MULTI-DOCUMENT SUMMARIZATION USING DEPENDENCY GRAMMARS

by

Şaziye Betül Bilgin

B.S., Computer Engineering, Boğaziçi University, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2014

MULTI-DOCUMENT SUMMARIZATION USING DEPENDENCY GRAMMARS

APPROVED BY:

Assist. Prof. Arzucan Özgür .....
(Thesis Supervisor)

Assoc. Prof. Ali Taylan Cemgil .....

Assoc. Prof. Banu Diri .....

DATE OF APPROVAL: 06.06.2014

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Assist. Prof. Arzucan Özgür for her valuable guidance, support, and kindness. Without her counseling and constant encouragement, I would have never completed this study.

I thank Assoc. Prof. Ali Taylan Cemgil and Assoc. Prof. Banu Diri for participating in my thesis committee and for their valuable revisions on the thesis. Special thanks to Prof. Dragomir Radev for his helpful comments and feedback.

I would like to give special thanks to my friend Hakime Öztürk for her valuable contributions to this study and her support. I thank my friends Esra Altınışık, Ayşenur Çor, Esra Çankaya , Şeyma Çetinkaya, Beyza Karakaya, and Zeynep Erçetin who never let me feel alone.

Finally and above all, I express my deepest gratitude to my dearest family. I am grateful to my father Abdullah Bilgin for always motivating his children to go further in academics and much more. I would like to give special thanks to my mother Güzide Bilgin for being always with me and for her endless love, patience and support. I am thankful to my dearest sisters Esma Fatıma Bilgin Taşdemir, Rümeysa Bilgin, Zehra Bilgin and to my brother Ahmed İlyas Bilgin for their true friendship, motivation, guidance, and support.

I acknowledge the support of TÜBİTAK in the form of a graduate scholarship.

ABSTRACT

MULTI-DOCUMENT SUMMARIZATION USING DEPENDENCY GRAMMARS

Information overload is one of the greatest challenges in recent years, especially due to the rapid increase of data produced on the Internet. Automatic summarization of documents about similar topics is a salient solution to overcome this problem. There are mainly two approaches for this task, extractive multi-document summarization where the summary is created by selecting salient sentences from documents, and abstractive multi-document summarization where new sentences are generated using natural language generation methods. Sentence similarity calculation is significant in most of the extractive multi-document summarization approaches. In this study we introduce the usage of dependency grammars to compute sentence similarity for extractive multi-document summarization. We adapt and investigate the effects of two untyped dependency tree based sentence similarity kernels, which have originally been proposed for relation extraction, to the multi-document summarization problem. In addition, we propose a series of new dependency grammar based kernels to better represent the syntactic and semantic similarities among the sentences. The proposed methods incorporate type information of dependency relations for sentence similarity calculation. Our best method achieves significantly better scores than the untyped dependency tree based kernels. We observe that using the dependency grammar representations of sentences leads to better results in finding the similarities between sentences and the type of dependency relations is crucial in identifying the important parts in sentences.

ÖZET

BAĞIMSAL DİLBİLGİSİ KULLANARAK ÇOKLU DOKÜMAN ÖZETLEME

İnternet ortamındaki elektronik belgelerin sayısındaki hızlı artış, bilgi yükü problemini de beraberinde getirmiştir. Otomatik çoklu doküman özetleme bu problemi çözmede gelecek vaadeden yöntemlerden biridir. Otomatik çoklu doküman özetleme için temelde iki yaklaşım vardır. Bunlar özetlenecek metinlerden önemli cümlelerin çeşitli yöntemlerle seçilmesi ile özet oluşturan *cümle seçerek özetleme* ve özetleme işini doğal dil işleme metotları kullanarak metinlerden yeni cümlelerin oluşturulmasıyla yapan *yorumlayarak özetleme* yaklaşımlarıdır. Cümle seçerek özetleme yöntemlerinin büyük bir kısmında cümleler arası benzerlik bulma işi önemli bir yer tutar. Bu çalışmada, bağımsal dilbilgisini cümleler arası benzerlik bulma probleminde kullandık ve oluşturduğumuz benzerlik yöntemini cümle seçerek çoklu doküman özetleme işine uyguladık. Bu iş için iki farklı bağımsal ağaç bazlı cümleler arası benzerlik bulma yöntemi kullandık. İlk olarak ilişki çıkarımı için önerilmiş bu yöntemler hesaplamalarında bağımsal dilbilgisindeki bağıntıların türlerinden yararlanmıyorlardı. Bu metotlarla çalıştıktan sonra bağıntıların türlerini de hesaba katan bir dizi yeni bağımsal dilbilgisi bazlı cümleler arası benzerlik bulma yöntemi geliştirdik. Tasarlanan bu yöntemler diğer iki yöntemden daha iyi sonuç verdiler. Bu çalışma sonucunda cümleler arası benzerlik bulmada bağımsal dilbilgisi kullanımının başarılı sonuçlar verdiğini gördük. Cümleleri bağımsal ağaç yapısını kullanarak göstermenin cümleler arasındaki benzer kısımları bulmakta iyi sonuç verdiğini gözlemledik. Ayrıca bağımsal ağaçlardaki bağıntıların türlerinin cümlelerdeki önemli kısımların bulunmasında önemli rol oynadığı sonucuna vardık.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . viii
LIST OF TABLES . . . xi
LIST OF SYMBOLS . . . xiii
LIST OF ACRONYMS/ABBREVIATIONS . . . xvi
1. INTRODUCTION . . . 1
1.1. Multi-document Summarization . . . 1
1.2. Dependency Grammars . . . 2
1.3. Motivation . . . 4
1.4. Thesis Organization . . . 4
2. SENTENCE SIMILARITY METHODS . . . 6
2.1. Dependency Tree Kernel . . . 6
2.2. Dependency Trigram Kernel . . . 10
2.3. Bigram Match Kernels . . . 12
2.3.1. Strict Bigram Match Kernel . . . 14
2.3.2. Approximate Bigram Match Kernels . . . 18
3. EXPERIMENTS . . . 26
3.1. Data Sets . . . 26
3.2. Evaluation Metric . . . 27
3.3. Tools . . . 27
3.4. Evaluation of the Sentence Similarity Kernels . . . 29
3.4.1. Evaluation of the Dependency Tree Kernel and the Dependency Trigram Kernel . . . 29
3.4.2. Evaluation of the Bigram Match Kernels . . . 32
4. RESULTS AND DISCUSSION . . . 34
4.1. Results of the Dependency Tree Kernel and the Dependency Trigram Kernel . . . 34

4.2. Results of the Bigram Match Kernels 36
5. CONCLUSIONS AND FUTURE WORK 43
APPENDIX A: SAMPLE SYSTEM GENERATED AND HUMAN GENERATED SUMMARIES 47
REFERENCES 50

LIST OF FIGURES

Figure 1.1. Dependency tree of the sentence *"Citing Hun Sens threats to arrest opposition politicians following two alleged attempts on his life, Ranariddh and Sam Rainsy have said they do not feel safe negotiating inside the country and asked the king to chair the summit at his residence in Beijing."* 2

Figure 2.1. Dependency tree of the sentence *"Pinochet, who turns 83 next month, was reported to have left London Bridge Hospital on Wednesday."* with the corresponding feature sets of the nodes in the tree 8

Figure 2.2. Two dependency trigram units: $Pinochet \rightarrow left \leftarrow Hospital$ and $Pinochet \rightarrow reported \leftarrow on$ 11

Figure 2.3. Typed dependency tree of the sentence *"Sam Rainsy said Wednesday that he was unsatised with the guarantee."* *nsubj* dependency relation is more informative than *det* dependency relation. 12

Figure 2.4. (a) Dependency trees of the sentences *"The man has killed his wife using a knife."* and *"Using a knife, a man killed his knife."*, respectively. Gray nodes with continuous lines constitute matching bigram units when *DTH* is used as the bigram type. The number of matching bigrams is 6. (b) When we use *DH* as the bigram type, the number of matching bigrams is 7. The non-matching bigrams {*using, xcomp, killed*} and {*using, partmod, killed*} is now matching as we omit the dependency type. 15

Figure 2.5. (a) Dependency trees of the sentences *"He refused to abolish his government."* and *"He refused to dissolve his current government."* respectively. Gray nodes with continuous lines constitute matching bigram units when *DTH* or *DH* is used as the bigram type. (b) When *DA* is used as the bigram type, three additional relations are formed. Only this type captures the relation between words *refused* and *government*. 16

Figure 2.6. Dependency trees of the sentences *"He refused to abolish his government."* and *"He refused to dissolve his current government."* respectively 18

Figure 2.7. D_1 : (a) Dependency tree of the first sentence S_1. (b) Dependency tree of the second sentence S_2. The subtrees with double-circled nodes denote the matching dependency relations. 23

Figure 4.1. ROUGE-1 scores of different runs of the Strict Bigram Match Kernel on DUC 2003 Task 2 data set 37

Figure 4.2. ROUGE-1 scores of different runs of the Simple Approximate Bigram Match Kernel using different u and θ combinations on DUC 2003 Task 2 data set 38

Figure 4.3. ROUGE-1 scores of different runs of the TF-IDF Based Approximate Bigram Match Kernel using different u and θ combinations on DUC 2003 Task 2 data set 38

Figure 4.4. ROUGE-1 scores of different runs of the TF-IDF Max Approximate Bigram Match Kernel using different u and θ combinations on DUC 2003 Task 2 data set 39

Figure A.1. System generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set 47

Figure A.2. First human generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set 48

Figure A.3. Second human generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set 48

Figure A.4. Third human generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set 49

Figure A.5. Fourth human generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set 49

LIST OF TABLES

Table 2.1. Feature Set of the Dependency Tree Kernel 7

Table 3.1. Parameter Setup of the LexRank System 30

Table 3.2. Unimportant dependency relation types. These types are considered as insignificant and nodes whose dependency types are one of them are ignored in sentence similarity calculations if these types are filtered. 31

Table 3.3. The effect of the degree of the common ancestor in trigram formation for the Dependency Trigram Kernel 32

Table 4.1. ROUGE-1 scores of the Dependency Tree Kernel with different settings in LexRank system on DUC 2003 Task 2 data set 34

Table 4.2. ROUGE-1 scores of the Dependency Tree Kernel with different settings in LexRank system on DUC 2004 Task 2 data set 35

Table 4.3. ROUGE-1 scores of the Dependency Trigram Kernel with different settings in LexRank system on DUC 2003 Task 2 data set 35

Table 4.4. ROUGE-1 scores of the Dependency Trigram Kernel with different settings in LexRank system on DUC 2004 Task 2 data set 36

Table 4.5. ROUGE-1 scores of the bigram match kernels, untyped dependency tree kernels, and baseline models in LexRank system on DUC 2003 Task 2 data set 41

Table 4.6. ROUGE-1 scores of the bigram match kernels, untyped dependency tree kernels, and baseline models in LexRank system on DUC 2004 Task 2 data set 42

LIST OF SYMBOLS

A_b	Bigram set of sentence A
${A_b}^i$	ith bigram of sentence A
${A^i}_T$	ith dependency trigram of sentence A
${A^i}_{T_c}$	Center node of the ith dependency trigram of sentence A
${A^i}_{T_l}$	Left node of the ith dependency trigram of sentence A
${A^i}_{T_r}$	Right node of the ith dependency trigram of sentence A
c	Binary function that returns θ if its arguments are equal and 1 otherwise
c_{r_i}	Children nodes of the node r_i
${d_A}^i$	Dependent node of the ith bigram of sentence A
d_i	Dependent word of the ith dependency relation
DA	Dependent-ancestor bigram unit
DH	Dependent-head bigram unit
DTH	Dependent-type-head bigram unit
DT-TH	Dependent-type, type-head bigram unit
F	Equality function defined over feature sets of dependency tree nodes
idf_w	Inverse document-frequency score of the word w
f_{m_i}	Matching feature set of node i
${h_A}^i$	Head node of the ith bigram of sentence A
K_1	Kernel function of the Strict Bigram Match Kernel
K_2	Kernel function of the Simple Approximate Bigram Match Kernel
K_3	Kernel function of the TF-IDF Based Approximate Bigram Match Kernel
K_4	Kernel function of the TF-IDF Max Approximate Bigram Match Kernel
K_5	Kernel function of the Matching Subtrees Kernel
K_c	Tf-idf based children kernel function of the Matching Subtrees Kernel

K_{c_2}	Fixed score children kernel function of the Matching Subtrees Kernel
K_{c_D}	Children kernel function of the Dependency Tree Kernel
K_D	Kernel function of the Dependency Tree Kernel
K_{mix}	Kernel function of the Composite Kernel
K_T	Kernel function of the Dependency Trigram Kernel
M	Match function of the Strict Bigram Match Kernel
$match$	Matching function of the Dependency Tree Kernel
max	Function that returns the maximum of its arguments
N	Tf-idf based normalizer function
N_f	Binary funtion which results in 1 when the fth features of two nodes match in the Dependency Trigram Kernel
s	Binary function that returns 1 if its arguments are equal and 0 otherwise
S	Score function of the Dependency Trigram Kernel
sim	Similarity function of the Strict Bigram Match Kernel
sim_D	Similarity function of the Dependency Tree Kernel
sim_t	Similarity function of the TF-IDF Based Approximate Bigram Match Kernel
r_i	The root node of a dependency tree i
${t_A}^i$	Type of the ith bigram of sentence A
t_i	Type of the ith dependency relation
tf_w	Term-frequency score of the word w
w_i	ith word of a sentence
α	Weight of the s function in K_{c_2} of the Composite Kernel
α_f	Weight of the feature f in the Dependency Trigram Kernel
β	Weight of K_4 in the Composite Kernel
δ	Weight of K_5 in the Composite Kernel
λ	Decaying factor of the kernel function K_{c_D}
θ	A constant that determines the inuence of a type match in bigram match kernels

ν	Decaying factor to prevent high increase in the similarity score in the Matching Subtrees Kernel

LIST OF ACRONYMS/ABBREVIATIONS

CK	Composite Kernel
DTK	Dependency Tree Kernel
MDS	Multi-document Summarization
MSK	Matching Subtrees Kernel
SABMK	Simple Approximate Bigram Match Kernel
SBMK	Strict Bigram Match Kernel
TABMK	TF-IDF Based Approximate Bigram Match Kernel
TABMK-Max	TF-IDF Max Approximate Bigram Match Kernel
TF-IDF	Term Frequency-Inverse Document Frequency
Trigram-K	Dependency Trigram Kernel

1. INTRODUCTION

The rapid increase in online data leads to the problem of information overload. As tremendous amount of information is being generated every day, the need of tools that organize and display information in an uderstandable way is constantly growing. Automatic summarization of multiple documents is one of the most promising solutions proposed to overcome this problem. An automated tool that represents multiple similar documents with a summary that includes the most important information in the documents would ease the analysis of the online documents. Multi-document summarization aims to create such automated tools.

1.1. Multi-document Summarization

Multi-document summarization (MDS) refers to the task of automatically generating a summary of multiple documents about the same topic without losing the most important information. This is a challenging research area because multiple documents about the same topic may contain both overlapping and contradicting information. The main tasks here are detecting and handling redundancy, while generating a coherent and complete summary [1].

There are mainly two approaches for text summarization, namely extractive summarization and abstractive summarization. Extractive summarization aims to select sentences which are most representative of the original document. On the other hand, the abstractive summarization approach tries to generate new sentences that capture the important points of the document using natural language generation methods. Most studies today, pursue extractive based approaches for text summarization.

Several methods including supervised approaches [2,3], topic driven models [4–6], and clustering based models [7, 8] have been proposed in the literature for the multi-document summarization problem. Recently, graph-based summarization methods have attracted increasing attention of researchers [9–11] and have been successful when

compared to the other state-of-the-art summarization approaches [12]. Graph-based methods represent documents as a graph where vertices are sentences and edges denote the similarity between the corresponding pairs of sentences.

1.2. Dependency Grammars

Dependency grammar is a way of representing syntactic dependencies between words in sentences. Figure 1.1 depicts the dependency tree of an example sentence for illustration purposes. Each word in a dependency tree is dependent to another word with its corresponding dependency relation type.

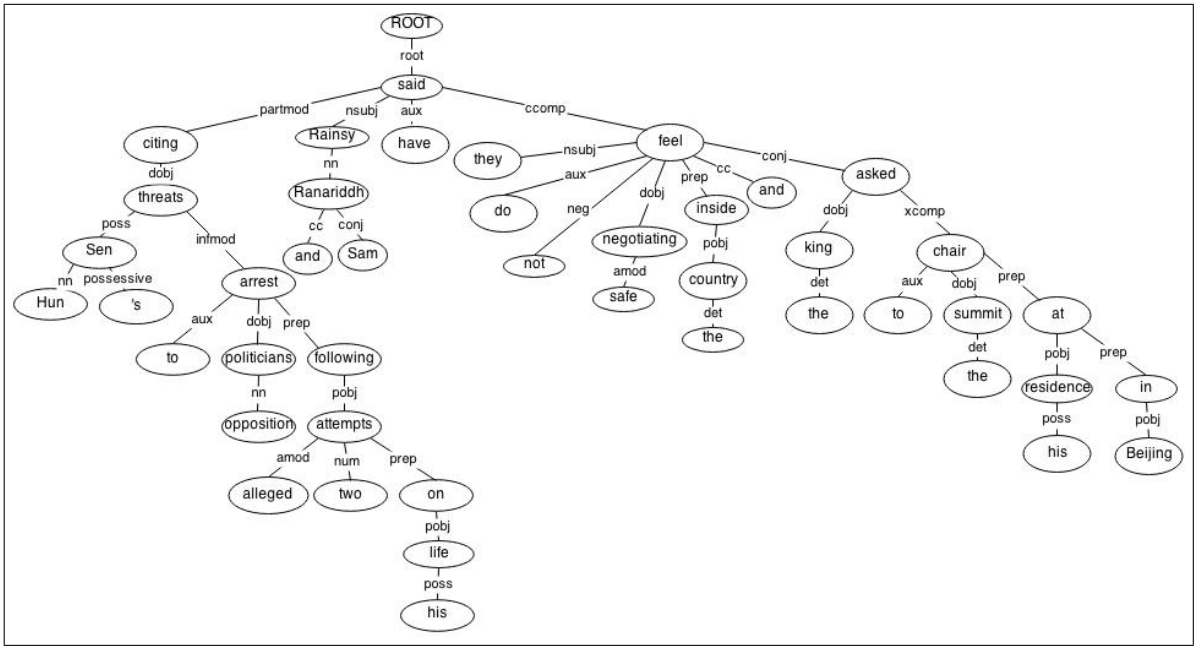

Figure 1.1. Dependency tree of the sentence *"Citing Hun Sens threats to arrest opposition politicians following two alleged attempts on his life, Ranariddh and Sam Rainsy have said they do not feel safe negotiating inside the country and asked the king to chair the summit at his residence in Beijing."*

There have been a number of studies on using dependency grammars for summarization. In [13], Barzilay and McKeow present the sentence fusion concept for the multi-document summarization task. Sentence fusion constructs a new sentence by extracting common information in a set of similar sentences. To find the common

information, they represent sentences using dependency trees and utilize a local multisequence alignment algorithm on dependency trees. Filippova and Strube [14] also propose a sentence fusion method that can be used in abstractive summarization. This method utilizes dependency grammar representations of sentences to build a dependency graph by aligning dependency trees of similar sentences. In [15], Zhuang *et al.* address the problem of movie review mining and summarization and propose multi-knowledge based approach which uses dependency grammar to find valid product feature - opinion pairs by mining dependency relations between features and relevant opinions in sentences. Somprasertsri and Lalitrojwong [16], also utilize dependency grammar for opinion summarization, but use Maximum Entropy classifier to identify valid product feature - opinion pairs. Fiszman *et al.* apply an abstractive approach for summarization of multiple medical documents in [17]. They identify concepts by matching noun phrases to a medical vocabulary and utilize dependency grammar for argument identification. In [18], Yousfi-Monod proposes an unsupervised method for sentence compression to be used in text summarization tasks. The method is based on removing incident sentence constituents to shorten the sentence without losing important information. For distinguishing incident and principal constituents, dependency grammar representation is employed. Galanis [19], proposes an extractive text summarization method which uses dependency grammar. The method first creates multiple candidate compressions by removing branches in the dependency tree of a sentence. Then Maximum Entropy classifier is used to eliminate the unlikely compressions and finally an SVR (Support Vector Regression) model is employed to select the best compressed sentence. In [20], Sakhare proposes a dependency grammar based noun phrase extraction method as a pre-processing step for text summarization instead of standard n-gram models. On the other hand, Liu and Zhou [21], use dependency grammar as a post-processing tool in their Chinese text summarization system for the purpose of improving the readability of the resulting summary. In [22], Blake *et al.* propose some strategies for multi-document text summarization. One of them is using dependency grammar representation for lexical simplification in sentences such that all tree branches other than the ones having both a subject and an object are removed from the tree.

To the best of our knowledge, none of the previous studies have used the dependency grammar concept to compute sentence similarity in a graph-based summarization approach.

1.3. Motivation

Sentence similarity calculation using bag-of-words representation of sentences computes similarity scores without capturing the semantics of the text and this may cause the final summary to be not informative enough [23].

To address this problem, we propose utilizing dependency grammar for sentence similarity computation. Using this representation method, concepts in multiple documents and relations between similar contents can be better captured.

In this study, we first adapt and investigate the effects of two dependency tree based sentence similarity kernels, [24] and [25] to the multi-document summarization problem. These similarity kernels were originally designed for relation extraction. They both utilize untyped dependency trees of sentences where types of dependency relations are not taken into account. They outperform other sentence similarity methods used in relation extraction that use bag-of-words representation of sentences. We then propose a number of new sentence similarity kernels based on typed dependency grammars and test these kernels on the LexRank multi-document summarization system [9] by replacing its tf-idf based cosine similarity method with each of the kernels. The proposed similarity kernels utilize binary dependency relations in sentences. With this approach, we aim to generate both coherent and complete summaries for multiple text documents.

1.4. Thesis Organization

The organization of this thesis is as follows. The adapted similarity kernels and our proposed similarity methods are described in Chapter 2. We describe the data sets and tools used in this study and present our experimental setup in Chapter 3. Chapter

4 discusses the results of the experiments performed in this study. Finally Chapter 5 presents the major conclusions of the work presented in this thesis and discusses possible future work.

2. SENTENCE SIMILARITY METHODS

Dependency tree representation of sentences allows us to find syntactic dependency relations of words and emphasizes informative parts in a sentence. Therefore, it is a more powerful approach than the bag-of-words representation for modeling the syntactic and semantic information in the sentences. Considering these strengths of the dependency grammars, we utilize two dependency tree kernels [24, 25] to measure the similarities among the nodes in a multi-document summarization sentence graph. In addition to these kernels, we design a series of new sentence similarity methods based on typed dependency grammars. The following sections describe these dependency tree based similarity methods in detail.

2.1. Dependency Tree Kernel

The *Dependency Tree Kernel* (DTK) was proposed by Culotta and Sorensen [24] for relation extraction. It is based on [26]. This kernel is used for finding a similarity score between dependecy trees of sentences that contain entities. It has been integrated into an SVM classifier to label the class of the relation of these entity pairs. The comparison of two dependency trees starts from root nodes and continues through the subtrees in a recursive manner. Each word in a sentence is represented with a node in a dependency tree and each node is described with a set of features. In [24], the features are selected as in Table 2.1.

We make use of this kernel to compute sentence similarity for multi-document summarization. Hence, we adapt their feature set for our task. In the task of relation extraction, this kernel only compares the nodes that are labeled as entities. However, in the multi-document summarization task, we include all the words into node comparison. So, using all of the feature types in node comparison can be misleading in terms of measuring the similarity. Two nodes whose *word* features do not match and have totally different meanings can be found as highly similar if the other features are matching. To reduce this misleading effect, we exclude the *chunk tag* and *entity level*

Table 2.1. Feature Set of the Dependency Tree Kernel

Index	**Feature Name**	**Example**
1	word	Rainsy, letter
2	part-of-speech (POS) tag	NNP, VBD
3	general POS tag	noun, verb
4	chunk tag	NP, VP
5	entity type	person, location
6	entity level	name, nominal
7	WordNet hypernyms	food, state
8	relation argument	ARG-A, ARG-B

features from the feature sets used in this study since the *chunk tag* feature is easy to match up with and the *entity level* feature is not necessary in the presence of the other features for our task. We also exclude the *relation argument* feature since it is a relation extraction specific feature. Figure 2.1 shows the dependecy tree and corresponding feature vector of an example sentence.

In [24], for any entity pairs in a sentence, the smallest common subtree that contains both entities is found and the kernel similarity is computed using these subtrees to reduce the effect of irrelevant words. For the task of multi-document summarization, we apply this kernel on the dependency trees of whole sentences.

For two dependency trees with root nodes r_1 and r_2, the similarity kernel is defined as follows by Culotta and Sorensen [24]:

$$K_D(r_1, r_2) = \begin{cases} 0 & \text{if } match(r_1, r_2) = 0 \\ sim_D(r_1, r_2) + & \\ K_{c_D}(c_{r_1}, c_{r_2}) & \text{otherwise} \end{cases} \tag{2.1}$$

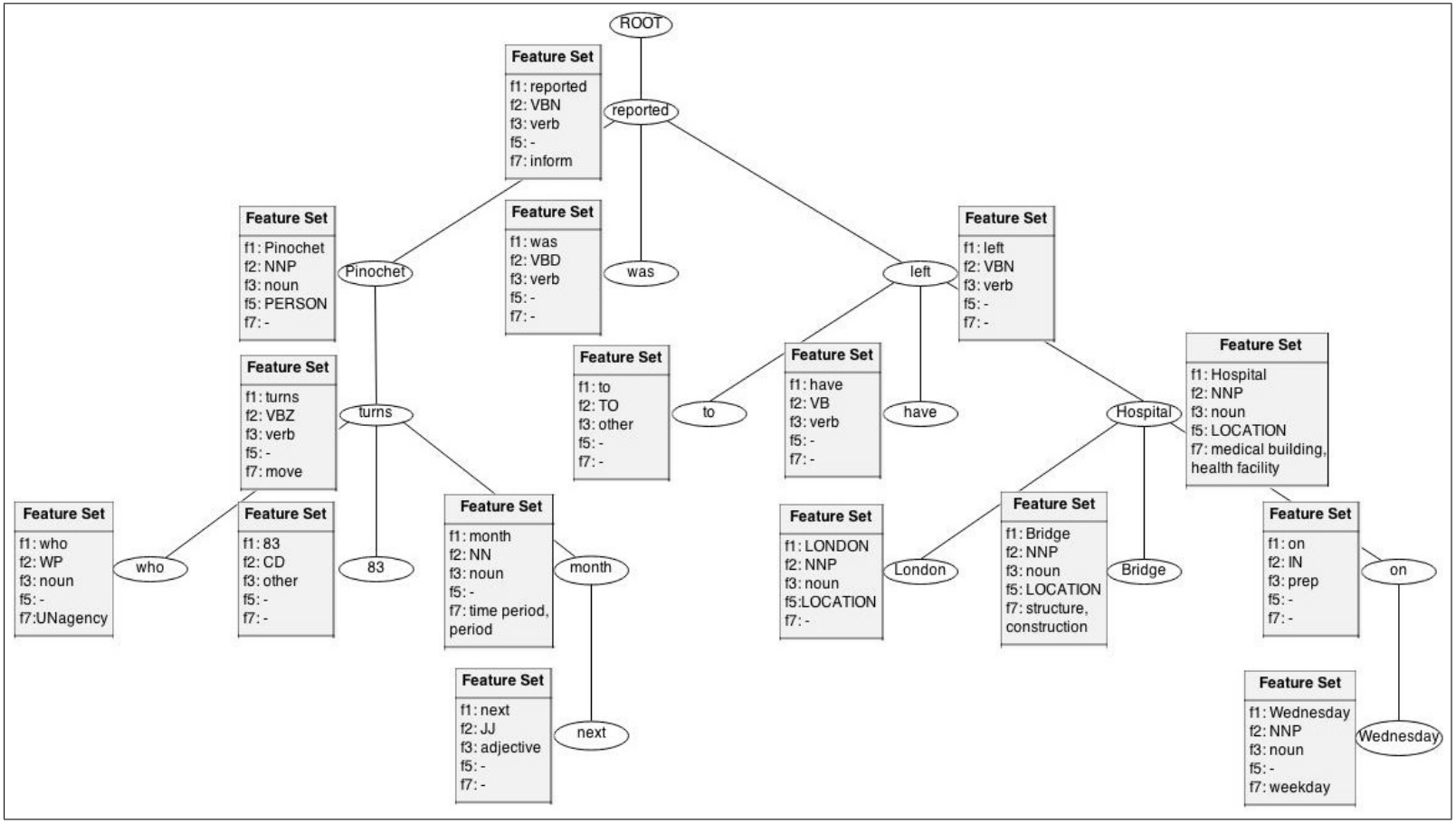

Figure 2.1. Dependency tree of the sentence *"Pinochet, who turns 83 next month, was reported to have left London Bridge Hospital on Wednesday."* with the corresponding feature sets of the nodes in the tree

K_{c_D} kernel is the kernel function defined over the children of two nodes t and z:

$$K_{c_D}(c_t, c_z) = \sum_{u \in c_t}^{n} \sum_{v \in c_z}^{m} \lambda^{n+m} \times K_D(u, v) \tag{2.2}$$

where c_t denotes the children of node t, and n and m are the number of children of t and z, respectively. λ is a constant that penalizes matching subtrees that are spread out in children nodes.

The matching function *match* is a function defined on {0,1} where it takes value of 1 if matching feature set of node t_i, $\mathbf{f_{m_i}}$ equals matching feature set of node t_j, $\mathbf{f_{m_j}}$, and 0 otherwise:

$$match(t_i, t_j) = \begin{cases} 1, & \text{if } \mathbf{f_{m_i}} = \mathbf{f_{m_j}} \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

The similarity function sim_D simply counts the number of common feature values of two nodes and is defined as follows:

$$sim_D(t_i, t_j) = \sum_{k} F(\mathbf{f_i}[k], \mathbf{f_j}[k]) \tag{2.4}$$

where F is a binary function that returns 1 if its argument feature values are equal, and 0 otherwise:

$$F(\mathbf{f_i}[k], \mathbf{f_j}[k]) = \begin{cases} 1, & \text{if } \mathbf{f_i}[k] = \mathbf{f_j}[k] \\ 0, & \text{otherwise} \end{cases} \tag{2.5}$$

2.2. Dependency Trigram Kernel

The *Dependency Trigram Kernel* (Trigram-K) was proposed by Choi and Kim [25] for the task of social relation extraction. The main idea of this kernel is to calculate the similarity between two sentences that contain social relations using their *dependency trigram* sets.

A dependency trigram is defined between the ith, jth, and kth words of a sentence as $w_i \rightarrow w_k \leftarrow w_j$ and w_i, w_j, and w_k are called the $left, right$, and *center* nodes respectively, if,

- w_k is a common ancestor of w_i and w_j and,
- index i is less than index j.

After creating the dependency trigram sets of the sentences, Choi and Kim [25] define a dependency kernel function as follows:

$$K_T(A, B) = \frac{\sum_{i=1}^{n} max(S(A_T^i, B_T^1), ..., S(A_T^i, B_T^m))}{n} \tag{2.6}$$

where A and B are the input sentences and n and m are the number of dependency trigrams of A and B, respectively. A_T^i refers to the ith dependency trigram of A and B_T^j is the jth trigram of B. S is the score function which is used to compute the similarity between two dependency trigram units.

$$\begin{aligned} S(A_T^i, B_T^j) = \ & \textstyle\sum_{f=1}^{F} \alpha_f N_f(A_{T_l}^i, B_{T_l}^j) \\ & \times \textstyle\sum_{f=1}^{F} \alpha_f N_f(A_{T_c}^i, B_{T_c}^j) \\ & \times \textstyle\sum_{f=1}^{F} \alpha_f N_f(A_{T_r}^i, B_{T_r}^j) \end{aligned} \tag{2.7}$$

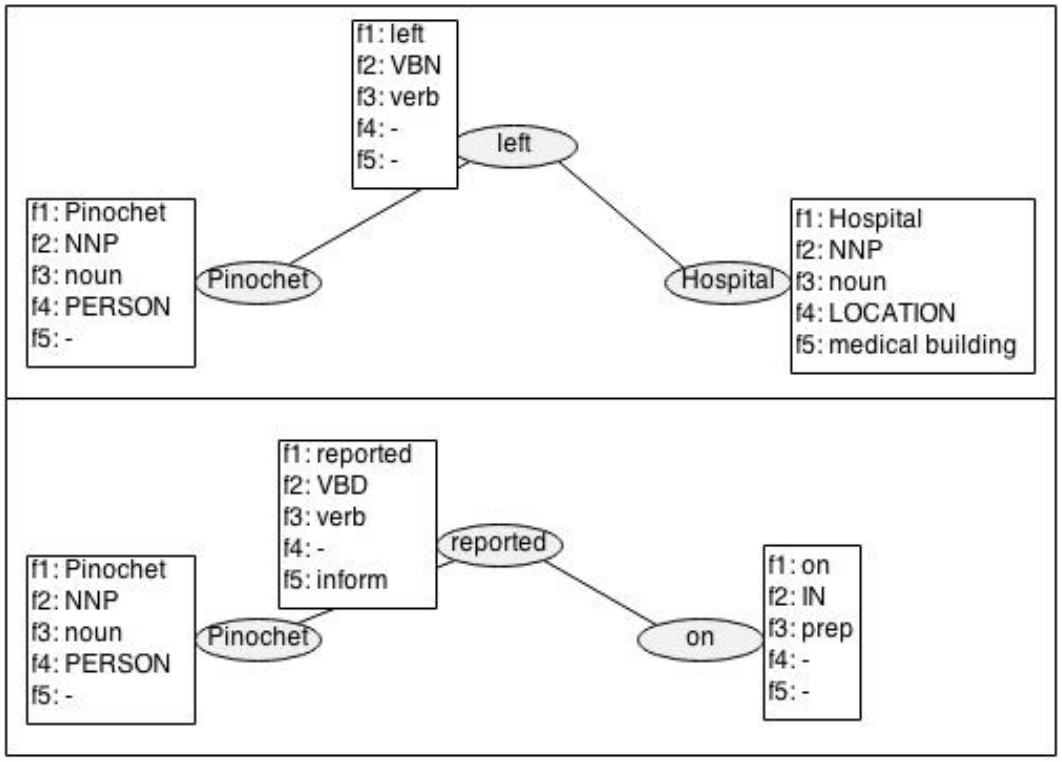

Figure 2.2. Two dependency trigram units: $Pinochet \rightarrow left \leftarrow Hospital$ and $Pinochet \rightarrow reported \leftarrow on$

where F is the number of features for each node. $A^i_{T_l}$, $A^i_{T_r}$, and $A^i_{T_c}$ denote the left node, the right node, and the center node of the dependency trigram unit A^i_T respectively. $N_f(A^i_{T_x}, B^j_{T_x})$ is a binary function which results in 1 when the fth attributes of the nodes match. α is the weight variable that is assigned for each attribute of a node which decides the effect of that attribute on the overall score. The same features are used as in the Dependency Tree Kernel for the feature sets of the trigram nodes.

For illustration purposes, let us consider the two example dependency trigram units in Figure 2.2. The $left$ nodes *Pinochet* and *Pinochet* have 4 features in common, the *center* nodes have 1 common features, and the *right* nodes have no common features. Then, the score function S computes the similarity of these two dependency trigram units as $4 \times 1 \times 0 = 0$. Although the $left$ nodes are totally matching and the *center* nodes have some matching feature, the similarity between these two trigram units is considered as 0 because of their nonmatching *right* nodes.

2.3. Bigram Match Kernels

DTK and Trigram-K do not take into account the types of dependencies in a sentence. They treat all dependencies in a dependency tree as having equal importance. However, there are different types of dependency relations in a tree and not all of them are equally important. For example, the dependency relation between a verb and its subject is in general more semantically significant than the dependency relation between a word and its determiner. This situation is visible in Figure 2.3.

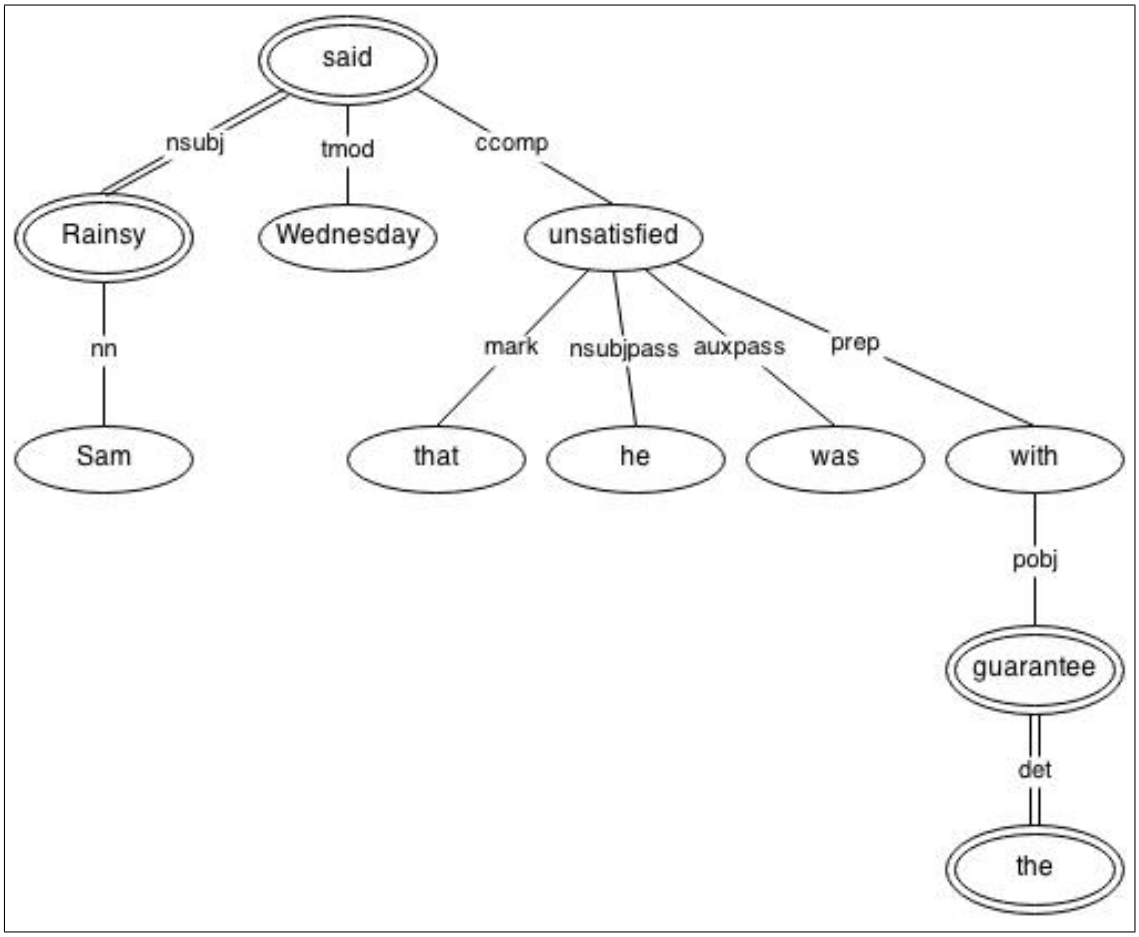

Figure 2.3. Typed dependency tree of the sentence *"Sam Rainsy said Wednesday that he was unsatised with the guarantee."* *nsubj* dependency relation is more informative than *det* dependency relation.

We design a series of new methods that make use of typed dependency grammars to compute sentence similarity. Our methods utilize dependency tree *bigram units*

and measure the similarity of two sentences using bigram unit matches. A bigram unit denotes a branch in the dependency tree, consisting of a *dependent* word, a *head* word, and the *type* of the dependency relation between them. To give an example, in Figure 2.3, the nodes *Rainsy* and *said* as well as the type of their dependency relation *nsubj* form the bigram unit $\{Rainsy, nsubj, said\}$. Measuring the similarity by only considering the word matches between sentences as in the bag-of-words model is not sufficient to capture the semantic similarity. However, bigram matches between two sentences can give more clues about their semantic similarities.

In addition to the default *DTH*: $\{dependent, type, head\}$ structure of a bigram unit, we also define two more relaxed versions as follows:

- *DH*: $\{dependent, head\}$: This version omits the *type* of the relation between *dependent* and *head* words. We cannot always identify similar sentences by looking only on *DTH* matches since searching an exact match of *DTH* structure in sentences is a very strict rule. The same relation between two words may be represented using different dependency relation types in similar sentences because of the intervention of other words in the sentences. This situation is visible in Figure 2.4. To overcome this defficiency, we introduce this second version by dropping the relation type from the *DTH* structure. The *DTH* representation of the dependency relation between the nodes *Rainsy* and *said* in Figure 2.3 is now represented as $\{Rainsy, said\}$ using *DH* bigram unit type.

- *DT-TH*: $\{dependent, type\}, \{type, head\}$: The third version actually includes a pair of bigram units: one consisting of the *dependent* word and the *type* of the relation that connects it to the *head* word, and one consisting of the *type* of the relation and *head* word. The intuition behind creating this version is that finding matching words which have the same task in the sentence can make sense in finding the similarities between sentences. The dependency relation between the nodes *Rainsy* and *said* in Figure 2.3 is now represented as $\{Rainsy, nsubj\}$, $\{nsubj, said\}$ using *DT-TH* bigram unit type.

- *DA*: {*dependent*, *ancestor*}: The previous versions take into account only the relation between a word and its parent in a dependency tree. However, the relations between a word and all of its ancestors may carry more information than the relation between the word and its parent. Two sentences may share some important common words, but the order and the type of the relations of these words may not be the same. Hence, we may not find matches of these words if we only consider word-parent relations. This situation is visible in Figure 2.5. To capture this kind of distant similarities, we define *DA* bigram version. For instance, in the second sentence of Figure 2.5, the *DA* bigram units of the subtree with the *head* node *dissolve* is: { {*government, dissolve*}, {*to, dissolve*}, {*his, dissolve*}, {*current, dissolve*}, {*his, government*}, {*current, government*} }.

We design five sentence similarity kernel methods which utilize these bigram unit structures. These kernels are described in the following subsections.

2.3.1. Strict Bigram Match Kernel

The first proposed kernel is the *Strict Bigram Match Kernel* (SBMK).

Let $A_b = \{{A_b}^1, {A_b}^2, ..., {A_b}^i, ..., {A_b}^{m-1}, {A_b}^m\}$ be the bigram set of sentence A where ${A_b}^i$ can be one of *DTH, DH, DT-TH, or DA*. Then we define the Strict Bigram Match Kernel as:

$$K_1(A, B) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} M({A_b}^i, {B_b}^j)}{m + n} \tag{2.8}$$

where A and B are the input sentences and m and n are the number of dependency

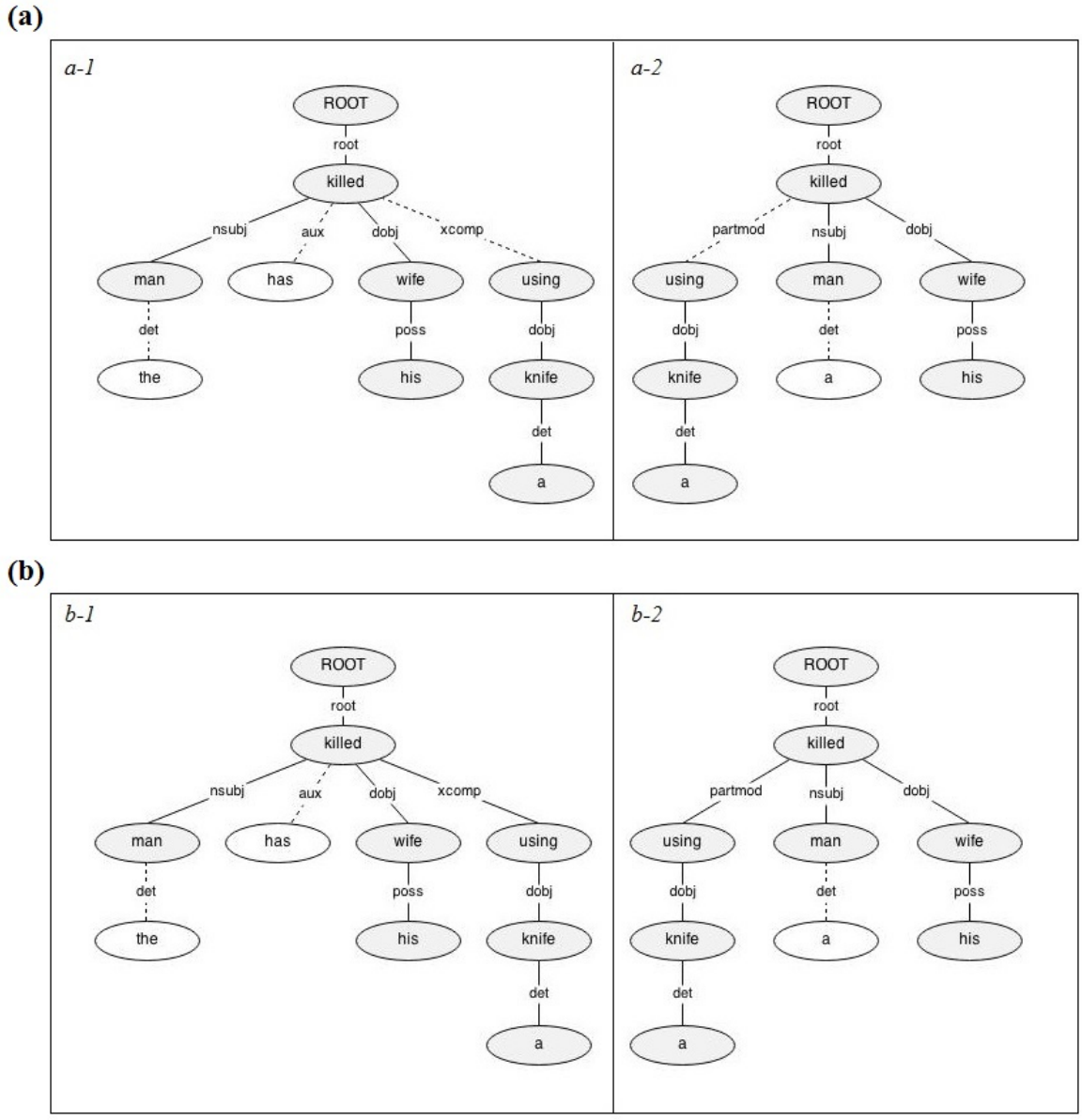

Figure 2.4. (a) Dependency trees of the sentences *"The man has killed his wife using a knife."* and *"Using a knife, a man killed his knife."*, respectively. Gray nodes with continuous lines constitute matching bigram units when *DTH* is used as the bigram type. The number of matching bigrams is 6. (b) When we use *DH* as the bigram type, the number of matching bigrams is 7. The non-matching bigrams {*using, xcomp, killed*} and {*using, partmod, killed*} is now matching as we omit the dependency type.

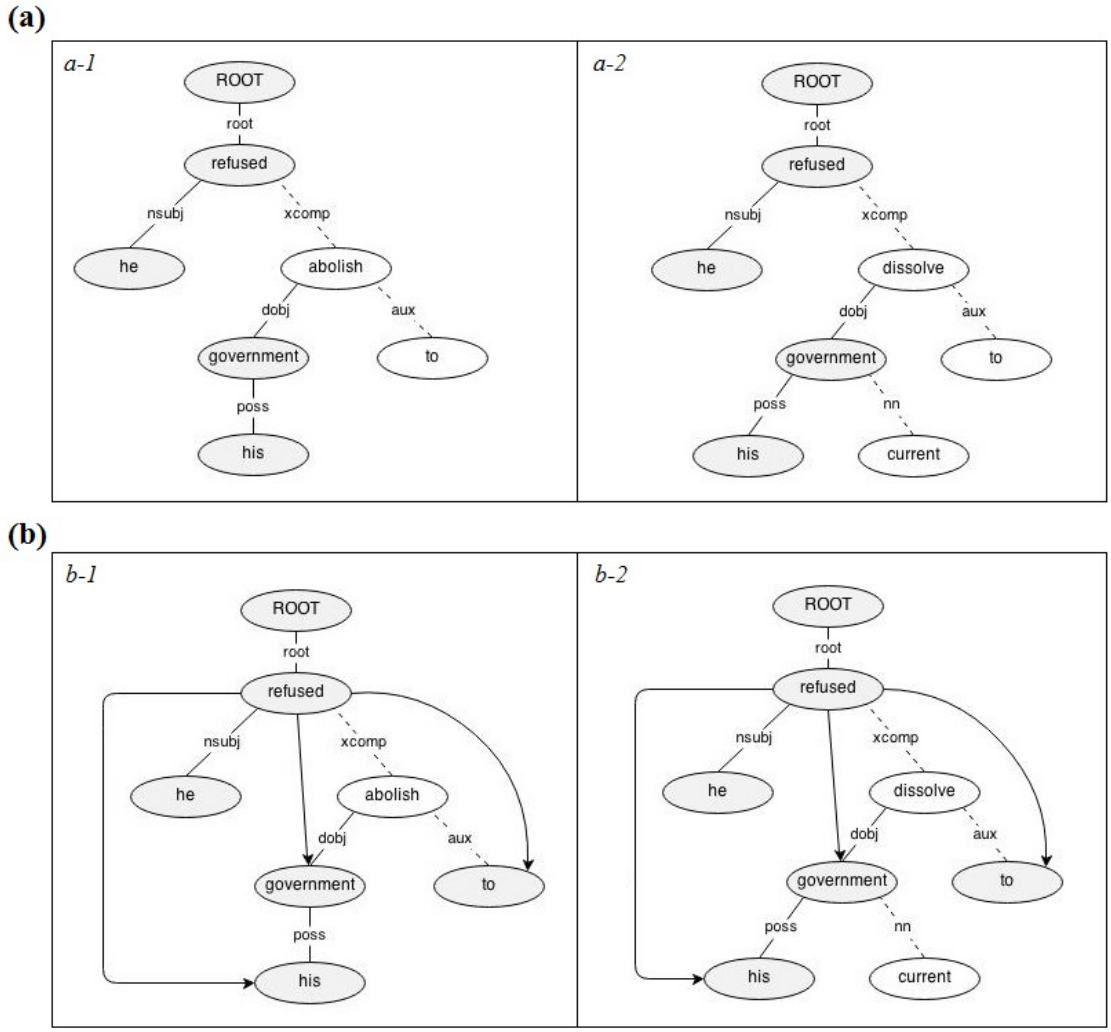

Figure 2.5. (a) Dependency trees of the sentences *"He refused to abolish his government."* and *"He refused to dissolve his current government."* respectively. Gray nodes with continuous lines constitute matching bigram units when DTH or DH is used as the bigram type. (b) When DA is used as the bigram type, three additional relations are formed. Only this type captures the relation between words *refused* and *government*.

bigrams of A and B, respectively. The function M is defined as:

$$M(a,b) = \begin{cases} 1, & if \;\; a = b \\ 0, & otherwise \end{cases} \tag{2.9}$$

This kernel counts the number of bigram matches between two sentences. The score is then normalized by the sum of cardinalities of bigram sets of the sentences. To create the bigram set of a sentence, we extract all *dependent-head* relations from its corresponding dependency tree and denote them using at least one of the bigram unit versions. For illustration purposes, let us consider the two sentences of Figure 2.5. The complete dependency trees of these sentences are depicted in Figure 2.6. If we use DTH as the bigram unit type, the bigram set of the first sentence would be:

$$A_b = \{\{refused, root, ROOT\}, \{he, nsubj, refused\}, \{abolish, xcomp, refused\}, \{government, dobj, abolish\}, \{his, poss, government\}, \{to, aux, abolish\}\}$$

and the second sentence would be represented with the following bigram unit set:

$$B_b = \{\{refused, root, ROOT\}, \{he, nsubj, refused\}, \{dissolve, xcomp, refused\}, \{government, dobj, dissolve\}, \{his, poss, government\}, \{current, nn, government\}, \{to, aux, dissolve\}\}$$

The number of matching bigram units between A_b and B_b is 3. So, SBMK would compute the similarity score between these two sentences as $3/(6+7) = 0.23$.

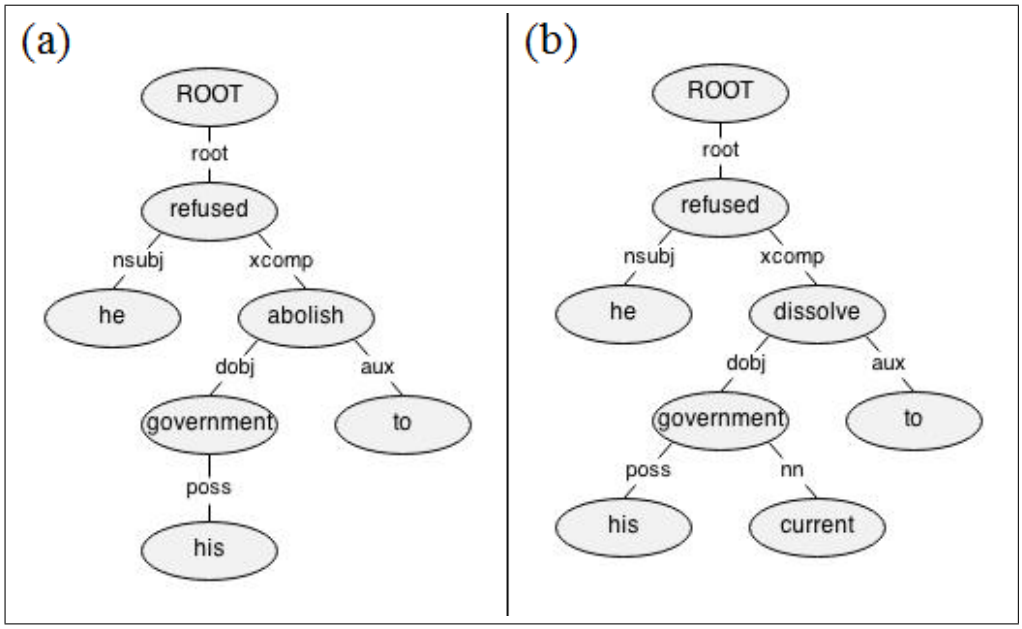

Figure 2.6. Dependency trees of the sentences *"He refused to abolish his government."* and *"He refused to dissolve his current government."* respectively

2.3.2. Approximate Bigram Match Kernels

The Strict Bigram Match Kernel entails overlapping of the whole bigram unit in order to count it as a match. Due to its strictness, this requirement may not capture the similarity between sentences that have different syntactic structures, but share a similar meaning. To overcome this problem, we design approximate bigram match kernels. As a first attempt, we relax the match function of the Strict Bigram Match Kernel and design an approximate bigram match kernel.

Let us first define ${A_b}^i = \{{d_A}^i, {t_A}^i, {h_A}^i\}$ as the ith bigram of sentence A where

${d_A}^i$: dependent node of ${A_b}^i$,
${t_A}^i$: type of ${A_b}^i$,
${h_A}^i$: head node of ${A_b}^i$.

Then, the *Simple Approximate Bigram Match Kernel* (SABMK) is defined as

follows:

$$K_2(A,B) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} sim({A_b}^i, {B_b}^j)}{m+n} \tag{2.10}$$

The function *sim* is defined as:

$$sim({A_b}^i, {B_b}^j) = \left[(s({d_A}^i, {d_B}^j) + s({h_A}^i, {h_B}^j)\right] \times c({t_A}^i, {t_B}^j) \tag{2.11}$$

where s and c are binary functions:

$$s(a,b) = \begin{cases} 1, & \text{if } a = b \\ 0, & \text{otherwise} \end{cases} \tag{2.12}$$

$$c(a,b) = \begin{cases} \theta, & \text{if } a = b \\ 1, & \text{otherwise} \end{cases} \tag{2.13}$$

and θ is a constant that determines the influence of a type match.

The function *sim* gives partial scores to bigram unit comparisons, even if they do not totally match with each other. With this function, bigram units that have only a common dependent word or a head word are considered as an approximate match. If their types are also overlapped, the similarity score is increased by a factor of θ. This approximate match structure of *sim* function actually covers all bigram unit types other than *DA*. To show how approximate bigram match kernel approach works, let us again

consider the two sentences in Figure 2.6. The Simple Approximate Bigram Match Kernel computes the numerator in equation 2.10 as:

$$(1+1)\cdot\theta + (1+1)\cdot\theta + (0+1)\cdot 1 + (0+1)\cdot 1 + (0+1)\cdot\theta$$
$$+ (1+0)\cdot\theta + (1+1)\cdot\theta + (0+1)\cdot 1 + (1+0)\cdot\theta = 9\theta + 3$$

If we take θ as 2, the similarity score would be $(9 \cdot 2 + 3)/(6 + 7) = 1.62$.

The Simple Approximate Bigram Match Kernel treats all words as having equal importance. However, this is hardly true for many cases. To model the importance of a word, we include the tf-idf (term frequency - inverse document frequency) values of the dependent and head words to the formula and form the *TF-IDF Based Approximate Bigram Match Kernel* (TABMK) as defined below:

$$K_3(A,B) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} sim_t({A_b}^i, {B_b}^j)}{N(A) \times N(B)} \tag{2.14}$$

where $N(A)$ is the normalizer function that uses tf-idf values of all dependent and head words in a dependency tree:

$$N(A) = \sqrt{\sum_{i=1}^{n} (tf_{{d_A}^i}\ idf_{{d_A}^i})^2 + (tf_{{h_A}^i}\ idf_{{h_A}^i})^2} \tag{2.15}$$

and sim_t is a tf-idf based similarity function defined between two dependency bigram

units as follows:

$$sim_t({A_b}^i, {B_b}^j) = \Big[(tf_{{d_A}^i}\ idf_{{d_A}^i} \times tf_{{d_B}^j}\ idf_{{d_B}^j}) \times s({d_A}^i, {d_B}^j) + (tf_{{h_A}^i}\ idf_{{h_A}^i} \times tf_{{h_B}^j}\ idf_{{h_B}^j}) \times s({h_A}^i, {h_B}^j)\Big] \times c({t_A}^i, {t_B}^j) \quad (2.16)$$

We define another version of this kernel by replacing the inner summation function in equation 2.14 with a max function that returns the maximum of its arguments. This version is named as *TF-IDF Max Approximate Bigram Match Kernel* (TABMK-Max) and defined as follows:

$$K_4(A, B) = \frac{\sum_{i=1}^{n} max(sim_t({A_b}^i, {B_b}^1), ..., sim_t({A_b}^i, {B_b}^m))}{N(A) \times N(B)} \quad (2.17)$$

For each bigram unit of the sentence A, K_4 looks at the similarity scores of it with each of the bigram units of the sentence B, finds the maximum of them, and sums these scores to compute the similarity score of sentences A and B.

The TF-IDF Based Approximate Bigram Match Kernel and the TF-IDF Max Approximate Bigram Match Kernel do not encourage consecutive bigram matches that form a subtree in the dependency trees of sentences. However, a common subtree in the dependency trees of two sentences means that these sentences contain similar subparts. For instance, let us consider the two sentences S_1 and S_2 written below.

S_1 : *Opposition leaders Prince Norodom Ranariddh and Sam Rainsy, citing Hun Sen's threats to arrest opposition figures after two alleged attempts on his life, said they could not negotiate freely in Cambodia and called for talks at Sihanouk's residence in Beijing.*

S_2 : *Citing Hun Sen's threats to arrest opposition politicians following two alleged attempts on his life, Ranariddh and Sam Rainsy have said they do not feel safe negotiating inside the country and asked the king to chair the summit at his residence in Beijing.*

Figure 2.7 depicts their corresponding dependency trees. The subtrees with double-circled nodes denote the same subparts of the sentences. This common subtrees indicate that these two sentences have very similar substructures. The TF-IDF Based Approximate Bigram Match Kernel and the TF-IDF Max Approximate Bigram Match Kernel cannot capture these similarities since they do not take into account the location of matching bigram units.

To emphasize this point, we design the *Matching Subtrees Kernel* (MSK) below:

$$K_5(A,B) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} sim_t({A_b}^i, {B_b}^j) + \sum_{k=1}^{K}\sum_{l=1}^{L} s({d_A}^i, {d_B}^j) \times K_c(c_{{d_A}^i}(k), c_{{d_B}^j}(l))}{N(A) \times N(B)} \quad (2.18)$$

where c_x denotes the set of children nodes of the node x and $c_x(i)$ is the ith child of the node x. K and L are the number of children of ${d_A}^i$ and ${d_B}^j$, respectively. Children kernel K_c is defined as:

$$K_c(n_i, n_j) = \begin{cases} sim_t(n_i, n_j) + \nu \cdot K_c(a_i, b_j) & \forall a_i \in c_{n_i} \text{ and } \forall b_j \in c_{n_j}, \\ & \text{if } d_i = d_j \text{ and } t_i = t_j \\ 0, & \text{otherwise} \end{cases} \quad (2.19)$$

Here, ν is a decaying factor to prevent high increase in the similarity score. In

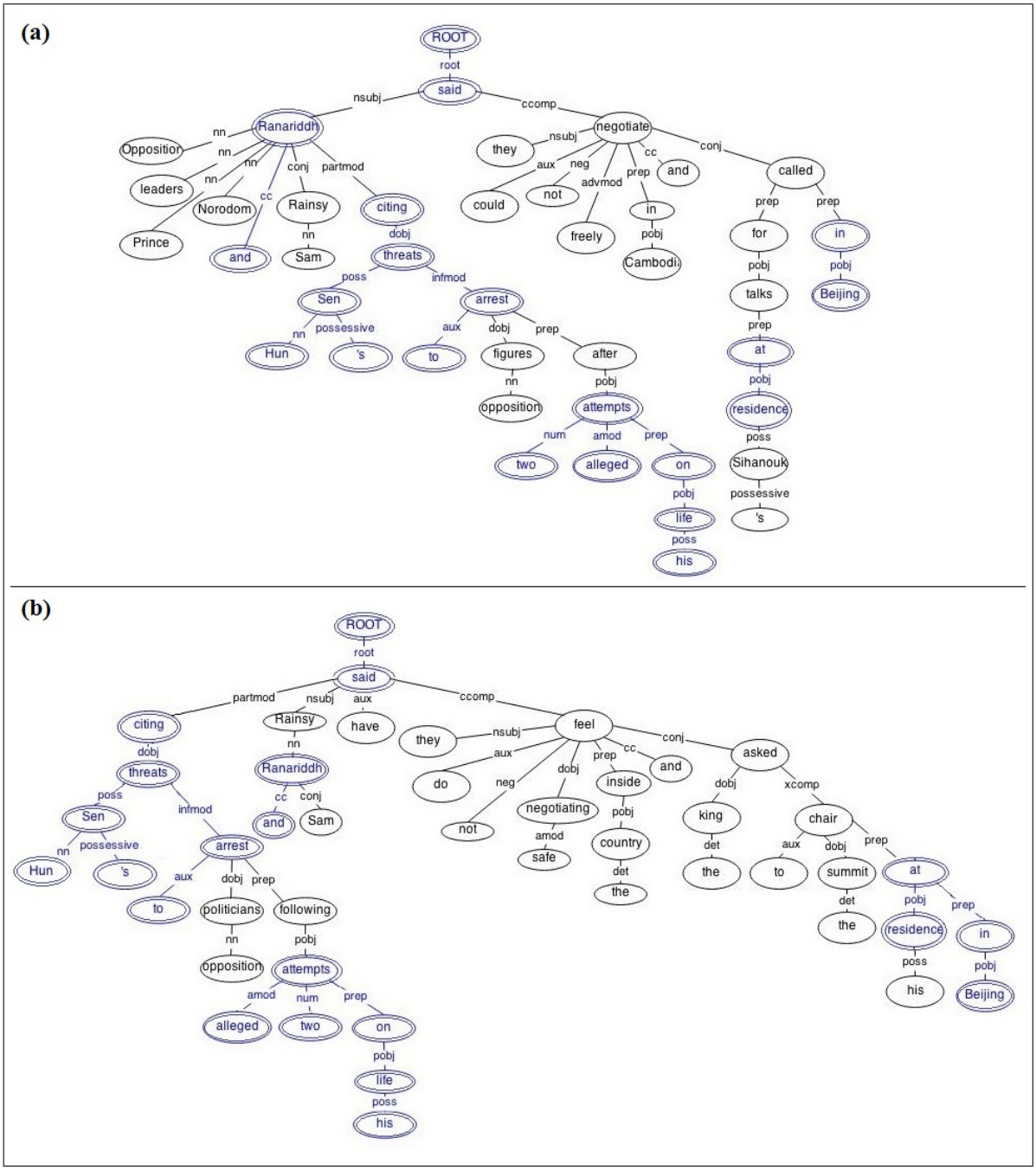

Figure 2.7. D_1 : (a) Dependency tree of the first sentence S_1. (b) Dependency tree of the second sentence S_2. The subtrees with double-circled nodes denote the matching dependency relations.

addition to comparing each bigram unit of the first sentence with each bigram unit of the second sentence, this kernel also tries to find matching subtrees by comparing children nodes of matching bigrams. Actually K_5 involves K_3 in it. K_c recursively compares children of a matching dependent word pair, and scores the comparison of two child nodes if their dependent words and types match.

A simpler version of $K_c(n_i, n_j)$ can be:

$$K_{c_2}(n_i, n_j) = \begin{cases} \alpha \cdot s(n_i, n_j) + \nu \cdot K_{c_2}(a_i, b_j) & \forall a_i \in c_{n_i} \text{ and } \forall b_j \in c_{n_j}, \\ & \text{if } d_i = d_j \text{ and } t_i = t_j \\ 0, & \text{otherwise} \end{cases} \tag{2.20}$$

Here, we give a fixed score of α to matching children nodes of a matching bigram pair instead of giving a score based on the tf-idf values of the words.

We also derive a *Composite Kernel* (CK) formula by combining the tf-idf based kernels as follows:

$$K_{mix}(A, B) = \beta K_4(A, B) + \delta K_5(A, B) \tag{2.21}$$

where β and δ determine the influence of the corresponding kernel in the calculation of sentence similarity.

Similarity scores returned by each kernel are normalized so that the highest similarity value gets the score 1 and the lowest similarity value gets the score 0. While computing the similarity scores of sentences, we multiply the score of matching bigram

units with a constant u if the bigram units contain a *dependent* or *head* word which begins with an uppercase letter based on a simple heuristic that words beginning with an uppercase letter are usually entity names and so are more important. We experiment with different values of u to measure the effect of this heuristic on similarity calculation.

3. EXPERIMENTS

We evaluated the performance of the untyped dependency tree kernels, the bigram match kernels, and the baseline approaches for multi-document summarization on two different data sets. In this chapter we describe the data sets, the evaluation metric, and the tools we used in our experiments. Then we define the steps of the experiments we performed for the untyped dependency tree kernels and the bigram match kernels.

3.1. Data Sets

We used two different data sets that are widely used in the multi-document summarization area. In this section we describe these data sets.

The first data set we used in our experiments is Task 2 data set of DUC 2003[1] which is a generic multi-document summarization task. This data set comprises 30 English document clusters and each cluster contains 10 newswire documents which were collected from the following news collections:

- AP newswire (1998-2000)
- New York Times newswire (1998-2000)
- English version of Xinhua News Agency (1996-2000)

The second data set used in this study is Task 2 of DUC 2004 data set[2] which is widely used in the generic multi-document summarization task. This data set contains 50 English newswire document clusters where each cluster consists of 10 documents on average. The newswire documents were gathered from the AP newsire and New York Times newswire.

[1] `http://duc.nist.gov/duc2003/tasks.html`
[2] `http://duc.nist.gov/duc2004/tasks.html`

3.2. Evaluation Metric

As the evaluation metric, ROUGE[3] (Recall Oriented Understudy of Gisting Evaluation) automated summary evaluation metric, which was developed by Chin-Yew Lin, is used. It is a recall-based metric and basically performs automatic n-gram matching to accomplish its task. It is defined formally in [27] as follows:

$$ROUGE\text{-}N = \frac{\sum_{S \in RefSum} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in RefSum} \sum_{gram_n \in S} Count(gram_n)} \tag{3.1}$$

Here $RefSum$ is the set of reference summaries, $gram_n$ represents an n-gram in the summary where n is the length of the n-gram, and $Count_{match}$ counts the maximum number of common n-grams between the candidate summary and reference summaries. It is shown that, the unigram based ROUGE score ROUGE-1 agrees most with the human judgements [28].

We generated 665-byte candidate summaries for each of the clusters in the data sets and computed ROUGE-1 scores using the reference summaries.

3.3. Tools

In our experiments, we made use of several different tools in the pre-processing and evaluation steps of the dependency tree kernels.

- We used the Stanford Parser [29], which is a Java coded package, for creating dependency grammars and for finding feature values used in the Dependency Tree Kernel and the Dependency Trigram Kernel.

[3] `http://www.berouge.com/`

- We used the Stanford Named Entity Recognizer (NER) tool which is available in the Stanford CoreNLP Package[4] in order to find the *entity type* features of the nodes in the dependency trees of the sentences.

- We also used JAWS[5] (Java API for WordNet Searching) to extract hypernym sets of a word from the WordNet database [30] in order to be used as a feature.

- We used Porter Stemmer [31] to stem all the words in the data sets we used in our experiments.

- We used the Dragon Toolkit[6] which is a development package for Information Retrieval and Text Mining [32], to develop the dependency grammar kernels embedded in the LexRank summarization method.

- We used the MEAD summarization platform [33] to evaluate the sentence similarity kernels. MEAD is a multi-document summarization system that performs extractive summarization on documents. The LexRank method is available in MEAD.

- We used the LexRank method which is available in MEAD. LexRank is a random-walk based method for computing sentence importance. It was proposed for generic summarization. The LexRank algorithm represents a document collection with a graph where vertices are sentences and edges between vertices denote the similarity among the corresponding pairs of sentences. In order to determine the centralities of the sentences, the LexRank uses the original PageRank method [34]. The most central sentences are considered to be the most important ones. The summary is created by selecting the sentences according to their importance. To deal with the redundancy problem, LexRank system uses a sentence reranker.

[4] `http://nlp.stanford.edu/software/corenlp.shtml`
[5] `http://lyle.smu.edu/~tspell/jaws/`
[6] `http://dragon.ischool.drexel.edu/`

- We used CSIS (Cross-Sentence Informational Subsumption) reranker [7] with the LexRank method. The main idea of the CSIS reranker is that, if the content of a sentence is present in another sentence with some additional information, then the former sentence is redundant in the presence of the latter, in other words the former sentence is subsumed by the latter. So the CSIS reranker reduces the importance scores of sentences that are already included in the summary by adjusting them according to the subsumption idea.

3.4. Evaluation of the Sentence Similarity Kernels

Erkan and Radev [9], evaluated their LexRank method inside the MEAD summarization system [33] where they combined the LexRank method with Position and Length features, and used the CSIS reranker to eliminate redundancy.

Since we tested our similarity in the LexRank system by replacing its own sentence similarity method, we used their environment in our experiments.

3.4.1. Evaluation of the Dependency Tree Kernel and the Dependency Trigram Kernel

The pre-processing steps of the experiments performed for untyped dependency tree kernels are as follows:

- We first generated the dependency parse trees of the sentences in our data sets using the Stanford Parser.

- Then, the feature set of each node in the tree was created. The *word, POS tag,* and *general POS tag* features were generated using the Stanford Parser. The *entity type* feature was created using the Stanford NER tool.The *WordNet hypernyms* feature was generated using the JAWS tool.

Table 3.1. Parameter Setup of the LexRank System

Parameter	**Value**
LexRank Centrality Weight	1
Position Feature Weight	1
Length Feature Cut-off	9
LexRank Centrality Threshold	continuous
Reranker Type	CSIS Reranker
Reranker Threshold	0.5

After these pre-processing steps, we ran the LexRank multi-document summarization method with the parameter set shown in Table 3.1.

Both the Dependency Tree Kernel and the Dependency Trigram Kernel are designed to use untyped dependency tree representations of sentences. To see the impact of the type of dependency relations in the sentence similarity calculation task, we integrated type information into these kernels using the methods outlined below.

- Introducing the dependency relation type as a new feature.

- Filtering the nodes that have unimportant dependency relation types in Table 3.2.

- Combining the dependency relation type with the *word* feature such that if the *word* features of two nodes match and they have the same dependency relation type, an extra score is given to that match.

- Including the dependency relation type as a matching feature (*specific to the Dependency Tree Kernel*).

Table 3.2. Unimportant dependency relation types. These types are considered as insignificant and nodes whose dependency types are one of them are ignored in sentence similarity calculations if these types are filtered.

Type Abbreviation	**Type Name**
det	determiner
expl	expletive
goeswith	goes with
possessive	possessive modifier
preconj	preconjunct
predet	predeterminer
prep	prepositional modifier
punct	punctuation
ref	referent

We also experimented with these two kernels by using only the *word* feature in feature comparisons. The motivation behind this experiment is to see whether the other features introduce noise to similarity calculation or not.

In [24], Culotta and Sorensen used *general-POS, entity-type,* and *relation argument* as the features of the *matching* function in the Dependency Tree Kernel. Since our aim is not finding entity relations, we only used the *general-POS* as the matching feature in the models other than the model that uses type information as a matching feature. In that model, we used *general-POS* and *type* features together as matching features.

In [25], Choi and Kim used the Dependency Trigram Kernel only on the sentences that include social relations. However, we adapted this kernel to work on all sentences on the data set. So, grouping each node with each of the other nodes in each dependency tree under a common ancestor and composing a trigram unit out of them results in relating irrelevant nodes with each other. This situation leads to a decline in the performance of the system. To overcome this problem, we put a restriction on the

common ancestor such that the common ancestor must be either the first degree or the second degree ancestor of two nodes in order for them to form a trigram unit. Table 3.3 shows the results of the experiments made using the first degree ancestor trigrams and the second degree ancestor trigrams.

Table 3.3. The effect of the degree of the common ancestor in trigram formation for the Dependency Trigram Kernel

Trigram-K	**Rouge-1 Precision**	**Rouge-1 Recall**	**Rouge-1 F1 score**
1st deg. ancestor	0.340	0.348	**0.344**
2nd deg. ancestor	0.341	0.347	**0.344**

The Dependency Trigram Kernel using only the first degree ancestor trigrams shows the same performance with the model that uses the first and the second degree ancestors. Therefore, we used this version in all of the experiments of the Dependency Trigram Kernel in order to reduce the time complexity.

3.4.2. Evaluation of the Bigram Match Kernels

We performed experiments on the DUC 2003 and DUC 2004 data sets to measure the performances of the bigram match kernels. In the pre-processing phase, we first stem all the words in the data sets using the Porter Stemmer and then extract the dependency bigram units from the dependency trees of the sentences in the data sets.

We evaluated the Strict Bigram Match Kernel with different bigram unit types. For the Simple Approximate Bigram Match Kernel, the TF-IDF Based Approximate Bigram Match Kernel, and TF-IDF Max Approximate Bigram Match Kernel, we experimented with different combinations of the uppercase word weight u and type weight θ. For u and θ, we tried the values in the ranges (1.0, 3.0) and (1.5, 3.5), respectively. For the Matching Subtrees Kernel, we experimented both with K_c and K_{c_2} by setting the decaying factor ν to 0, 0.5, and 1. For all of the bigram match kernels, we filtered the dependency relation types in Table 3.2.

The results were then compared with the original LexRank system that uses tf-idf based cosine similarity function as well as the Lead-based summarization approach, which selects sentences by using only the Position feature [9], and the Random approach which composes a summary by selecting sentences in a random manner as the baseline approaches.

4. RESULTS AND DISCUSSION

In this chapter, the results of the experiments we performed for the different sentence similarity methods on DUC 2003 and DUC 2004 Task 2 data sets are presented.

4.1. Results of the Dependency Tree Kernel and the Dependency Trigram Kernel

Tables 4.1 and 4.2 show the ROUGE-1 scores of the Dependency Tree Kernel without including dependency type *(untyped)*, with dependency type as a similarity feature *(type as a sim ft.)*, with dependency type used as a match feature *(type as a match ft.)*, with filtering unimportant types *(unimp. type filter.)*, with including the type into the *word* feature *(word and type)*, and with using only the *word* feature as a feature set *(only word ft.)* on DUC 2003 and DUC 2004 Task 2 data sets. According to these results, filtering the unimportant dependency relation types, using only the *word* feature, and using type information as a matching feature show the best performances in terms of ROUGE-1 score.

Table 4.1. ROUGE-1 scores of the Dependency Tree Kernel with different settings in LexRank system on DUC 2003 Task 2 data set

DUC 2003 Task 2			
DTK	**Rouge-1 Precision**	**Rouge-1 Recall**	**Rouge-1 F score**
untyped	0.327	0.342	0.334
type as a sim ft.	0.327	0.341	0.333
type as a match ft.	0.329	0.344	0.336
unimp. type filter.	0.330	0.344	**0.337**
word and type	0.325	0.340	0.332
only word ft.	0.328	0.344	0.336

Tables 4.3 and 4.4 show the ROUGE-1 scores of the Dependency Trigram Kernel models on DUC 2003 and DUC 2004 Task 2 data sets. The best ROUGE-1 score is

Table 4.2. ROUGE-1 scores of the Dependency Tree Kernel with different settings in LexRank system on DUC 2004 Task 2 data set

DUC 2004 Task 2			
DTK	**Rouge-1 Precision**	**Rouge-1 Recall**	**Rouge-1 F score**
untyped	0.340	0.346	**0.343**
type as a sim ft.	0.340	0.345	0.342
type as a match ft.	0.340	0.346	**0.343**
unimp. type filter.	0.340	0.346	**0.343**
word and type	0.339	0.344	0.341
only word ft.	0.339	0.347	**0.343**

reached when we use only the *word* feature in feature comparisons on DUC 2003. On DUC 2004 data set, filtering unimportant dependency relation types gives the best score while using only the *word* feature gives the second best score.

Table 4.3. ROUGE-1 scores of the Dependency Trigram Kernel with different settings in LexRank system on DUC 2003 Task 2 data set

DUC 2003 Task 2			
Trigram-K	**Rouge-1 Precision**	**Rouge-1 Recall**	**Rouge-1 F score**
untyped	0.327	0.345	0.336
type as a sim ft.	0.325	0.342	0.333
unimp. type filter.	0.324	0.342	0.332
word and type	0.329	0.348	0.338
only word ft.	0.334	0.348	**0.341**

By looking at the results of DUC 2003 and DUC 2004 data sets, we can say that features other than the *word* itself may include misleading information to the system. Considering the computational burden of *entity type* and *hypernym* features, we recommend that using only the *word* feature is sufficient for both of the kernels. Besides, filtering the unimportant dependency relation types when using only the *word*

Table 4.4. ROUGE-1 scores of the Dependency Trigram Kernel with different settings in LexRank system on DUC 2004 Task 2 data set

DUC 2004 Task 2			
Trigram-K	**Rouge-1 Precision**	**Rouge-1 Recall**	**Rouge-1 F score**
untyped	0.341	0.348	0.344
type as a sim ft.	0.343	0.350	0.346
unimp. type filter.	0.345	0.352	**0.348**
word and type	0.341	0.348	0.344
only word ft.	0.344	0.352	0.347

feature can increase their performances.

The results show that the Dependency Trigram Kernel performs better than the Dependency Tree Kernel in terms of ROUGE-1 score. The trigram unit comparison method better captures the similar substructures in sentences. However, this method is too strict to work on multi-document summarization. So, we came up with the bigram unit approach which has a more flexible structure than trigram units.

4.2. Results of the Bigram Match Kernels

We conducted experiments on the DUC 2003 data set with different parameter combinations for each of the bigram match kernels.

Figure 4.1 depicts the ROUGE-1 scores of the Strict Bigram Match Kernel with different combinations of bigrams. The best score is reached using *DH* as the bigram unit type.

For the Simple Approximate Bigram Match Kernel, TF-IDF Based Approximate Bigram Match Kernel, and TF-IDF Max Approximate Bigram Match Kernel, we measured the effect of the uppercase word weight u with different θ combinations on the DUC 2003 data set. Figures 4.2, 4.3, and 4.4 depict the changes in ROUGE-1 scores

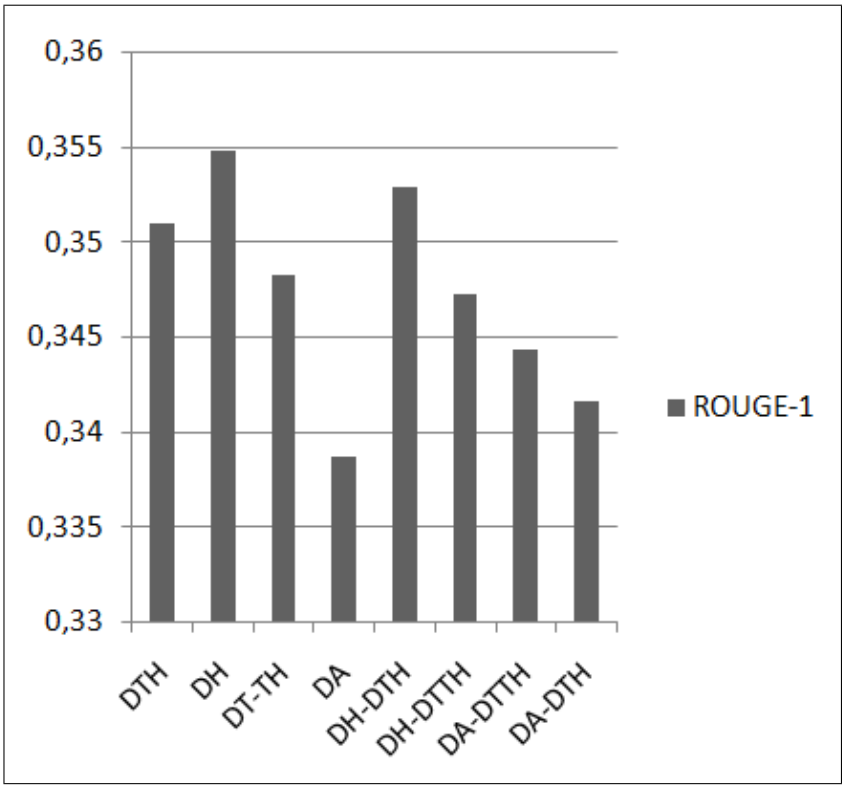

Figure 4.1. ROUGE-1 scores of different runs of the Strict Bigram Match Kernel on DUC 2003 Task 2 data set

of these kernels with different u-θ combinations.

We observed that setting θ to 1.5 gives the best ROUGE-1 scores for all u values except 3 in the Simple Approximate Bigram Match Kernel.

For the TF-IDF Based Approximate Bigram Match Kernel, the best performance is achieved when θ is 3 and the uppercase word weight u is 1.6 and 1.7.

TF-IDF Max Approximate Bigram Match Kernel shows the best performance when both θ and u are 1.5. According to the results in Figure 4.4, setting θ to 1.5 or 2 gives better results in terms of ROUGE-1 scores.

Tables 4.5 and 4.6 show the ROUGE-1 scores for the experiments made for the bigram match kernels, the best models of the untyped dependency tree kernels, and the baseline models on DUC 2003 and DUC 2004 Task 2 data sets. The Strict Bigram

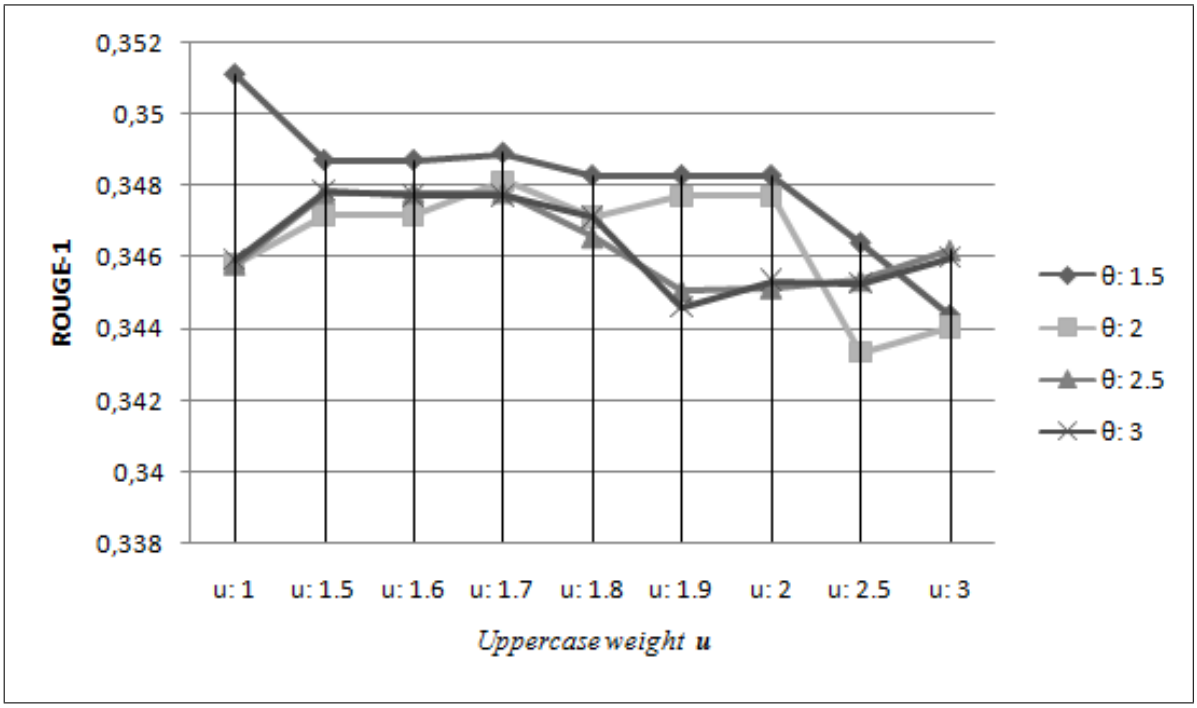

Figure 4.2. ROUGE-1 scores of different runs of the Simple Approximate Bigram Match Kernel using different u and θ combinations on DUC 2003 Task 2 data set

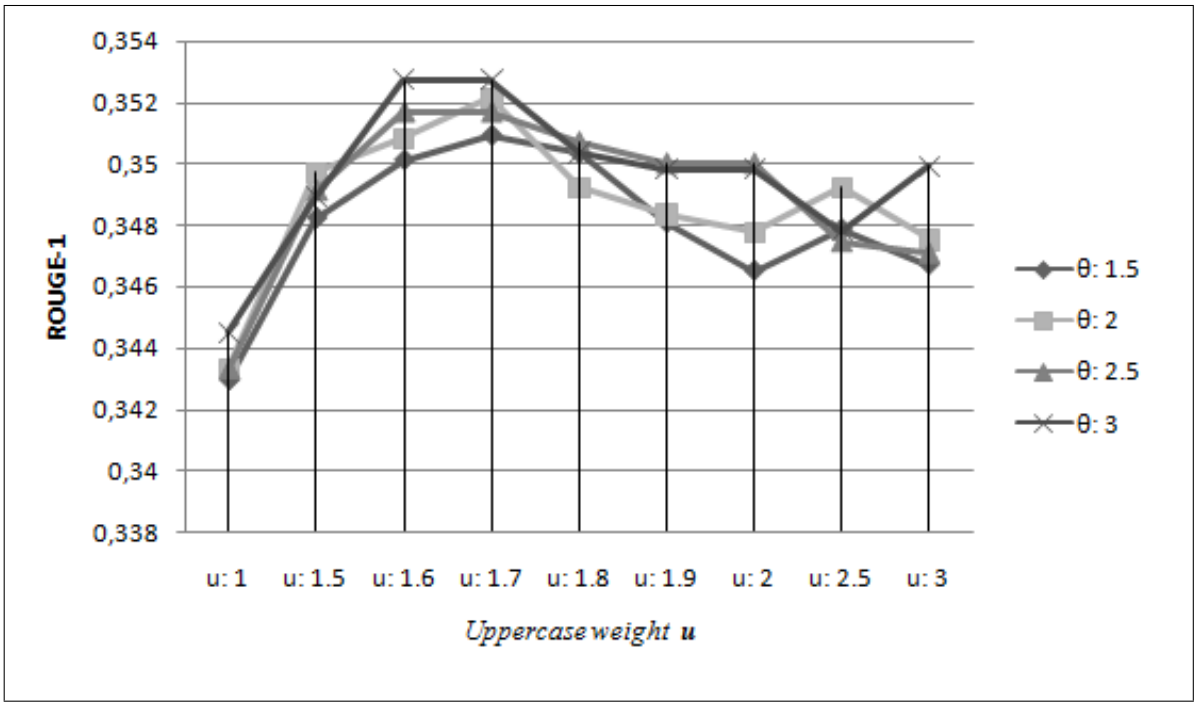

Figure 4.3. ROUGE-1 scores of different runs of the TF-IDF Based Approximate Bigram Match Kernel using different u and θ combinations on DUC 2003 Task 2 data set

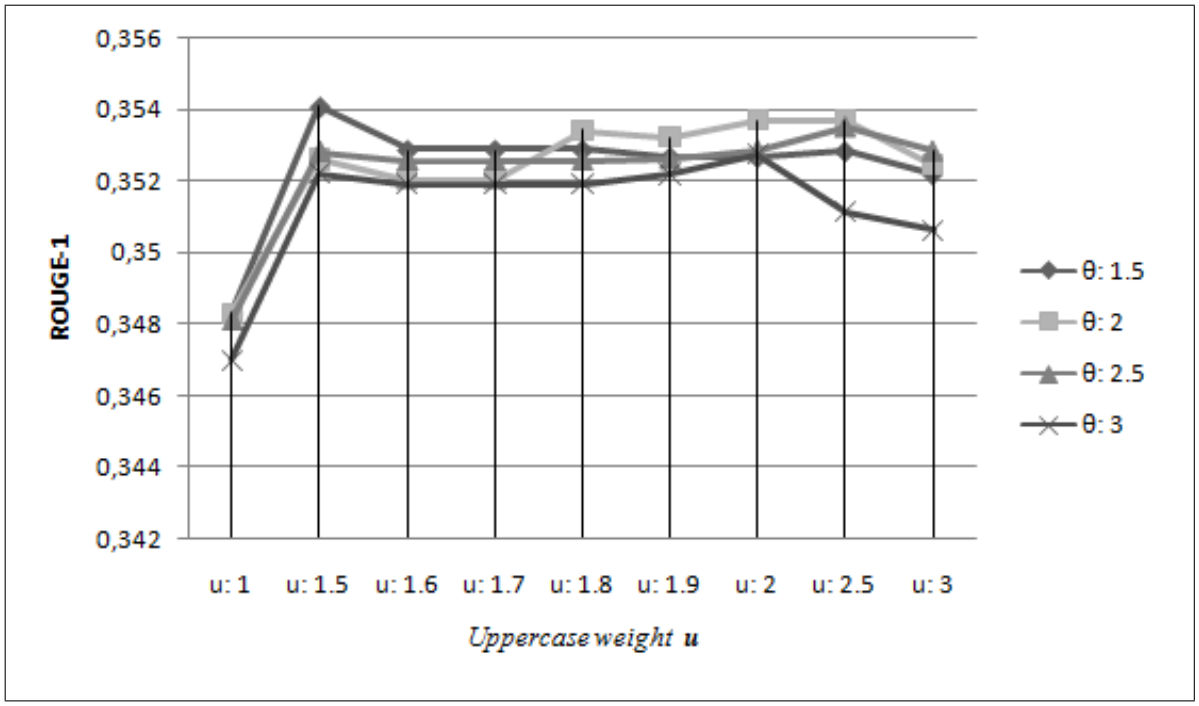

Figure 4.4. ROUGE-1 scores of different runs of the TF-IDF Max Approximate Bigram Match Kernel using different u and θ combinations on DUC 2003 Task 2 data set

Match Kernel outperforms all of the other methods on DUC 2003 Task 2 data set but TF-IDF Based Approximate Bigram Match Kernel and TF-IDF Max Approximate Bigram Match Kernel show almost the same performance with the Strict Bigram Match Kernel. All of our approximate bigram match kernels achieve better ROUGE-1 scores than the untyped dependency tree kernels and LexRank's original similarity method. However, the Strict Bigram Match Kernel shows worse performance than LexRank's original similarity method on DUC 2004. The Composite Kernel shows the best performance on DUC 2004 Task 2 data set. The Dependency Tree Kernel outperforms the lead-based method only on DUC 2004. The Dependency Trigram Kernel shows better performance than the lead-based method on both data sets. However, these untyped dependency tree based approaches failed to achieve higher scores than the original similarity method of the LexRank system.

In Appendix A, we present a sample 665-byte system generated summary and the corresponding human generated summaries of the $23rd$ cluster of DUC 2004 Task

2 data set. The system generated summary is generated using the Composite Kernel as the sentence similarity kernel.

Table 4.5. ROUGE-1 scores of the bigram match kernels, untyped dependency tree kernels, and baseline models in LexRank system on DUC 2003 Task 2 data set

DUC 2003 Task 2				
All Systems	**Rouge-1 Prec.**	**Rouge-1 Rec.**	**Rouge-1 F**	**Std. Dev.**
SBMK	0.348	0.362	**0.355**	0.052
SABMK	0.343	0.360	0.351	0.044
TABMK	0.347	0.362	0.354	0.051
TABMK-Max	0.346	0.363	0.354	0.050
MSK	0.346	0.360	0.353	0.050
CK	0.345	0.360	0.352	0.052
DTK (best)	0.330	0.344	0.337	0.049
Trigram-K (best)	0.334	0.348	0.341	0.053
LexRank (tf-idf)	0.341	0.360	0.350	0.048
Lead-based	0.331	0.345	0.338	0.052
Random	0.281	0.299	0.289	0.046

Table 4.6. ROUGE-1 scores of the bigram match kernels, untyped dependency tree kernels, and baseline models in LexRank system on DUC 2004 Task 2 data set

DUC 2004 Task 2				
All Systems	**Rouge-1 Prec.**	**Rouge-1 Rec.**	**Rouge-1 F**	**Std. Dev.**
SBMK	0.353	0.359	0.356	0.044
SABMK	0.360	0.366	0.363	0.037
TABMK	0.361	0.369	0.365	0.040
TABMK-Max	0.362	0.367	0.364	0.045
MSK	0.361	0.368	0.364	0.041
CK	0.363	0.369	**0.366**	0.043
DTK (best)	0.340	0.346	0.343	0.044
Trigram-K (best)	0.345	0.352	0.348	0.040
LexRank (tf-idf)	0.359	0.363	0.361	0.043
Lead-based	0.337	0.343	0.340	0.041
Random	0.293	0.308	0.300	0.033

5. CONCLUSIONS AND FUTURE WORK

The rapid increase in the amount of electronic documents produced every day has emphasized the need of automatic tools that arrange and present data to the end users in an understandable and neat way. Multi-document summarization proposes a salient solution to this necessity.

Multi-document summarization is a process that aims to summarize multiple documents about the same topic by producing a complete and coherent summary of these documents. Many approaches to the multi-document summarization problem pursue extractive based methods where a summary is created by extracting sentences that best explain the documents. Computing similarities between sentences is a crucial task for most of these approaches in terms of identifying important sentences and eliminating redundant information. In this study, we consider this sentence similarity measure task. We discuss two different untyped dependency tree based sentence similarity kernels previously proposed for relation extraction and present our new sentence similarity methods that use typed dependency tree representations of sentences in Chapter 2. We employ dependency grammars because bag-of-words representation is a poor descriptor of sentences and cannot capture similar substructures in sentences.

We adapt the two untyped dependency tree based sentence similarity kernels the Dependency Tree Kernel and the Dependency Trigram Kernel to the multi-document summarization task and compare their performances in a graph-based multi-document summarization system named LexRank. We evaluate these kernels with and without including type information of dependency relations. Our results show that incorporating type information into these kernels improves their performances for the multi-document summarization task in terms of ROUGE-1 score. However, they fail to outperform the original bag-of-words based sentence similarity method of LexRank. The reason behind the worse performance of the Dependency Tree Kernel is that it overrates matching nodes and cannot capture syntactic relations in sentences as a whole. On the other hand, the Dependency Trigram Kernel fails because its strict trigram structure makes

it hard to match for similar substructures in sentences. Even for the sentences that have similar meanings, it is hard to find triple node groups that are common in both of the sentences. Bigram unit approach is more flexible and more suitable for multi-document summarization. We explain the new sentence similarity methods that utilize this bigram unit approach in Chapter 2. The experiment results presented in Chapter 4 reveal that the approximate bigram match kernels achieve better performances than both untyped dependency tree kernels and LexRank's original similarity method.

In Chapter 4, we compare different bigram unit types for the Strict Bigram Match Kernel. The best ROUGE-1 score is reached using *DH* as the bigram unit type. According to the results, usage of *DA* shows worse performance than the other bigram unit types. This low performance of *DA* is caused by its method of creating bigrams. Relating a node with its distant ancestors results in misleading bigram units where two nodes are considered as related although there is no semantic relation between them. Setting a limit to the degree of ancestors a node can be related to, can increase the performance of *DA*. Our results show that using the bigram type *DT-TH* also shows poor performance since it cannot capture similar parts between sentences. Usage of *DTH* better detects similarities between sentences but the more relaxed structure of *DH* outperforms *DTH* in capturing meaningful relations between nodes.

The second proposed kernel the Simple Approximate Bigram Match Kernel shows better performance than the Strict Bigram Match Kernel when we look at the overall results. the Simple Approximate Bigram Match Kernel is an approximate bigram match kernel method and it allows approximate match of bigram units in contrast to the Strict Bigram Match Kernel which entails matching of whole bigram units. Results show that approximate bigram match techniques give better results in terms of ROUGE-1 score. The tf-idf based approximate bigram match kernels perform better than the Simple Approximate Bigram Match Kernel. Tf-idf is a good measure that indicates importance of a word. Using tf-idf measure in the bigram unit similarity comptutation step increases the performance. TF-IDF Based Approximate Bigram Match Kernel, TF-IDF Max Approximate Bigram Match Kernel, and the Matching Subtrees Kernel show almost the same success rates. In contrast to our expectations,

the Matching Subtrees Kernel does not outperform TF-IDF Based Approximate Bigram Match Kernel and TF-IDF Max Approximate Bigram Match Kernel. One reason for this situation might be that the structure of the Matching Subtrees Kernel is more complex and hence its parameters should be fine-tuned in order to improve its performance.

Filtering the unimportant dependency relation types increases the performance of the typed dependency tree kernels. The reason for this increase is that dependency relations with unimportant types introduce noise to the system. Another method that increases the performance of the typed dependency tree kernels is giving an extra score to matching bigram units whose dependent or head words start with an uppercase letter. This approach improves the performance because the words which start with an uppercase letter are usually entity names and so are more significant. Rewarding matching bigrams with the same dependency relation type also improves the results for bigram match kernels.

We conclude that using dependency grammar representation of sentences leads to better results in measuring similarities between sentences for the task of multi-document summarization. The type of dependency relations is crucial in identifying the important parts in sentences and utilizing the dependency tree structure of sentences helps us to find similar substructures between sentences.

As future work, we will apply our typed dependency tree based sentence similarity kernels to other natural language processing areas such as text mining or question-answering. We want to measure the performance of our sentence similarity kernels in other systems that need a sentence similarity computation step. We are also planning to work on the Dependency Trigram Kernel by applying the approximate bigram match technique to its trigram structure. Relaxing the trigram similarity function of this kernel by replacing its multiplication structure with a less strict model can improve its performance for the multi-document summarization problem. Performing multi-document summarization on Turkish documents using these dependency tree based sentence similarity kernels together with the LexRank system is another future

work that would be interesting. Turkish has a free word order structure and so this language is more suitable to dependency grammar representation analysis. Applying the methods presented in this study to Turkish can give better summarization performance.

APPENDIX A: SAMPLE SYSTEM GENERATED AND HUMAN GENERATED SUMMARIES

Ariel Sharon, the hawkish former general tipped to be Israel's next foreign minister, said in an interview published Sunday that if he gets the job, he won't shake the hand of Palestinian leader Yasser Arafat.

The New York Times said in an editorial on Saturday, Oct. 10: Just days before heading to the United States for critical negotiations with Palestinian leaders, Prime Minister Benjamin Netanyahu jolted the Middle East peace effort with the appointment of Ariel Sharon as Israeli foreign minister.

Ariel Sharon's appointment as the Israeli foreign minister serves as ``the bullet of mercy'' for the Middle East peace process, an official Syrian newspaper said

Figure A.1. System generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set

Prior to his appointment as foreign minister by Israeli Prime Minister Natanyahu in 1998, seasoned warrior Ariel Sharon announced that if appointed, he would not shake hands with Palestinian leader Yasser Arafat.

The remark fit Sharon's long history of unyielding bellicosity towards Palestine.

When he was appointed some thought that Netanyahu was placating Israel's far right wing while striving for peace.

Others saw it as a move to scuttle the peace effort.

Arab commentary deemed the appointment "tantamount to disaster" and "the bullet of mercy" to the peace process.

Peace talks in the U.S. were in the offing with the outcome in doubt.

Figure A.2. First human generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set

In a move widely viewed as an effort to placate the far right as he moves to withdraw from more West Bank land, Israeli Prime Minister Netanyahu named hardliner Ariel Sharon foreign minister and chief peace negotiator.

Sharon, a military leader with legendary victories in the 1967 and 1973 Mideast wars, is infamous in the Arab world as the defense minister in the 1982 invasion of Lebanon during which Lebanese Christian militiamen, Israeli allies, slaughtered hundreds of unarmed Palestinians.

His appointment as lead negotiator was denounced as a "disaster" in the Lebanese press and "a bullet of mercy to the peace process" in a Syrian paper.

Figure A.3. Second human generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set

Ariel Sharon was named Israeli foreign minister just weeks before a scheduled peace summit in the US, aimed at negotiating an Israeli withdrawal from 13% of the West Bank.

Arabs reacted strongly and said the hard-line, right-wing, warrior and former defense minister would kill any progress for a settlement in the mid-East.

Many blame Sharon for a 1982 massacre of Palestinian civilians at refugee camps in Lebanon.

Others believed his appointment was an effort to placate the far right and gain their support for the Netanyahu government and the concessions necessary for the peace process.

Sharon vowed not to shake Yasser Arafat's hand.

Figure A.4. Third human generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set

Appointment of Ariel Sharon as Foreign Minister by PM Netanyahu does not bode well for the peace process.

His appointment is surprising in that the two have differed on withdrawal from Palestinian territory and is seen as an effort to placate the right.

Sharon, 70, is an unapologetic warrior on the battlefield and in partisan politics.

He brings heavy baggage: right-wing intransigence and responsibility for failing to prevent the killings of 460 unarmed Palestinians in refugee camps during the Lebanon incursion.

Sharon has said that he will not shake Arafat's hand and his appointment is seen as a "bullet of mercy" for the Middle East peace process.

Figure A.5. Fourth human generated summary of the 23*rd* cluster of DUC 2004 Task 2 data set

REFERENCES

1. Li, S., Y. Ouyang, W. Wang and B. Sun, "Multi-document Summarization Using Support Vector Regression", *Proceedings of the 7th Document Understanding Conference*, DUC, Rochester, New York, USA, 2007.

2. Das, D. and A. F. Martins, "A Survey on Automatic Text Summarization", *Literature Survey for the Language and Statistics II course at CMU*, Vol. 4, pp. 192–195, 2007.

3. Pei, Y., W. Yin, Q. Fan and L. Huang, "A Supervised Aggregation Framework for Multi-Document Summarization", *Proceedings of the 24th International Conference on Computational Linguistics*, pp. 2225–2242, COLING, Indian Institute of Technology Bombay, Bombay, India, 2012.

4. Nastase, V., "Topic-driven Multi-document Summarization with Encyclopedic Knowledge and Spreading Activation", *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 763–772, ACL, 2008.

5. Hennig, L. and D. Labor, "Topic-based Multi-document Summarization with Probabilistic Latent Semantic Analysis", *Proceedings of the International Conference on Recent Advances in NLP (RANLP)*, pp. 144–149, 2009.

6. Wang, D., S. Zhu, T. Li and Y. Gong, "Multi-document Summarization Using Sentence-based Topic Models", *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pp. 297–300, ACL, Suntec, Singapore, 2009.

7. Radev, D. R., H. Jing, M. Styś and D. Tam, "Centroid-based Summarization of Multiple Documents", *Information Processing & Management*, Vol. 40, No. 6, pp. 919–938, 2004.

8. Aliguliyev, R. M., "Clustering Techniques and Discrete Particle Swarm Optimiza-

tion Algorithm for Multi-document Summarization", *Computational Intelligence*, Vol. 26, No. 4, pp. 420–448, 2010.

9. Erkan, G. and D. R. Radev, "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization", *Journal of Artificial Intelligence Research(JAIR)*, Vol. 22, No. 1, pp. 457–479, 2004.

10. Wan, X. and J. Yang, "Multi-document Summarization Using Cluster-based Link Analysis", *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 299–306, ACL, Singapore, 2008.

11. Shen, C. and T. Li, "Multi-document Summarization via the Minimum Dominating Set", *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 984–992, ACL, Beijing, China, 2010.

12. Mihalcea, R., "Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization", *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, p. 20, ACL, Barcelona, Spain, 2004.

13. Barzilay, R. and K. R. McKeown, "Sentence Fusion for Multidocument News Summarization", *Computational Linguistics*, Vol. 31, No. 3, pp. 297–328, 2005.

14. Filippova, K. and M. Strube, "Sentence Fusion via Dependency Graph Compression", *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 177–185, ACL, Waikiki, Honolulu, Hawaii, 2008.

15. Zhuang, L., F. Jing and X.-Y. Zhu, "Movie Review Mining and Summarization", *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pp. 43–50, ACM, Arlington, VA, USA, 2006.

16. Somprasertsri, G. and P. Lalitrojwong, "Mining Feature-Opinion in Online Customer Reviews for Opinion Summarization", *The Journal of Universal Computer*

Science, Vol. 16, No. 6, pp. 938–955, 2010.

17. Fiszman, M., T. C. Rindflesch and H. Kilicoglu, "Abstraction Summarization for Managing the Biomedical Research Literature", *Proceedings of the HLT-NAACL Workshop on Computational Lexical Semantics*, pp. 76–83, ACL, Boston, MA, USA, 2004.

18. Yousfi-Monod, M., V. V. Prince *et al.*, "Sentence Compression as a Step in Summarization or an Alternative Path in Text Shortening", *Proceedings of the 22nd International Conference on Computational Linguistics*, pp. 137–140, ACL, Manchester, UK, 2008.

19. Galanis, D., *Automatic Generation of Natural Language Summaries*, Ph.D. thesis, Athens University Of Economics and Business, 2012.

20. Sakhare, M. D., "Dependency Grammar Feature Based Noun Phrase Extraction for Text Summarization", *International Journal of Computer Trends and Technology*, Vol. 1, No. 2, pp. 309–313, 2011.

21. Liu, W. and J. Zhou, "Building a Chinese Text Summarizer with Phrasal Chunks and Domain Knowledge", *Proceedings of Research on Computational Linguistics Conference XIII*, pp. 87–96, Taipei, Taiwan, 2000.

22. Blake, C., J. Kampov, A. K. Orphanides, D. West and C. Lown, "UNC-CH at DUC 2007: Query Expansion, Lexical Simplification and Sentence Selection Strategies for Multi-Document Summarization", *Proceedings of the 7th Document Understanding Conference*, DUC, Rochester, New York, USA, 2007.

23. Kumar, Y. J. and N. Salim, "Automatic Multi Document Summarization Approaches", *Journal of Computer Science*, Vol. 8, No. 1, p. 133, 2011.

24. Culotta, A. and J. Sorensen, "Dependency Tree Kernels for Relation Extraction", *Proceedings of the 42nd Annual Meeting on Association for Computational Lin-*

guistics, p. 423, ACL, Barcelona, Spain, 2004.

25. Choi, M. and H. Kim, "Social Relation Extraction from Texts Using a Support-vector-machine-based Dependency Trigram Kernel", *Information Processing & Management*, Vol. 49, No. 1, pp. 303–311, 2013.

26. Zelenko, D., C. Aone and A. Richardella, "Kernel Methods for Relation Extraction", *The Journal of Machine Learning Research*, Vol. 3, pp. 1083–1106, 2003.

27. Lin, C.-Y., "Rouge: A Package for Automatic Evaluation of Summaries", *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pp. 74–81, Barcelona, Spain, 2004.

28. Lin, C.-Y. and E. Hovy, "Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics", *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Vol. 1, pp. 71–78, ACL, Edmonton, Canada, 2003.

29. Klein, D. and C. D. Manning, "Accurate Unlexicalized Parsing", *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Vol. 1, pp. 423–430, ACL, 2003.

30. Miller, G. A., "WordNet: A Lexical Database for English", *Communications of the ACM*, Vol. 38, No. 11, pp. 39–41, 1995.

31. Porter, M. F., "An Algorithm for Suffix Stripping", *Program: Electronic Library and Information Systems*, Vol. 14, No. 3, pp. 130–137, 1980.

32. Zhou, X., X. Zhang and X. Hu, "Dragon Toolkit: Incorporating Auto-learned Semantic Knowledge into Large-Scale Text Retrieval and Mining", *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, Vol. 2, pp. 197–201, IEEE, Patras, Greece, 2007.

33. Radev, D., T. Allison, S. Blair-Goldensohn, J. Blitzer, A. Celebi, S. Dimitrov,

E. Drabek, A. Hakim, W. Lam, D. Liu *et al.*, "MEAD - A Platform for Multidocument Multilingual Text Summarization", *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC)*, Lisbon, Portugal, 2004.

34. Page, L., S. Brin, R. Motwani and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web", *Proceedings of the 7th International World Wide Web Conference*, pp. 161–172, Brisbane, Australia, 1998.