ARTICLE IN PRESS

Information Processing and Management xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels

Yllias Chali [a], Sadid A. Hasan [a,*], Shafiq R. Joty [b]

[a] University of Lethbridge, Lethbridge, AB, Canada
[b] University of British Columbia, Vancouver, BC, Canada

ARTICLE INFO

Article history:
Received 29 March 2009
Received in revised form 27 September 2010
Accepted 3 October 2010
Available online xxxx

Keywords:
Complex question answering
Graph-based method
Syntactic kernel
Shallow semantic kernel
Extended string subsequence kernel

ABSTRACT

The task of answering complex questions requires inferencing and synthesizing information from multiple documents that can be seen as a kind of topic-oriented, informative multi-document summarization. In generic summarization the stochastic, graph-based random walk method to compute the relative importance of textual units (i.e. sentences) is proved to be very successful. 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. This paper presents the impact of syntactic and semantic information in the graph-based random walk method for answering complex questions. Initially, we apply tree kernel functions to perform the similarity measures between sentences in the random walk framework. Then, we extend our work further to incorporate the Extended String Subsequence Kernel (ESSK) to perform the task in a similar manner. Experimental results show the effectiveness of the use of kernels to include the syntactic and semantic information for this task.

1. Introduction

The size of the publicly indexable world-wide-web has probably surpassed several billions of documents and as yet growth shows no sign of leveling off. The demand for access to different types of information have led to a renewed interest in a broad range of Information Retrieval (IR) related areas such as question answering, topic detection and tracking, summarization, multimedia retrieval (e.g., image, video and music), chemical and biological informatics, text structuring, text mining, genomics, etc.

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. QA research attempts to deal with a wide range of question types including fact, list, definition, how, why, hypothetical, semantically-constrained, and cross-lingual questions. Search collections vary from a small local document collections, to an internal organization documents, to compiled newswire reports, to the World Wide Web.

Some questions are easier to answer which we call *simple questions*. For example, the question, "Who is the president of Bangladesh?" asks for a person name. This type of questions (i.e. factoid) requires small snippets of text as the answers. Again, the question "Which countries has Pope John Paul II visited?" asks for a list of small snippets (i.e. list questions) of text. Finding answers to these questions are easier than questions that have complex information needs. After having made substantial headway in factoid and list questions, researchers have turned their attention to more complex information

* Corresponding author.
E-mail addresses: chali@cs.uleth.ca (Y. Chali), hasan@cs.uleth.ca (S.A. Hasan), rjoty@cs.ubc.ca (S.R. Joty).

0306-4573/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.ipm.2010.10.002

needs that cannot be answered by simply extracting named entities (persons, organization, locations, dates, etc.) from documents. Unlike 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 (Amigo, Gonzalo, Peinado, Peinado, & Verdejo, 2004).

The graph-based methods (such as LexRank Erkan & Radev, 2004, TextRank Mihalcea & Tarau, 2004) are applied successfully to generic, multi-document summarization. A topic-sensitive LexRank is proposed in Otterbacher, Erkan, and Radev (2005). 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, it cannot distinguish between "The hero killed the villain" and "The villain killed the hero". For 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 (Chali & Joty, 2008; Chali & Joty, 2008).

In this paper, we extensively study the impact of syntactic and semantic information in measuring similarity between the sentences in the random walk framework for answering complex questions. We apply the tree kernel functions and Extended String Subsequence Kernel (ESSK) to include syntactic and semantic information. We run our experiments on the DUC 2007 data and based on this we argue that for the complex question answering task, similarity measures based on syntactic and semantic information perform 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.

We organize the paper as follows: Section 2 focuses on the related work, Section 3 describes the graph-based model, Section 4 presents the methods to encode syntactic and shallow semantic structures, Section 5 discusses the syntactic and shallow semantic kernels, Section 6 discusses the theory of Extended String Subsequence Kernel, Section 7 describes the redundancy checking and summary generation module whereas Section 8 presents the experimental details with evaluation results and finally, Section 9 concludes the paper by cuing some future directions.

2. Related work

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. Indeed, this motivated us to include syntactic and semantic features to get the structural similarity between sentences. In MacCartney, Grenager, de Marneffe, Cer, and Manning (2006), they use typed dependency graphs (same as dependency trees) to represent the text and the hypothesis. They try to find a good partial alignment between the typed dependency graphs representing the hypothesis (contains n nodes) and the text (graph contains m nodes) in a search space of $O((m+1)n)$. They use an incremental beam search combined with a node ordering heuristic to do approximate global search in the space of possible alignments. A locally decomposable scoring function was chosen such that the score of an alignment is the sum of the local node and edge alignment scores. The scoring measure is designed to favor alignments which align semantically similar subgraphs, irrespective of polarity. For this reason, nodes receive high alignment scores when the words they represent are semantically similar. Synonyms and antonyms receive the highest score and unrelated words receive the lowest. Alignment scores also incorporate local edge scores which are based on the shape of the paths between nodes in the text graph which correspond to adjacent nodes in the hypothesis graph. In the final step they make a decision about whether or not the hypothesis is entailed by the text conditioned on the typed dependency graphs as well as the best alignment between them. To make this decision they use a supervised statistical logistic regression classifier (with a feature space of 28 features) with a Gaussian prior parameter for regularization.

The importance of syntactic and semantic features in finding textual similarity is described by Zhang and Lee (2003), Moschitti, Quarteroni, Basili, and Manandhar (2007) and Moschitti and Basili (2006). An effective way to integrate syntactic and semantic structures in machine learning algorithms is the use of *tree kernel* functions (Collins & Duffy, 2001) which has been successfully applied to question classification (Zhang & Lee, 2003; Moschitti & Basili, 2006). To the best of our knowledge, no study has used tree kernel functions to encode syntactic/semantic information for more complex tasks such as computing the relatedness between the query sentences and the document sentences. Another good way to encode some shallow syntactic information is the use of Basic Elements (BE) (Hovy, Lin, Zhou, & Fukumoto, 2006) which uses dependency relations. Moreover, the study of shallow semantic information such as predicate argument structures annotated in the PropBank (PB) project (Kingsbury & Palmer, 2002) is a promising research direction.

In Hirao, Suzuki, Isozaki, and Maeda (2004), they 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

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

sentences. They also introduce Extended String Subsequence Kernel (ESSK) to incorporate semantics in DTPs. In Kouylekov and Magnini (2005), they 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. In Punyakanok, Roth, and Yih (2004), they 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 Bag-Of-Words (BOW) scoring methods.

3. Graph-based methods for summarization

In 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 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. Fig. 1 shows an example of a similarity graph for four sentences.

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 Otterbacher et al. (2005). 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.

3.1. Relevance to the question

Following (Otterbacher et al., 2005), 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.

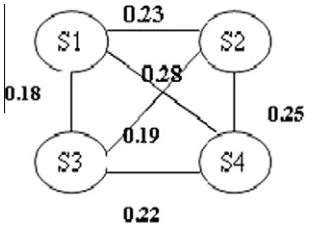

Fig. 1. LexRank similarity.

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

3.2. 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 (Otterbacher et al., 2005):

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

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 (Otterbacher et al., 2005), Eq. (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 ith column are proportional to $rel(i|q)$. **B** is also a square matrix such that each entry $\mathbf{B}(i,j)$ 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 Eq. (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 (Erkan & Radev, 2004). 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:

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

where $KERSIM(s,q)$ 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 three sections, we describe how we can encode syntactic and semantic structures in calculating the similarity between sentences.

4. 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 (Charniak, 1999) and then we calculate the similarity between the two trees using the *tree kernel* (discussed in Section 5.1).

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 (Moschitti et al., 2007).

Initiatives such as PropBank (PB) (Kingsbury & Palmer, 2002) have made possible the design of accurate automatic Semantic Role Labeling (SRL) systems like ASSERT (Hacioglu, Pradhan, Ward, Martin, & Jurafsky, 2003). Attempting an appli-

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

cation 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:

[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 Fig. 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 Fig. 3. As we can see in Fig. 3A, 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 Fig. 3B. 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 Fig. 3C. We refer to this kind of nested STs as STNs (Semantic Tree Networks).

5. Syntactic and semantic kernels for text

5.1. 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 ith element $v_i(T)$ is the number of occurrences of the ith tree fragment in tree T. The tree fragments of a tree are all of its sub-trees which include at least one production with the restriction that no production rules can be broken into incomplete parts.

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

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, (Collins & Duffy, 2001) 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 $v(T_2)$:

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

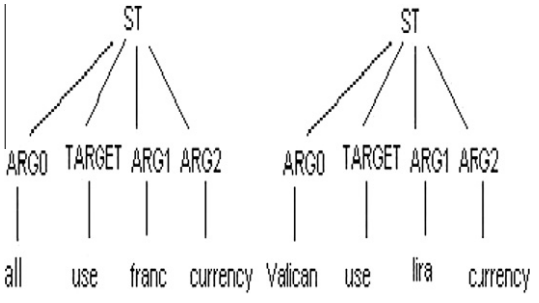

Fig. 2. Example of semantic trees.

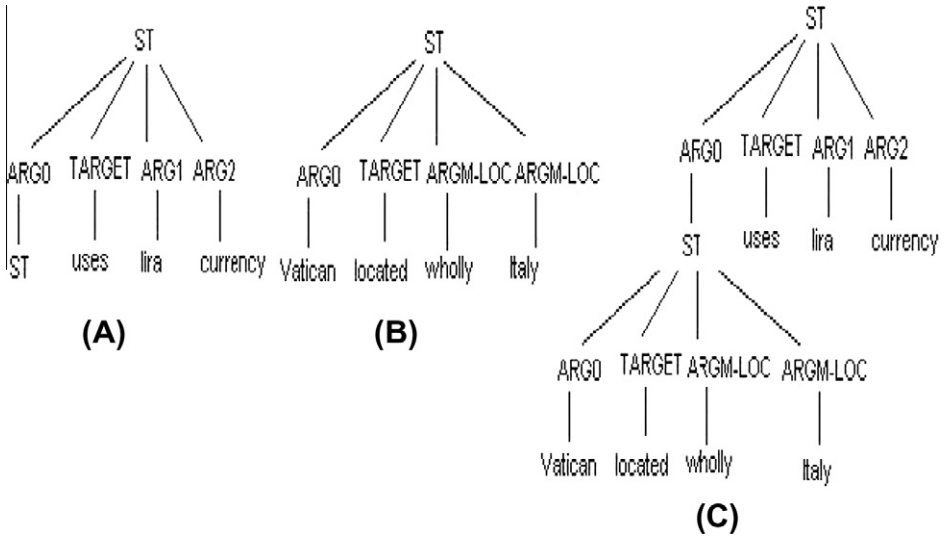

Fig. 3. Two STs composing a STN.

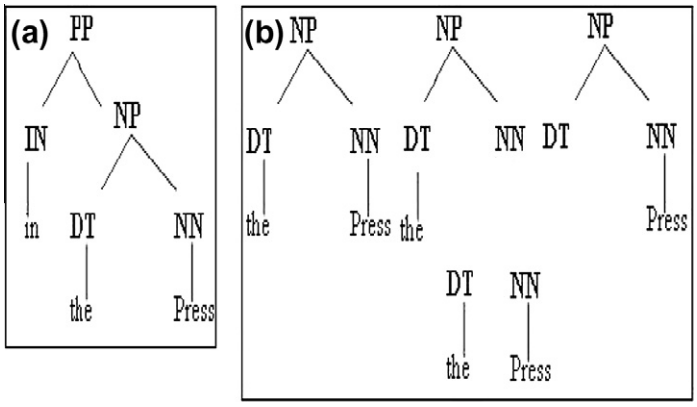

Fig. 4. (a) An example tree. (b) The sub-trees of the NP covering "the press".

We define the indicator function $I_i(n)$ to be 1 if the sub-tree 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:

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

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,

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

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 ith child-node of n_1 is $ch(n_1,i)$.

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

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 4. For instance, although the two STs of Fig. 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. Moschitti et al. (2007) solve this problem by designing the Shallow Semantic Tree Kernel (SSTK) which allows to match portions of a ST.

5.2. Shallow Semantic Tree Kernel (SSTK)

The SSTK is based on two ideas: first, it changes the ST, as shown in Fig. 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 Fig. 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).

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.

We can see that the above discussed kernels are designed by either choosing an explicit mapping function and incorporating it into an inner product or by directly defining the kernel function while making sure that it complies with the requirement of being a positive semi-definite function. A kernel $k: \chi \times \chi \rightarrow R$ is a symmetric and a positive semi-definite function, which simply computes an inner product in a reproducing kernel Hilbert space (Scholkopf & Smola, 2002). The constructed kernel matrices for the proposed kernel functions can be proved positive and semi-definite along with the kernel function's symmetricity using the theorem in Shin and Kuboyama (2008).

6. Extended String Subsequence Kernel (ESSK)

The ESSK is a simple extension of the Word Sequence Kernel (WSK) (Cancedda, Gaussier, Goutte, & Renders, 2003) and String Subsequence Kernel (SSK) (Lodhi, Saunders, Shawe-Taylor, Cristianini, & Watkins, 2002) that can incorporate semantic information with the use of word senses. WSK receives two sequences of words as input and maps each of them into a high-dimensional vector space. WSK's value is just the inner product of the two vectors. But, WSK disregards synonyms, hyponyms, and hypernyms. On the other hand, SSK measures the similarity between two sequences of "alphabets". In ESSK, each "alphabet" in SSK is replaced by a disjunction of an "alphabet" and its alternative (Hirao, Suzuki, Isozaki, & Maeda, 2003).

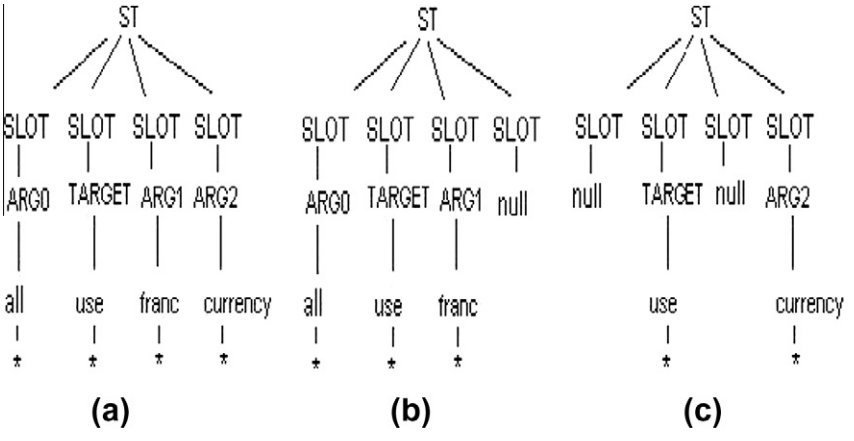

Fig. 5. Semantic tree with some of its fragments.

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

Here, each word in a sentence is considered an "alphabet", and the alternative is its disambiguated sense that we find using the WSD (Word Sense Disambiguation) System of Chali and Joty (2007). The use of word sense yields flexible matching even when paraphrasing is used for the sentences (Hirao et al., 2004).

We calculate the similarity score $sim(T_i,U_j)$ using ESSK where T_i and U_j are two sentences. Formally, ESSK is defined as follows (Hirao et al., 2004):

$$K_{esk}(T,U) = \sum_{m=1}^{d} \sum_{t_i \in T} \sum_{u_j \in U} K_m(t_i,u_j)$$

$$K_m(t_i,u_j) = \begin{cases} val(t_i,u_j) & \text{if } m = 1 \\ K'_{m-1}(t_i,u_j) \cdot val(t_i,u_j) \end{cases}$$

Here, $K'_m(t_i,u_j)$ is defined below. t_i and u_j are nodes of T and U, respectively. Each node includes a word and its disambiguated sense.[1] The function $val(t,u)$ returns the number of attributes common to given nodes t and u.

$$K'_m(t_i,u_j) = \begin{cases} 0 & \text{if } j = 1 \\ \lambda K'_m(t_i,u_{j-1}) + K''_m(t_i,u_{j-1}) \end{cases}$$

Here, λ is the decay parameter for the number of skipped words. $K''_m(t_i,u_j)$ is defined as:

$$K''_m(t_i,u_j) = \begin{cases} 0 & \text{if } i = 1 \\ \lambda K''_m(t_{i-1},u_j) + K_m(t_{i-1},u_j) \end{cases}$$

Finally, the similarity measure is defined after normalization as below:

$$sim_{esk}(T,U) = \frac{K_{esk}(T,U)}{\sqrt{K_{esk}(T,T)K_{esk}(U,U)}}$$

7. Redundancy checking and generating a 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. Goldstein, Kantrowitz, Mittal, and Carbonell (1999) observed this in what the authors called "Maximum-Marginal-Relevancy (MMR)". Following (Hovy et al., 2006), 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.

8. Experiments

8.1. 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.

[1] We use a dictionary based disambiguation approach assuming one sense per discourse. To accomplish this task, we use WordNet (Fellbaum, 1998) that is a widely used semantic lexicon for the English language. It groups English words (i.e. nouns, verbs, adjectives and adverbs) into sets of synonyms called synsets, provides short, general definitions (i.e. gloss definition), and records the various semantic relations between these synonym sets.

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

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 four 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.

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 five systems defined below:

(1) *TF*IDF:* This system is the original topic-sensitive LexRank described in Section 3 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 5.1. The mixture model for this system is:

$$p(s|q) = d \times SYNSIM(s,q) + (1-d) \times \sum_{v \in C} SYNSIM(s,v) \times p(v|q) \quad (7)$$

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 5.2. Therefore, the mixture model for this system is:

$$p(s|q) = d \times SEMSIM(s,q) + (1-d) \times \sum_{v \in C} SEMSIM(s,v) \times p(v|q) \quad (8)$$

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:

$$p(s|q) = d \times SYNSEM(s,q) + (1-d) \times \sum_{v \in C} SYNSEM(s,v) \times p(v|q) \quad (9)$$

where

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

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

(5) *ESSK:* This system measures the similarity between the sentences using the *Extended String Subsequence Kernel (ESSK)* defined in Section 6. Therefore, the mixture model for this system is:

$$p(s|q) = d \times ESSK(s,q) + (1-d) \times \sum_{v \in C} ESSK(s,v) \times p(v|q) \quad (10)$$

where $ESSK(s,q)$ is the normalized similarity score between the *query* (q) and *the document sentence* (s) and C is the set of all sentences in the collection.

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

8.2. Evaluation results

8.2.1. Automatic evaluation

We carried out automatic evaluation of our summaries using ROUGE (Lin, 2004) 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 ($n = 1$–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 (Lin & Hovy, 2003). We showed four of the ROUGE metrics in the

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

Table 1
ROUGE measures for TF*IDF system.

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 2
ROUGE measures for SYN 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 3
ROUGE measures for SEM 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 4
ROUGE measures for SYNSEM.

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

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

Tables 1 and 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. Tables 3 and 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 measure. The SYNSEM system performs better

Table 5
ROUGE measures for ESSK system.

Measures	ROUGE-1	ROUGE-L	ROUGE-W	ROUGE-SU
Precision	0.392012	0.360215	0.192815	0.146032
Recall	0.357786	0.328751	0.094569	0.121505
F-score	0.374077	0.343727	0.126854	0.132594

Table 6
ROUGE *F*-scores for all systems.

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
ESSK	0.374077	0.343727	0.126854	0.132594

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

Table 7
95% Confidence intervals for different systems.

Systems	ROUGE-1
Baseline	0.326680–0.342330
Best system	0.431680–0.445970
TF*IDF	0.355371–0.400891
SYN	0.359389–0.407745
SEM	0.363465–0.416964
SYNSEM	0.356249–0.420719
ESSK	0.358846–0.390068

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. Table 5 shows the ROUGE scores of the ESSK system. Here, we find that the ESSK system performs better than the TF*IDF and SYN systems whereas it works worse than that of the SEM and SYNSEM systems.

The comparison between the systems in terms of their F-scores is given in Table 6. 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 with the semantic feature degrades the performance. On the other hand, the ESSK system improves the ROUGE-1, ROUGE-L, ROUGE-W, and ROUGE-SU scores over the TF*IDF system by 4.07%, 2.64%, 2.12%, and 1.52%, and over the SYN system by 1.19%, 2.10%, 0%, and 2.70% respectively.

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.

Confidence Interval We show the 95% confidence interval of the important evaluation metrics for our systems to report significance for doing meaningful comparison. We use the ROUGE tool for this purpose. ROUGE uses a randomized method named bootstrap resampling to compute the confidence interval. We used 1000 sampling points in the bootstrap resampling. We also include the 95% confidence interval scores of one baseline system and the best system in DUC-2007 in order to show the level of performance our systems achieve. The baseline system generates summaries by returning all the leading sentences (up to 250 words) in the ⟨*TEXT*⟩ field of the most recent document(s). Table 7 reports the 95% confidence intervals of ROUGE-1 F-scores for all the systems. Our systems could not beat the best system in DUC-2007 because of the fact that during the summary generation step they concatenated the sentences in the order found in the source documents (Pingali & Varma, 2007). Besides, they dereferenced repeating entities (person names and organization names) which we did not focus on while producing a summary.

8.2.2. Manual evaluation

For a sample of 25 summaries[2] drawn from our different systems' generated summaries we conduct an extensive manual evaluation in order to analyze the effectiveness of our approaches. The manual evaluation comprised a Pyramid-based evaluation of contents and a user evaluation to get the assessment of linguistic quality and overall responsiveness.

Pyramid Evaluation In the DUC 2007 main task, 23 topics were selected for the optional community-based pyramid evaluation. Volunteers from 16 different sites created pyramids and annotated the peer summaries for the DUC main task using the given guidelines.[3] Eight sites among them created the pyramids. We used these pyramids to annotate our peer summaries to compute the modified pyramid scores.[4] We used the *DUCView.jar*[5] annotation tool for this purpose. Table 8 shows the modified pyramid scores of all our systems. The baseline system's score and the best DUC-2007 system's scores are also reported. The peer summaries of the baseline system are generated by returning all the leading sentences (up to 250 words) in the ⟨ *TEXT*⟩ field of the most recent document(s). From these results we see that all our systems perform better than the baseline system and inclusion of semantic information always yields better scores whereas SYN and SYNSEM systems perform decently. All our systems' scores are better than that of the best DUC-2007 system as because we conducted our manual evaluation on a small subset of peer summaries.

User Evaluation Some university graduate students judged the summaries for linguistic quality and overall responsiveness. The given score is an integer between 1 (very poor) and 5 (very good) and is guided by consideration of the following

[2] Randomly, we chose five summaries for each of these systems.
[3] http://www1.cs.columbia.edu/becky/DUC2006/2006-pyramid-guidelines.html.
[4] This equals the sum of the weights of the Summary Content Units (SCUs) that a peer summary matches, normalized by the weight of an ideally informative summary consisting of the same number of contributors as the peer.
[5] http://www1.cs.columbia.edu/ani/DUC2005/Tool.html

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

Table 8
Modified pyramid scores for all systems.

Systems	Modified pyramid scores
Baseline	0.13874
Best system	0.34870
TF*IDF	0.51205
SYN	0.48485
SEM	0.57275
SYNSEM	0.46156
ESSK	0.54697

Table 9
Linguistic quality and responsive scores for all systems.

Systems	Linguistic quality	Overall responsiveness
Baseline	4.24	1.80
Best system	4.11	3.40
TF*IDF	4.16	3.00
SYN	4.15	2.75
SEM	4.12	2.80
SYNSEM	4.10	2.50
ESSK	4.20	2.75

factors: 1. Grammaticality, 2. Non-redundancy, 3. Referential clarity, 4. Focus and 5. Structure and Coherence. They also assigned a content responsiveness score to each of the automatic summaries. The content score is an integer between 1 (very poor) and 5 (very good) and is based on the amount of information in the summary that helps to satisfy the information need expressed in the topic narrative. These measures were used at DUC 2007. Table 9 presents the average linguistic quality and overall responsive scores of all our systems. Again, the baseline system's scores and the best DUC-2007 system's scores are given for meaningful comparison.

9. Conclusion

In this paper, we have used the syntactic and 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. We have also extended our work using the Extended String Subsequence Kernel (ESSK) to include semantic information by word senses.

We evaluated our systems automatically using ROUGE and report the significance of our results through 95% confidence intervals. We conducted two types of manual evaluation: (1) Pyramid and (2) User Evaluation to further analyze the performance of our systems. Our experiments suggest the following: (a) similarity measures based on the syntactic tree and/or shallow semantic tree and Extended String Subsequence Kernel (ESSK) outperform the similarity measures based on the TF*IDF and (b) similarity measures based on the shallow semantic tree performs best for this problem.

In future, we plan to experiment with a shallow syntactic sentence similarity measurement approach like Basic Elements (BE) to check how it performs for this task.

Acknowledgments

We thank the anonymous reviewers for their useful comments on the earliest version of this paper. The research reported here was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada – discovery grant and the University of Lethbridge.

References

Amigo, E., Gonzalo, J., Peinado, V., Peinado, A., & Verdejo, F. (2004). An empirical study of information synthesis tasks. In: *Proceedings of the 42nd annual meeting of the association for computational linguistics* (pp. 207–214). Barcelona, Spain.

Cancedda, N., Gaussier, E., Goutte, C., & Renders, J. M. (2003). Word sequence kernels. *Journal of Machine Learning Research, 3*, 1059–1082.

Chali, Y., & Joty, S. R. (2007). Word sense disambiguation using lexical cohesion. In: *Proceedings of the 4th international conference on semantic evaluations* (pp. 476–479). Prague: ACL.

Chali, Y., & Joty, S. R. (2008). Improving the performance of the random walk model for answering complex questions. In: *Proceedings of the 46th annual meeting of the association for computational linguistics: human language technologies*. OH, USA: ACL.

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002

Chali, Y., & Joty, S. R. (2008). Exploiting syntactic and shallow semantic kernels to improve random walks for complex question answering. In: *Proceedings of the 20th IEEE international conference on tools with artificial intelligence (ICTAI), IEEE, Dayton, OH, USA*.

Charniak, E. (1999). A maximum-entropy-inspired parser. In: *Technical Report CS-99-12*. Brown University, Computer Science Department.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. In: *Proceedings of Neural Information Processing Systems, Vancouver, Canada* (pp. 625–632).

Erkan, G., & Radev, D. R. (2004). LexRank: graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research, 22*, 457–479.

Fellbaum, C. (1998). *WordNet – an electronic lexical database*. Cambridge, MA: MIT Press.

Goldstein, J., Kantrowitz, M., Mittal, V., & Carbonell, J. (1999). Summarizing text documents: sentence selection and evaluation metrics. In: *Proceedings of the 22nd international ACM conference on research and development in information retrieval, SIGIR, Berkeley, CA* (pp. 121–128).

Hacioglu, K., Pradhan, S., Ward, W., Martin, J.H., & Jurafsky, D. (2003). Shallow semantic parsing using support vector machines. In: *Technical Report TR-CSLR-2003-03*. University of Colorado.

Hirao, T., Suzuki, J., Isozaki, H., & Maeda, E. (2003). NTT's multiple document summarization system for DUC2003. In: *Proceedings of the document understanding conference*.

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

Hovy, E., Lin, C. Y., Zhou, L., & Fukumoto, J. (2006). Automated summarization evaluation with basic elements. In: *Proceedings of the Fifth Conference on Language Resources and Evaluation, Genoa, Italy*.

Kingsbury, P., & Palmer, M. (2002). From treebank to propbank. In: *Proceedings of the International Conference on Language Resources and Evaluation, Las Palmas, Spain*.

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

Lin, C. Y. (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, Barcelona, Spain* (pp. 74–81).

Lin, C.Y., & Hovy, E. (2003). Automatic evaluation of summaries using *n*-gram co-occurrence statistics. In: *Proceedings of the annual meeting of the North American association for computational linguistics, Edmonton, Canada* (pp. 150–156).

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research, 2*, 419–444.

MacCartney, B., Grenager, T., de Marneffe, M., Cer, D., & Manning, C. D. (2006). Learning to recognize features of valid textual entailments. In: *proceedings of the human language technology conference of the North American chapter* (pp. 41–48). New York, USA: ACL.

Mihalcea, R., & Tarau, P. (2004). TextRank: bringing order into texts. In: *Proceedings of the conference of empirical methods in natural language processing, Barcelona, Spain*.

Moschitti, A., & Basili, R. (2006). 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*.

Moschitti, A., Quarteroni, S., Basili, R., Manandhar, S. (2007). Exploiting syntactic and shallow semantic kernels for question/answer classificaion. In: *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics* (pp. 776–783). Prague, Czech Republic: ACL.

Otterbacher, J., Erkan, G., & Radev, D. R. (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* (pp. 915–922). Canada: Vancouver.

Pingali, P., Rahul, K., & Varma, V. (2007). IIIT Hyderabad at DUC 2007. In: *Proceedings of the document understanding conference, NIST, Rochester*.

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

Scholkopf, B., & Smola, A. (2002). *Learning with kernels*. Cambridge, MA: MIT Press.

Shin, K., & Kuboyama, T. (2008). A generalization of haussler's convolution kernel: mapping kernel. In: *ICML* (pp. 944–951).

Zhang, A., & Lee, W. (2003). Question classification using support vector machines. In: *Proceedings of the special interest group on information retrieval* (pp. 26–32). Toronto, Canada: ACM.

Please cite this article in press as: Chali, Y., et al. Improving graph-based random walks for complex question answering using syntactic, shallow semantic and extended string subsequence kernels. *Information Processing and Management* (2010), doi:10.1016/j.ipm.2010.10.002