Intelligent Information Management, **2013, 5, 182-190**
Published Online November 2013 (http://www.scirp.org/journal/iim)
http://dx.doi.org/10.4236/iim.2013.56020

Scientific Research

Using AdaBoost Meta-Learning Algorithm for Medical News Multi-Document Summarization

Mahdi Gholami Mehr
Department of Computer and Information Technology, Malek-Ashtar University of Technology, Tehran, Iran
Email: Mahdi.Gholami.Mehr@gmail.com

Received September 26, 2013; revised October 25, 2013; accepted November 6, 2013

ABSTRACT

Automatic text summarization involves reducing a text document or a larger corpus of multiple documents to a short set of sentences or paragraphs that convey the main meaning of the text. In this paper, we discuss about multi-document summarization that differs from the single one in which the issues of compression, speed, redundancy and passage selection are critical in the formation of useful summaries. Since the number and variety of online medical news make them difficult for experts in the medical field to read all of the medical news, an automatic multi-document summarization can be useful for easy study of information on the web. Hence we propose a new approach based on machine learning meta-learner algorithm called AdaBoost that is used for summarization. We treat a document as a set of sentences, and the learning algorithm must learn to classify as positive or negative examples of sentences based on the score of the sentences. For this learning task, we apply AdaBoost meta-learning algorithm where a C4.5 decision tree has been chosen as the base learner. In our experiment, we use 450 pieces of news that are downloaded from different medical websites. Then we compare our results with some existing approaches.

Keywords: Multi-Document Summarization; Machine Learning; Decision Trees; AdaBoost; C4.5; Medical Document Summarization

1. Introduction

Nowadays there are lots of online medical news on the web and study of these huge amount of information is not possible for experts in medical field [1]. Medical information on the web such as news, articles, clinical trial reports is an important source to help clinicians in patient treatment. Usually, clinicians go through author-written abstracts or summaries available in the medical domain and then decide whether articles are relevant to them for in-depth study. Since all types of medical articles do not come with authors written abstracts or summaries. An automatic muti-document summarization can be useful for help clinicians or medical students to find their relevant information on the web.

Text summarization is the process to produce a condensed representation of the content of its input for human consumption. In existing categorization for summarization with respect to the number of input documents, the summarization is divided into two types, namely single and multi documents. Automatic multi document summarization refers to the production process of a compressed summary of documents while the content, readability, and cohesion are maintained [2]. Considering further complexities of multi document summarization than single document summarization, we face with some challenges among them. The most significant ones are as following [3]:

- The rate of information redundancy is higher in a group of subject-related texts;
- The need for devoting great attention to the extraction of unknown perspectives in the documents and covering all of them;
- Difficulty of producing a highly readable summary from documents that address the same subject from different perspectives;
- Difficulty of ordering extractive phrases for production of the final summary.

In another categorization based on the type of summary, if the summarized text was obtained through extracting some phrases from the original text, the summarization would be "extractive" or "selective", and if the text summary is generated after understanding the available content in the original text, the summarization

would be "abstractive" [4]. Both types of summarizations face different challenges. In extracting method, main challenges are identifying important sentences in the text, distinguishing and extracting key words, and analyzing the main text with the purpose of preparing a readable and coherent summary. In abstract method, first, the main text must be understood; then based on the meaning of the text, a meaningful summary is produced. In this method, the main challenges are natural language processing and the analysis of the meaning of the text with the purpose of comprehension.

In this paper, we present a machine learning based model for a sentence extraction based, Multi document, and informative text summarization in the medical domain (This work is an improvement of the study proposed in [5]). In our work, we approach automatic text summarization as a supervised learning task. We treat a document as a set of sentences, which must be classified as positive or negative examples of sentences based on the summary worthiness of sentences where a sentence is represented by a feature set, which includes a number of features used in the summarization literature and some other features specific to the medical domain.

Thus, this summarization task can be formulated as the classical machine-learning problem of learning from examples. There are several unusual aspects to this classification problem. For example, the size of positive examples in the training set is relatively small compared to the size of the entire training set because a summary size is roughly less than one-fourth of the size of the source document. It has been generally thought that a summary should be no shorter than 15% and no longer than 35% of the source text [6].

C4.5 is typically applied to more balanced class distributions. In our experiment, we found that AdaBoost improves and performs significantly and uniformly well, when combined with C4.5. In general, AdaBoost works by repeatedly running a given weak learning algorithm on various distributions over the training data, and then combining the classifiers produced by the weak learner into a single composite classifier. There seem to be two separate reasons for the improvement in performance that is achieved by boosting. The first and better understood effect of boosting is that it generates a hypothesis whose error on the training set is small by combining many hypotheses whose error may be large (but still better than random guessing). It seems that boosting may be helpful to learning problems having either of the following two properties. The first property, which holds for many real-world problems, is that the observed examples tend to have varying degrees of hardness. For such problems, the boosting algorithm tends to generate distributions that concentrate on the harder examples, thus challenging the weak learning algorithm to perform well on these harder parts of the sample space. The second property is that the learning algorithm is sensitive to changes in the training examples so that significantly different hypotheses are generated for different training sets.

For text summarization applications, we need to rank sentences based on its summary scores. So, for the sentence ranking, we have to follow a new methodology to combine decisions (discuss in Section 4).

We adopted and designed ten features to characterize sentences (taken as basic linguistic units) in the documents.

The paper is organized as follows. Section 2 provides related work. In Section 3, we discuss how to extract and use features. In Section 4, the summarization method has been discussed. We present the evaluation and the experimental results in Section 5.

2. Related Work

In this section, we discuss about some previous works that are used in text summarization. Radev *et al*. [7] suggested a multi-document extracting summary maker that extracts the sentences of the summary from several texts. The extraction is based on center clusters. To increase coherence, Harry Hilda [8] and Mitra [9] extracted paragraphs instead of sentences from documents. Knight and Marcu [10] presented two algorithms for sentence compression which are based on "Noisy Channel" and "Decision Tree". The input of the algorithm of "Decision Tree" is a long sentence, and the output is supposed to be a shorter sentence but with more meaningful content. Barzilay *et al.* [11] presented an algorithm for the fusion of information. This algorithm tries to combine similar sentences of the documents to create a new sentence based on "Language Generation Technology". Although this method simulates human behavior in summarization process to some extent, it is heavily dependent on external sources such as "Dependency Parser", "Production Rules", and etc. Therefore, the portability of this method is limited. In sentence extracting strategy, clustering is introduced with the purpose of eliminating redundant information which is due to using multi documents [12]. But this technique cannot solve the problem of redundancy entirely because some sentences can be in more than one cluster. To solve this problem some researchers predefine the number of clusters, or determine a threshold level for similarities. Even if the number of sentences is predefined, it is probable that the sentences with highest score in clusters are not the best sentences. To solve this problem GA algorithm [13] is used; in this algorithm the favorable summary is chosen from a group of summaries that are created by combined sentences obtained from the main documents. Four properties which are length of sentence, cover criterion, information criterion, and similarity are used as fitness function of GA algo-

rithm for summarization.

MEAD [6] a popular summarization system ranks sentences based on its similarity to the centroid, position in the text, similarity to the first sentence of the article and length. It uses linear combination of features whose values are normalized between 0 and 1 for sentence ranking. Redundancy is removed by a variation of MMR (Maximal Marginal Relevance) algorithm [14].

Some machine learning approaches to extractive summarization have already been investigated. In [15] sentence extraction is viewed as a Bayesian classification task. To our knowledge, there are few attempts to use machine learning algorithm for medical document summarization task. Most of the researchers extend to the medical domain the summarization techniques already used in other domains. One of the projects in medical domain is [16]. MiTAP (MITRE Text and Audio Processing) monitors infectious disease outbreaks or other biological threats by monitoring multiple information sources. The work presented in [17] exploits extractive techniques, which ranks the extracted sentences according to the so-called cluster signature of the document. The abstracts and full texts from the Journal of the American Medical Association were used for their experiments. TRESTLE (Text Retrieval Extraction and Summarization Technologies for Large Enterprises) is a system, which produces single sentence summaries of pharmaceutical newsletters [18]. TRESTLE generates summaries by filling the templates by the Information Extraction process. The system Helpful Med [19] helps professional and advanced users to access medical information on the Internet and in medical related databases. An ontology based summarization approach has been proposed in [20]. A query based medical information summarization system that exploits ontology knowledge has been proposed in [21].

The work presented in [21] uses ontology to expand query words and assigns scores to sentences based on number of original keywords (query words) and expanded keywords. Most recently a variation of lexical chaining method [22] called bio-chain [23] is used in biomedical text summarization.

Compared to the above-mentioned approaches, we develop a machine learning based model for medical document summarization that also exploits domain knowledge.

3. Summarization Method

In extractive text summarization approach, the main task is to identify sentences in a source text, which are relevant to the users while simultaneously reducing information redundancy. Sentences are scored based on a set of features. The top-n highest scoring sentences in a text are then extracted where n is an upper bound, which is determined by the compression rate. Finally the selected sentences are presented to the user in their order of appearance in the original source text [24].

The proposed summarization method consists of three primary parts that shows in **Figure 1**.

The preprocessing task includes formatting the document, removal of punctuation marks (except dots at the sentence boundaries).

3.1. Using AdaBoost for Sentence Extraction

We apply a meta-learner called AdaBoost for sentence extraction, where a C4.5 decision tree [25] has been chosen as the base learner.

The boosting algorithm takes as input a training set of m examples $S = \left((x_1, y_1) \cdots (x_m, y_m)\right)$ where x_i is an instance drawn from some space X and represented in some manner (typically, a vector of attribute values), and $y_i \in Y$ is the class label associated with x_i. In this paper, we always assume that the set of possible labels Y is of finite cardinality k.

In addition, the boosting algorithm has access to another unspecified learning algorithm, called the weak learning algorithm, which is denoted generically as **Weak Learn**. The boosting algorithm calls **Weak Learn** repeatedly in a series of rounds. On round t, the booster provides **Weak Learn** with a distribution D_t over the training set S. In response, **Weak Learn** computes a classifier or hypothesis h_t: $X \rightarrow Y$ which should correctly classify a fraction of the training set that has large probability with respect to D_t. That is, the weak learner's goal is to find a hypothesis h_t which minimizes the (training) error $\in_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right]$. Note that this error is measured with respect to the distribution D_t that was provided to the weak learner. This process continues for T rounds, and, at last, the booster combines the weak hypotheses $h_1 \ldots h_T$ into a single final hypothesis h_{fin}. Still unspecified are: 1) the manner in which D_t is computed on each round, and 2) how h_{fin} is computed. Different boosting schemes answer these two questions in different ways. **AdaBoost** uses the simple rule shown in **Figure 2**. The initial distribution D_1 is uniform over S so $D_1(i) = 1/m$ for all i. To compute distribution D_{t+1} from D_t and the last weak hypothesis h_t, we multiply the weight of example i by some number $\beta_t \in [0,1]$ if h_t classifies x_i correctly, and otherwise the weight is left unchanged. The weights are then renormalized by dividing by the normalization constant weights are then renormalized by dividing by the normalization constant Z_t. Effectively, "easy" examples that are correctly classified by many of the previous weak hypotheses get lower weight, and "hard" examples which tend often to be misclassified get higher weight. Thus, **AdaBoost** focuses the most weight on the examples which seem to be hardest for **Weak Learn**.

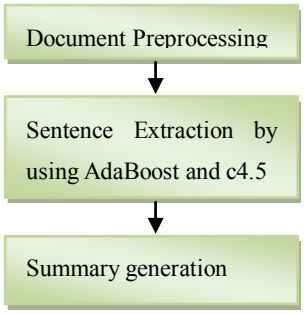

Figure 1. Summarization method.

Algorithm AdaBoost

Input: sequence of m examples $\{(x_1, y_1),...,(x_m, ym)\}$

with lables $y_i \in Y = \{1,...,k\}$

weak learning algorithm WeakLearn

integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i

Do for t = 1,2,...,T:

1) Call WekLearn, providing it with the distribution D_t.

2) Get back a hypothesis $h_t : X - Y$.

3) Calculate the error of h_t: $\epsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$

If $\epsilon_t > 1/2$, *then* set T = t-1 and abort loop.

4) Set $\beta_t = \epsilon_t / (1-\epsilon_t)$.

5) Update distribution D_t :

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$

Where Z_t is a normalization constant

(chosen so that D_t+1 will be a distribution).

Output the final hypothesis:

$$h_{fin}(x) = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log\frac{1}{\beta_t}$$

Figure 2. The algorithm AdaBoost.

The number β_t is computed as shown in the figure as a function of ϵ_t. The final hypothesis h_{fin} is a weighted vote (*i.e.*, a weighted linear threshold) of the weak hypotheses. That is, for a given instance *x*, h_{fin} outputs the label *y* that maximizes the sum of the weights of the weak hypotheses predicting that label. The weight of hypothesis h_t is defined to be $\log(1/\beta_t)$ so that the greater weight is given to hypotheses with lower error. The important theoretical property about **AdaBoost** is stated in the following theorem. This theorem shows that if the weak hypotheses consistently have error only slightly better than 1/2, then the training error of the final hypothesis h_{fin} drops to zero exponentially fast. For binary classification problems, this means that the weak hypotheses need be only slightly better than random.

Theorem 1: suppose the weak learning algorithm **Weak Learn**, when called by **AdaBoost**, generates hypotheses with errors $\epsilon_1,\cdots,\epsilon_T$, where ϵ_t is as defined in **Figure 1**. Assume each $\epsilon_t \le 1/2$ and let

$$\gamma_t = 1/2 - \epsilon_t .$$

Then the following upper bound holds on the error of the final hypothesis h_{fin}:

$$\frac{|i : h_{fin}(x_i) \neq y_i|}{m} \le \prod_{t=1}^{T} \sqrt{1-4\gamma_t^2}\ \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$$

Theorem 1 implies that the training error of the final hypothesis generated by **AdaBoost** is small. This does not necessarily imply that the test error is small. However, if the weak hypotheses are "simple" and *T* "not too large", then the difference between the training and test errors can also be theoretically bounded.

Traditionally, the component learners are of the same general form. In our case, all component learners are decision trees. In general, decision tree induction algorithms have low bias but high variance.

Boosting multiple trees improves performance by reducing variance and this procedure appears to have relatively little impact on bias.

To train a learning algorithm, we need to establish a set of features and a training corpus of document/extract pairs. In our work, the main goal is to train a booster of decision trees with the set of features and the multiple versions of training set *D* and combine the decisions of those trained decision trees to classify a sentence as summary worthy (positive) or not (negative example). After completion of training, the trained learning algorithm is tested on unseen instances, which is not part of training corpus.

3.1.1. Building Corpus

The training and test corpus are built by downloading medical news form different websites. Then the summaries are manually created for that news. A total of 450 news documents that downloaded from different websites.

For each news, two manual summaries (model summaries) are created by human abstractors. Since summaries are very subjective and user sensitive, for each news we decide to have three different model summaries created by three different human abstractors. Human abstractors are faculty members and postgraduate students of our institute. Though human abstractors have been instructed to create abstracts for each article by copying material from the original text, they freely used their own sentence construction while summarizing news.

But, to apply machine learning algorithm, we need to have extracts instead of abstracts because for extracting features for summary sentences we need to match the manual summary sentences to the sentences in the original document. So, for each manually created abstract, we

create an extract by selecting the sentences from the original document that best match the sentences in the abstract. The average size of an extract is 25% of the source document. We choose relatively long medical news documents in our study because the summarization becomes more useful for long news documents. The reason behind choosing medical news articles for our experiment is that the medical news does not come with any abstract or summaries. Though some features of the medical news articles and general newspaper articles may overlap, we found that the medical news have some features such as medical terms, medical cue phrases which may be absent in the general newspaper articles. It is a common practice to evaluate a machine learning algorithm using *k*-fold cross validation, where the entire dataset is divided into *k* subsets, and each time, one of the *k* subsets is used as the test set and the other $k-1$ subsets are put together to form a training set. Thus, every data point gets to be in a test set exactly once, and gets to be in a training set $k-1$ times. For the evaluation of the proposed learning based system, the entire dataset are divided into 2 folds where each fold consists of one training set of 300 documents and a test set of 150 documents.

3.1.2. Features

To characterize the sentences in the medical documents we have designed and adopted a number of features such as: centroid overlap, similarity to subject, similarity of sentences to each other, positive and negative cue phrases, acronyms, sentence position, sentence length, numeral data.

For normalization of each feature value, we divide the value by the maximum of the scores obtained by the sentences due to the feature. The following we discuss the features in detail.

- **Centroid:** this criterion is used to calculate the similarity of the sentences to the central sentence of the text. The following procedure is used to find the central sentence of the text.
  - With the use of sentence similarity matrix, total similarity of a sentence to the rest of the sentences is calculated with the following method:

$$\text{SimSen}_i = \sum_{j=1}^{n} \text{SimMat}[i, j]$$

SimMat [*i*,*j*] shows the similarity of *i*th sentence to *j*th sentence. After calculation, maximum value for SimSeni is found and the position of the sentence-sentence index is placed in index-centroid variable.

  - With the use of cosine similarity formula, similarity of each sentence to the center is calculated as follows:

$$\text{Centroid\_Sen}_i = \text{Sim}\left(S_i, S_{\text{index\_centroid}}\right)$$

Then the above relation is normalized between 0 and 1.

- **Similarity to title:** A good summary consists of sentences that are similar to the title [26,27]. This means that if sentence S_i is the most similar sentence to the title, in comparison to other sentences in the document, sentence S_i can be considered more important than other sentences or the most important sentence. Calculating this criterion is as follows:

$$\text{Title\_Sen}_i = \text{Sim}\left(S_i, S_{\text{title}}\right)$$

Then the above relation is normalized between 0 and 1. The more the value (closer to 1), the greater the similarity is.

- **Sentence position**: A summery which contains the first and the last sentence of a document is a good summary [28]. For this reason, the position of the sentences in a text is very important. The following formula is used to calculate the position of the sentence in a document:

$$\text{Pos\_Score}_{i,j} = \left(1 - 0.5 * \text{Sin}\left(\pi * i / \left(N-1\right)\right)\right) \times \text{DocRank}_j, \quad j \in D$$

N is the total number of documents. DocRank_j is the parameter that is used for giving higher score to the first and the last sentences which have more importance. Therefore, documents need to be ranked. For ranking, the number of keywords in each document is divided to the total number of words in the document. The score of the document with more keywords is closer to one while score closer to zero represent fewer keywords.

$$\text{DockRank}_j = \frac{\text{Num of Keyword}_j}{\max\left\{\text{Num of Keyword}_i, i \in D\right\}}$$

- **Positive and negative cue phrases in medical domain:** A sentence gets score of n if it contains n positive cue phrases and gets score of $-n$ if it contains n negative cue phrases from our knowledge base.
- **Acronyms**: A sentence gets a score based on the number of acronyms it contains. In medical articles, authors frequently use acronyms for important complex medical terms, perhaps it help them memorize the things better. So, we consider acronym as an important feature for medical document summarization task. If some letters (at least two letters) of a term are capital, we treat the term as an acronym (gene names, medical instruments etc.).
- **Sentence length** long or short sentences are not suitable for the summary. Therefore, to calculate this criterion, first the lengths of all sentences in the text are calculated. Afterward the average length called Len_{avg} is calculated. Now, the following formula is used to score each sentence of the text for the sentence length

criteria:

$$\text{Sen\_Len}_i = \begin{cases} 1 & \text{Len}_i = \text{Len}_{\text{avg}} \\ \dfrac{\text{Len}_{\text{avg}}}{\left|\text{Len}_i - \text{Len}_{\text{avg}}\right| + \text{Len}_{\text{avg}}} & \text{else} \end{cases}$$

- **Numerical data:** Numerical data like date and time is important in news. Hence a sentence gets score (+1) if it contains numerical data.

3.1.3. Sentence Extraction

Training a learning algorithm for summary sentence extraction requires document sentences to be represented as feature vectors. For this purpose, we write a computer program for automatically extracting values for the features characterizing the sentences in the documents. For each sentence in the given document we extract the feature values from the source document using the measures discussed in Sub-Section 3.1.2. If the sentence under consideration is found in both the extracts, extract1 and extract 2, which are created from the human abstracts (discussed in 3.1.1), we label the sentence as "Summary Worthy" sentence. If it is found in one of these extracts, we label the sentence as "Moderately Summary Worthy" and if it is not found in any one of these extracts we label the sentence as "Summary Unworthy". Thus each sentence vector looks like {<a_1 a_2 a_3 ⋯ a_n>, <label>} which becomes an instance (example) for a base learner C4.5 decision tree, where a_1, a_2, ⋯, a_n, indicate feature values for a sentence. All the documents in our corpus are converted to a set of instances of the above form. We divide the entire data set into 3 folds where each fold consists of one training set corresponding to a set of training documents and a test set corresponding to a set of test documents. After preparation of a training set, the multiple decision trees are trained with the different versions of the training set and the decisions of those trained decision trees are combined to classify a sentence as one of three categories: "Summary Worthy", "Moderately Summary Worthy" and "Summary Unworthy". For each fold, a model is built from a training set using the boosting technique and then the learned model is applied to the test set. For our experiments, we have used Weka (www.cs.waikato.ac.nz/ml/weka) machine learning tools. Initially, for each fold, we submit the training data set and the test data set to Weka. Then we select the option "boosting" under meta-classifier folder in Weka. We chose J48 (Weka's implementation of Quinlan's C4.5 decision tree) as a base learner and set the number-of-base learners to the default value which is 10.

Though all the attribute values of the instances in the training and test sets are continuous, we did not apply any separate discretization algorithm because C4.5 is capable of handling continuous attribute values. We configure WEKA in such a way that for each test instance, we can get the predicted class and the probability estimate for the class. The trained learning algorithm will assign one of three labels: "Summary Worthy" (SW), "Moderately Summary Worthy" (MSW), "Summary Unworthy" (SU) to a test instance corresponding to a sentence in a test document. It is possible to save the output in a separate file. We save the output produced by WEKA in a file and then collect the classification output for the sentences belonging to each test document. Then we design a sentence-ranking algorithm based on the classification output and the probability estimates for the classes. The algorithm for sentence ranking is given below.

Sentence Ranking Algorithm

Input:

An output file produced by WEKA, which contains the sentences of a test document with their classifications and the probability estimates of the classes to which the sentences belong.

Output: A file containing the ranked sentences

Begin

Read the input file.

- Select those sentences, which have been classified as "Summary Worthy" (SW) and reorder the selected sentences in decreasing order of the probability estimates of their classes. Save the selected sentences in the output file and delete them from the input file.
- Select those sentences, which have been classified as "Moderately Summary Worthy" (MSW) and reorder the selected sentences in decreasing order of the probability estimates of their classes. Save the selected sentences in the output file and delete them from the input file.
- For the rest of the sentences, which are classified as "Summary Unworthy", we order the sentences in increasing order of the probability estimates of the class labels. In effect, the sentence for which the probability estimate is minimum (that is, the sentence is minimum "Summary Unworthy") comes at the top. Append the ordered sentences to the output file.
- Close the output file.

End of the Algorithm

The sentence-ranking algorithm has three major steps. At the first step, the sentences, which are classified as "Summary Worthy", we undoubtedly select those sentences in the summary.

If the number of sentences selected at Step 1 is less than the desired number of sentences, we consider those sentences which are not selected in the summary at the first step. At the second step, the sentences, which are classified as "Moderately Summary Worthy", are considered. If the number of sentences selected at Step 1 and Step 2 are less than desired number of sentences, we con-

sider the sentences, which have been classified as "Summary Unworthy" and order those sentences in increasing order of the probability estimates of the class labels, that is, the sentences are ordered from minimum summary unworthiness (maximum summary worthiness) to maximum summary unworthiness (minimum summary worthiness). They are selected in this order one by one in the summary until the desired summary length is reached.

4. Summary Generation

After ranking the sentences, n top ranked sentences are selected to generate the final summary. Value of n depends on the compression rate. But, the summary produced in this way may contain some redundant information, that is, some sentences in the summary may entail partially or fully the concept embodied in other sentences. This restricts the summary to be more informative when the summary length is a restriction. Moreover, a user who is used to just looking at first few sentences representing the same concept will prefer to see something different information, though marginally less relevant. To keep the sentences in the summary sufficiently dissimilar from each other, the diversity based re-ranking method called Maximal Marginal Relevance (MMR) is a well-known measure. This approach uses a ranking parameter that allows the user to slide between relevance to the query and diversity from the sentences seen so far. The MMR algorithm is most suitable to apply in query-focused summarization where the summary will be focused toward the user's query. But in our generic summarization environment where only one generic summary will be produced for a text document, we have used a variant of the MMR algorithm to remove redundancy in the summary. This algorithm works as follows:

- Rank the sentences using the ranking algorithm discussed in Sub-Section 3.1.3.
- Select the top ranked sentence first.
- Select the next sentence from the ordered list and include into the summary if this sentence is sufficiently dissimilar to all of the previously selected sentences.
- Continue selecting sentences one by one until the predefined summary length is reached.

The similarity between two sentences is measured using cosine similarity metric. If the cosine similarity between two sentences is greater (less) than a threshold, we say that the sentences are similar (dissimilar). The cosine similarity between two sentences is measured by the following formula as stated in [29].

$$\text{Idf-modified-cosine}(x,y) = \frac{\sum_{\omega \in x,y} tf_{\omega,x} * tf_{\omega,y} * (idf_{\omega})^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} * idf_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (tf_{y_i,y} * idf_{y_i})^2}}$$

where $f_{\omega,S}$ is the number of occurrences of the word ω in the sentence S, idf_{ω} inverse document frequency of the word ω and x_i is the *i*-th word in the sentence *x* and y_i is the *i*-th word in the sentence *y*. idf value of a word is computed on a corpus of documents using the formula: log(*N*/*df*) where *N* is the number of documents in the corpus and *df* is the number of documents in the corpus that contain the word. Finally, the sentences selected in the above-mentioned manner are reordered using text order (sorted in the order in which they appear in the input texts) to increase the readability of the summary.

5. Experimental Results

To evaluate our summarization system, 450 medical news articles have been downloaded from a number of online medical news sources. From the downloaded articles, the images and other links are manually removed and only the news content is considered.

Traditionally, for each system generated summary, more than one model summaries are used for evaluation because the human abstractors may disagree with each other in producing the summary of the document. But, manual summary creation is a tedious task. In our experiments, we have used two reference summaries for evaluating a system generated summary.

For system evaluation, we have used precision and recall.

Precision and recall: Precision and recall are the well-known evaluation measures in the information retrieval settings. Since our system extracts sentences from the source document to form a summary, we define precision and recall as follows:

$$\text{Precision} = \frac{N}{K}$$

where, *N* = number of extracted sentences matched with a reference summary and *K* = number of sentences extracted by the system.

$$\text{Recall} = \frac{N}{M}$$

where, *N* = number of extracted sentences matched with a reference summary and *M* = number of sentences in the reference summary. Since we have used two reference summaries for evaluating a system generated summary, we have compared the system summary to each of the reference summaries and computed the precision and recall. Thus for each system generated summary, we get one pair of precision and recall values for the first reference summary and another pair of precision and recall values for the second reference summary. We define the average precision_{R1R2} and the average recall_{R1R2} as follows:

$$\text{Average precision}_{R1R2} = \frac{P_{R_1} + P_{R_2}}{2}$$

$$\text{Average recall}_{R1R2} = \frac{R_{R_1} + R_{R_2}}{2}$$

where P_{R_1} = average precision of a system, where the precision is computed by comparing the system generated summary and the first reference summary for a document, P_{R_2} = average precision of a system, where the precision is computed by comparing the system generated summary and the second reference summary for a document, P_{R_1} = average recall of a system, where the recall is computed by comparing the system generated summary and the first reference summary for a document, P_{R_2} = average recall of a system, where the recall is computed by comparing the system generated summary and the second reference summary for a document.

For evaluating the system using precision and recall, we set the compression ratio to 15% and 20%. Compression ratio r% means r percent of the total sentences in the source documents are extracted as a summary. To measure the overall performance of the proposed learning based summarization system, our experimental dataset consisting of 450 documents are divided into 3 folds for 3-fold cross validation where each fold contains two independent sets: a training set of 300 documents and a test set of 150 documents. For each fold, a separate model is built from 300 documents and the learned model is applied to the test set of 150 documents. Thus, for each task, if we consider all three folds, we can get a summary for each of 450 documents in our corpus. For other systems such as MEAD and the lead baseline system (which simply takes the first n words or n sentences of the document) and Bagging Method to which the proposed system is compared, we run the systems on the entire 450 documents in our corpus to collect 450 summaries for each task.

Table 1 shows the results in terms of precision and recall for the compression ratio set to 15% and **Table 2** shows the results for the compression ratio set to 20%.

By analyzing **Table 1**, we find that for 15% summary generation task, the learning based system performs better than the Bagging Method, lead baseline and MEAD, but MEAD performs worse than the lead baseline and Bagging method.

Table 2 shows that for 20% summary generation task, MEAD performs better than the lead baseline whereas the learning based system performs better than MEAD and Bagging method.

Table 1. Precision and recall for 15% summary generation task on the test data set.

	Average Precision$_{R1R2}$	**Average Recall$_{R1R2}$**
Proposed Method	0.69	0.26
Mead	0.54	0.24
Baseline-Lead	0.58	0.25
Bagging Method	0.63	0.29

Table 2. Precision and recall for 20% summary generation task on the test data set

	Average Precision$_{R1R2}$	Average Recall$_{R1R2}$
Proposed Method	0.61	0.34
Mead	0.54	0.31
Baseline-Lead	0.47	0.27
Bagging Method	0.59	0.35

6. Conclusion

This paper discusses a machine learning based model for text summarization in medical domain. Most of previous works on text summarization in the medical domain extends the various features used in other domains to the medical domain. In our work, we have combined several medical domain specific features with some other features used in the state-of-art summarization approaches. A machine-learning tool has been used for effective feature combination. The proposed approach performs better than the systems it is compared to.

REFERENCES

[1] S. D. Afantenos, V. Karkaletsis and P. Stamatopoulos, "Summarization from Medical Documents: A Survey," *Journal of Artificial Intelligence in Medicine*, Vol. 33, No. 2, 2005, pp. 157-177. http://dx.doi.org/10.1016/j.artmed.2004.07.017

[2] I. Mani, "Automatic Summarization," John Benjamins Publishing Company, Amsterdam/Philadelphia, 2001.

[3] X. Wan, J. Yang and J. Xiao, "Manifold-Ranking Based Topic-Focused Multi Document Summarization," *Proceedings of the* 20*th International Joint Conference on Artifical Intelligence*, Hyderabad, 6-12 January 2007, pp. 2903-2908.

[4] H. Jing and K. McKeown, "Cut and Paste Based Text Summarization," *Proceedings of the* 1*st Meeting of the North American Chapter of the Association for Computational Linguistics* (*NAACL* 2000), Seatle, Washington DC, June 2000.

[5] K. Knight and D. Marcu, "Summarization Beyond Sentence Extraction: A Probablistic Approach to Sentence Compression," *Artificial Intelligence*, Vol. 139, No. 1, 2002, pp. 91-107.

[6] E. H. Hovy, "Automated Text Summarization," In: R. Mitkov, Ed., *The Oxford Handbook of Computational Linguistics*, Oxford University Press, Oxford, 2005, pp. 583-598. http://dx.doi.org/10.1093/oxfordhb/9780199276349.013.0032

[7] D. R. Radev, H. Jing, M. Sty and D. Tam, "Centroid-

Based Summarization of Multiple Documents," *Journal of Information Processing and Management*, Vol. 40, No. 6, 2004, pp. 919-938. http://dx.doi.org/10.1016/j.ipm.2003.10.006

[8] H. Hilda, "Cross-Document Summarization by Concept Classification," *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, 11-15 August 2002, pp. 121-128.

[9] M. Mitra, S. Amit and B. Chris, "Automatic Text Summarization by Paragraph Extraction," *ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization*, Madrid, 20 July 1997, pp. 31-36.

[10] K. Knight and D. Marcu, "Summarization beyond Sentence Extraction: A Probablistic Approach to Sentence Compression," *Artifcial Intelligence*, Vol. 139, No. 1, 2002, pp. 91-107.

[11] B. Regina, K. R. McKeown and M. Elhadad, "Information Fusion in the Context of Multi Document Summarization," *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, Morristown, 20-26 June 1999, pp. 550-557.

[12] J. Manuel and MANA-LOPEZ, "Multi-Document Summarization: An Added Value to Clustering in Interactive Retrieval," *ACM Transactions on Information Systems*, Vol. 22, No. 2, 2004, pp. 215-241.

[13] Y.-X. He, D.-X. Liu, D.-H. Ji and C. Teng, "MSBGA: A Multi-Document Summarization System Based on Genetic Algorithm," *Proceedings of the 5th International Conference on Machine Learning and Cybernetics*, Daliban, 13-16 August 2006, pp. 2659-2664.

[14] G. J. Carbonell and J. Goldstein, "The Use of MMR, Diversity-Based Re-Ranking for Reordering Documents and Producing Summaries," *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, 24-28 August 1998, pp. 335-336.

[15] J. Kupiec, J. O. Pedersen and F. Chen, "A Trainable Document Summarizer," *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seattle, Washington DC, 1995, pp. 68-73.

[16] L. D. Day, L. Hirschman, R. Kozierok, S. Mardis, T. McEntee, *et al*., "Real Users, Real Data, Real Problems: The MiTAP System for Monitoring Bio Events," *Proceedings of the Conference on Unified Science & Technology for Reducing Biological Threats & Countering Terrorism* (*BTR* 2002), University of New Mexico, Mexico, 2002, pp. 167-177.

[17] D. B. Johnson, Q. Zou, J. D. Dionisio, V. Z. Liu and W. W. Chu, "Modeling Medical Content for Automated Summarization," *Annals of the New York Academy of Sciences*, Vol. 980, 2002, pp. 247-258.

[18] R. Gaizauskas, P. Herring, M. Oakes, M. Beaulieu, P. Willett, H. Fowkes, *et al*., "Intelligent Access to Text: Integrating Information Extraction Technology into Text Browsers," *Proceedings of the Human Language Technology Conference* (*HLT* 2001), San Diego, 2001, pp. 189-193.

[19] H. Chen, A. Lally, B. Zhu and M. Chau, "HelpfulMed: Intelligent Searching for Medical Information over the Internet," *Journal of American Society for Information Science and Technology* (*JASIST*), Vol. 54, No. 7, 2003, pp. 683-694. http://dx.doi.org/10.1002/asi.10260

[20] M. Fiszman, T. Rindflesch and H. Kilicoglu, "Abstraction Summarization for Managing the Biomedical Research Literature," *Proceedings of the HLT-NAACL Workshop on Computational Lexical Semantics*, Stroudsburg, 2004, pp. 76-83.

[21] P. Chen and R. Verma, "A Query-Based Medical Information Summarization System Using Ontology Knowledge," *Proceedings of the 19th IEEE Symposium on Computer Based Medical Systems*, Salt Lake City, 2006, pp. 37-42.

[22] R. Barzilay and M. Elhadad, "Using Lexical Chains for Text Summarization," In: I. Mani and M. T. Maybury, Eds., *Advances in Automatic Text Summarization*, The MIT Press, Cambridge, 1999, pp. 111-121.

[23] L. H. Reeve, H. Han and A. D. Brooks, "The Use of Domain-Specific Concepts in Biomedical Text Summarization," *Journal of Information Processing and Management*, Vol. 43, No. 6, 2007, pp. 1765-1776. http://dx.doi.org/10.1016/j.ipm.2007.01.026

[24] R. Barzilay, N. Elhadad and K. McKeown, "Sentence Ordering in Multi-Document Summarization," *Proceedings of the Human Language Technology Conference*, San Diego, 2001, pp. 1-7.

[25] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann, California, 1993.

[26] G. Salton and C. Buckley, "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing and Management*, Vol. 24, No. 5, 1998, pp. 513-523. http://dx.doi.org/10.1016/0306-4573(88)90021-0

[27] J. Silla, C. Nascimento, G. L. Pappa, A. A. Freitas and C. A. A. Kaestner, "Automatic Text Summarization with Genetic Algorithm-Based Attribute Selection," Lecture Notes in Artificial Intelligence, 2004.

[28] C. Y. Lin and E. Hovy, "Identifying Topics by Position," *Proceedings of the 5th Applied Natural Language Processing Conference*, 1997, pp. 283-290.

[29] G. Erkan and D. R. Radev, "LexRank: Graph-Based Lexical Centrality as Salience in Text Summarization," *Journal of Artificial Intelligence Research* (*JAIR*), Vol. 22, 2004, pp. 457-479.