Cross-document Structural Relationship Identification Using Supervised Machine Learning

Yogan Jaya Kumar[a,b], Naomie Salim[b], Basit Raza[c]

[a]*Faculty of Information and Communication Technology, University Teknikal Malaysia Melaka, 76100 Melaka, Malaysia*
[b]*Faculty of Computer Science and Information Systems, University Teknologi Malaysia, 81310, Skudai, Johor, Malaysia*
[c]*Department of Computer Science and Software Engineering , International Islamic University Islamabad, Paksitan*

Abstract

Multi document analysis has been a field of interest for decades and is still being actively researched until today. One example of such analysis could be for the task of multi document summarization which is meant to represent the concise description of the original documents. In this paper, we will focus on some special properties that multi document articles hold, specifically news articles. Information across news articles reporting on the same story are often related. Cross-document Structure Theory (CST) gives several relationships between pairs of sentences from different documents. Among them, we focus on four relations namely "Identity", "Overlap", "Subsumption", and "Description". Our aim is to automatically identify these CST relationships. We applied three machine learning techniques, i.e. SVM, Neural Network and our proposed Case-based reasoning (CBR) model. Comparison between these techniques shows that the proposed CBR model yields better results.

***Keywords*:** cross-document structure theory (CST); multi document; machine learning; support vector machine; neural network, case-based reasoning.

1. Introduction

Two sentences from topically related documents can be similar or different in a number of ways. Studying the existence of such inter-document relationships can lead to the identification of rhetorical relations. These rhetorical relations are based on the CST model (Cross-document Structure Theory) [1]. Documents which are related to the same topic usually contains semantically related sentences which can be described by CST relations. Examples of such semantic connections are "Equivalence", "Contrast", "Elaboration", "Agreement" and etc. Fig. 1 shows some examples of sentence pairs that hold CST relationship. The CST model relations are shown in Table 1. The full description of these relations are given in [1].

As far as multi document is concern, especially news documents, its information contents are closely related eventhough the news story comes from various sources. By referring to the list of CST relations shown in Table 1, these types of relations can be essential for the analysis of redundancy, complementarity and contradiction among different information sources. Thus, the ability to automatically identify the types of CST relationship will definitely be handy for tasks related to multi documents, e.g. multi document summarization [2]. In this work, we will discuss on the identification of CST relations using machine learning techniques.

Contradiction

S1: There were 122 people on the downed plane.
S2: 126 people were aboard the plane.

Subsumption

S1: Green Bay has 3 wins this year.
S2: With 3 wins this year, Green Bay has the best record in the NFL.

Follow-up

S1: So far, no casualties from the quake have been confirmed.
S2: 102 casualties have been reported in the earthquake region.

Fig. 1. Examples of CST relationship between sentences.

The rest of this paper is organized as follows: Section 2 presents related works on CST. Section 3 outlines the

automatic identification of CST relationships using the three supervised machine learning techniques i.e. SVM, Neural Network and our proposed CBR model. The experimental setting and results of each technique is given in Section 4, while the discussions of results are given in Section 5. We finally end with conclusions in Section 6.

Table 1
CST relations

Identity	Judgment
Identity	Overlap
Equivalence	Fulfillment
Translation	Description
Subsumption	Reader profile
Contradiction	Change of perspective
Historical background	Follow-up
Modality	Elaboration
Attribution	Citation
Summary	Indirect speech

2. Related works

The work on CST can be put inline with Rhetorical Stucture Theory (RST) [3]. The difference between these two theories is that RST is aimed to capture the rhetorical relation between a span of adjacent text units while CST goes across topically related documents to describe its rhetorical relation. This allows CST to facilitate tasks related to multi document.

A number of research works have addressed the benefits of CST for summarization task. In the work proposed by Zhang et al. [4], they first generate the summary using MEAD, a summarization system developed by Radev et al. [5]. Then they use CST to replace low-salience sentences with sentences that maximize total number of CST relationships in the final summary. The impact of CST on summarization system has also been experimented by Jorge et al. [6], where they rank sentence according to the number of CST relations it has. The limitation of the above works is that the CST relationships need to be manually annotated by human expert.

However, there have been efforts put to learn the CST relationships in texts. Zhang et al. [7] used boosting, a classification algorithm to identify the presence of six CST relationship types between sentences. Their classifier was able to identify sentence pairs with no relationship very well, but showed poor performance in classifying the six CST relationship types. In another work, Miyabe et al. [18] investigated on two CST relationship types, i.e. equivalence and transition using cluster-wise classification with SVM classifier. The authors propose to use the detected equivalence relations to address the task of transition identification.

Closely related to our work is the approach by Zahri and Fukumoto [8]. The authors determined five types of CST relation between sentences using SVM. They used the identified CST relations to determine the directionality between sentences for PageRank [30] computation. However there were no experimental results shown on the performance of their CST classification, specifically. This is essential because the performance of the classification has direct implication on the final results of the system. Here in our work, we have taken the effort to investigate the performance of the well known SVM and Neural Network classification techniques and compare its result with our proposed CBR model.

3. Automatic identification of CST relationships using supervised machine learning

Relying on manually annotated text for CST relationship identification consumes a lot of time and resources. Thus it is favorable to have a system which can automatically identify the existence of CST relation between pairs of sentences. However, at this point of time, we are only considering four types of CST relations namely "Identity", "Overlap", "Subsumption", and "Description". Details of these relations are given in Table 2. Futher details with examples can be found in [4].

Using the dataset from CSTBank [9], we are able to obtain CST annotated sentence pairs. Based on this available dataset, we prepared our training set which comprises of the features between sentences with its corresponding CST relationship. We also manually selected 100 pairs of sentences that poses no CST relationship for our training and test data. We experimented with the following features computed from sentences pair (S_1, S_2) :

Cosine similarity – cosine similarity is used to measure how similar two sentences are. Here the sentences are represented as word vectors having words with tf-idf as its element value:

$$\cos(S_1, S_2) = \frac{\sum S_{1,i} \cdot S_{2,i}}{\sqrt{\sum (S_{1,i})^2} \cdot \sqrt{\sum (S_{2,i})^2}} \quad (1)$$

Word overlap – this feature represents the measure on the numbers of words overlap in the two sentences (after stemming process). This measure is not sensitive to the word order in the sentences [8]:

$$overlap(S_1, S_2) = \frac{\#commonwords(S_1, S_2)}{\#words(S_1) + \#words(S_2)} \quad (2)$$

Length type of S_1 – this feature gives the length type of the first sentence when the lengths of two sentences are compared:

$$lengtype(S_1) = \begin{cases} 1 & \text{if } length(S_1) > length(S_2), \\ -1 & \text{if } length(S_1) < length(S_2), \\ 0 & \text{if } length(S_1) = length(S_2) \end{cases} \quad (3)$$

NP similarity – this feature represents the noun phrase (NP) similarity between two sentences. The similarity between the NPs was calculated according to Jaccard coefficient as defined in the following equation:

$$NP(S_1, S_2) = \frac{NP(S_1) \cap NP(S_2)}{NP(S_1) \cup NP(S_2)} \quad (4)$$

VP similarity – this feature represents the verb phrase (VP) similarity between two sentences. The similarity between the VPs was calculated according to Jaccard coefficient as defined in the following equation:

$$VP(S_1, S_2) = \frac{VP(S_1) \cap VP(S_2)}{VP(S_1) \cup VP(S_2)} \quad (5)$$

Kotsiantis [19] has came out with a good review on various supervised machine learning techniques which have been applied to classification. The basic idea of machine learning is to automatically learn or make decisions from data (training examples) so as to be able to produce a useful output in new cases. If the training examples are given with known labels (the corresponding correct outputs) then the learning is called supervised.

In the following sub-sections, we will discuss the implementation of three supervised machine learning techniques namely SVM, Neural Network and our proposed CBR model for CST relationship identification task.

Table 2
CST relations used in this work

CST Type	Description
Identity	The same text appears in more than one location
Subsumption	S1 contains all information in S2, plus additional information not in S2
Description	S1 describes an entity mentioned in S2
Overlap (partial equivalence)	S1 provides facts X and Y while S2 provides facts X and Z; X, Y, and Z should all be non-trivial.

3.1 Support vector machine

Support Vector Machine (SVM) by Vapnik [15] is a supervised machine learning technique commonly used for classification and regression analysis [21]. Some good references on SVM can be found in the articles written by Burges [16] and Cristianini and Taylor [17]. Thus in this study, we will give a brief description of SVM and show its implementation in our classification problem.

SVM are inherently binary or two-class classifiers. Given a set of training examples with outputs belonging to one of its two classes, the SVM classifier assigns new examples into one class or the other. Theoretically, a support vector machine constructs a hyperplane that separates data into two, positioning them to either side of the hyperplane corresponding to its classes.

Given *n* training data in the form

$$(x_i, y_i) \quad \text{for } i=1\ldots n, \quad y_i \in \{-1, +1\}, \quad x_i \in \Re^D$$

where x_i is the input (a *D*-dimensional real vector) and y_i is its binary output (either +1 or −1) which indicates the class to which x_i belongs. Assuming the training data is linearly separable, a hyperplane can be described as

$$w \cdot x - b = 0 \quad (3)$$

where *w* is normal to the hyperplane and $b/\|w\|$ is the perpendicular distance from the hyperplane to the origin. Refer to Fig. 2. The following constrains implies the position of an input data in its respective class.

$$w \cdot x_i - b \geq +1 \quad \text{for class } y_i = +1 \quad (4)$$

$$w \cdot x_i - b \geq -1 \quad \text{for class } y_i = -1 \quad (5)$$

The SVM learning algorithm works towards maximizing the margin or distance between the separating hyperplane and the two data classes. The hyperplane plays a decisive role as it determines the binary target value for future predictions.

However, in most real-world problems, the data involved is not always linearly separable. Thus it is not always possible to find a hyperplane that can separate two data instances successfully.

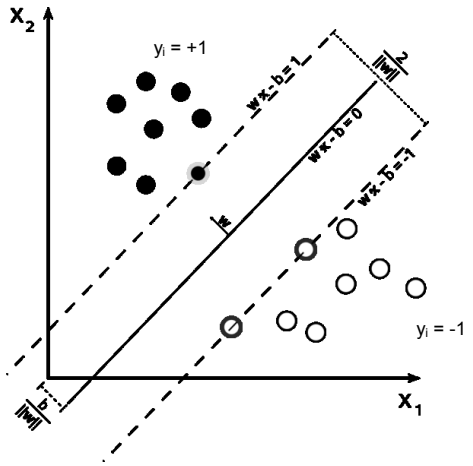

Fig. 2. Hyperplane separating two classes.

In order to handle non-linearly separable data, one solution is to map the data onto a higher dimensional space (also known as feature space) in which we can form a separating hyperplane. This can be done by applying kernel function [20] where it is used to map non-linearly separable data onto a feature space for classification.

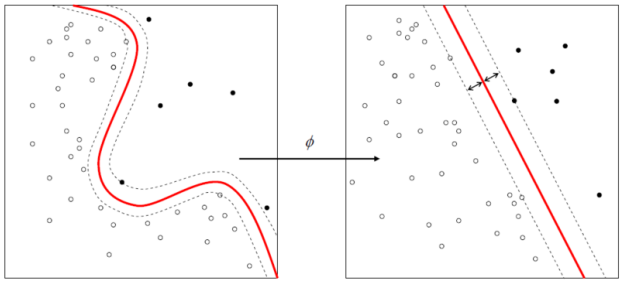

Fig. 3. A non-linear mapping from the input space to some feature space.

As shown in Fig. 3 (left image), the data instances initially were not linearly separable in the original data space. After applying kernel function which transforms the position of the data instances from the original space onto a higher dimensional space, it become linearly separable. Refer to Fig. 3 (right image).

Once a SVM model has been trained, the same kernel function is applied to new instance (input) to locate its position in the feature space so that the binary target value or the class to which the new instance belongs to can be determined. The success behind Support Vector Machine lies in this kernel trick. Genton [22] has described several classes of kernels. Some popular kernel functions which are being extensively used are given below:

linear: $K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)$.

polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \mathbf{x}_j + r)^d, \gamma > 0$.

RBF: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0$.

sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^T \mathbf{x}_j + r)$.

where γ, r and d are kernel parameters.

What we have discussed thus far is the classification of binary type output or two class classification. How does SVM handle multi-class classification as in our (5-class) classification problem? There are a number of ways to handle SVM multiclass classification problem [23]. One of the common approach is to build one-versus-all classifiers (also known as "one-versus-rest"). Another approach is to build a set of one-versus-one classifiers, where the target class is determined by choosing the class that is selected by the most classifiers. We apply the latter approach in our work. Fig. 4 shows the training and classification stages.

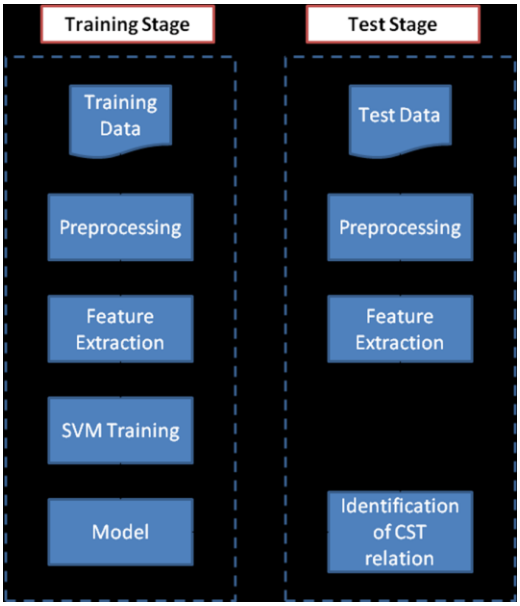

Fig. 4. The training and classification stages using SVM.

We first prepocess the text by stopword filtering and word stemming. After computing the feature values for every sentence pair from the training set, we input them for the training of SVM. Once the training is completed, the resulting classifier model will be tested with test data to measure its performance. Section 4 gives the experimental setting and results of SVM.

3.2 Neural network

Zhang [12] have written a survey on neural network for classification task. A more comprehensive foundation of neural network can be found in [11] and will be outlined briefly here. A generally accepted definition of neural network (NN) is a network of many simple processors. These simple processing elements are referred to as units, nodes, or neurons. These neurons are interconnected and it receive, process and transmit numeric data via the connections.

NN works by training its network which is fed with a set of examples (the training set) containing the input and its the corresponding target (correct) output. It learns by comparing the network output and target output and makes adjustments on the weights (connections between neurons) in order to move the network outputs closer to the targets. This process is depicted in Fig. 5. The network trains until the network output matches the target or achieves error below certain threshold value.

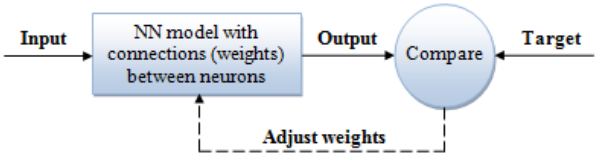

Fig. 5. Neural network's general process paradigm.

To look further into the process which takes place in a neuron, we explain the neuron model and how it functions. A simple neuron model is shown in Fig. 6.

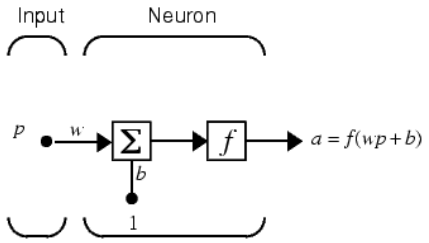

Fig. 6. Neuron Model

When a neuron receives input *P*, it multiplies its strength by weigth *w*. This weighted input is then added with bias *b* which is much like a weight, having constant value of 1. This summed value is then passed to the transfer function *f* and produces the output *a*. There are various types of transfer functions and some of the commonly used are hard-limit transfer function, linear transfer function and sigmoid transfer function. Note that all input, weight, bias and output are numeric data.

A more complex structure of NN is the multi layer neural network. The layers of a multilayer network can have different numbers of neurons. A layer that produces the network output is called an output layer. The intermediate layers between the inputs and output layer are called hidden layers. An example three-layer network is shown in Fig. 7. These networks are usually trained by adapting learning algorithms and one that is very well known to this field is the back propagation algorithm [24].

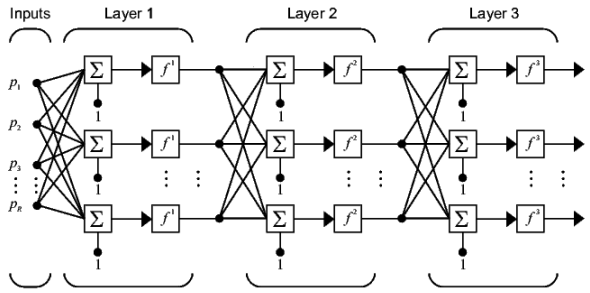

Fig. 7. Example: Multi Layer Neural Network

In our CST relationship identification problem, we applied the Feed-forward neural networks (a standard NN model). Feed-forward neural networks are usually trained by the back propagation algorithm. Some researchers have imposed hybrid learning to train the network [29]. In our work, we have used the Levenberg-Marquardt backpropagation (a variant of back propagation algorithm). Our network model is shown in Fig. 8.

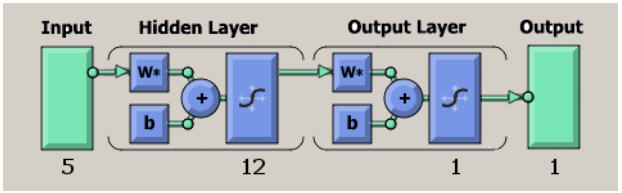

Fig. 8. Generated network model for CST relationship classification.

The numbers in the figure indicates 5 inputs features (cosine similarity, word overlap, length type, NP similarity and VP similarity), 12 hidden neuron, 1 output neuron dan 1 final output (CST relation type). The experimental setting and results of NN is given in Section 4.

3.3 Case-based reasoning

"Case-Based Reasoning (CBR) is the usual name given to problem solving methods which make use of specific past experiences. It is a form of problem solving by analogy in which a new problem is solved by recognizing its similarity to a specific known problem, then transferring the solution of the known problem to the new one" [28]. Using its memory of past experiences, CBR could be applied to solve various real world problems such as course timetabling, solving legal cases [25] and classifying the disease of a patient [14]. Some of the prominent CBR systems which has been used extensively are applications such as Appliance Call Center automation at General Electric [26] and many other reasoning systems with major application area in the health sciences [27].

The general process of CBR can be represented by the CBR cycle as shown in Fig. 9. It consists of four major phases, namely Retrieve, Reuse, Revise, and Retain that links to a central repository called the casebase or knowledgebase. CBR is based on the theory that similar cases have similar solution. For example, when a new case is input into the CBR cycle, the following steps will be taken to solve it.

1. Retrieve – the most similar cases from the casebase;
2. Reuse – the solutions from the retrieved cases;
3. Revise – the solution for the new case if necessary;
4. Retain – adapt revised new cases into the casebase.

An interesting characteristic of this method is that it is capable to adapt new cases to its casebase, whereby this method does not require retraining of data which is necessary for most supervised machine learning techniques.

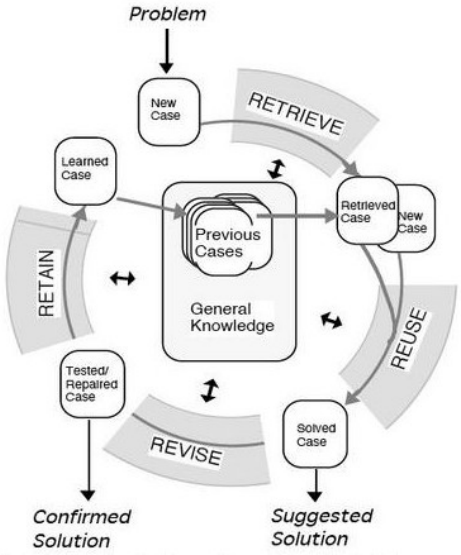

Fig. 9. The CBR cycle, depicted from [13].

Based on the general CBR model described above, we modeled the framework of CBR for CST relationship identification. This framework is depicted in Fig. 10. As explained in previous sections, first we perform text preprocessing and feature extraction on the sentence pairs from the dataset. Once we have extracted all the features from each sentence pair, we represent them as feature vectors (inputs). These inputs together with their respective outputs (CST relationship types) represent the cases in the casebase. Refer to Table 3.

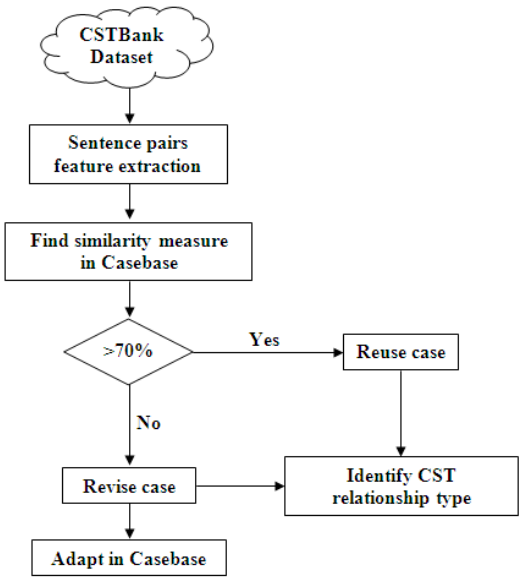

Fig. 10. CBR process for CST relationship identification.

Table 3
An example of case representation

Cases	Input features(F_i)					Output
	F_1	F_2	F_3	F_4	F_5	
Case 1	0.2	0.3	0	0.2	0.5	Description
Case 2	0.4	0.3	1	0.1	0.3	Subsumption

Next, to identify the relationship type of a new case, we will compare the input feature vector of the new case with existing cases in casebase. Here we use the following cosine measure between two cases (C_i, C_j) to retrieve the similar cases.

$$cos(C_i, C_j) = \frac{C_i \cdot C_j}{|C_i||C_j|}, \quad (6)$$

where $C =$ input feature vector.

An example is shown in Table 4, where a new case is being matched with Case 1 and Case 2 from the casebase by using the cosine measure. If the similarity value is more than the predefined threshold value, the model will reuse the solution. Thus, the solution (relationship type) to the new case will then be the output of the most similar case retrieved from the casebase.

If the similarity value is less than the threshold value, the model will revise the new case as "No relation" type and retain the revised new case into the casebase. This is important because not all sentence pairs with similarity can be regarded as having CST relationship. Morever it is not practical to have all variants of "No relation" type cases in the casebase. Thus the adaptive nature of this model will enable the casebase to be revised without much human intervention.

Table 4
An example of similarity measure between cases

Input features	New case	Case 1	Case 2
Cosine Similarity	0.6	0.2	0.4
Word Overlap	0.5	0.3	0.3
Length Type	1	0	1
NP Similarity	0.4	0.2	0.5
VP Similarity	0.4	0.1	0.3
Similarity with new case		0.66	0.97

4. Experimental settings and results

In conducting the experiments, we used the dataset taken from CSTBank[9] – a corpus consisting clusters of English news articles annotated with CST relationships. Our training and testing set consist of sentence pairs with

its corresponding CST type label. We selected 476 sentence pairs for training and 206 sentence pairs for testing. These includes a sample of 100 pairs of sentences that have no CST relationship. First we performed text preprocessing on each of the sentences. Here, two important processes are carried out, namely stop word removal and stemming.

Stop words do not give much meaning but appears too frequent in a document. Examples of stop words are 'the', 'a', 'and', 'to', 'at' and 'on'. To avoid being considered as potential or important words, the stop words are removed from the sentences.

Stemming is a technique to find the root of words, so that the text processing is conducted on the roots and not on the original words. For example, by using a stemming algorithm, words such as 'playing', 'played' and 'plays' will be reduced to the root word 'play'. Stemming proofs to be useful in information retrieval process like in pattern or string matching where the existence of variance in word form can be handled i.e. allowing more terms to be related.

After preprocessing, the features (as described in Section 3) will be extracted from each sentence pairs. These features will then form the instances for the training set where each intance is represented as feature vector with its corresponding CST relationship type label.

For evaluation procedure, we employ the evaluation measures commonly used in classification tasks – Precision, Recall and F-measure. Given the actual class and the predicted class (in this case, the CST relation type), for each class *C*, the following measures are applied:

$$\text{Precision}=\frac{\#\text{instances correctly labeled as class } C}{\text{total } \#\text{instances labeled as class } C} \quad (7)$$

$$\text{Recall}=\frac{\#\text{instances correctly labeled as class } C}{\text{total } \#\text{instances actually belong to class } C} \quad (8)$$

$$\text{F-measure}=2\cdot\frac{\text{precision}\cdot\text{recall}}{\text{precision}+\text{recall}} \quad (9)$$

4.1 Experiment and results using SVM

We trained the data using the LibSVM tool developed by Chang and Lin [10] on MATLAB. LibSVM is an integrated software which is extensively used for solving multiclass classification problems. For our kernel selection, we chose the RBF kernel function as it gives better accuracy and as stated by Hsu et al. [31], it has several advantages over the other kernel functions. The SVM model best parameters were chosen after applying 5-fold cross validation. Table 5 and Fig. 11 shows the precision, recall, and F-measure of SVM classification.

Table 5
Precision, recall, and F-measure of SVM classification

CST Type	Precision	Recall	F-Measure
No relation	0.923077	0.8	0.8571429
Identity	1	0.966667	0.9830508
Subsumption	0.706897	0.82	0.7592593
Description	0.666667	0.894737	0.7640449
Overlap	0.809524	0.586207	0.68

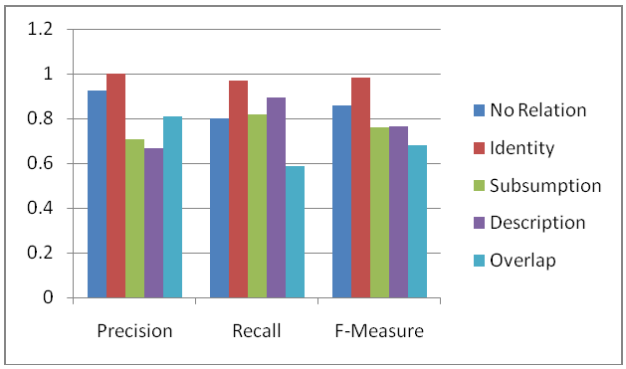

Fig. 11. Performance of SVM classification.

4.2 Experiment and results using NN

To perform the experiment for NN, we trained the data using the Neural Network tool on MATLAB. We use a feed-forward network with the default tan-sigmoid transfer function in the hidden layer and linear transfer function in the output layer. The number of hidden nodes H_i is initially set to 1. The accuracy of the network is then recorded for H_i after training it. Then H_i is incremented and the process continues. The process ends when the result of H_i is better than H_{i+1} and H_{i+2}. After determining the best H, we fixed it as the number of hidden node in the network hidden layer. Table 6 and Fig. 12 shows the precision, recall, and F-measure of NN classification.

Table 6
Precision, recall, and F-measure of NN classification.

CST Type	Precision	Recall	F-Measure
No relation	1	0.8	0.8888889
Identity	1	0.966667	0.9830508
Subsumption	0.754717	0.8	0.776699
Description	0.711111	0.842105	0.7710843
Overlap	0.727273	0.689655	0.7079646

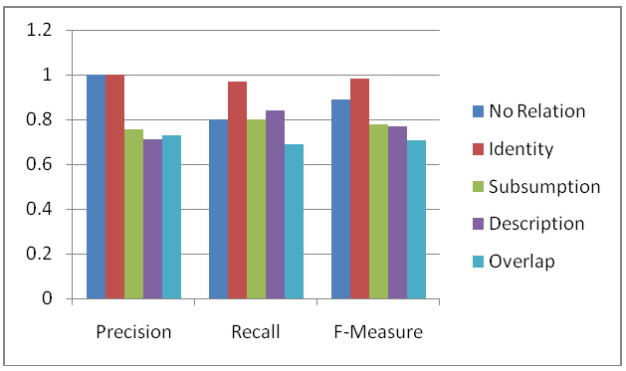

Fig. 12. Performance of NN classification.

4.3 Experiment and results using CBR

We created a MATLAB program to perform the four phases of our CBR model. Both casebase and testing set was represented in matrix form where each rows represent the cases and the columns represent its input features. For similarity measure, we used the commonly used cosine distance function. If there is more than one case with similarity value greater than the threshold, the program selects the highest value among them to be then retrieved. The retrieved case will then be reused to classify the test case. The CBR process (as shown in Fig.10) continues until all test cases have been classified. Table 7 and Fig. 13 shows the precision, recall, and F-measure of CBR classification.

Table 7
Precision, recall, and F-measure of CBR classification

CST Type	Precision	Recall	F-Measure
No relation	0.956522	0.733333	0.8301887
Identity	0.966667	0.966667	0.9666667
Subsumption	0.788462	0.82	0.8039216
Description	0.686275	0.921053	0.7865169
Overlap	0.78	0.672414	0.7222222

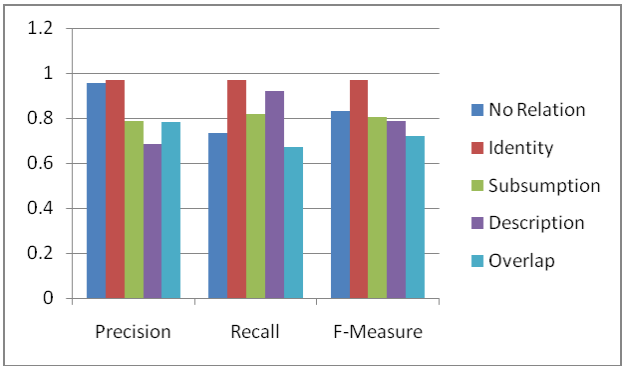

Fig. 13. Performance of CBR classification.

5. Discussions

It is important to compare different machine learning techniques on the same datasets, to see if the performance of the technique being proposed is comparable or performs better than the other techniques. In this work, i.e. the automatic identification of CST relationships, we have compared the performance of our proposed CBR model with Neural Network and Support Vector Machine, which are two popular machine learning techniques used for classification tasks.

The experiments results for CST relationship identification described in the previous section using SVM, NN and CBR gives us an insight into the performance of these supervised machine learning techniques. From Table 5, Table 6 and Table 7, we can observe the performance of each classifier in identifying the CST relationship types. We can see that all three techniques give good performance (i.e. > 90%) for the relationship type "Identity" and (> 80%) for "No Relation". This result is probably due to the characteristics of "Identity" type sentences which have high similarity in terms of words and length while "No Relation" possess the complete opposite characteristics.

Table 8 and Fig. 14 shows the comparison of F-measures between the three techniques. It can be seen in Fig. 15 that overall, CBR performs better than SVM and NN. In addition, the accuracy of each method is shown in Fig. 16. We could observe that the accuracy of SVM is lesser than CBR and NN. The poor performance of SVM is probably due to number of features used as SVM normally performs well with high dimensionality datasets. This may possibly explain why SVM could not well differentiate between the different classes.

The ability of CBR to be superior than SVM and NN for CST relationship type identification could closely be related to nature of its learning method itself, i.e. lazy learning. As opposed to eager learning methods which need to generalize the training data to classify new cases, lazy learning is a learning method which performs classifcation based on the similarity of that problem with already known problems. Concerning our problem domain, since texts data have high variability, a key advantage of lazy learning is that instead of estimating the target function once for the entire instance space, this method can estimate it locally for each new instance to be classifed.

On the whole, our proposed CBR technique for CST relationship type identification has able to outperform the popular SVM and NN technique. On top of that, CBR will better fit our CST relationship identification problem since it is capable to adapt new cases into its casebase. This will not require retraining of data as opposed to SVM and NN.

Table 8
Comparison of F-measures between SVM, NN and CBR

CST Type	F-Measure		
	SVM	NN	CBR
No relation	0.857143	0.888889	0.830189
Identity	0.983051	0.983051	0.966667
Subsumption	0.759259	0.776699	0.803922
Description	0.764045	0.771084	0.786517
Overlap	0.68	0.707965	0.722222

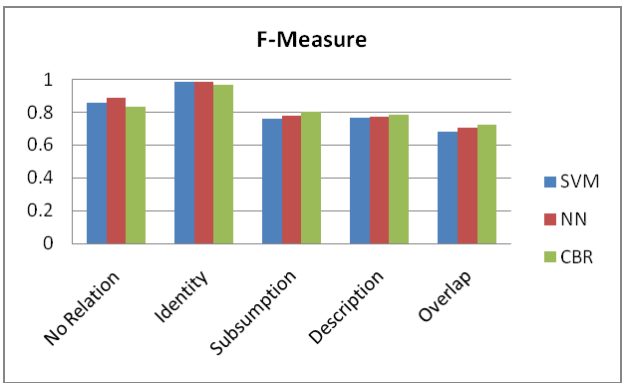

Fig. 14. Comparison of F-measures between SVM, NN and CBR.

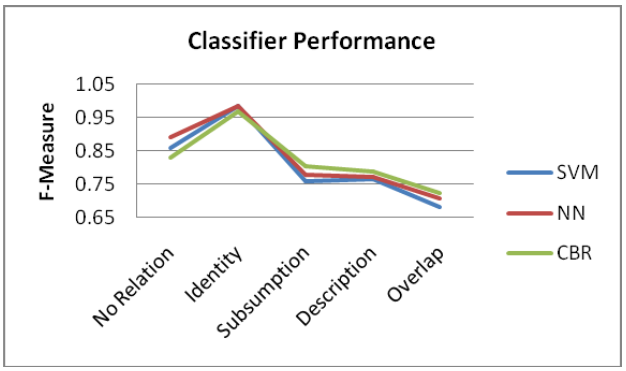

Fig. 15. Performance comparison between SVM, NN and CBR.

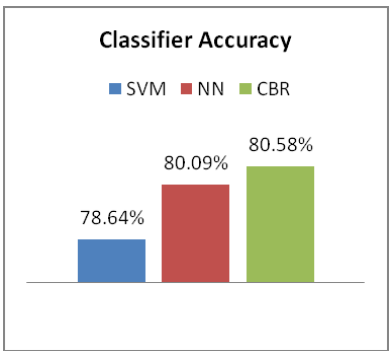

Fig. 16. Accuracy comparison between SVM, NN and CBR.

6. Conclusions and Future Work

This work provides the study on CST relationship identification between sentences in topically related documents. The ability to automatically identify the types of CST relationship will definitely be handy for tasks related to multi documents, e.g. multi document analysis for text summarization. Relying on manually annotated text for this task consumes a lot of time and resources. With the motivation to have a system which can automatically identify the existence of CST relation between sentences, we propose a supervised machine learning technique using case-based reasoning (CBR) model.

We have experimented using the dataset obtained from CSTBank which comprises human annotated CST relations. We also described in this work the implementation of two popular classification techniques, i.e. support vector machine and neural network and compared its performance with our proposed method. Comparison between these techniques shows that the proposed CBR model yields better results.

We belief that the CBR model has the potential to be further enhanced to improve its performance. As the performance of CBR model is highly sensitive to its similarity function, the quality of features has a significant impact on the learning algorithm. In existing setting, all features are assumed to hold equal importance by the learning algorithm. Therefore scaling the relevance of each feature based on feature weighting could improve the similarity-based selection of relevant cases. We regard this as our future work.

Acknowledgements

The researcher is sponsored by IDF and Ministry of Science Technology and Innovation under the university research grant vote number 01H74, Universiti Teknologi Malaysia.

References

[1] Radev, D.R., A Common Theory of Information Fusion from Multiple Text Sources Step One: Cross-Document Structure, Proceeding SIGDIAL 10 (2000) 74-83.

[2] Kumar, Y.J. and N. Salim, Automatic multi document summarization approaches, J. Comput. Sci. 8 (2011) 133-140.

[3] Taboada, M. and W.C. Mann, Rhetorical Structure Theory: Looking Back and Moving Ahead, Discourse Studies 8 (2006) 423-459.

[4] Zhang, Z., Blair-Goldensohn, S., and Radev, D.R., Towards CST-Enhanced Summarization, In Proceedings of AAAI/IAAI (2002) 439-446.

[5] Radev, D.R., S. Blair-Goldensohn, Z. Zhang, Experiments in single and multidocument summarization using MEAD, In Proceedings of the Document Understanding Conference, (2001).

[6] Jorge, M.L.C., Pardo, T.S., Experiments with CST-based Multidocument Summarization, Workshop on Graph-based Methods for Natural Language Processing, ACL, Uppsala, Sweden, (2010) 74–82.

[7] Zhang, Z., Otterbacher, J., and Radev, D.R., Learning cross-document structural relationships using boosting. In Proceedings of CIKM. (2003) 124-130.

[8] Zahri, N.A.H.B. and Fukumoto, F., Multi-document Summarization Using Link Analysis Based on Rhetorical Relations between Sentences, In Proceedings of CICLing (2) (2011) 328-338.

[9] Radev, D.R., Otterbacher, J.: CSTBank PhaseI, (2003), http://tangra.si.umich.edu/clair/CSTBank/

[10] Chang, C. and Lin, C., LIBSVM: a library for support vector machine, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1--27:27.

[11] Simon S. Haykin, "Neural Networks: A Comprehensive Foundation", Macmillan, New York, 1994.

[12] Zhang, G., Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics Part C 30 (4) (2000) 451-462.

[13] Aamodt, A. and Plaza, E., Case-based reasoning: foundational issues, methodological variations and system approaches. AI Communications 7 (1) (1994) 39 – 59.

[14] Fan, C., Chang, P., Lin, J., and Hsieh, J.C., A hybrid model combining case-based reasoning and fuzzy decision tree for medical data classification, Applied Soft Computing 11 (1) (2011) 632-644.

[15] Vladimir N. Vapnik, The Nature of Statistical Learning Theory. Springer, 1995.

[16] Burges, C., A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2 (2) (1998) 1-47.

[17] Cristianini, N. & Shawe-Taylor, J., An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, Cambridge, (2000).

[18] Y. Miyabe, H. Takamura, and M. Okumura, Identifying cross-document relations between sentences. InProc. of IJCNLP (2008) 141–148.

[19] Sotiris B. Kotsiantis. Supervised Machine Learning: A Review of Classification Techniques. Emerging Artificial Intelligence Applications in Computer Engineering (2007) 3-24.

[20] B. Schölkopf and A. J. Smola. Learning with Kernels. MIT Press, 2002.

[21] Chen-Chia Chuang, Zne-Jung Lee, Hybrid robust support vector machines for regression withoutliers Applied Soft Computing 11 (1) January (2011) 64-72.

[22] Genton, M., Classes of Kernels for Machine Learning: A Statistics Perspective. Journal of Machine Learning Research 2 (2001) 299-312.

[23] Duan, K. and Keerthi, S.S., Which Is the Best Multiclass SVM Method? An Empirical Study. In Proceedings of Multiple Classifier Systems. (2005) 278-285.

[24] Y. Chauvin and D. E. Rumelhart, Eds., Backpropagation: Theory, Architectures, and Applications, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1995.

[25] V. Raman, P. Ayyappan. Computer Aided Legal Support System: Methodology to Automatically Convert Legal Text into Cases for Building Case Base Reasoning Proceedings of International Conference on Computer Engineering and Applications(ICCEA) (2009) 260-264.

[26] Cheetham, W., Goebel, K., Appliance Call Center: A Successful Mixed-Initiative Case Study, Artificial Intelligence Magazine 28 (2) (2007) 89 – 100.

[27] Begum, S.; M. U Ahmed, P. Funk, Ning Xiong, M. Folke, Case-Based Reasoning Systems in the Health Sciences: A Survey of Recent Trends and Developments, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 41 (4) (2011) 421–434.

[28] Bareis R., Exemplar-Based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification, and Leaerning. Academic Press. 1989.

[29] Mounir Ben Nasr, Mohamed Chtourou, A self-organizing map-based initialization for hybrid training of feedforward neural networks, Applied Soft Computing 11 (8) (2011) 4458-4464.

[30] Erkan, G. and Radev D.R., LexPageRank: Prestige in multi-document text summarization, In Proceedings of EMNLP (2004) 365-371.

[31] C. W. Hsu, C. C. Chang and C. J. Lin, A Practical Guide to Support Vector Classification, Technical Report, Department of Computer Science and Information Engineering, University of National Taiwan, Taipei (2003) 1-12.