Exploiting Syntactic and Shallow Semantic Kernels to Improve Random Walks for Complex Question Answering

Yllias Chali and Shafiq R. Joty
Department of Computer Science
University of Lethbridge
Lethbridge, Alberta, Canada, T1K 3M4
{chali,jotys}@cs.uleth.ca

Abstract

*We consider the problem of answering complex questions that require inferencing and synthesizing information from multiple documents and can be seen as a kind of topic-oriented, informative multi-document summarization. The stochastic, graph-based method for computing the relative importance of textual units (i.e. sentences) is very successful in generic summarization. In this method, a sentence is encoded as a vector in which each component represents the occurrence frequency (TF*IDF) of a word. However, the major limitation of the TF*IDF approach is that it only retains the frequency of the words and does not take into account the sequence, syntactic and semantic information. In this paper, we study the impact of syntactic and shallow semantic information in the graph-based method for answering complex questions. Experimental results show the effectiveness of the syntactic and shallow semantic information for this task.*

1. Introduction

Automated Question Answering (QA) -the ability of a machine to answer questions, simple or complex, posed in ordinary human language-is perhaps the most exciting technological development of the past six or seven years. 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, organization, locations, dates, etc.) from documents. Unlike informationally-simple factoid questions, complex questions often seek multiple different types of information simultaneously and do not presuppose that one single answer could meet all of its information needs. For example, with a factoid question like "How accurate are HIV tests?", it can be safely assumed that the submitter of the question is looking for a number or a range of numbers. However, with complex questions like "What are the causes of AIDS?", the wider focus of this question suggests that the submitter may not have a single or well-defined information need and therefore may be amenable to receiving additional supporting information that is relevant to some (as yet) undefined informational goal. This type of questions require inferencing and synthesizing information from multiple documents. This information synthesis in Natural Language Processing (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. Unlike indicative summaries (which help to determine whether a document is relevant to a particular topic), informative summaries must be helpful to answer, for instance, factual questions about the topic [1].

Recently, the graph-based methods (such as LexRank [4], TextRank [11]) are applied successfully to generic, multi-document summarization. A topic-sensitive LexRank is proposed in [14]. In this method, a sentence is mapped to a vector in which each element represents the occurrence frequency (TF*IDF) of a word. However, the major limitation of the TF*IDF approach is that it only retains the frequency of the words and does not take into account the sequence, syntactic and semantic structure thus cannot distinguish between "The hero killed the villain" and "The villain killed the hero". Textual similarity measure is a key part in NLP algorithms. If a well-suited measure improves performance, a bad measure will definitely lead to irrelevant results. Defining a good measure is not a straight-forward process. Indeed, the measure should give a fine indication of how much two sentences are similar.

The task like *answering complex questions* that requires the use of more complex syntactic and semantics, the approaches with only TF*IDF are often inadequate to perform fine-level textual analysis. The importance of syntactic and semantic features in this context is described by [15], [13] and [12]. An effective way to integrate syntactic and semantic structures in machine learning algorithms is the use of *tree kernel* functions [3] which has been successfully applied to question classification [15], [12]. In more complex tasks such as computing the relatedness between query sentences and document sentences in order to rank sentences for answering complex questions, to our knowledge no study uses kernel functions to encode syntactic/semantic information. Moreover, the study of shallow semantic information such as predicate argument structures annotated in the PropBank (PB) project [8] is a promising research direction.

In this paper, we extensively study the impact of syntactic and shallow semantic information in measuring similarity between the sentences in the random walk framework for answering complex questions. We argue that for this task, similarity measures based on syntactic and semantic information performs better and can be used to characterize the relation between a question and a sentence (answer) in a more effective way than the traditional TF*IDF based similarity measures.

2. Graph-based Methods for Summarization

In [4], 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 nodes measure the cosine similarity between the respective pair of sentences where each sentence is represented as a vector of term specific weights. The term specific weights in the sentence vectors are products of local and global parameters. The model is known as term frequency-inverse document frequency (tf-idf) model. The weight vector for a sentence s is $\vec{v_s} = [w_{1,s}, w_{2,s}, \ldots, w_{N,s}]^T$, where

$$w_{t,s} = tf_t \times \log \frac{|S|}{|\{t \in s\}|}$$

and

- tf_t is term frequency (tf) of term t in sentence s (a local parameter)
- $\log \frac{|S|}{|\{t \in s\}|}$ is inverse document frequency (idf) (a global parameter). $|S|$ is the total number of sentences in the corpus; $|\{t \in s\}|$ is the number of sentences containing the term t.

The degree of a given node is an indication of how much important the sentence is. Figure 1 shows an example of a similarity graph for 4 sentences.

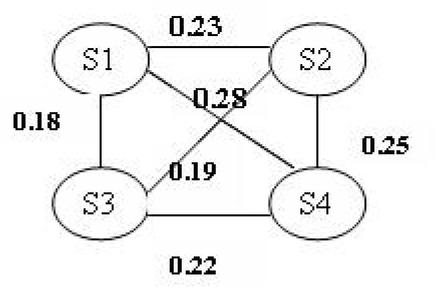

Figure 1. LexRank similarity

Once the similarity graph is constructed, the sentences are then ranked according to their eigenvector centrality. The LexRank performed well in the context of generic summarization.

To apply LexRank to query-focused context, a topic-sensitive version of LexRank is proposed in [14]. 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.

2.1. Relevance to the question

Following [14], we first stem out all the sentences in the collection and compute the word IDFs using the following formula:

$$idf_w = log\left(\frac{N+1}{0.5+sf_w}\right)$$

Where, *N* is the total number of sentences in the document cluster, and sf_w is the number of sentences that the word *w* appears in.

We also stem out the questions and remove the stop words. The relevance of a sentence s to the question q is computed by:

$$rel(s|q) = \sum_{w \in q} log\,(tf_{w,s}+1) \times log\,(tf_{w,q}+1) \times idf_w$$

Where, $tf_{w,s}\ and\ tf_{w,q}$ are the number of times w appears in s and q, respectively.

2.2. Mixture Model

In the previous section, we measured the relevance of a sentence to the question but a sentence that is similar to the high scoring sentences in the cluster should also have a high score. For instance, if a sentence that gets high score based on the question relevance model is likely to contain an answer to the question, then a related sentence, which may not be similar to the question itself, is also likely to contain an answer. This idea is captured by the following mixture model [14]:

$$p(s|q) = d \times \frac{rel(s|q)}{\sum_{z \in C} rel(z|q)} + (1-d) \times \sum_{v \in C} \frac{sim(s,v)}{\sum_{z \in C} sim(z,v)} \times p(v|q) \quad (1)$$

Where, $p(s|q)$ is the score of a sentence *s* given a question *q*, is determined as the sum of its relevance to the question and the similarity to the other sentences in the collection. C is the set of all sentences in the collection. The value of the parameter *d* which we call “bias”, is a trade-off between two terms in the equation and is set empirically. For higher values of *d*, we prefer the relevance to the question to similarity to other sentences. The denominators in both terms are for normalization. We measure the cosine similarity weighted by word IDFs as the similarity between two sentences in a cluster:

$$sim(x,y) = \frac{\sum_{w \in x,y} tf_{w,x} tf_{w,y} (idf_w)^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} idf_{x_i})^2} \sqrt{\sum_{y_i \in y} (tf_{y_i,y} idf_{y_i})^2}}$$

Following [14], Equation 1 can be written in matrix notation as follows:

$$\mathbf{p} = [d\mathbf{A} + (1-d)\mathbf{B}]^T \mathbf{p} \quad (2)$$

A is the square matrix such that for a given index *i*, all the elements in the i^{th} column are proportional to $rel(i|q)$. **B** is also a square matrix such that each entry **B***(i,j)* is proportional to *sim(i,j)*. Both matrices are normalized so that row sums add up to 1. Note that as a result of this normalization, all rows of the resulting square matrix $Q = [dA + (1-d)B]$ also add up to 1. Such a matrix is called *stochastic* and defines a Markov chain. If we view each sentence as a state in a Markov chain, then Q(i,j) specifies the transition probability from state *i* to state *j* in the corresponding Markov chain. The vector **p** we are looking for in equation 2 is the stationary distribution of the Markov chain. An intuitive interpretation of the stationary distribution can be understood by the concept of a random walk on the graph representation of the Markov chain.

With probability *d*, a transition is made from the current node to the nodes that are similar to the query. With probability *(1-d)*, a transition is made to the nodes that are lexically similar to the current node. Every transition is weighted according to the similarity distributions. Each element of the vector **p** gives the asymptotic probability of ending up at the corresponding state in the long run regardless of the starting state. The stationary distribution of a markov chain can be computed by a simple iterative algorithm, called power method [4]. The power method starts with a uniform distribution. At each iteration, the eigenvector is updated by multiplying with the transpose of the stochastic matrix. Since the Markov chain is irreducible and aperiodic, the algorithm is guaranteed to terminate.

We claim that for a complex task like answering complex questions where the relatedness between the query sentences and the document sentences is an important factor, the graph-based method of ranking sentences would perform better if we could encode the syntactic and semantic information instead of just the BOW (i.e. TF*IDF) information in calculating the similarity between sentences. Thus, our mixture model for answering complex questions is:

$$p(s|q) = d \times TREESIM(s,q) + (1-d) \times \sum_{v \in C} TREESIM(s,v) \times p(v|q) \quad (3)$$

Where *TREESIM(s,q)* is the normalized syntactic (and/or semantic) similarity between the *query (q)* and *the document sentence (s)* and *C* is the set of all sentences in the collection. In cases where the query is composed of two or more sentences, we compute the similarity between the document sentence (s) and each of the query-sentences (q_i) then we take the average of the scores. In the next two sections, we describe how we can encode syntactic and semantic structures in calculating the similarity between sentences.

3. Encoding Syntactic and Shallow Semantic Structures

Encoding syntactic structure is easier and straight forward. Given a sentence (or query), we first parse it into a syntactic tree using a parser like [2] and then we calculate the similarity between the two trees using the *tree kernel* (discussed in Section 4.1).

Though introducing syntactic information gives an improvement on BOW by the use of 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 [13].

Initiatives such as PropBank (PB) [8] have made possible the design of accurate automatic Semantic Role Labeling (SRL) systems like ASSERT [6]. Attempting an application of SRL to QA hence seems natural, as pinpointing the answer to a question relies on a deep understanding of the semantics of both.

For example, consider the PB annotation:

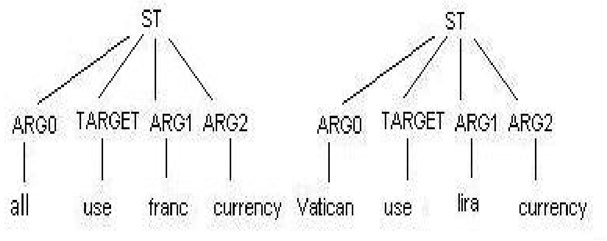

Figure 2. Example of semantic trees

```
[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 (or query) using the tree structures like Figure 2 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. We look for noun first, then verb, then adjective, then adverb to find the semantic head in the argument. If none of these is present, we take the first word of the argument as the semantic head. This reduces the data sparseness with respect to a typical BOW representation.

However, sentences rarely contain a single predicate: it happens more generally that propositions contain one or more subordinate clauses. For instance, let us consider a slight modification of the second sentence: "the Vatican, located wholly within Italy uses the Italian lira as their currency." Here, the main predicate is "uses" and the subordinate predicate is "located". The SRL system outputs the following two annotations:

```
(1) [ARG0 the Vatican located wholly
within Italy][TARGET uses][ARG1 the
Italian lira][ARG2 as their currency]
(2) [ARG0 the Vatican][TARGET located]
[ARGM-LOC wholly][ARGM-LOC within Italy]
uses the Italian lira as their currency
```

giving the STs in Figure 3. As we can see in Figure 3(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 3(B). If taken separately, such STs do not express

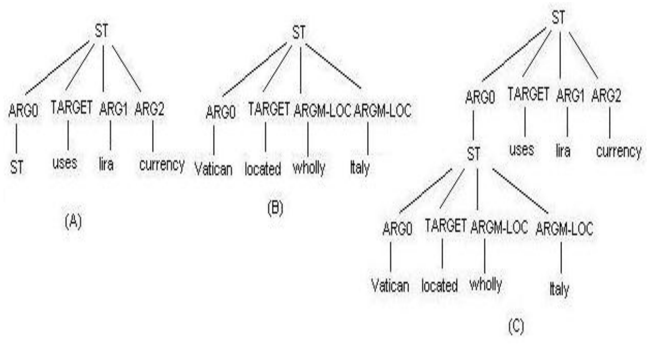

Figure 3. Two STs composing a STN

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 3(C). We refer to this kind of nested STs as STNs.

4. Syntactic and Semantic Kernels for Text

4.1. Tree Kernels

Once we build the trees (syntactic or semantic), our next task is to measure the similarity between the trees. For this, every tree T is represented by an m dimensional vector $v(T) = (v_1(T), v_2(T), \cdots v_m(T))$, where the i-th element $v_i(T)$ is the number of occurrences of the i-th tree fragment in tree T. The tree fragments of a tree are all of its subtrees which include at least one production with the restriction that no production rules can be broken into incomplete parts.

Figure 4 shows an example tree and a portion of its subtrees.

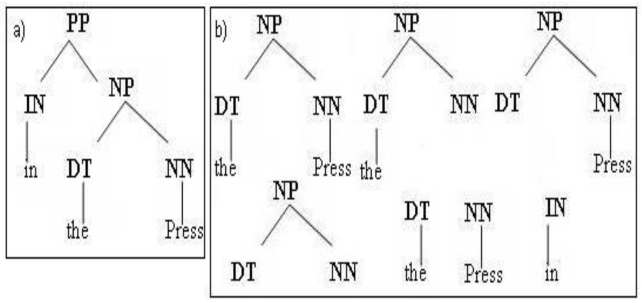

Figure 4. (a) An example tree (b) The subtrees of the NP covering "the press".

Implicitly we enumerate all the possible tree fragments $1, 2, \cdots, m$. These fragments are the axis of this m-dimensional space. Note that this could be done only implicitly, since the number m is extremely large. Because of this, [3] defines the tree kernel algorithm whose computational complexity does not depend on m.

The tree kernel of two trees T_1 and T_2 is actually the inner product of $v(T_1)$ and $v(T_2)$:

$$TK(T_1, T_2) = v(T_1).v(T_2) \tag{4}$$

We define the indicator function $I_i(n)$ to be 1 if the subtree i is seen rooted at node n and 0 otherwise. It follows:

$$v_i(T_1) = \sum_{n_1 \in N_1} I_i(n_1)$$
$$v_i(T_2) = \sum_{n_2 \in N_2} I_i(n_2)$$

Where N_1 and N_2 are the set of nodes in T_1 and T_2 respectively. So, we can derive:

$$\begin{aligned} TK(T_1, T_2) &= v(T_1).v(T_2) \\ &= \sum_i v_i(T_1) v_i(T_2) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) I_i(n_2) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} C(n_1, n_2) \qquad (5) \end{aligned}$$

where we define $C(n_1, n_2) = \sum_i I_i(n_1) I_i(n_2)$. Next, we note that $C(n_1, n_2)$ can be computed in polynomial time, due to the following recursive definition:

1. If the productions at n_1 and n_2 are different then $C(n_1, n_2) = 0$
2. If the productions at n_1 and n_2 are the same, and n_1 and n_2 are pre-terminals, then $C(n_1, n_2) = 1$
3. Else if the productions at n_1 and n_2 are not pre-terminals,

$$C(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + C(ch(n_1, j), ch(n_2, j))) \tag{6}$$

where, $nc(n_1)$ is the number of children of n_1 in the tree; because the productions at n_1 and n_2 are the same, we have $nc(n_1) = nc(n_2)$. The i-th child-node of n_1 is $ch(n_1, i)$.

Note that, the tree kernel (TK) function 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. 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 in Section 3. For instance, although the two STs of Figure 2 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 steps (1), (2) and (3) 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. [13] solve this problem by designing the Shallow Semantic Tree Kernel (SSTK) which allows to match portions of a ST.

4.2. Shallow Semantic Tree Kernel (SSTK)

The SSTK is based on two ideas: first, it changes the ST, as shown in Figure 5 by adding *SLOT* nodes. These accommodate argument labels in a specific order i.e. it provides a fixed number of slots, possibly filled with *null* arguments, that encode all possible predicate arguments. Leaf nodes are filled with the wildcard character * but they may alternatively accommodate additional information. The slot nodes are used in such a way that the adopted TK function can generate fragments containing one or more children like for example those shown in frames (b) and (c) of Figure 5. As previously pointed out, if the arguments were directly attached to the root node, the kernel function would only generate the structure with all children (or the structure with no children, i.e. empty).

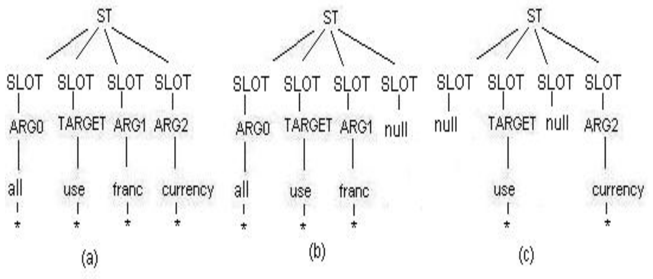

Figure 5. Semantic tree with some of its fragments

Second, as the original tree kernel would generate many matches with slots filled with the null label, we have set a new step 0 in the TK calculation:

(0) if n_1 (or n_2) is a pre-terminal node and its child label is *null*, $C(n_1, n_2) = 0$;

and subtract one unit to $C(n_1, n_2)$, in step 3:

$$(3)\ C(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + C(ch(n_1, j), ch(n_2, j))) - 1$$

The above changes generate a new C which, when substituted (in place of original C) in Eq. 5, gives the new SSTK.

5 Redundancy Checking and Generating Summary

Once the sentences are scored by the mixture model, the easiest way to create summaries is just to output the topmost N sentences until the required summary length is reached. In that case, we are ignoring other factors: such as redundancy and coherence.

As it is described that text summarization clearly entails selecting the most salient information and putting it together in a coherent summary. The answer or summary consists of multiple separately extracted sentences from different documents. Obviously, each of the selected text snippets should individually be important. However, 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 two levels of analysis: first, a content level, where every sentence is scored according to the features or concepts it covers and second, a textual level, when, 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. [5] observed this in what the authors called "Maximum-Marginal-Relevancy (MMR)". Following [7], 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. Experiments

6.1. Evaluation Setup

Over the past three years, complex questions have been the focus of much attention in both the automatic question-answering and multi-document summarization (MDS) communities. While most current complex QA evaluations (including the 2004 AQUAINT Relationship QA Pilot, the 2005 Text Retrieval Conference (TREC) Relationship QA Task, and the 2006 GALE Distillation Effort) require systems to return unstructured lists of candidate answers in response to a complex question, recent MDS evaluations (including the 2005, 2006 and 2007 Document Understanding Conferences (DUC)) have tasked systems with returning paragraph-length answers to complex questions that are responsive, relevant, and coherent.

The DUC conference series is run by the National Institute of Standards and Technology (NIST) to further progress in summarization and enable researchers to participate in large-scale experiments. We used the main task of 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, including the developer of the topic. 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 [9] toolkit (i.e. ROUGE-1.5.5 in this study) for evaluation, which has been widely adopted by DUC for automatic summarization evaluation. It measures summary quality by counting overlapping units such as the n-gram (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 toolkit reports separate scores for n-grams (n=1, 2, 3, and 4), longest common subsequence (LCS), weighted longest common subsequence (WLCS) co-occurrences and skip bi-gram (SB) co-occurrences. Among these different scores, unigram-based ROUGE score (ROUGE-1) has been shown to agree with human judgment most [10]. We showed four of the ROUGE metrics in the experimental results: ROUGE-1 (unigram), ROUGE-L (LCS), ROUGE-W (weighted LCS with $weight = 1.2$) and ROUGE-SU (skip bi-gram).

ROUGE parameters were set as the same as DUC 2007 evaluation setup. All the ROUGE measures were calculated by running ROUGE-1.5.5 with stemming but no removal of stopwords.

ROUGE run-time parameters: ROUGE-1.5.5.pl -2 -1 -u -r 1000 -t 0 -n 4 -w 1.2 -m -l 250 -a

The purpose of our experiments is to study the impact of the syntactic and semantic representation introduced earlier for complex question answering task. To accomplish this, we generate summaries for 20 topics of DUC 2007 by each

of our four systems defined as below:
(1) TF*IDF: This system is the original topic-sensitive LexRank described in Section 2 that uses the similarity measures based on tf*idf (BOW) and does not consider the syntactic/semantic information. The mixture model for this system is given in Eq 1.
(2) SYN: This system measures the similarity between the sentences using the *syntactic tree* and the *general tree kernel* function defined in Section 4.1. The mixture model for this system is:

$$\begin{aligned} p(s|q) &= d \times SYNSIM(s,q) + (1-d) \\ &\times \sum_{v \in C} SYNSIM(s,v) \times p(v|q) \end{aligned} \tag{7}$$

Where *SYNSIM(s,q)* is the normalized syntactic similarity between the *query (q)* and *the document sentence (s)* and C is the set of all sentences in the collection.
(3) SEM: This system measures the similarity between the sentences using the *shallow semantic tree* and the *shallow semantic tree kernel* function defined in Section 4.2. Therefore, the mixture model for this system is:

$$\begin{aligned} p(s|q) &= d \times SEMSIM(s,q) + (1-d) \\ &\times \sum_{v \in C} SEMSIM(s,v) \times p(v|q) \end{aligned} \tag{8}$$

Where *SEMSIM(s,q)* is the normalized shallow semantic similarity between the *query (q)* and *the document sentence (s)* and C is the set of all sentences in the collection.
(4) SYNSEM: This system measures the similarity between the sentences using both the *syntactic* and *shallow semantic* trees and their associated *kernels*. Hence, the mixture model for this system is:

$$\begin{aligned} p(s|q) &= d \times SYNSEM(s,q) + (1-d) \\ &\times \sum_{v \in C} SYNSEM(s,v) \times p(v|q) \end{aligned} \tag{9}$$

Where,

$$SYNSEM(s,q) = \frac{SYNSIM(s,q) + SEMSIM(s,q)}{2}$$

$$SYNSEM(s,v) = \frac{SYNSIM(s,v) + SEMSIM(s,v)}{2}$$

In our experiments, we set 0.7 as the value of d (bias) and R (overlap ratio).

6.2. Evaluation Results

Table 1 and Table 2 show the ROUGE scores of the TF*IDF and SYN systems respectively. It can be noticed that almost in every cases (except ROUGE-SU) the SYN system outperforms the TF*IDF system which proves the effectiveness of syntactic similarity over the tf*idf based similarity. Table 3 and Table 4 show the ROUGE scores of the SEM system and the SYNSEM system respectively. SEM system outperforms the TF*IDF and SYN systems in every measures. The SYNSEM system performs better than SYN system but not as good as the SEM system. It indicates that the shallow semantic similarity is more effective than the syntactic and tf*idf based similarity.

Measures	ROUGE-1	ROUGE-L	ROUGE-W	ROUGE-SU
Precision	0.376242	0.350540	0.185705	0.143747
Recall	0.344227	0.320673	0.093375	0.119785
F-score	0.359458	0.334882	0.124226	0.130603

Table 1. ROUGE measures for TF*IDF system

Measures	ROUGE-1	ROUGE-L	ROUGE-W	ROUGE-SU
Precision	0.384994	0.350604	0.188580	0.140672
Recall	0.355740	0.323993	0.095684	0.119528
F-score	0.369677	0.336673	0.126890	0.129109

Table 2. ROUGE measures for SYN system

Measures	ROUGE-1	ROUGE-L	ROUGE-W	ROUGE-SU
Precision	0.409580	0.374844	0.198894	0.161490
Recall	0.372191	0.340608	0.099266	0.133258
F-score	0.389865	0.356792	0.132378	0.145859

Table 3. ROUGE measures for SEM system

The comparison between the systems in terms of their F-scores is given in Table 5. The SYN system improves the ROUGE-1, ROUGE-L and ROUGE-W scores over the TF*IDF system by 2.84%, 0.53% and 2.14% respectively. The SEM system improves the ROUGE-1, ROUGE-L, ROUGE-W, and ROUGE-SU scores over the TF*IDF system by 8.46%, 6.54%, 6.56%, and 11.68%, and over the SYN system by 5.46%, 5.98%, 4.33%, and 12.97% respectively. The SYNSEM system improves the ROUGE-1, ROUGE-L, ROUGE-W, and ROUGE-SU scores over the TF*IDF system by 4.64%, 1.63%, 2.15%, and 4.06%, and over the SYN system by 1.74%, 1.09%, 0%, and 5.26% respectively. The SEM system improves the ROUGE-1, ROUGE-L, ROUGE-W, and ROUGE-SU scores over the SYNSEM system by 3.65%, 4.84%, 4.32%, and 7.33% respectively which indicates that including syntactic feature

Measures	ROUGE-1	ROUGE-L	ROUGE-W	ROUGE-SU
Precision	0.392235	0.354921	0.188973	0.148440
Recall	0.361455	0.327040	0.095581	0.125499
F-score	0.376126	0.340330	0.126894	0.135901

Table 4. ROUGE measures for SYNSEM

with the semantic feature degrades the performance.

Our experimental results show that, the graph-based random walk method of generating query-relevant summaries performs best when we measure the similarity between the sentences (and query) using their semantic structures. Similarity measure based on the syntactic structure also outperforms the traditional tf*idf based measure for this task.

Systems	ROUGE-1	ROUGE-L	ROUGE-W	ROUGE-SU
TF*IDF	0.359458	0.334882	0.124226	0.130603
SYN	0.369677	0.336673	0.126890	0.129109
SEM	0.389865	0.356792	0.132378	0.145859
SYNSEM	0.376126	0.340330	0.126894	0.135901

Table 5. ROUGE F-scores for systems

7. Conclusion

In this paper, we have used the syntactic and shallow semantic structures and discussed their impacts in measuring the similarity between the sentences in the random walk framework for answering complex questions. We parsed the sentences into the syntactic trees using the Charniak parser and applied the general tree kernel function to measure the similarity between sentences. We have redefined shallow semantic trees (STs and STNs) to represent predicate argument relations, which we automatically extract using the ASSERT SRL system. We have used the shallow semantic tree kernel to measure the semantic similarity between two semantic trees.

Our experiments suggest the following: (a) similarity measures based on the syntactic tree and/or shallow semantic tree outperforms the similarity measures based on the TF*IDF and (b) similarity measures based on the shallow semantic tree performs best for this problem.

References

[1] E. Amigo, J. Gonzalo, V. Peinado, A. Peinado, and F. Verdejo. An Empirical Study of Information Synthesis Tasks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 207–es, Barcelona, Spain, 2004.

[2] E. Charniak. A Maximum-Entropy-Inspired Parser. In *Technical Report CS-99-12*, Brown University, Computer Science Department, 1999.

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

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

[5] J. Goldstein, M. Kantrowitz, V. Mittal, and J. Carbonell. Summarizing Text Documents: Sentence Selection and Evaluation Metrics. In *Proceedings of the 22nd International ACM Conference on Research and Development in Information Retrieval, SIGIR*, pages 121–128, Berkeley, CA, 1999.

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

[7] E. Hovy, C. Y. Lin, L. Zhou, and J. Fukumoto. Automated Summarization Evaluation with Basic Elements. In *Proceedings of the Fifth Conference on Language Resources and Evaluation*, Genoa, Italy, 2006.

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

[9] C. Y. Lin. 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, 2004.

[10] C. Y. Lin and E. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the Annual Meeting of the North American Association for Computational Linguistics*, pages 150–156, Edmonton, Canada, 2003.

[11] R. Mihalcea and P. Tarau. TextRank: Bringing Order into Texts. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing*, Barcelona, Spain, 2004.

[12] A. Moschitti and R. Basili. A Tree Kernel approach to Question and Answer Classification in Question Answering Systems. In *Proceedings of the 5th international conference on Language Resources and Evaluation*, Genoa, Italy, 2006.

[13] A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar. 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, 2007. ACL.

[14] J. Otterbacher, G. Erkan, and D. R. Radev. 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, 2005.

[15] A. Zhang and W. Lee. Question Classification using Support Vector Machines. In *Proceedings of the Special Interest Group on Information Retrieval*, pages 26–32, Toronto, Canada, 2003. ACM.