Selecting Sentences for Answering Complex Questions

Yllias Chali
University of Lethbridge
4401 University Drive
Lethbridge, Alberta, Canada, T1K 3M4
chali@cs.uleth.ca

Shafiq R. Joty
University of British Columbia
2366 Main Mall
Vancouver, B.C. Canada V6T 1Z4
rjoty@cs.ubc.ca

Abstract

Complex questions that require inferencing and synthesizing information from multiple documents can be seen as a kind of topic-oriented, informative multi-document summarization. In this paper, we have experimented with one empirical and two unsupervised statistical machine learning techniques: k-means and Expectation Maximization (EM), for computing relative importance of the sentences. However, the performance of these approaches depends entirely on the feature set used and the weighting of these features. We extracted different kinds of features (i.e. lexical, lexical semantic, cosine similarity, basic element, tree kernel based syntactic and shallow-semantic) for each of the document sentences in order to measure its importance and relevancy to the user query. We used a local search technique to learn the weights of the features. For all our methods of generating summaries, we have shown the effects of syntactic and shallow-semantic features over the bag of words (BOW) features.

1 Introduction

After having made substantial headway in factoid and list questions, researchers have turned their attention to more complex information needs that cannot be answered by simply extracting named entities (persons, organizations, locations, dates, etc.) from documents. For example, the question: *"Describe the after-effects of cyclone Sidr-Nov 2007 in Bangladesh"* requires inferencing and synthesizing information from multiple documents. This information synthesis in NLP can be seen as a kind of topic-oriented, informative multi-document summarization, where the goal is to produce a single text as a compressed version of a set of documents with a minimum loss of relevant information.

In this paper, we experimented with one empirical and two well-known unsupervised statistical machine learning techniques: k-means and EM and evaluated their performance in generating topic-oriented summaries. However, the performance of these approaches depends entirely on the feature set used and the weighting of these features. We extracted different kinds of features (i.e. lexical, lexical semantic, cosine similarity, basic element, tree kernel based syntactic and shallow-semantic) for each of the document sentences in order to measure its importance and relevancy to the user query. We have used a gradient descent local search technique to learn the weights of the features. Traditionally, information extraction techniques are based on the BOW approach augmented by language modeling. But when the task requires the use of more complex semantics, the approaches based on only BOW are often inadequate to perform fine-level textual analysis. Some improvements on BOW are given by the use of dependency trees and syntactic parse trees (Hirao et al., 2004), (Punyakanok et al., 2004), (Zhang and Lee, 2003), but these, too are not adequate when dealing with complex questions whose answers are expressed by long and articulated sentences or even paragraphs. Shallow semantic representations, bearing a more compact information, could prevent the sparseness of deep structural approaches and the weakness of BOW models (Moschitti et al., 2007). Attempting an application of syntactic and semantic information to complex QA hence seems natural, as pinpointing the answer to a question relies on a deep understanding of the semantics of both. In more complex tasks such as computing the relatedness between the query sentences and the document sentences in order to generate query-focused summaries (or answers to complex questions), to our knowledge no study uses tree kernel functions to encode syntactic/semantic information. For all our methods of generating summaries (i.e. empirical, k-means and EM), we have shown the effects of syntactic and shallow-semantic features over the BOW features.

This paper is organized as follows: Section 2 focuses on the related work, Section 3 describes how the features are extracted, Section 4 discusses the scoring approaches, Section 5 discusses how we remove the redundant sentences before adding them to the summary, Section 6 describes our experimental study. We conclude and give future directions in Section 7.

2 Related Work

Researchers all over the world working on query-based summarization are trying different directions to see which methods provide the best results. The *LexRank* method addressed in (Erkan and Radev, 2004) was very successful in generic multi-document summarization. A topic-sensitive LexRank is proposed in (Otterbacher et al., 2005). As in LexRank, the set of sentences in a document cluster is represented as a graph, where nodes are sentences and links between the nodes are induced by a similarity relation between the sentences. Then the system ranked the sentences according to a random walk model defined in terms of both the inter-sentence similarities and the similarities of the sentences to the topic description or question.

The summarization methods based on lexical chain first extract the nouns, compound nouns and named entities as candidate words (Li et al., 2007). Then using WordNet, the systems find the semantic similarity between the nouns and compound nouns. After that, lexical chains are built in two steps: 1) Building single document strong chains while disambiguating the senses of the words and, 2) building multi-chain by merging the strongest chains of the single documents into one chain. The systems rank sentences using a formula that involves a) the lexical chain, b) keywords from query and c) named entities.

(Harabagiu et al., 2006) introduce a new paradigm for processing complex questions that relies on a combination of (a) question decompositions; (b) factoid QA techniques; and (c) Multi-Document Summarization (MDS) techniques. The question decomposition procedure operates on a Marcov chain, by following a random walk with mixture model on a bipartite graph of relations established between concepts related to the topic of a complex question and subquestions derived from topic-relevant passages that manifest these relations. Decomposed questions are then submitted to a state-of-the-art QA system in order to retrieve a set of passages that can later be merged into a comprehensive answer by a MDS system. They show that question decompositions using this method can significantly enhance the relevance and comprehensiveness of summary-length answers to complex questions.

There are approaches that are based on probabilistic models (Pingali et al., 2007) (Toutanova et al., 2007). (Pingali et al., 2007) rank the sentences based on a mixture model where each component of the model is a statistical model:

$$Score(s) = \alpha \times QIScore(s) + (1 - \alpha) \times QFocus(s, Q)$$

Where, Score(s) is the score for sentence s. Query-independent score (QIScore) and query-dependent score (QFocus) are calculated based on probabilistic models. (Toutanova et al., 2007) learns a log-linear sentence ranking model by maximizing three metrics of sentence goodness: (a) ROUGE oracle, (b) Pyramid-derived, and (c) Model Frequency. The scoring function is learned by fitting weights for a set of feature functions of sentences in the document set and is trained to optimize a sentence pair-wise ranking criterion. The scoring function is further adapted to apply to summaries rather than sentences and to take into account redundancy among sentences.

There are approaches in “Recognizing Textual Entailment”, “Sentence Alignment” and “Question Answering” that use syntactic and/or semantic information in order to measure the similarity between two textual units. (MacCartney et al., 2006) use typed dependency graphs (same as dependency trees) to represent the text and the hypothesis. Then they try to find a good partial alignment between the typed dependency graphs representing the hypothesis and the text in a search space of $O((m + 1)n)$

where hypothesis graph contains n nodes and a text graph contains m nodes. (Hirao et al., 2004) represent the sentences using Dependency Tree Path (DTP) to incorporate syntactic information. They apply String Subsequence Kernel (SSK) to measure the similarity between the DTPs of two sentences. They also introduce Extended String Subsequence Kernel (ESK) to incorporate semantics in DTPs. (Kouylekov and Magnini, 2005) use the tree edit distance algorithms on the dependency trees of the text and the hypothesis to recognize the textual entailment. According to this approach, a text T entails a hypothesis H if there exists a sequence of transformations (i.e. deletion, insertion and substitution) applied to T such that we can obtain H with an overall cost below a certain threshold. (Punyakanok et al., 2004) represent the question and the sentence containing answer with their dependency trees. They add semantic information (i.e. named entity, synonyms and other related words) in the dependency trees. They apply the approximate tree matching in order to decide how similar any given pair of trees are. They also use the edit distance as the matching criteria in the approximate tree matching. All these methods show the improvement over the BOW scoring methods.

Our Basic Element (BE)-based feature used the dependency tree to extract the BEs (i.e. head-modifier-relation) and ranked the BEs based on their log-likelihood ratios. For syntactic feature, we extracted the syntactic trees for the sentence as well as for the query using the Charniak parser and measured the similarity between the two trees using the tree kernel function. We used the ASSERT semantic role labeler system to parse the sentence as well as the query semantically and used the shallow semantic tree kernel to measure the similarity between the two shallow-semantic trees.

3 Feature Extraction

The sentences in the document collection are analyzed in various levels and each of the document-sentences is represented as a vector of feature-values. The features can be divided into several categories:

3.1 Lexical Features

3.1.1 N-gram Overlap

N-gram overlap measures the overlapping word sequences between the candidate sentence and the query sentence. With the view to measure the N-gram (N=1,2,3,4) overlap scores, a *query pool* and a *sentence pool* are created. In order to create the query (or sentence) pool, we took the query (or document) sentence and created a set of related sentences by replacing its important words[1] by their first-sense synonyms. For example given a stemmed document-sentence: "John write a poem", the sentence pool contains: "John compose a poem", "John write a verse form" along with the given sentence. We measured the recall based n-gram scores for a sentence P using the following formula:

n-gramScore(P) = $max_i(max_j$ N-gram$(s_i, q_j))$

$$\text{N-gram(S,Q)} = \frac{\sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{gram_n \in S} Count(gram_n)}$$

Where, n stands for the length of the n-gram $(n = 1, 2, 3, 4)$ and $Count_{match}\,(gram_n)$ is the number of n-grams co-occurring in the query and the candidate sentence, q_j is the j^{th} sentence in the query pool and s_i is the i^{th} sentence in the sentence pool of sentence P.

3.1.2 LCS, WLCS and Skip-Bigram

A sequence $W = [w_1, w_2, ..., w_n]$ is a subsequence of another sequence $X = [x_1, x_2, ..., x_m]$, if there exists a strict increasing sequence $[i_1, i_2, ..., i_k]$ of indices of X such that for all $j = 1, 2, ..., k$ we have $x_{i_j} = w_j$. Given two sequences, $S_1\ and\ S_2$, the Longest Common Subsequence (LCS) of $S_1\ and\ S_2$ is a common subsequence with maximum length. The longer the LCS of two sentences is, the more similar the two sentences are.

The basic LCS has a problem that it does not differentiate LCSes of different spatial relations within their embedding sequences (Lin, 2004). To improve the basic LCS method, we can remember the length of consecutive matches encountered so far to a regular two dimensional dynamic program table computing LCS. We call this weighted LCS (WLCS) and use k to indicate the length of the current consecutive matches ending at words x_i and y_j. Given two sentences X and Y, the WLCS score of X and Y can be computed using the similar dynamic programming procedure as stated in (Lin, 2004). We computed the LCS and WLCS-based F-measure following (Lin, 2004) using both the query pool and the sentence pool as in the previous section.

Skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps. Skip-bigram measures the overlap of skip-bigrams between a candidate sentence and a query sentence. Following (Lin, 2004), we computed the skip bi-gram score using both the sentence pool and the query pool.

[1] hence forth important words are the nouns, verbs, adverbs and adjectives

3.1.3 Head and Head Related-words Overlap

The number of head words common in between two sentences can indicate how much they are relevant to each other. In order to extract the heads from the sentence (or query), the sentence (or query) is parsed by Minipar [2] and from the dependency tree we extract the heads which we call exact head words. For example, the head word of the sentence: "John eats rice" is "eat".

We take the synonyms, hyponyms and hypernyms[3] of both the query-head words and the sentence-head words and form a set of words which we call head-related words. We measured the exact head score and the head-related score as follows:

ExactHeadScore = $\frac{\sum_{w_1 \in HeadSet} Count_{match}(w_1)}{\sum_{w_1 \in HeadSet} Count(w_1)}$

HeadRelatedScore = $\frac{\sum_{w_1 \in HeadRelSet} Count_{match}(w_1)}{\sum_{w_1 \in HeadRelSet} Count(w_1)}$

Where HeadSet is the set of head words in the sentence and $Count_{match}$ is the number of matches between the HeadSet of the query and the sentence. HeadRelSet is the set of synonyms, hyponyms and hypernyms of head words in the sentence and $Count_{match}$ is the number of matches between the head-related words of the query and the sentence.

3.2 Lexical Semantic Features

We form a set of words which we call *QueryRelatedWords* by taking the important words from the query, their first-sense synonyms, the nouns' hypernyms/hyponyms and important words from the nouns' gloss definitions.

Synonym overlap measure is the overlap between the list of synonyms of the important words extracted from the candidate sentence and the *QueryRelatedWords*. *Hypernym/hyponym overlap* measure is the overlap between the list of hypernyms and hyponyms of the nouns extracted from the sentence and the *QueryRelatedWords*, and *gloss overlap* measure is the overlap between the list of important words that are extracted from the gloss definitions of the nouns of the sentence and the *QueryRelatedWords*.

3.3 Statistical Similarity Measures

Statistical similarity measures are based on the co-occurance of similar words in a corpus. We have used two statistical similarity measures: *1. Dependency-based similarity measure* and *2. Proximity-based similarity measure*.

Dependency-based similarity measure uses the dependency relations among words in order to measure the similarity. It extracts the dependency triples then uses statistical approach to measure the similarity. *Proximity-based similarity measure* is computed based on the linear proximity relationship between words only. It uses the information theoretic definition of similarity to measure the similarity.

We used the data provided by Dr. Dekang Lin[4]. Using the data, one can retrieve most similar words for a given word. The similar words are grouped into clusters. Note that, for a word there can be more than one cluster. Each cluster represents the sense of the word and its similar words for that sense.

For each query word, we extract all of its clusters from the data. Now, in order to determine the right cluster for a query word, we measure the overlap score between the *QueryRelatedWords* and the *clusters of words*. The hypothesis is that, the cluster that has more words common with the *QueryRelatedWords* is the right cluster. We chose the cluster for a word which has the highest overlap score.

Once we get the clusters for the query words, we measured the overlap between the cluster words and the sentence words as follows:

Measure = $\frac{\sum_{w_1 \in SenWords} Count_{match}(w_1)}{\sum_{w_1 \in SenWords} Count(w_1)}$

Where, SenWords is the set of important words extracted from the sentence and $Count_{match}$ is the number of matches between the sentence words and the clusters of similar words of the query words.

3.4 Graph-based Similarity Measure

In LexRank (Erkan and Radev, 2004), the concept of graph-based centrality is used to rank a set of sentences, in producing generic multi-document summaries. A similarity graph is produced for the sentences in the document collection. In the graph, each node represents a sentence. The edges between the nodes measure the cosine similarity between the respective pair of sentences. The degree of a given node is an indication of how much important the sentence is. Once the similarity graph is

[2] http://www.cs.ualberta.ca/ lindek/minipar.htm

[3] hypernym and hyponym levels are restricted to 2 and 3 respectively

[4] http://www.cs.ualberta.ca/ lindek/downloads.htm

constructed, the sentences are then ranked according to their eigenvector centrality. To apply LexRank to query-focused context, a topic-sensitive version of LexRank is proposed in (Otterbacher et al., 2005). We followed a similar approach in order to calculate this feature. The score of a sentence is determined by a mixture model of the relevance of the sentence to the query and the similarity of the sentence to other high-scoring sentences.

3.5 Syntactic and Semantic Features:

So far, we have included the features of type Bag of Words (BOW). The task like *query-based summarization* that requires the use of more complex syntactic and semantics, the approaches with only BOW are often inadequate to perform fine-level textual analysis. We extracted three features that incorporate syntactic/semantic information.

3.5.1 Basic Element (BE) Overlap Measure

The "head-modifier-relation" triples, extracted from the dependency trees are considered as BEs in our experiment. The triples encode some syntactic/semantic information and one can quite easily decide whether any two units match or not- considerably more easily than with longer units (Zhou et al., 2005). We used the BE package distributed by ISI[5] to extract the BEs for the sentences.

Once we get the BEs for a sentence, we computed the Likelihood Ratio (LR) for each BE following (Zhou et al., 2005). Sorting BEs according to their LR scores produced a BE-ranked list. Our goal is to generate a summary that will answer the user questions. The ranked list of BEs in this way contains important BEs at the top which may or may not be relevant to the user questions. We filter those BEs by checking whether they contain any word which is a *query word* or a *QueryRelatedWords* (defined in Section 3.2). The score of a sentence is the sum of its BE scores divided by the number of BEs in the sentence.

3.5.2 Syntactic Feature

Encoding syntactic structure is easier and straight forward. Given a sentence (or query), we first parse it into a syntactic tree using a syntactic parser (i.e. Charniak parser) and then we calculate the similarity between the two trees using the *tree kernel* defined in (Collins and Duffy, 2001).

3.5.3 Shallow-semantic Feature

Though introducing BE and syntactic information gives an improvement on BOW by the use of dependency/syntactic parses, but these, too are not adequate when dealing with complex questions whose answers are expressed by long and articulated sentences or even paragraphs. Shallow semantic representations, bearing a more compact information, could prevent the sparseness of deep structural approaches and the weakness of BOW models (Moschitti et al., 2007).

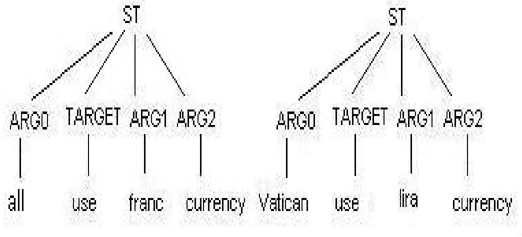

Figure 1: Example of semantic trees

Initiatives such as PropBank (PB) (Kingsbury and Palmer, 2002) have made possible the design of accurate automatic Semantic Role Labeling (SRL) systems like ASSERT (Hacioglu et al., 2003). For example, consider the PB annotation:

[ARG0 all][TARGET use][ARG1 the french franc][ARG2 as their currency]

Such annotation can be used to design a shallow semantic representation that can be matched against other semantically similar sentences, e.g.

[ARG0 the Vatican][TARGET use][ARG1 the Italian lira][ARG2 as their currency]

In order to calculate the semantic similarity between the sentences, we first represent the annotated sentence using the tree structures like Figure 1 which we call Semantic Tree (ST). In the semantic tree, arguments are replaced with the most important word-often referred to as the semantic head.

The sentences may contain one or more subordinate clauses. For example the sentence, "the Vatican, located wholly within Italy uses the Italian lira as their currency." gives the STs as in Figure 2. As we can see in Figure 2(A), when an argument node corresponds to an entire subordinate clause, we label its leaf with ST, e.g. the leaf of ARG0. Such ST node is actually the root of the subordinate clause in Figure 2(B). If taken separately, such STs do not express the whole meaning of the sentence, hence it is more accurate to define a single structure encoding the dependency between the two predicates as in Figure 2(C). We refer to this kind of nested STs as STNs.

Note that, the tree kernel (TK) function defined in (Collins and Duffy, 2001) computes the number of common subtrees between two trees. Such subtrees are subject to the constraint that their nodes are taken with all or none of the children they have in the original tree.

[5] BE website:http://www.isi.edu/ cyl/BE

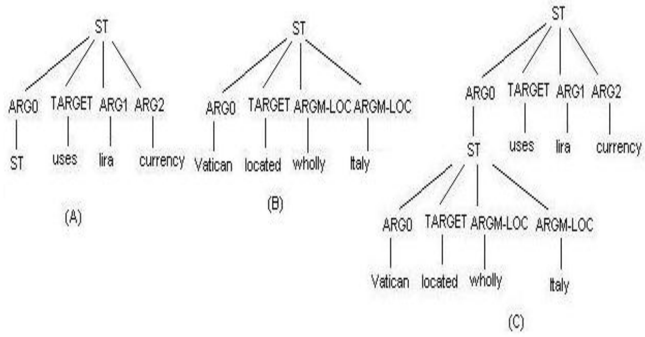

Figure 2: Two STs composing a STN

Though, this definition of subtrees makes the TK function appropriate for syntactic trees but at the same time makes it not well suited for the semantic trees (ST) defined above. For instance, although the two STs of Figure 1 share most of the subtrees rooted in the *ST* node, the kernel defined above computes only one match (ST ARG0 TARGET ARG1 ARG2) which is not useful.

The critical aspect of the TK function is that the productions of two evaluated nodes have to be identical to allow the match of further descendants. This means that common substructures cannot be composed by a node with only some of its children as an effective ST representation would require. (Moschitti et al., 2007) solve this problem by designing the Shallow Semantic Tree Kernel (SSTK) which allows to match portions of a ST. We followed the similar approach to compute the SSTK.

4 Ranking Sentences

In this section, we describe the scoring techniques in detail.

4.1 Learning Feature-weights: A Local Search Strategy

In order to fine-tune the weights of the features, we used a local search technique with simulated annealing to find the global maximum. Initially, we set all the feature-weights, $w_1, \cdots, w_n$, as equal values (i.e. 0.5) (see Algorithm 1). Based on the current weights we score the sentences and generate summaries accordingly. We evaluate the summaries using the automatic evaluation tool ROUGE (Lin, 2004) (described in Section 6) and the ROUGE value works as the feedback to our learning loop. Our learning system tries to maximize the ROUGE score in every step by changing the weights individually by a specific step size (i.e. 0.01). That means, to learn weight w_i, we change the value of w_i keeping all other weight values ($w_j \forall_{j \neq i}$) stagnant. For each weight w_i, the algorithm achieves the local maximum of ROUGE value. In order to find the global maximum we ran this algorithm multiple times with different random choices of initial values (i.e. simulated annealing).

```
Input: Stepsize $l$, Weight Initial Value $v$
Output: A vector $\vec{w}$ of learned weights
Initialize the weight values $w_i$ to $v$.
for $i \leftarrow 1 \ to \ n$ do
    $rg1 = rg2 = prev = 0$
    while (true) do
        scoreSentences($\vec{w}$)
        generateSummaries()
        $rg2$ = evaluateROUGE()
        if $rg1 \leq rg2$ then
            prev = $w_i$
            $w_i + = l$
            $rg1 = rg2$
        else
            break
        end
    end
end
return $\vec{w}$
```

Algorithm 1: Tuning weights using Local Search technique

Once we have learned the feature-weights, our *empirical* method computes the final scores for the sentences using the formula:

$$score_i = \vec{x_i}.\vec{w} \qquad (1)$$

Where, $\vec{x_i}$ is the feature vector for i-th sentence, $\vec{w}$ is the weight vector and $score_i$ is the score of i-th sentence.

4.2 K-means Learning

We start with a set of initial cluster centers and go through several iterations of assigning each object to the cluster whose center is closest. After all objects have been assigned, we recompute the center of each cluster as the centroid or mean (μ) of its members.

Once we have learned the means of the clusters using the k-means algorithm, our next task is to rank the sentences according to a probability model. We have used Bayesian model in order to do so. Bayes' law says:

$$P(q_k|\vec{x}, \Theta) = \frac{p(\vec{x}|q_k, \Theta)P(q_k|\Theta)}{\sum_{k=1}^{K} p(\vec{x}|q_k, \Theta)p(q_k|\Theta)} \qquad (2)$$

where q_k is a class, $\vec{x}$ is a feature vector representing a sentence and Θ is the parameter set of all class models. We set the weights of the clusters as equiprobable (i.e. $P(q_k|\Theta) = 1/K$). We calculated

$p(x|q_k, \Theta)$ using the gaussian probability distribution. The gaussian probability density function (pdf) for the d-dimensional random variable $\vec{x}$ is given by:

$$p_{(\mu,\Sigma)}(\vec{x}) = \frac{e^{\frac{-1}{2}(\vec{x}-\mu)^T\Sigma^{-1}(\vec{x}-\mu)}}{\sqrt{2\pi}^d\sqrt{det(\Sigma)}} \tag{3}$$

where μ, the mean vector and Σ, the covariance matrix are the parameters of the gaussian distribution. We get the means (μ) from the k-means algorithm and we calculate the covariance matrix using the unbiased covariance estimation:

$$\hat{\Sigma} = \frac{1}{N-1}\sum_{i=1}^{N}(x_j - \mu_j)(x_i - \mu_i)^T \tag{4}$$

4.3 EM Learning

EM is an iterative two step procedure: 1. Expectation-step and 2. Maximization-step. In the expectation step, we compute expected values for the hidden variables $h_{i,j}$ which are cluster membership probabilities. Given the current parameters, we compute how likely an object belongs to any of the clusters. The maximization step computes the most likely parameters of the model given the cluster membership probabilities. The data-points are considered to be generated by a mixture model of k-gaussians of the form:

$$P(\vec{x}) = \sum_{i=1}^{k} P(C = i)P(\vec{x}|\mu_i, \Sigma_i) \tag{5}$$

Where the total likelihood of model Θ with k components given the observed data points, $X = x_1, \cdots, x_n$ is:

$$\begin{aligned} L(\Theta|X) &= \prod_{i=1}^{n}\sum_{j=1}^{k} P(C = j)P(x_i|\Theta_j) \\ &= \prod_{i=1}^{n}\sum_{j=1}^{k} w_j P(x_i|\mu_j, \Sigma_j) \\ &\Leftrightarrow \sum_{i=1}^{n} log \sum_{j=1}^{k} w_j P(x_i|\mu_j, \Sigma_j) \end{aligned}$$

where P is the probability density function (i.e. eq 3). μ_j and Σ_j are the mean and covariance matrix of component j, respectively. Each component contributes a proportion, w_j, of the total population, such that: $\sum_{j=1}^{K} w_j = 1$.

However, a significant problem with the EM algorithm is that it converges to a local maximum of the likelihood function and hence the quality of the result depends on the initialization. In order to get good results from using random starting values, we can run the EM algorithm several times and choose the initial configuration for which we get the maximum log likelihood among all configurations. Choosing the best one among several runs is very computer intensive process. So, to improve the outcome of the EM algorithm on gaussian mixture models it is necessary to find a better method of estimating initial means for the components. To achieve this aim we explored the widely used "k-means" algorithm as a cluster (means) finding method. That means, the means found by k-means clustering above will be utilized as the initial means for EM and we calculate the initial covariance matrices using the unbiased covariance estimation procedure (eq:4).

Once the sentences are clustered by EM algorithm, we filter out the sentences which are not query-relevant by checking their probabilities, $P(q_r|x_i, \Theta)$ where, q_r denotes the cluster "query-relevant". If for a sentence x_i, $P(q_r|x_i, \Theta) > 0.5$ then x_i is considered to be query-relevant.

Our next task is to rank the query-relevant sentences in order to include them in the summary. This can be done easily by multiplying the feature vector $\vec{x_i}$ with the weight vector $\vec{w}$ that we learned by the local search technique (eq:1).

5 Redundancy Checking

When many of the competing sentences are included in the summary, the issue of information overlap between parts of the output comes up, and a mechanism for addressing redundancy is needed. Therefore, our summarization systems employ a final level of analysis: before being added to the final output, the sentences deemed to be important are compared to each other and only those that are not too similar to other candidates are included in the final answer or summary. Following (Zhou et al., 2005), we modeled this by BE overlap between an intermediate summary and a to-be-added candidate summary sentence. We call this overlap ratio R, where R is between 0 and 1 inclusively. Setting $R = 0.7$ means that a candidate summary sentence, s, can be added to an intermediate summary, S, if the sentence has a BE overlap ratio less than or equal to 0.7.

6 Experimental Evaluation

6.1 Evaluation Setup

We used the main task of Document Understanding Conference (DUC) 2007 for evaluation. The task was: *"Given a complex question (topic description) and a collection of relevant documents, the task is to synthesize a fluent, well-organized 250-word summary of the documents that answers the question(s) in the topic."*

NIST assessors developed topics of interest to them and choose a set of 25 documents relevant (document cluster) to each topic. Each topic and its document cluster were given to 4 different NIST assessors. The assessor created a 250-word summary of the document cluster that satisfies the information need expressed in the topic statement. These multiple "reference summaries" are used in the evaluation of summary content.

We carried out automatic evaluation of our summaries using ROUGE (Lin, 2004) toolkit, which has been widely adopted by DUC for automatic summarization evaluation. It measures summary quality by counting overlapping units such as the n-grams (ROUGE-N), word sequences (ROUGE-L and ROUGE-W) and word pairs (ROUGE-S and ROUGE-SU) between the candidate summary and the reference summary. ROUGE parameters were set as the same as DUC 2007 evaluation setup.

One purpose of our experiments is to study the impact of different features for complex question answering task. To accomplish this, we generated summaries for the topics of DUC 2007 by each of our seven systems defined as below:

The **LEX** system generates summaries based on only lexical features: n-gram (n=1,2,3,4), LCS, WLCS, skip bi-gram, head, head synonym. The **LSEM** system considers only lexical semantic features: synonym, hypernym/hyponym, gloss, dependency-based and proximity-based similarity. The **COS** system generates summary based on the graph-based method. The **SYS1** system considers all the features except the BE, syntactic and semantic features. The **SYS2** system considers all the features except the syntactic and semantic features. The **SYS3** considers all the features except the semantic and the **ALL**[6] system generates summaries taking all the features into account.

6.2 Evaluation Results

Table 1[7] to Table 3, Table 4 to Table 6 and Table 7 to Table 9 show the evaluation measures for k-means, EM and empirical approaches respectively. As Table 1 shows, in k-means, SYS2 gets 0-21%, SYS3 gets 4-32% and ALL gets 3-36% improvement in *ROUGE-2* scores over the SYS1 system. We get best *ROUGE-W* (Table 2) scores for SYS2 (i.e. including BE) but SYS3 and ALL do not perform well in this case. SYS2 improves the ROUGE-W F-score by 1% over SYS1. We do not get any improvement in *ROUGE-SU* (Table 3) scores when we include any kind of syntactic/semantic structures.

The case is different for EM and empirical approaches. Here, in every case we get a significant amount of improvement when we include the syntactic and/or semantic features. For EM (Table 4 to Table 6), the ratio of improvement in F-scores over SYS1 is: 1-3% for SYS2, 3-15% for SYS3 and 2-24% for ALL. In our empirical approach (Table 7 to Table 9), SYS2, SYS3 and ALL improve the F-scores by 3-11%, 7-15% and 8-19% over SYS1 respectively. These results clearly indicate the positive impact of the syntactic/semantic features for complex question answering task.

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.074	0.077	0.086	0.075	0.075	0.078	0.077
P	0.081	0.084	0.093	0.081	0.098	0.107	0.110
F	0.078	0.080	0.089	0.078	0.085	0.090	0.090

Table 1: ROUGE-2 measures in k-means learning

Table 10 shows the F-scores of the ROUGE measures for one baseline system, the best system in DUC 2007 and our three scoring techniques considering all features. The baseline system gener-

[6] SYS2, SYS3 and ALL systems show the impact of BE, syntactic and semantic features respectively

[7] R stands for Recall, P stands for Precision and F stands for F-score

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.098	0.097	0.101	0.099	0.101	0.097	0.097
P	0.195	0.194	0.200	0.237	0.233	0.241	0.237
F	0.130	0.129	0.134	0.140	0.141	0.139	0.138

Table 2: ROUGE-W measures in k-means learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.131	0.127	0.139	0.136	0.135	0.135	0.135
P	0.155	0.152	0.162	0.176	0.171	0.174	0.174
F	0.142	0.139	0.150	0.153	0.151	0.152	0.152

Table 3: ROUGE-SU in k-means learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.089	0.080	0.087	0.085	0.085	0.089	0.091
P	0.096	0.087	0.094	0.092	0.095	0.116	0.138
F	0.092	0.083	0.090	0.088	0.090	0.101	0.109

Table 4: ROUGE-2 measures in EM learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.103	0.096	0.101	0.102	0.101	0.102	0.101
P	0.205	0.193	0.200	0.203	0.218	0.222	0.223
F	0.137	0.128	0.134	0.136	0.138	0.139	0.139

Table 5: ROUGE-W measures in EM learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.146	0.128	0.138	0.143	0.144	0.145	0.144
P	0.171	0.153	0.162	0.168	0.177	0.186	0.185
F	0.157	0.140	0.149	0.154	0.159	0.163	0.162

Table 6: ROUGE-SU measures in EM learning

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.086	0.080	0.087	0.087	0.090	0.095	0.099
P	0.093	0.087	0.094	0.094	0.112	0.115	0.116
F	0.089	0.083	0.090	0.090	0.100	0.104	0.107

Table 7: ROUGE-2 in empirical approach

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.102	0.096	0.101	0.102	0.102	0.104	0.105
P	0.203	0.193	0.200	0.204	0.239	0.246	0.247
F	0.135	0.128	0.134	0.137	0.143	0.147	0.148

Table 8: ROUGE-W in empirical approach

Score	LEX	LSEM	COS	SYS1	SYS2	SYS3	ALL
R	0.144	0.129	0.138	0.145	0.146	0.149	0.150
P	0.169	0.153	0.162	0.171	0.182	0.195	0.197
F	0.155	0.140	0.150	0.157	0.162	0.169	0.170

Table 9: ROUGE-SU in empirical approach

ates summaries by returning all the leading sentences (up to 250 words) in the $\langle TEXT \rangle$ field of the most recent document(s). It shows that the empirical approach outperforms the other two learning techniques and EM performs better than k-means algorithm. EM improves the F-scores over k-means by 0.7-22.5%. Empirical approach improves the F-scores over k-means and EM by 5.9-20.2% and 3.5-6.5% respectively. Comparing with the DUC 2007 participants our systems achieve top scores and for some ROUGE measures there is no statistically significant difference between our system and the best DUC 2007 system.

System	ROUGE-1	ROUGE-2	ROUGE-W	ROUGE-SU
Baseline	0.335	0.065	0.114	0.113
Best	0.438	0.122	0.153	0.174
k-means	0.390	0.090	0.138	0.152
EM	0.399	0.109	0.139	0.162
Empirical	0.413	0.107	0.148	0.170

Table 10: F-measures for different systems

7 Conclusion and Future Work

Our experiments show the following: (a) our approaches achieve promising results, (b) empirical approach outperforms the other two learning and EM performs better than the k-means algorithm for this particular task, and (c) our systems achieve better results when we include BE, syntactic and semantic features.

In future, we have the plan to decompose the complex questions into several simple questions before measuring the similarity between the document sentence and the query sentence. We expect that by decomposing complex questions into the sets of subquestions that they entail, systems can improve the average quality of answers returned and achieve better coverage for the question as a whole.

References

M. Collins and N. Duffy. 2001. Convolution Kernels for Natural Language. In *Proceedings of Neural Information Processing Systems*, pages 625–632, Vancouver, Canada.

G. Erkan and D. R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. 2003. Shallow Semantic Parsing Using Support Vector Machines. In *Technical Report TR-CSLR-2003-03*, University of Colorado.

S. Harabagiu, F. Lacatusu, and A. Hickl. 2006. Answering complex questions with random walk models. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 220 – 227. ACM.

T. Hirao, , J. Suzuki, H. Isozaki, and E. Maeda. 2004. Dependency-based sentence alignment for multiple document summarization. In *Proceedings of Coling 2004*, pages 446–452, Geneva, Switzerland. COLING.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *Proceedings of the international conference on Language Resources and Evaluation*, Las Palmas, Spain.

M. Kouylekov and B. Magnini. 2005. Recognizing textual entailment with tree edit distance algorithms. In *Proceedings of the PASCAL Challenges Workshop: Recognising Textual Entailment Challenge*.

J. Li, L. Sun, C. Kit, and J. Webster. 2007. A Query-Focused Multi-Document Summarizer Based on Lexical Chains. In *Proceedings of the Document Understanding Conference*, Rochester. NIST.

C. Y. Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of Association for Computational Linguistics*, pages 74–81, Barcelona, Spain.

B. MacCartney, T. Grenager, M.C. de Marneffe, D. Cer, and C. D. Manning. 2006. Learning to recognize features of valid textual entailments. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, page 4148, New York, USA.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. 2007. Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classificaion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 776–783, Prague, Czech Republic. ACL.

J. Otterbacher, G. Erkan, and D. R. Radev. 2005. Using Random Walks for Question-focused Sentence Retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 915–922, Vancouver, Canada.

P. Pingali, Rahul K., and V. Varma. 2007. IIIT Hyderabad at DUC 2007. In *Proceedings of the Document Understanding Conference*, Rochester. NIST.

V. Punyakanok, D. Roth, and W. Yih. 2004. Mapping dependencies trees: An application to question answering. In *Proceedings of AI & Math*, Florida, USA.

K. Toutanova, C. Brockett, M. Gamon, J. Jagarlamudi, H. Suzuki, and L. Vanderwende. 2007. The PYTHY Summarization System: Microsoft Research at DUC 2007 . In *proceedings of the Document Understanding Conference*, Rochester. NIST.

D. Zhang and W. S. Lee. 2003. A Language Modeling Approach to Passage Question Answering. In *Proceedings of the Twelfth Text REtreival Conference*, pages 489–495, Gaithersburg, Maryland.

L. Zhou, C. Y. Lin, and E. Hovy. 2005. A BE-based Multi-dccument Summarizer with Query Interpretation. In *Proceedings of Document Understanding Conference*, Vancouver, B.C., Canada.