2 Experiment 8_1_1

Input: Lucene's score, for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: even numbered queries from 04.testset first **40** docs for each query, queries that have 3 terms.

9.4.3 Experiment 8_1_2

Input: Lucene's score, for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: even numbered queries from 04.testset first **100** docs for each query, queries that have 3 terms.

9.4.4 Experiment 8_1_3

Input: Lucene's score, for each t of q - TF Normalized (tf/terms(doc)), idf

FIS: Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: **all** queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.4.5 Experiment 8_2

Input: **No Lucene's score**, for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

Comments:
Surprisingly taking out Lucene's score as input did not significantly affect performance of the system, and in fact made it a little bit better for P@5

9.4.6 Experiment 8_2_1

Input: **No Lucene's score**, for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: even numbered queries from 04.testset first **40** docs for each query, queries that have 3 terms.

9.4.7 Experiment 8_2_2

Input: **No Lucene's score**, for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: **all** queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.4.8 Experiment 8_2_3

Input: **No Lucene's score**, for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: **all** queries from 04.testset first **40** docs for each query, queries that have 3 terms.

9.4.9 Experiment 8_2_4

Input: **No Lucene's score**, for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed 2 simple obvious rules high-> high; low->low; **adjusted mf ranges**
Testing data: **all** even numbered queries from 04.testset first **40** docs for each query, queries that have 3 terms.

9.4.10 Experiment 8_3

Input: For each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Tweaked mf functions to pick at the end of range
Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: even numbered queries from 04.testset first 20 docs for each query, queries that have 3 terms.

9.4.11 Experiment 8_3_1

Experiment 8_3_1
Input: For each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Tweaked mf functions to pick at the end of range
Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: even numbered queries from 04.testset first **40** docs for each query, queries that have 3 terms.

9.4.12 Experiment 8_3_2

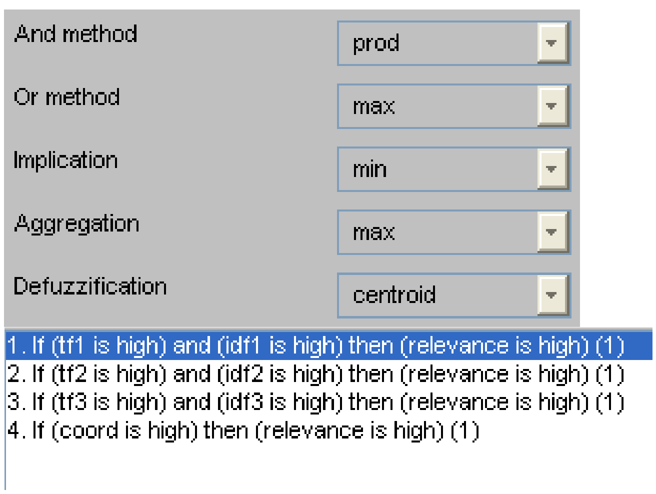
Input: For each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Tweaked mf functions to pick at the end of range
Manually constructed 2 simple obvious rules high-> high; low->low
Testing data: even numbered queries from 04.testset first **100** docs for each query, queries that have 3 terms.

9.4.13 Experiment 8_4

Input: For each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Tweaked mf functions to pick at the end of range
Manually constructed **5** simple obvious rules high-> high; low->low
Testing data: even numbered queries from 04.testset first **100** docs for each query, queries that have 3 terms.

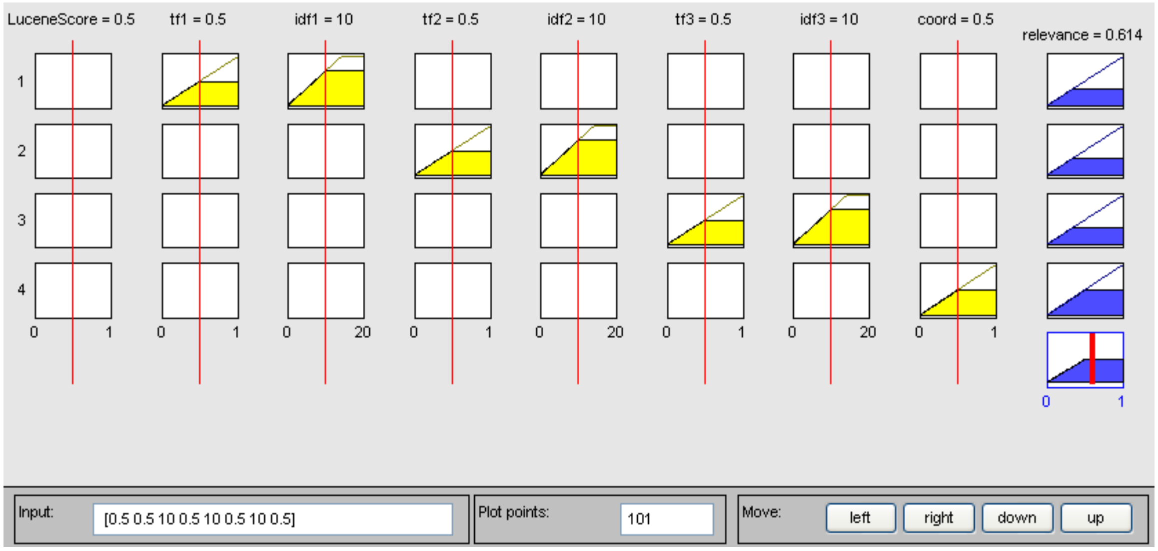
9.4.14 Experiment 11_1

Input:
FIS:
Testing data: queries w/ 3 terms (1000)



Data:
0.670000000000
0.670000000000
0.670000000000
………………..
0.670000000000
0.67000000000
0.641601285324
…………………
0.641601285324

Due to Aggregation = max; coord was dominating the scoring since usually its value is the largest.



9.4.15 Experiment 11_2

Aggregation = sum

9.4.16 Experiment 11_3

coord rule weight = 0

9.4.17 Experiment 11_4

Implication = prod

9.4.18 Experiment 11_5

Aggregation: sum
Scale weights for the rules (0.33) to add to 1.

9.4.19 Experiment 11_6

Add the not rules

Conclusion: not rules have improved it by quite a bit.

9.4.20 Experiment 11_7

And method = min

9.4.21 Experiment 11_8

Activate coord rules with w=0.1

9.4.22 Experiment 11_9

w(coord rules) = 0.2

9.4.23 Experiment 11_10

w(coord rules) = 0.05

9.4.24 Experiment 11_11

w(coord rules) = 0.025

9.4.25 Experiment 11_12

Defuzification = bisector

9.4.26 Experiment 11_13

Defuzification = mom

9.4.27 Experiment 11_14

Defuzification = som

9.4.28 Experiment 11_15

Output mf type = gbell

9.4.29 Experiment 11_16

input mf type = sigm

Note: sigm is much harder to tune correctly – is appropriate for machine learning application.

9.4.30 Experiment 11_17

Based on ex11_10
tf1, idf1, relevance, have 2 mfs (high; low) symmetrical.
Rule changed to if low(tf1) & low(idf1) -> low(relevance)

Note: As expected adding inversely symmetrical mf has exactly the same effect as adding appropriate not rules. Adding not rules is more simple; however if opposing function is not inversely symmetrical a separate mf would need to be added.

9.4.31 Experiment 11_18

Based on 11_16: tuned tf mf to stretch monotonously.

Note: Almost as good as linear mf.

9.4.32 Experiment 11_19

Centered tf mf at 0.5 and then stretched to 0.1

9.4.33 Experiment 11_20

Centered tf mf at 0.5 and then stretched to **0.2**

9.4.34 Experiment 11_21

Centered tf mf at 0.5 and then stretched to 0.3

9.4.35 Experiment 11_22

Centered tf mf at 0.5 and then stretched to 0.05

9.4.36 Experiment 11_23

Centered tf mf at 0.5 and then stretched to 0

9.4.37 Experiment 11_24

Centered tf mf at 0.4 and then stretched to 0.1

9.4.38 Experiment 11_25

Centered tf mf at 0.4 and then stretched to 0.2

9.4.39 Experiment 11_26

Centered tf mf at 0.4 and then stretched to 0.05

9.4.40 Experiment 11_27

Centered tf mf at 0.3 and then stretched to 0.05

9.4.41 Experiment 11_28

Centered tf mf at 0.7 and then stretched to 0.05

9.4.42 Experiment 11_29

Centered tf mf at 0.7 and then stretched to

9.4.43 Experiment 12_1

Based on 11_10; added handling for 4 terms; title run full.

If terms < 4; then tf, idf = 0 of non existing terms

9.4.44 Experiment 12_2

Stretch idf function completely

9.4.45 Experiment 13_1

title-run with normalized idf (3 terms)

9.4.46 Experiment 13_2

title-run with normalized idf (3 terms); gausian idf mf

9.4.47 Experiment 14_1

description-run

Note: This performance is quite bad; possible reasons coord value is fixed but terms are not.

9.4.48 Experiment 14_2

Remove coord rule

9.5 Training Knowledge Based FIS with ANFIS

9.5.1 Experiment 9_1

Input:	Lucene Score, and for each t of q - TF Normalized (tf/terms(doc)), idf
FIS:	Manually constructed **2** simple rules; converted to Sugeno and trained with ANFIS.
Testing data:	even numbered queries from 04.testset first **40** docs for each query, queries that have 3 terms.

9.5.2 Experiment 9_2

Input:	Lucene Score, and for each t of q - TF Normalized (tf/terms(doc)), idf
FIS:	Manually constructed **2** simple rules; converted to Sugeno and trained with ANFIS.

Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.5.3 Experiment 10_1

Input: **No Lucene Score**, and for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed **4** simple rules; **madami**.
Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.5.4 Experiment 10_2

Input: **No Lucene Score**, and for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed **4** simple rules; **sugeno w/ Anfis**
Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.5.5 Experiment 10_3

Input: **Lucene Score**, and for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed **5** simple rules; **sugeno w/ Anfis**
Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.5.6 Experiment 10_4

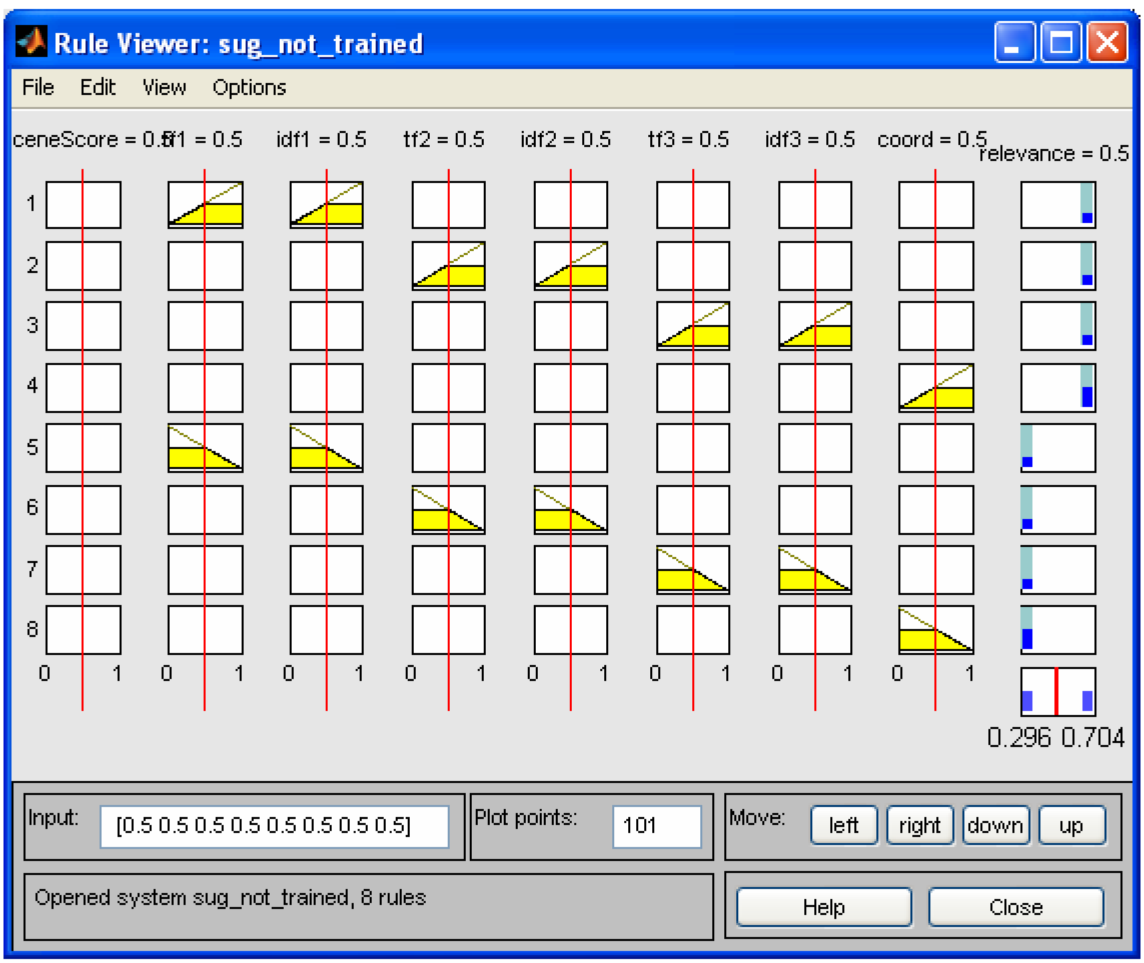
Input: **Lucene Score**, and for each t of q - TF Normalized (tf/terms(doc)), idf
FIS: Manually constructed **4** simple rules; **sugeno w/ Anfis; took out rule for LScore separately.**
Testing data: even numbered queries from 04.testset first **20** docs for each query, queries that have 3 terms.

9.5.7 Experiment 15_1

Train one of the best manual FIS with ANFIS. Based on 13_01 not the best one, 12_02 is; but 13_1 has a normalized idf which is more desirable. Removed tf4, idf4 since running only with 3 terms (simpler).

Definitions: x / y where x – training size; y – testing size.

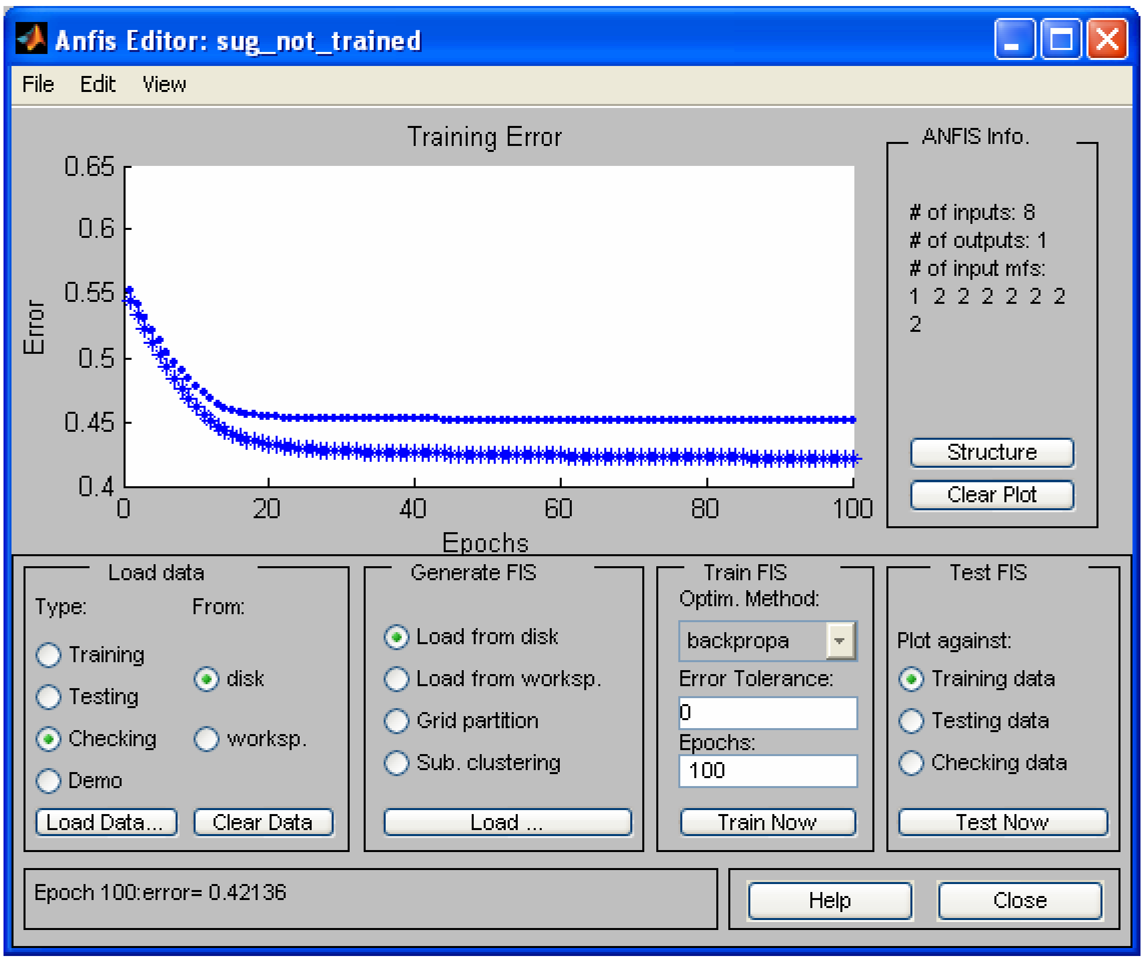
Sugeno converted from mam fis. This is a baseline run; in order to see how much actual improvement was done; since certain things changed during conversion from mam -> sug; weights were reset to 1.

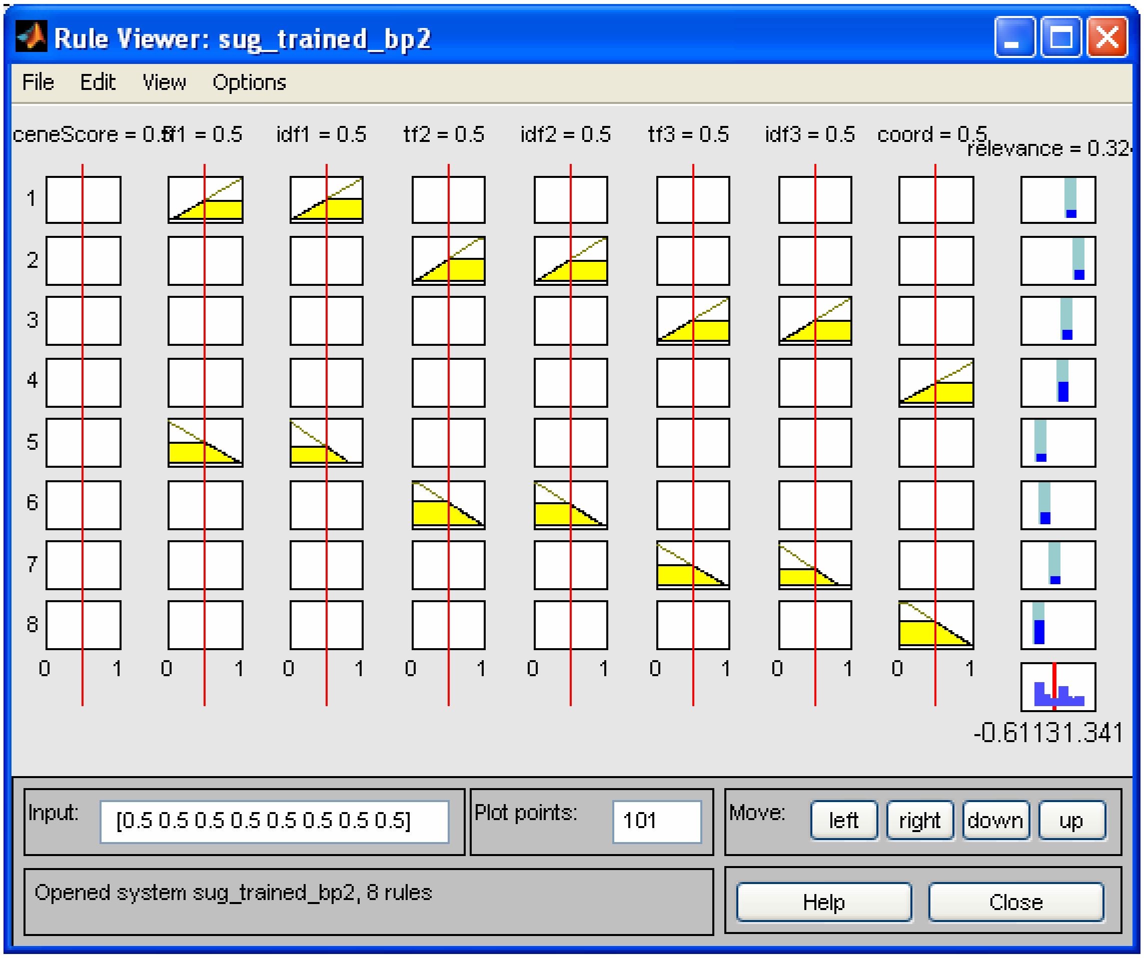


9.5.8 Experiment 15_2

100/100

Back prop 100 epochs

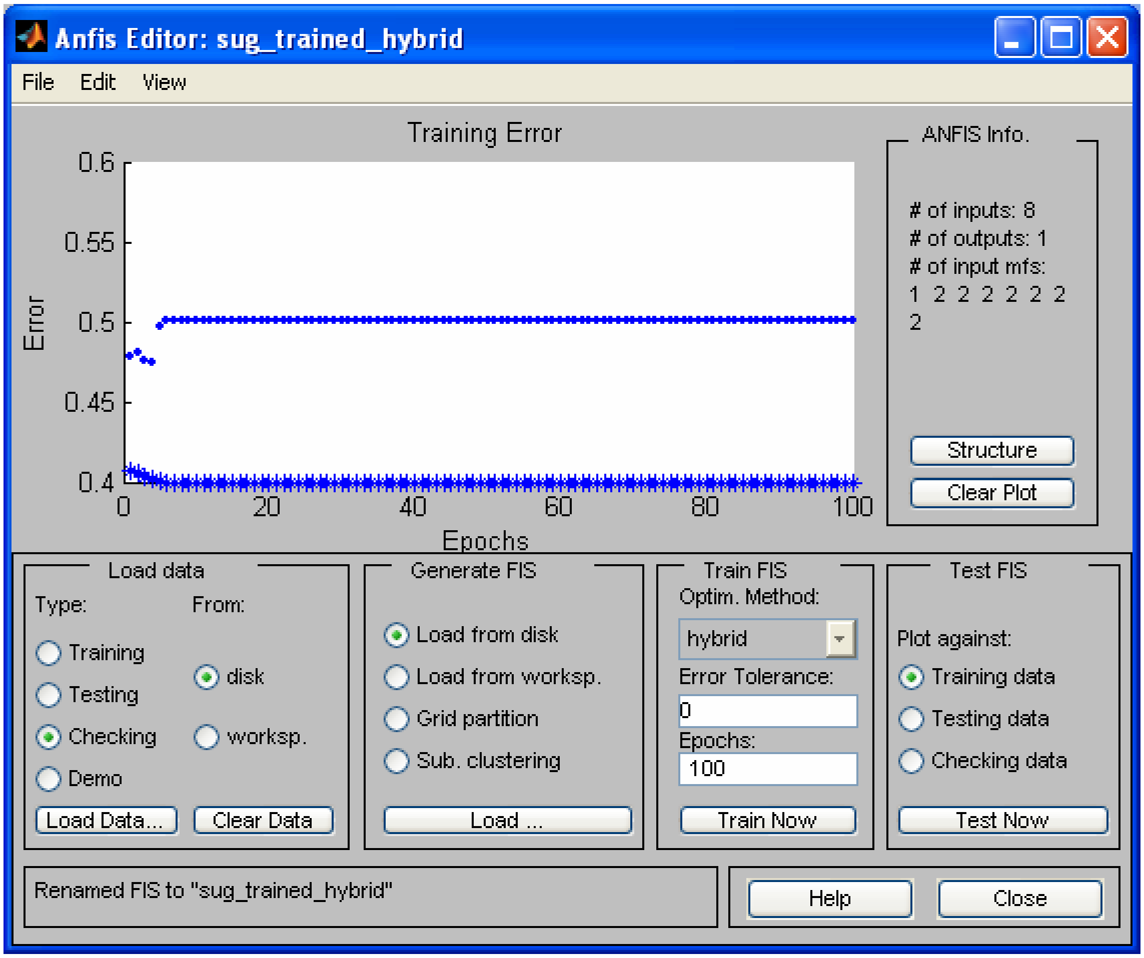


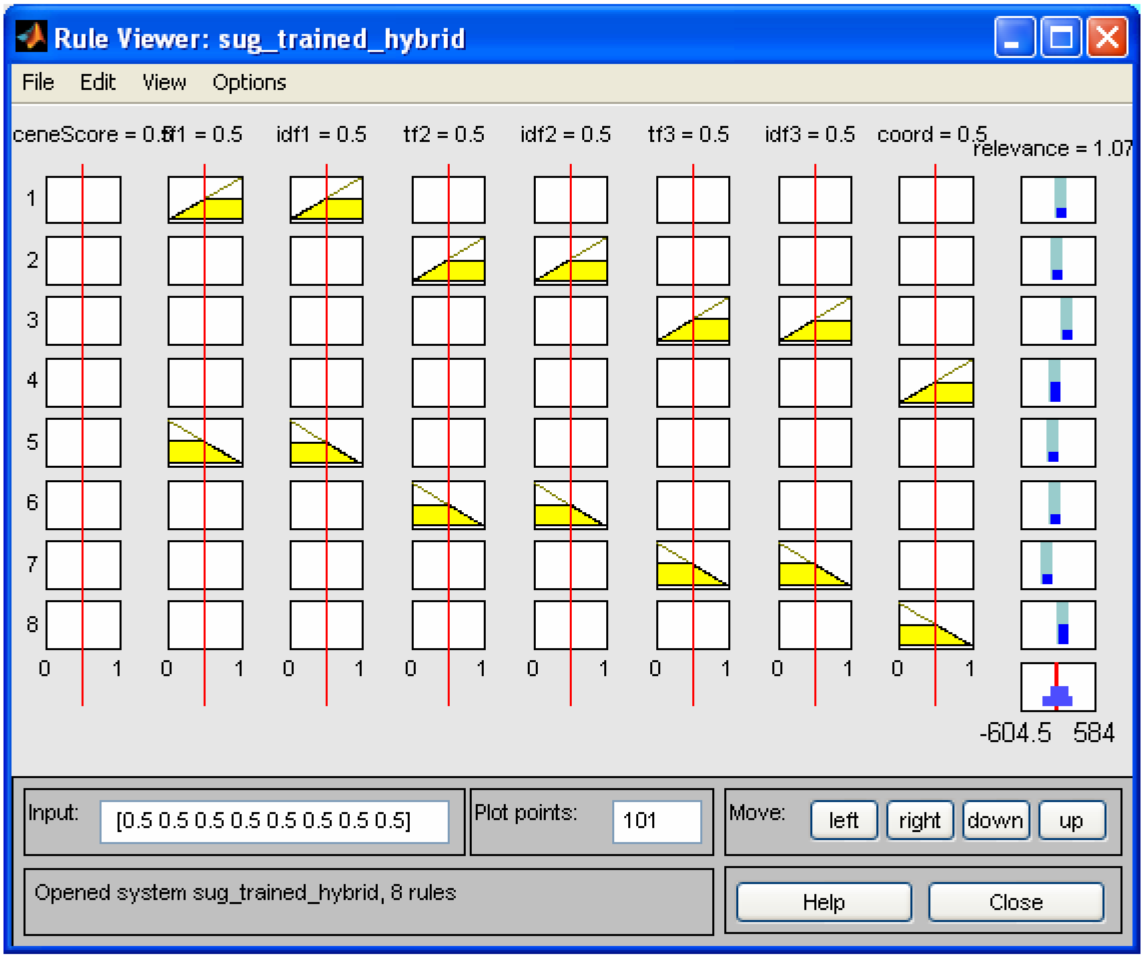


9.5.9 Experiment 15_3

100/100

hybrid 100 epochs


Anfis Editor: sug_trained_hybrid
File Edit View
Training Error
0.6
0.55
0.5
0.45
0.4
Error
0 20 40 60 80 100
Epochs
ANFIS Info.
# of inputs: 8
# of outputs: 1
# of input mfs:
1 2 2 2 2 2 2
2
Structure
Clear Plot
Load data
Type:
From:
Training
Testing
Checking
Demo
disk
worksp.
Load Data...
Clear Data
Generate FIS
Load from disk
Load from worksp.
Grid partition
Sub. clustering
Load ...
Train FIS
Optim. Method:
hybrid
Error Tolerance:
0
Epochs:
100
Train Now
Test FIS
Plot against:
Training data
Testing data
Checking data
Test Now
Renamed FIS to "sug_trained_hybrid"
Help
Close



9.5.10 Experiment 16_1

100/100

Problem discovered ANFIS sets all the weights of the rules to 1 which throws off balance between coord and tfidf; so the training does not go well.

Fix weight of coord and rerun experiments

9.5.11 Experiment 16_2

Bp 100/100

9.5.12 Experiment 16_3

hybrid 100/100

9.5.13 Experiment 17_1

baseline 100/40

9.5.14 Experiment 17_2

100/40

Bp a little bit better then 100/100

9.5.15 Experiment 17_3

100/40

hybrid

9.5.16 Experiment 18_1

baseline 100/20

9.5.17 Experiment 18_2

Bp 100/20

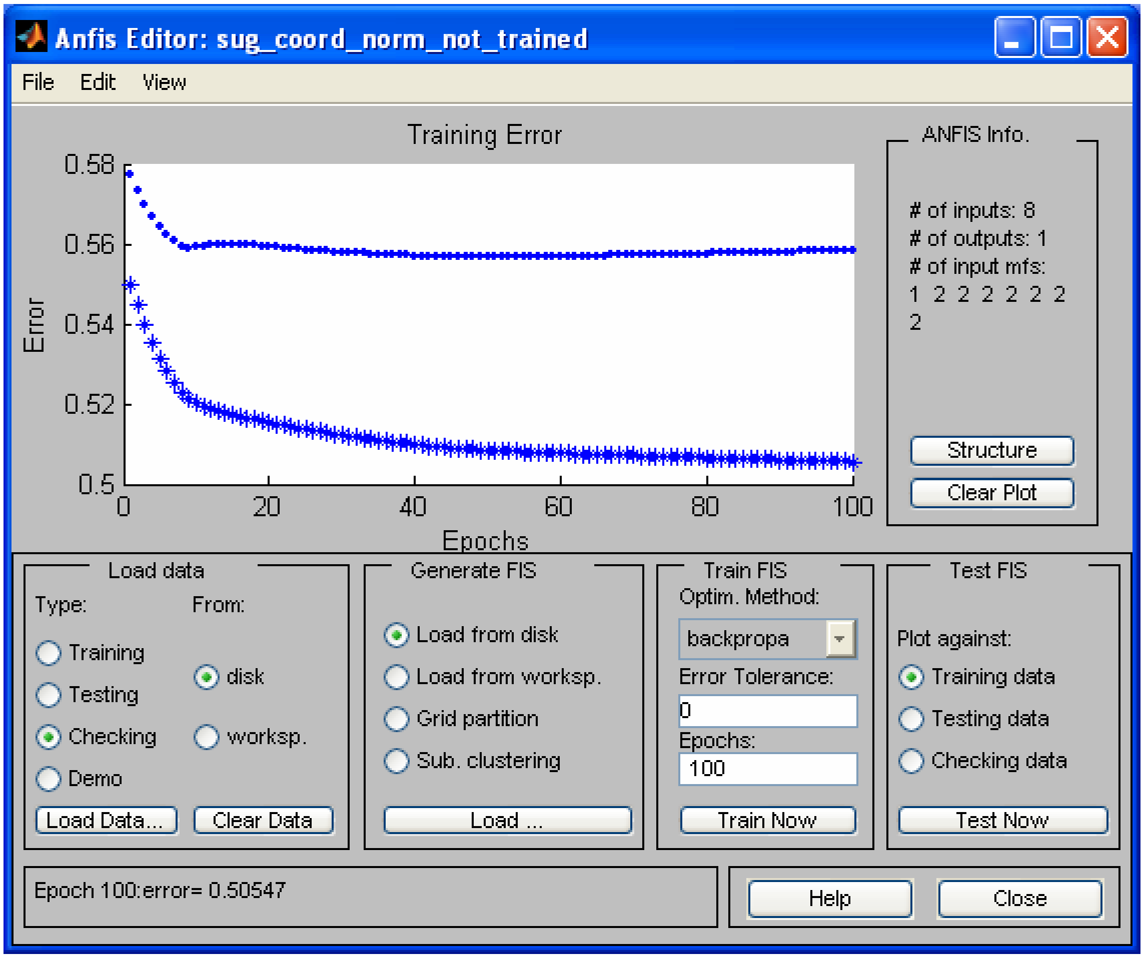
9.5.18 Experiment 18_3

Hybrid 100/20

9.5.19 Experiment 19_1

40/40

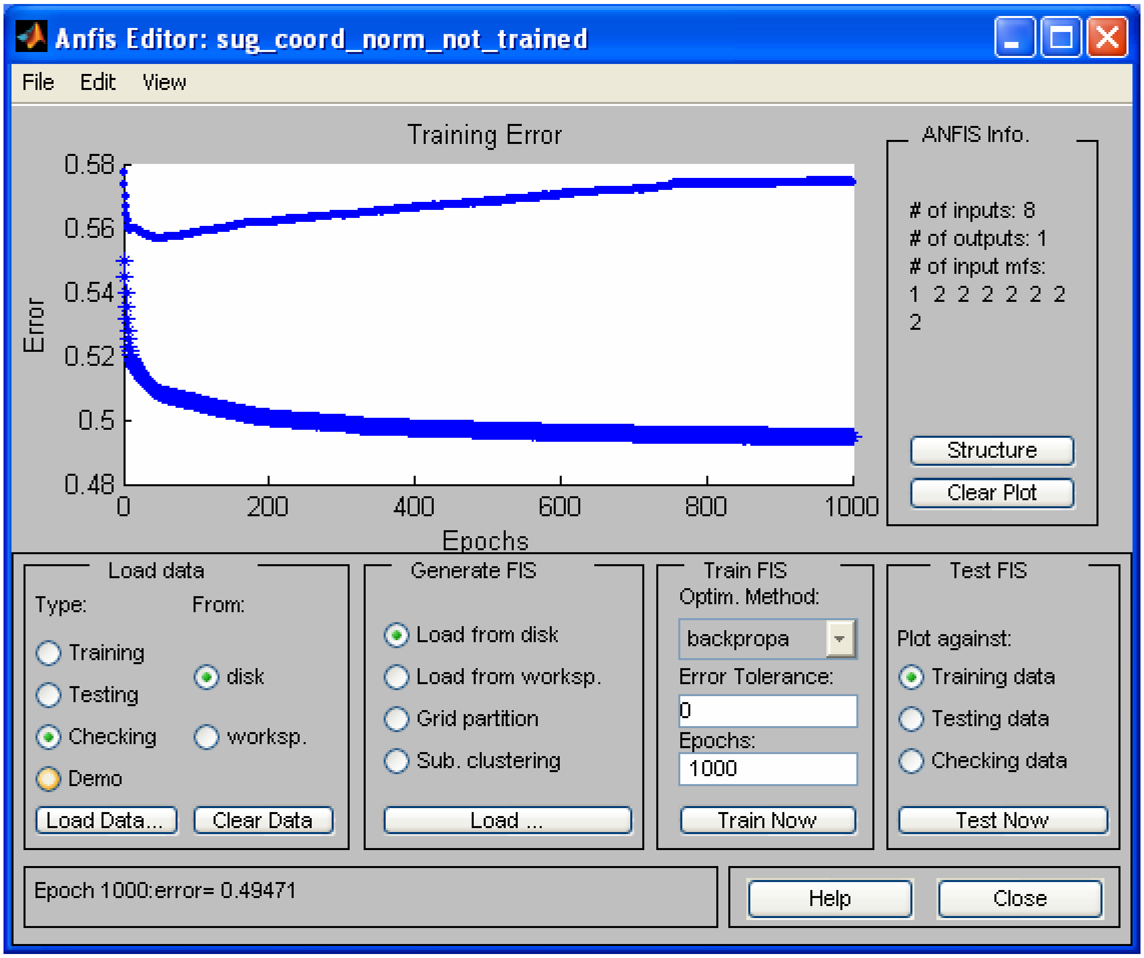
Bp 100



9.5.20 Experiment 19_2

40/40

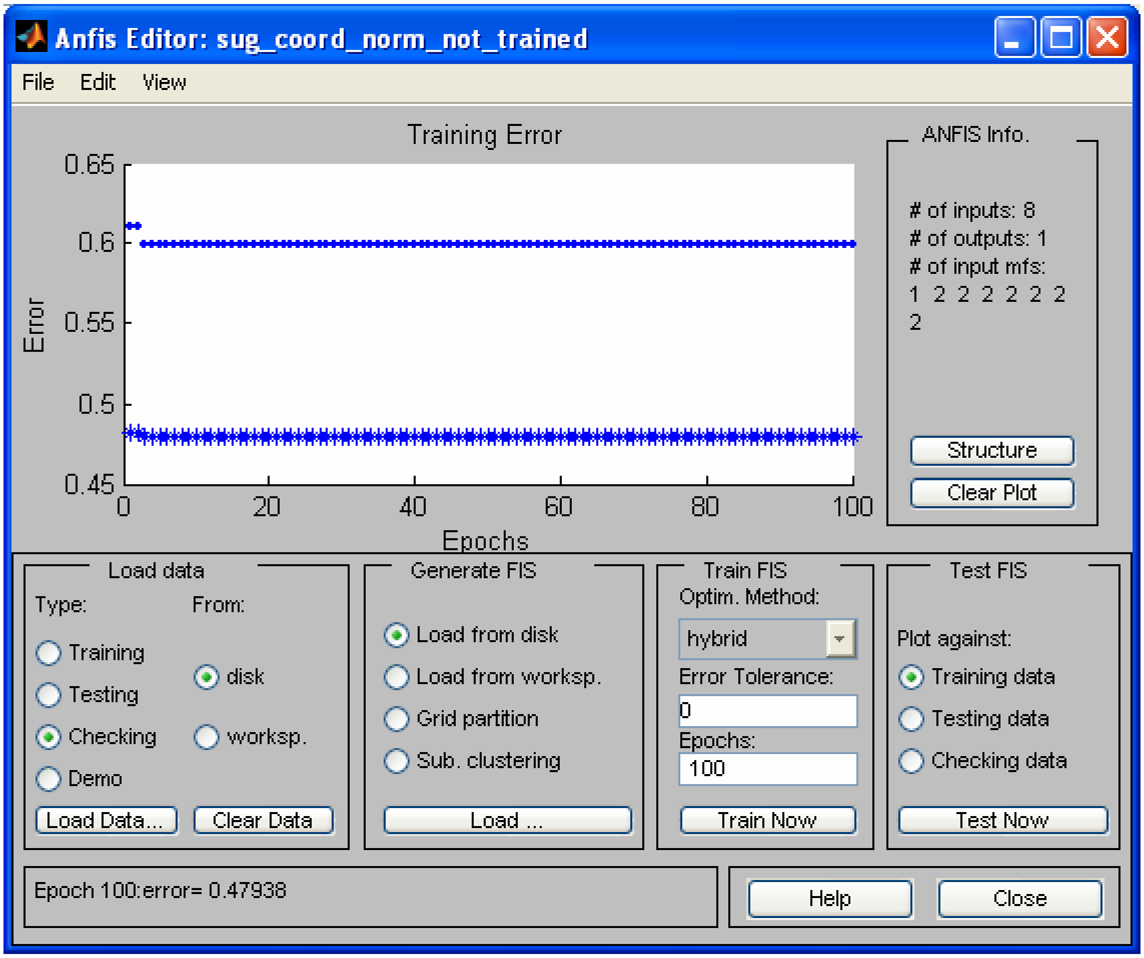
Bp 1000



9.5.21 Experiment 19_3

40/40

hybrid 100



9.5.22 Experiment 20_1

40/20

Baseline

9.5.23 Experiment 20_2

40/20

BP

9.5.24 Experiment 20_3

40/20

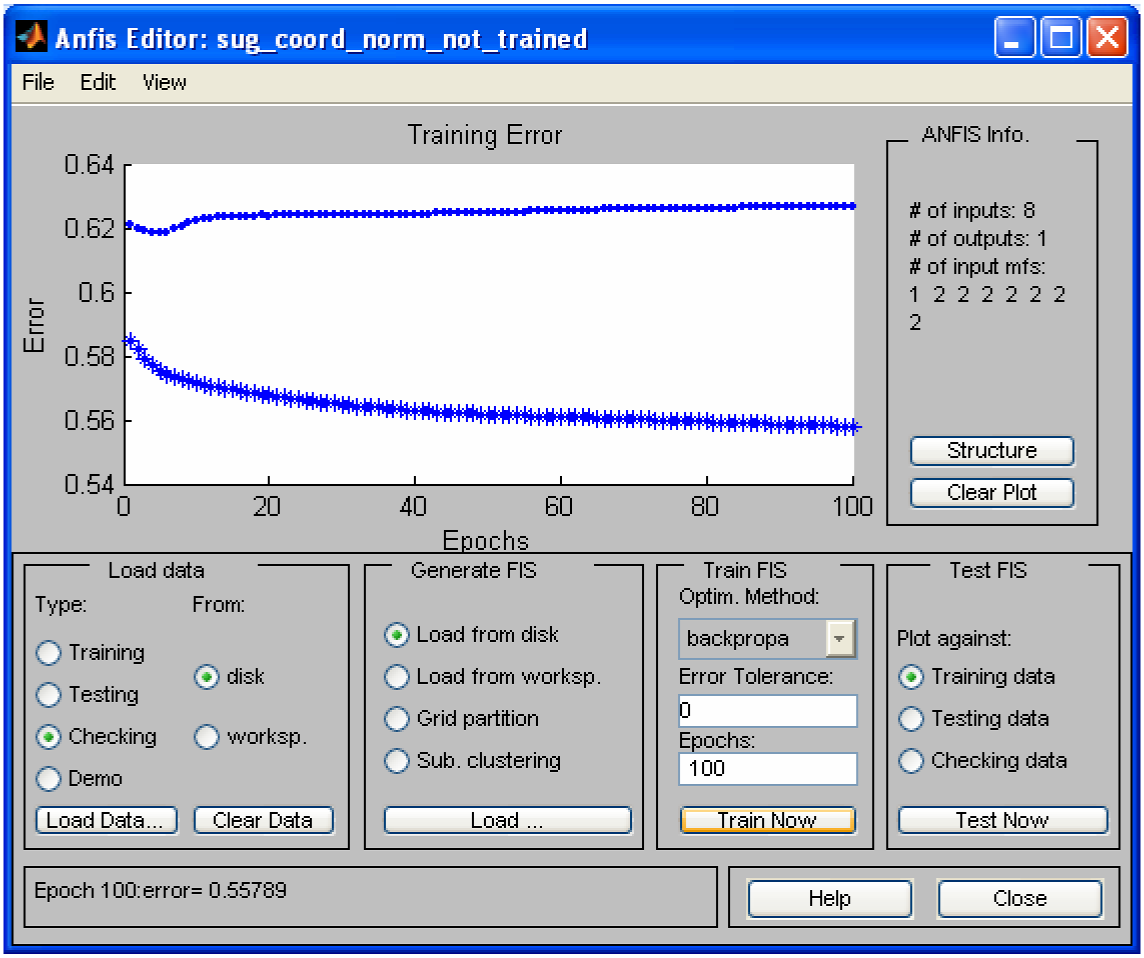
Hybrid

9.5.25 Experiment 21_1

base 20/20

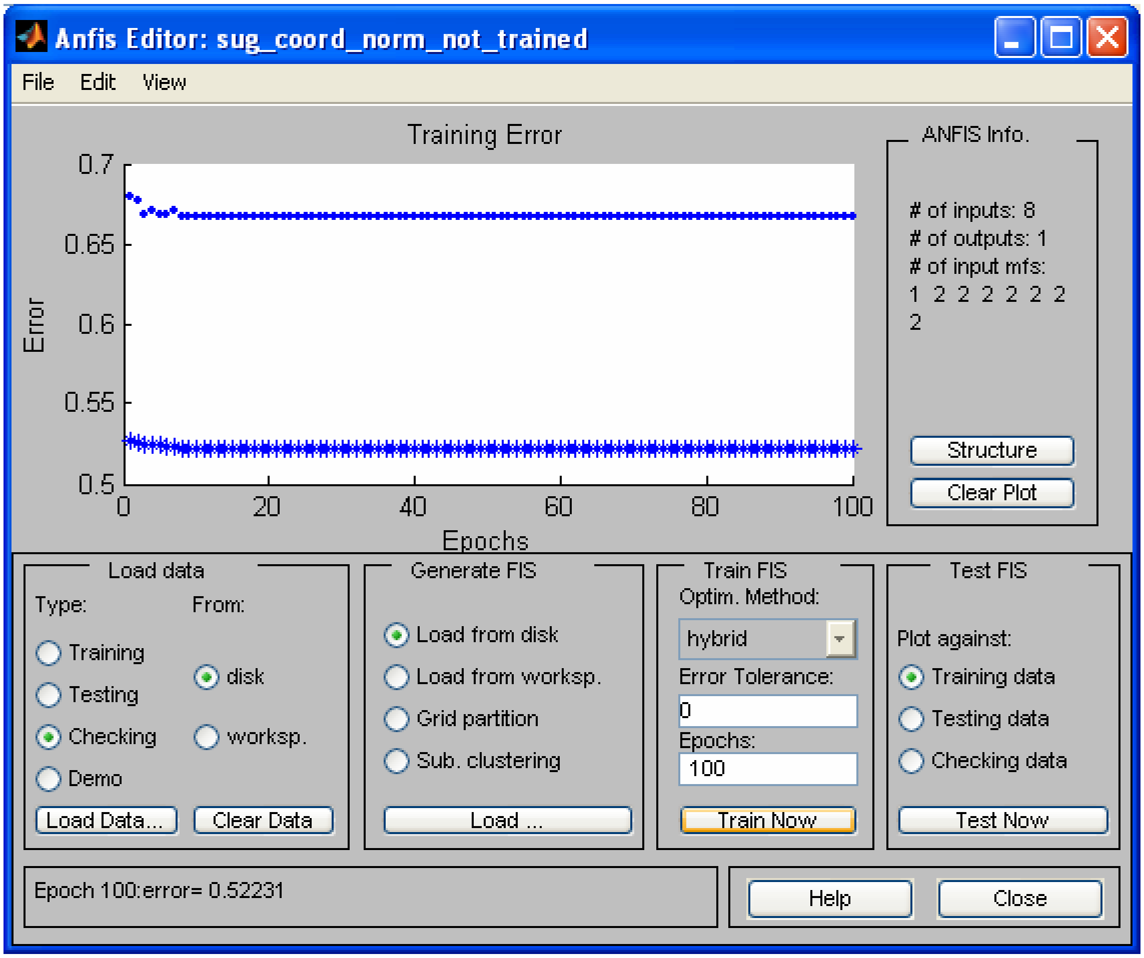
9.5.26 Experiment 21_2

20/20



9.5.27 Experiment 21_3

20/20



9.6 Experiments - F-TFIDF Construction – Merging ANFIS / Knowledge Based R-FIS / Lucene result sets

9.6.1 Experiment 22_1

Anfis 17_2 w/ Lucene

9.6.2 Experiment 22_2

13_01 mam; luc

10 References

[1] G. Amati, C. Carpineto, and G. Romano. Fondazione Ugo Bordoni at TREC 2004. In TREC 13, 2004.

[2] R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. ACM Press, New York, 1999.

[3] G. Clare and K. Hendrikse. The NexTrieve Search System in TREC 2001. NIST Special Publication 500-250: The Tenth Text REtrieval Conference, 2001.

[4] Fuzzy Logic Toolbox User's Guide. The MathWorks Inc., Jun 2004.

[5] Y. Horng, S. Chen, and C. Lee. A fuzzy information retrieval method using fuzzy-valued concept networks. Tools with Artificial Intelligence, 1998. Proceedings. Tenth IEEE International Conference, Nov. 1998.

[6] Jakarta Lucene 1.4.final API. http://jakarta.apache.org/lucene/docs/api/index.html, 2004.

[7] K.L. Kwok, L. Grunfeld, H.L. Sun, P. Deng and N. Dinstl. Queens College, CUNY, TREC 2004 Robust Track Experiments Using PIRCS. In TREC 13, 2004.

[8] S. Liu, C. Sun, C. Yu. University of Illinois at Chicago at TREC 2004: Robust Track. In TREC 13, 2004.

[9] Y. Ogawa, T. Morita, and K. Kobayashi. A fuzzy document retrieval system using the keyword connection matrix and a learning method. Fuzzy Sets and Systems, 39:163-179, 1991.

[10] V. Plachouras, B. He, and I. Ounis. University of Glasgow at TREC 2004:Experiments in Web, Robust, and Terabyte Tracks with Terrier. In TREC 13, 2004.

[11] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-4. In the Proceedings of the Fourth Text Retrieval Conference, D. K. Harman, Ed., 1996, pp. 73-97.

[12] P. Subtil, N. Mouaddib and O. Foucaut. A fuzzy information retrieval and management system and its applications. In Proceedings of the 1996 ACM symposium on Applied Computing, Feb 1996.

[13] M. Sugeno. Industrial applications of fuzzy control. Elsevier Science Pub. Co., 1985.

[14] E.M. Voorhees. Overview of the TREC 2004 Robust Retrieval Track. In TREC 13, 2004.

[15] E.M. Voorhees. TREC 2004 Robust Track Guidelines. In TREC 13, 2004.

[16] Li Wang, Weiguo Fan, Wensi Xi, Edward A. Fox ;Virginia Tech, University of Michigan;Can We Get A Better Retrieval Function From Machine? In TREC 13, 2004.

[17] L. A. Zadeh. Foreword to Fuzzy Logic Toolbox User’s Guide. The MathWorks Inc., Jun 2004.

[18] L.A. Zadeh. Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets and Systems 90 (2), 1997.