USING GENETIC ALGORITHMS WITH LEXICAL CHAINS FOR AUTOMATIC TEXT SUMMARIZATION

by

Mine Berker

B.S., Computer Engineering, Boğaziçi University, 2003

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
2011

USING GENETIC ALGORITHMS WITH LEXICAL CHAINS FOR AUTOMATIC TEXT SUMMARIZATION

APPROVED BY:

Assoc. Prof. Tunga Güngör (Thesis Supervisor)	.....
Prof. Fikret Gürgen	.....
Prof. Sumru A. Özsoy	.....

DATE OF APPROVAL: .....

ABSTRACT

USING GENETIC ALGORITHMS WITH LEXICAL CHAINS FOR AUTOMATIC TEXT SUMMARIZATION

With the rapid increase in the amount of online text information, it became more important to have tools that would help users distinguish the important content. Automatic text summarization attempts to address this problem by taking an input text and extracting the most important content of it. However, the determination of the salience of information in the text depends on different factors and remains as a key problem of automatic text summarization.

In the literature, there are some studies that use lexical chains as an indicator of lexical cohesion in the text and as an intermediate representation for text summarization. Also, some studies make use of genetic algorithms in order to examine some manually generated summaries and learn the patterns in the text which lead to the summaries by identifying relevant features which are most correlated with human generated summaries.

In this study, we combine these two approaches of summarization. Firstly, lexical chains are computed to exploit the lexical cohesion that exists in the text. Then, this deep level of knowledge about the text is combined with other higher level analysis results. Finally, all these results that give different levels of knowledge about the text are combined using genetic algorithms to obtain a general understanding.

ÖZET

OTOMATİK METİN ÖZETLEME İÇİN GENETİK ALGORİTMALARIN SÖZCÜK ZİNCİRLERİ İLE KULLANIMI

Elektronik ortamda bulunan bilginin miktarının hızla artmasıyla, kullanıcılara bu bilginin içindeki önemli içeriği ayırt etmekte yardımcı olacak araçlar önem kazandı. Otomatik metin özetleme, bir girdi metni alıp içindeki en önemli içeriği seçip çıkartarak bu probleme hitap etmeyi amaçlamaktadır. Ancak metindeki belli başlı bilginin belirlenmesi değişik unsurlara dayanmakta ve otomatik metin özetlemenin önemli sorunlarından birini oluşturmaya devam etmektedir.

Literatürde bazı çalışmalar metindeki sözcüksel bağlılığın göstergesi ve metin özetlemenin ara gösterimi olarak sözcük zincirlerini kullanmıştır. Ayrıca, elle yaratılmış özetlerle en çok ilintili metin özelliklerini ayırt ederek, özetlere götüren kalıpları öğrenmek için genetik algoritmalardan faydalanan çalışmalar da bulunmaktadır.

Bu çalışmada, özetlemenin bu iki yaklaşımını birleştiriyoruz. Öncelikle, metinde bulunan sözcüksel bağlılıktan yararlanmak için sözcük zincileri hesaplanıyor. Ardından, metinle ilgili bu derin seviyedeki bilgi, daha üst seviye analiz sonuçları ile birleştiriliyor. Sonunda, metinle ilgili değişik seviyelerde bilgi veren bütün bu sonuçlar, genetik algoritmalar kullanılarak birleştiriliyor.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

ATS	Automatic Text Summarization
CAST	Computer-Aided Summarization Tool
NE	NamedEntity
synset	synonym set
WCL	Word Control List
GAs	Genetic Algorithms
TF	Term Frequency
TF-IDF	Term Frequency - Inverse Document Frequency
PoS	Part of Speech

1. INTRODUCTION

1.1. Automatic Text Summarization

The document of ISO 215 standards in 1986 formally defines a summary as a "brief restatement within the document (usually at the end) of its salient findings and conclusions" that "is intended to complete the orientation of a reader who has studied the preceding text" [1]. It contains the most important information about the document. Automatic text summarization (ATS) is the process where a computer automatically produces such a summary.

The goal of automatic summarization is to take an information source, extract content from it, and present the most important content to the user in a condensed form and in a manner sensitive to the user's or application's needs [2].

ATS has been an active research area for many years. Although it is very difficult to implement the automation of human skills needed in summarization, the exponential growth in the quantity and complexity of information sources on the internet has exposed the need to quickly process and digest this huge amount of information. This situation has attracted researchers' attention. Even if they can not perfectly imitate the human summarizers, automatic text summarizers can be used, for instance, to select the most relevant information from the many results that a search engine returns for a search.

1.2. Summarization Types

The most fundamental distinction that can be made between summarization types is the one between ***extracts*** and ***abstracts***. An extract is a summary consisting entirely of material copied from the input. On the other hand, an abstract is a summary at least some of whose material is not present in the input [2].

Extracts are generally produced by shallow approaches, where the sentences of the text are analyzed to a syntactic level. These approaches extract salient parts of the source text and present them. Here, the analysis phase of summarization is important. The classic work of Edmundson (1969) set a framework for work on extraction [3]. He considered different text features like cue words, title words and sentence location, and gave scores to sentences based on these features. He adjusted the feature weights by hand, however subsequent work used machine learning approaches using a training corpus to adjust feature weights. For example, [4] used genetic algorithms and mathematical regression models to train their summarizer and learn the feature weights, while [5] used genetic programming.

On the other hand, abstracts are produced by deeper approaches. These approaches analyze the source text to a sentential semantics level. In order to retrieve important information from the text, approaches like template filling [6], term rewriting [7] and concept hierarchy [8] are used. After the analysis phase, these approaches go through a synthesis phase, which usually involves natural language generation.

Most of the studies in this area are based on extraction. While abstraction deals heavily with natural language processing, extraction can be viewed as selecting the most important parts of the original document and concatenating them to form the summary.

1.3. Evaluation

Evaluation is an essential part of ATS. Summarization is currently a practical discipline where no deep theory exists. There are, however, theoretical frameworks that are being investigated. For this reason, evaluating the outputs of these frameworks is very important.

Intrinsic evaluation methods test the ATS system within itself. Typically, an *ideal* summary is created (generally by human summarizers) and then the output of the automatic summarizer is compared with this ideal summary. The quality of the

summaries is mostly measured by precision and recall [9].

Extrinsic methods try to determine the effect of summarization on some other tasks. These tasks may include, for instance, relevance assessment. That is, a subject is asked to determine the relevance of a given topic to the given text. Or, another task may be executing the instructions, if the summary is of a detailed technical manual. It is possible to measure the efficiency in executing the instructions by simply following the summary, when compared to following the original manual. The variety of tasks is in fact very large.

There are also new summarization areas that need large-scale formal evaluation, like summaries as answers to questions, narrative summarization, multi-lingual summarization, etc.

1.4. Motivation

In extract generation, scoring sentences based on text features and using machine learning methods to learn the feature weights has been studied before. Moreover, some studies used lexical chains as an intermediate representation in the text summarization process, in order to detect the lexical cohesion between sentences and select the most semantically related ones [10] [11] [12]. In this thesis, we combined these two approaches to generate extract based summaries. In addition to shallow, syntactic text features, we used lexical chains as a feature to score sentences. These chains are expected to identify the cohesion that exists throughout the text, and assign higher scores to the sentences that are semantically related to each other.

1.5. Thesis Organization

The remainder of this thesis is organized as follows:

In section 2, some background information is introduced. Together with the logic behind using lexical chains for automatic text summarization, genetic algorithms and

their use in summarization are explained. We describe the details of our study in sections 3, 4 and 5. In section 6 we present the results of our evaluations. Then we conclude our thesis with section 7.

2. BACKGROUND

2.1. Lexical Chains

Cohesion can be defined as the way certain words or grammatical features of a sentence can connect it to its predecessors and successors in a text. Cohesion occurs where the interpretation of some element in the discourse is dependent on the interpretation of another element [13]. As described in Haliday and Hasan [14], cohesion is a device for "sticking together" different parts of the text. It is achieved through the use of semantically related terms, reference, ellipsis and conjunctions in the text. Among these types, lexical cohesion, which is created by using semantically related words, is the most frequent one [13].

Lexical cohesion can be classified into reiteration category and collocation category. Reiteration occurs when one lexical item reminds the meaning of an earlier item in the text. It can be created by using repetition, synonyms and hyponyms. Collocation refers to words that tend to co-occur in the text.

Lexical cohesion can occur not only between two terms but also among sequences of related words. These sequences of words are called *lexical chains*. Lexical chains can be distributed over sentences and different text parts. Words may be grouped in the same lexical chain when: [11]

- Two noun instances are identical and are used in the same sense.
  (The house on the wood is large. The house is made of wood.)
- Two noun instances are used in the same sense (i.e., synonyms).
  (The car is fast. My automobile is faster.)
- The senses of two noun instances have a hypernym/hyponym relation between them.
  (John owns a car. It is a Toyota.)
- The senses of two noun instances are siblings in the hypernym/hyponym tree.

(The truck is fast. The car is faster.)

Barzilay and Elhadad used lexical chains first [10], as an intermediate step in the text summarization process to extract important concepts from a document. They showed that cohesion is one of the surface signs of discourse structure and lexical chains can be used to identify it. They relied on WordNet [15] to provide sense possibilities for word instances as well as semantic relations among them. Senses in the WordNet database are represented relationally by synonym sets (synsets) which are the sets of all the words sharing a common sense. Words of the same category are linked through semantic relations like synonymy and hyponymy.

Lexical chains were constructed in three steps:

(i) Select a set of candidate words
(ii) For each candidate word, find an appropriate chain according to a relatedness criterion
(iii) If such a chain is found, insert the word into the chain and update the chain

Once the chains were constructed, they showed that picking the concepts represented by strong lexical chains gives a better understanding of the central topic of a text than picking only the most frequent words in the text. Finally they used these strong chains to extract sentences from the original text to construct a summary.

After Barzilay and Elhadad, many researchers followed this approach to use lexical chains in text summarization. Silber-McCoy proposed a new algorithm to compute lexical chains that was based on Barzilay-Elhadad method but was linear in space and time [11]. Since the method proposed in [10] had exponential complexity, it was hard to compute lexical chains for large documents. For this purpose, Silber and McCoy recompiled the WordNet noun database into a binary format and memory-mapped it. Then, they created *metachains* that represent every possible representation of the text. These metachains were used to disambiguate word senses and to create the lexical chains. Since WordNet was recompiled into a new format, it could be accessed as

a large array, and this allowed the algorithm to compute the lexical chains in linear time. After the chains were computed, the strong chains were selected, and summary sentences were extracted like [10] did.

Brunn-Chali-Pinchak also used the lexical cohesion approach as the driving engine of their summarization system [12]. In order to identify the most important portions of the text which are topically most salient, they used the degree of connectiveness among the chosen text portions. The architecture of their model consisted of a preprocessor (a text segmenter + a part of speech tagger + a parser), a noun filtering module that removed nouns that did not contribute to the subject, a lexical chainer and a sentence extractor. The lexical chainer first selected candidate words and then represented each word sense by distinct sets considered as levels. Then chains were constructed according to the semantic relationships between word senses. Finally, longest chains were determined according to the following preference criterion:

word repetition $\gg$ synonym/antonym $\gg$ ISA-1/INCLUDE-1 $\gg$ ISA-2/INCLUDE-2

Moreover, Li *et al.* proposed a model for a query-focused multi-document summarizer based on lexical chains [16]. A set of lexical chains were built for each document by creating a chain for each sense of a candidate word. Chains were created until half of the candidate words had been processed. The resulting chains were then merged. In the second step for multi-chain building, all strongest chains from each document were merged into a new chain set as the final result of lexical chain building. Two chains were merged if they had at least a common word with the same sense. Finally, candidate sentences were scored based on their inclusion of strong chain words, and sentences with the highest scores were extracted.

Furthermore, Fuentes and Rodriguez proposed a system that combined lexical chains, co-reference chains and NamedEntity (NE) chains [17]. Lexical chains was the primary source for ranking text segments, the others were complementary. The extractive unit in their work was paragraph.

2.2. Summarization With Genetic Algorithms

One of the biggest problems in text summarization is to find a salience function which determines what information in the source is important and should be included in the summary. Usually, sentence scoring methods are used for this purpose. These methods score sentences in the source text based on some text features.

These features may include for example location. The sentences that lead the text are assumed to give the most important information about the text and they are included in the summary. Edmundson (1969) combined positional importance and word frequency with the presence of cue words (presence of words like significant, or hardly) and the skeleton of the document (whether the sentence is a title or heading) [3]. This was the earliest work in extraction. Later on, Paice and Jones (1993) used stylistic clues and constructs that can be gained during a scan of a source text [18]. They claimed that as highly structured technical papers' content can be organized using a semantic frame, candidate fillers for the slots of this frame could be extracted using these clues.

Another feature may be similarity between sentences. Skorokhodko (1972) proposed an adaptive method that used relationships between sentences depending on the semantic relatedness of their words in order to set up a graphical representation of the text [19]. Pollock and Zamora (1975) used a word control list (WCL) in order to determine the presence or absence of certain syntactic features in a sentence [20]. It is based on the idea that the information content of a word is less if the word occurs rather commonly. Kupiec, Pedersen, and Chen (1995) checked the presence of proper names [21].

Miike *et al.* (1994) claimed that the nuclei of a rhetorical structure tree could provide a summary of the text for which that tree was built [22]. Marcu (1997) confirmed this hypothesis and showed that one can build extractive summaries of short texts at high levels of performance by using a scoring schema that assigned higher importance to the discourse units found closer to the root of a rhetorical structure tree than to the

units found at lower levels in the tree [23].

Statistical measures of term prominence that is derived from word frequency and distribution were also used. The first steps in this area were made by Luhn (1958) [24] and Brandow *et al.* (1995) followed that work [25].

Moreover, measures of prominence of certain semantic concepts and relationships were used as a text feature. Maybury (1995) described a system on summarizing events [26], whereas Fum, Guida, and Tasso (1985) used a set of rules to assign importance values to the different parts of a text [27].

Generally, a number of features drawn from different levels of analysis may contribute to the salience of a sentence. For this reason, a summarization system must have an automatic way of finding out how to combine different text features for a given summarization task.

One of the approaches to this problem is to use machine learning on a training corpus of documents and their extraction based summaries. There have been studies that applied statistical learning approaches [21] [28]. However, they require a very large amount of training sets to learn accurately. An alternative way is to use evolutionary methods as the learning mechanism. In many researches, Genetic Algorithms (GAs) are employed to learn the importance of different text features for a given summarization task.

For example, [29] used machine learning on a training corpus of documents and their abstracts to discover salience functions which describe what combination of features is optimal for a given summarization task. Here, the aim was to construct rules or functions which label any new sentence as a summary sentence or not. Later on, Alfonseca and Rodriguez used genetic algorithms to select the best summary among summaries that are composed of randomly selected sentences [30]. They were scoring the summaries. Moreover, [1] [5] and [31] scored sentences based on some text features and used genetic algorithms to learn the appropriate set of feature weights to score the

source sentences. In these works, the summarization task is seen as a two-class classification problem, where a sentence is labeled "summary" if it belongs to the extractive reference summary or as "non-summary" otherwise. The trainable system is expected to learn the patterns which lead to the summaries by identifying relevant features which are most correlated with the classes "summary" and "non-summary". When a new document is given to the system for summarization, the "learned" patterns are used to classify that sentence into one of the two classes. This classification will be used to decide which sentences will be included in the summary.

3. PROPOSED APPROACH

3.1. Genetic Algorithms with Lexical Chains

The aim of this study is to combine these two approaches of summarization. Firstly, lexical chains are computed to exploit the lexical cohesion that exists in the text. Then, this deep level of knowledge about the text is combined with other higher level analysis results such as location analysis and thematic analysis. Finally, all these results that give different levels of knowledge about the text are combined to obtain a general understanding.

In this thesis, we use a sentence extraction procedure that makes use of these properties of the text to weight the sentences. Each sentence in a text is given a sentence score that is calculated using the different text feature scores. After that, the sentences are sorted in descending order of their score values. And then appropriate number of highest score sentences are selected from the text to form the summary, according to the summarization ratio.

While weighting the sentences, not all the properties of the text will have the same importance. However, weighting the text feature scores with predetermined constant weights does not seem to be powerful enough for a good summarization. For this reason, the system first goes through a training phase, where the weights of each text feature are learned using machine learning methods.

In order to be able to learn the weights of different text features, a set of manually summarized documents is used. These human generated extracts are expected to give an idea about the patterns which lead to the summaries. In this study, we use the manually summarized documents from the CAST (Computer-Aided Summarization Tool) corpus [32]. The documents in the corpus are taken from the Reuters corpus and consist of 20 sentences in the average.

Each document in the corpus is represented by a document object in the system. A document object consists of a title, and a list of sentences. Every sentence in a document has an importance flag. This flag is set to "summary" if the sentence is selected as a summary sentence by human summarizers; it is set to "non-summary" otherwise.

After the feature score weights are learned through the training phase, the system will go through a testing phase where new documents are introduced to the system for summarization. In this phase, sentence scores will be calculated for each sentence in a document using the text feature scores for that sentence and their respective score weights. Then the sentences will be sorted in a descending order of their score values, and the highest score sentences will be selected to form the extractive summary.

3.2. Text Features

In this system, the sentences are modeled as vectors of features extracted from the text. The system uses 12 text features to score sentences. For each sentence of a document, a sentence score will be calculated using the feature scores of these text features for that sentence. Each feature score can have a value between 0 and 1.

The text features used in this system are grouped into three classes, according to their level of text analysis. Table 3.1 shows the features and their corresponding classes.

There is a sample source text to be summarized shown in Figure 3.1. The text consists of a title and 11 sentences. Assume that the sentences of this text are being scored using the 12 text features of this study. The last sentence of the sample text will be used as an example in order to show the calculation of the feature scores.

Table 3.1. Text features.

Location Features	Sentence Location
	Sentence Relative Length
Thematic Features	Average TF
	Average TF-IDF
	Sentence Resemblance to Title
	Sentence Centrality
	Sentence Inclusion of Emphasize Words
	Sentence Inclusion of Name Entities
	Sentence Inclusion of Numerical Data
Cohesion Features	Number of Synonym Links
	Number of Co-occurrence Links
	Lexical Chain Score

Rite Aid Q4 Net Lower, but Oper Net Up.

(1) For the quarter, income from operations before nonrecurring charges and extraordinary loss was $87,640,000 or 75 cents per share compared to $56,994,000 or $.68 per share last year.
(2) Pre-tax nonrecurring charges were $52,000,000 or 28 cents per share on an after-tax basis.
(3) Income before nonrecurring charges and extraordinary loss was $202,897,000 or $2.20 per share, compared with $158,947,000 or $1.90 per share last year.
(4) Net income for the year ended March 1, 1997, including nonrecurring charges for the fourth quarter, and $9,923,000 or 12 cents per share in the first quarter for costs associated with the attempted acquisition of Revco D.S. Inc., was $115,377,000 or $1.25 per share.
(5) During the fourth quarter, the company completed a tender offer for 12_1/4% notes due 2004 of its subsidiary Thrifty PayLess Inc.
(6) The early extinguishment of these notes and other indebtedness resulted in an extraordinary loss of $45,157,000, net of taxes or 49 cents per share.
(7) The LIFO method of valuing inventory had the effect of reducing net income 1 cent per share for the quarter and 11 cents per share for the year ended March 1, 1997.
(8) For the comparable periods last year, the LIFO adjustments were 4 cents per share for the quarter and 13 cents per share for the year.
(9) Common stock issued in connection with the acquisition of Thrifty PayLess Holdings Inc., is included in the calculation of weighted average shares since the date of acquisition.
(10) Weighted average shares were 33,151,000 for the quarter and 8,288,000 for the year.
(11) ***As a result of the weighting differences for the quarterly and annual periods, aggregation of quarterly earnings per share data do not equal annual earnings per share amounts.***

Figure 3.1. Sample source text to be summarized.

3.3. Location Features

These features exploit the structure of the text at a shallow level of analysis. Depending on the location and length of the sentence, the importance of its content is tried to be predicted. Based on this prediction, a sentence will be given a higher or a lower score.

3.3.1. Sentence Location

This feature scores the sentences according to their position in the text. In this work, we assume that the first sentences of the text are the most important ones. So, the first sentence of a document gets a score value of 1, the second sentence gets 0.8, the fifth sentence gets 0.2 and the rest of the sentences get 0. The example sentence will get a score value of 0 since it is the 11^{th} sentence of the text.

3.3.2. Sentence Relative Length

This feature uses the sentence length to score a sentence, assuming that longer sentences contain more information and have a higher possibility to be in the summary. Thus, shorter sentences are penalized. The feature score is calculated as follows for the sentence s in the document d:

$$SRL(s,d) = \frac{length(s)}{maxSentenceLength(d)} \tag{3.1}$$

The length of the example sentence is 28, and the length of the longest sentence in the document, which is sentence number 4, is 44. So, the example sentence will get a score value of 0.63.

3.4. Thematic Features

These features study the text more deeply to analyze the term based properties. The term frequencies of each document and each sentence are calculated, after removing the stop words. The list of the stop words used in this work can be found in Appendix A.

3.4.1. Average TF

This feature calculates the Term Frequency (TF) score for each term in a sentence and takes their average. The TF metric makes two assumptions:

(i) Multiple appearances of a term in a document are more important than single appearances.

(ii) Length of the document should not affect the importance of the terms.

The TF score for a term t in the document d is calculated as follows:

$$TF(t,d) = \frac{frequencyOfTermInDocument(t,d)}{maxTermFrequency(d)} \tag{3.2}$$

So, the feature score for a sentence s is the average of the TF scores of all the terms in s. The example sentence has an average TF score of 0.23.

3.4.2. Average TF-IDF

This feature calculates the Term Frequency - Inverse Document Frequency (TF-IDF) score for each term in a sentence and takes their average. The TF-IDF metric makes one more assumption in addition to the two assumptions of the TF metric:

(iii) Rare terms are more important than frequent terms.

This metric takes into account not only the frequency of a term within a document but also the frequency of a term throughout all the documents in the corpus. This way, if a term exists in too many documents, its importance in a single document is decreased proportionally.

The TF-IDF score for a term t in the document d given a corpus c is calculated as follows:

$$TF - IDF(t, d, c) = TF(t, d) * log(\frac{n}{df(t)}) \tag{3.3}$$

where n is the total number of documents in corpus c, and df(t) is the number of documents in corpus c in which term t occurs.

So, the feature score for a sentence s is the average of the TF-IDF scores of all the terms in s.

3.4.3. Sentence Resemblance to Title

This feature considers the vocabulary overlap between a sentence and the document title. If a sentence has many words in common with the document title, it is assumed to be related to the main topic of the document. So, it is assumed to have more chance to be in the summary.

The feature score is calculated as follows for a sentence s:

$$SRT(s) = \frac{|m \cap k|}{|m \cup k|} \tag{3.4}$$

where m is the number of terms that occur in sentence s, and k is the number of terms that occur in the title. Since the example sentence and the title of the sample text have no common words, the example sentence gets a feature score of 0.

3.4.4. Sentence Centrality

This feature considers the vocabulary overlap between a sentence and the other sentences in the document. If a sentence has many words in common with the rest of the document, it is assumed to be about an important topic in the document. So, it is assumed to have more chance to be in the summary.

The feature score is calculated as follows for a sentence s in the document d:

$$SC(s, d) = \frac{m}{k} \tag{3.5}$$

where m is the number of terms that occur both in sentence s and in a sentence of document d other than s, and k is the total number of terms in document d.

In the example sentence, the words *result*, *weight*, *period*, *per*, and *share* exist respectively in the sentences 6, 9, 8, 1 and 1. So, m is 5 and k is 41. For this reason, the sentence gets a score value of 0.12.

3.4.5. Sentence Inclusion of Emphasize Words

This feature counts the number of emphasize words (such as *especially*, *certainly* etc.) in a sentence, assuming that a sentence that contains these words is an important one and will probably be included in the summary. The list of the emphasize words used in the system can be found in Appendix B.

The feature score is calculated as follows:

$$SEW(s) = \frac{numEmphasizeWordsInSentence(s)}{length(s)} \tag{3.6}$$

Since there are no emphasize words in the example sentence, its score value will be 0.

3.4.6. Sentence Inclusion of Name Entities

This feature counts the number of name entities (proper nouns) in a sentence, assuming that a sentence that contains name entities is an important one and will probably be included in the summary. In this work, name entities are recognized using the University of Illinois Named Entity Tagger [33].

The feature score is calculated as follows:

$$SNE(s) = \frac{numNameEntityInSentence(s)}{length(s)} \tag{3.7}$$

Since there are no name entities in the example sentence, its score value will be

0.

3.4.7. Sentence Inclusion of Numerical Data

This feature counts the number of numerical terms in a sentence, assuming that a sentence that contains numerical data is an important one and will probably be included in the summary. The feature score is calculated as follows:

$$SND(s) = \frac{numNumericalDataInSentence(s)}{length(s)} \tag{3.8}$$

Since there are no numerical data in the example sentence, its score value will be 0.

3.5. Cohesion Features

Cohesion can be defined as the way certain words or grammatical features of a sentence can connect it to its predecessors and successors in a text [13]. Cohesion is brought about by linguistic devices such as repetition, synonymy, anaphora and ellipsis [29]. In this system, three cohesion based features are used.

3.5.1. Number of Synonym Links

In order to compute this feature, first the nouns in a sentence are extracted by the LingPipe part-of-speech tagger [34]. Then nouns in the given sentence s are compared to the nouns in other sentences in the document d. This comparison is made by taking two nouns from the two sentences and looking whether they have a synset in common in WordNet. For instance, if a noun from sentence s has a synset in common with a noun from another sentence t, this means there is a synonym link between the sentences s and t.

So, the feature score is calculated as follows for a sentence s in the document d:

$$NSL(s) = \frac{n}{k} \tag{3.9}$$

where n is the number of synonym links of sentence s (i.e., the number of sentences t) and k is the total number of sentences in document d.

The word *share* in the example sentence appears in every sentence in the text, except the sentence 5. Moreover, the word quarter appears both in the sentence 5 and in the example sentence. So, the example sentence shares a term with every other sentence in the text. Sharing the exact same word is not necessary for two sentences to have a synonym link, but of course, it counts. For this reason, the example sentence has 10 synonym links, and its feature score is 0.90.

This is a very high feature score value. However, the score values of each sentence in this text for this feature will be likewise high, since 10 of the sentences share a common word. So, this feature will not be a distinctive property for sentences in this text, even though their score values will be very high.

3.5.2. Number of Co-occurrence Links

In order to compute this feature, first all the bigrams in the document are considered and their frequencies are calculated. If a bigram in a document has a frequency greater than one, then this bigram is assumed to be a collocation. This is the case for the bigram "per share" in the sample text.

Secondly, terms of the given sentence s are compared to the terms in other sentences in the document d. This comparison procedure checks if a term from sentence s forms a collocation with a term from another sentence. If it does, this means there

is a co-occurrence link between this sentence and the sentence s.

So, the feature score is calculated as follows for a sentence s in the document d:

$$NCL(s) = \frac{n}{k} \tag{3.10}$$

where n is the number of co-occurrence links of sentence s and k is the total number of sentences in document d.

The word "per" in the example sentence forms a collocation with every instance of the word "share" in each of the sentences 1, 2, 3, 4, 6, 7, 8, 9, and 10. So the number of co-occurrence links for the example sentence is 9, and the score value is 0.81. Like the synonym links feature, the score value for this feature will be very high for almost every sentence in the sample text, and this feature will not be distinctive. However, this is because the saying "per share" appears too much in this text. In another text where this much of repetition does not exist, these features may give a better understanding about the importance of the sentences.

3.5.3. Lexical Chain Score

In order to use lexical chains as a means for scoring the sentences of a document, first the chains are computed for the whole document. Then these constructed chains are scored and the strongest ones among them are selected. Finally, sentences of the document are scored according to their inclusion of strong chain words.

The details of the lexical chain computing and scoring processes are explained in section 4. When the sample text is put through these lexical chain computing and scoring processes, the following chains are selected to be the strongest ones:

- income, earnings, net
- year, periods, quarter
- acquisition, loss
- amounts, shares

So, after the chains are constructed and scored for a document d, the lexical chain score of a sentence s is as follows:

$$LC(s) = \frac{\sum_{i} frequency(i) | i \in s \text{ and i is a word in a strong chain}}{maxLCScore(d)} \tag{3.11}$$

The example sentence contains the words *earning*, *amount* and *share* which appear in strong chains. When the frequencies of these words are added up and the maximum lexical chain score of sentences in the sample text are computed, the lexical chain feature score value for the example sentence turns out to be 0.5.

4. COMPUTING LEXICAL CHAIN SCORES

Lexical chains are composed of words that have a lexical relation. In order to find these relations among words, WordNet lexical knowledge base is used. In WordNet, words have a number of meanings corresponding to different senses. Each sense of a word belongs to a synset (a set of words that are synonyms). This means, ambiguous words may be present in more than one synset. Synsets may be related to each other with different types of relations (like hyponym, hypernym, antonym, etc.).

In computing lexical chains, each word must belong to exactly one lexical chain. There are two challenges for this. First, there may be more than one sense for ambiguous words and a heuristic must be used to determine the correct sense of the word. Second, a word may be related to words in different chains. For example, a word may be in the same synset with a word in one lexical chain, while having a hyponym/hypernym relationship with another word in another chain. The aim here is to find the best way of grouping words that will result in the longest and strongest lexical chains.

This process consists of four steps:

- Selecting candidate words
- Constructing lexical chains from these words
- Scoring these chains
- Selecting the strong chains

4.1. Selecting Candidate Words

Candidate words for lexical chains are the nouns. So, firstly, the text is put through part of speech (PoS) tagging. This tagging process is necessary to determine the nouns in the document. After the nouns are determined, they are added to the lexical chain candidate words list.

The part of speech tagger used in this work comes from LingPipe project version 3.8.1 [34].

4.2. Constructing Lexical Chains from Candidate Words

When the candidate words list is constructed, the words in the list are sorted in ascending order of their number of senses. This way, the words with the least number of senses (i.e., the least ambiguous ones) are treated first.

For each word, the system tries to find an appropriate chain that the candidate word can be added, according to a relatedness criterion among the members of the chain and the candidate word. This search continues for every sense of the candidate word, until an appropriate chain is found. If such a chain is found, the current sense of the candidate word is set to be the disambiguated sense, and the word is added to the lexical chain.

This relatedness criterion compares each member of the chain to the candidate word to find out if

- the sense of the lexical chain word belongs to the same synset as the sense of the candidate word
- the synset of the lexical chain word has a hyponym relation with the synset of the candidate word
- the synset of the lexical chain word has a hypernym relation with the synset of the candidate word
- the synset of the lexical chain word share the same parent with the synset of the candidate word in a hyponym relation
- the synset of the lexical chain word share the same parent with the synset of the candidate word in a hypernym relation

If the system cannot find an appropriate lexical chain to add the candidate word for any sense of the word, a new chain is constructed for every sense of the word. For

instance, this will create five new lexical chains in the system for a word that has five different senses. This way, when a new candidate word is compared to these chains, it will be possible to find a relation between the new candidate word and any of these five senses of the previous word.

The problem here is that, there may be more than one chain in the system for the same word, that continue growing at the same time. For example a word with two senses will create two different lexical chains. When a second word arrives, it may be related to the first sense of the first word and be added to the first chain. After that, if a third word arrives and is related to the second sense of the first word, it will be added to the second chain and the two chains will continue growing independently. This will conflict the requirement that says each word must belong to exactly one lexical chain.

This problem is eliminated by removing the rest of the chains for the word in the system, as soon as a second word is related with one of the senses of the word. This is illustrated in Figure 4.1. At the beginning, there are no chains in the system. When the first word *plant* arrives, since there are no existing chains, four new lexical chains are created for the four different senses of the word. When the second word *flower* arrives, it is compared to these four chains successively. No relations can be found with the first chain. However, one sense of the word *flower* (a plant cultivated for its blooms or blossoms) is related to the sense of the word *plant* in the second chain (a living organism lacking the power of locomotion) with a hypernym relation. So, the word *flower* is added to the second chain. This means both the word *flower* and the word *plant* are disambiguated. So, the chains that contain only the word *plant* are removed from the system.

4.3. Scoring the Chains

Once the lexical chains are computed, each chain is given a score number that shows its strength. This score number will be used to select the strongest chains of the document and the sentences that contain words that occur in strong chains will be given a higher sentence score.

(i) No chains

(ii) Chain1 ⇒ **plant:** buildings for carrying on industrial labor
Chain2 ⇒ **plant:** a living organism lacking the power of locomotion
Chain3 ⇒ **plant:** something planted secretly for discovery by another
Chain4 ⇒ **plant:** an actor situated in the audience whose acting is rehearsed but seems spontaneous to the audience

(iii) Chain2 ⇒ **plant:** a living organism lacking the power of locomotion; **flower:** a plant cultivated for its blooms or blossoms

Figure 4.1. Lexical chain management.

The score of a chain depends both on its length and on its homogeneity. The length of a chain is the number of occurrences of members of the chain. Its homogeneity is inversely related with its diversity. For instance, if there are three distinct words in a chain that has seven members, this chain is assumed to be stronger than a chain with the same number of members, but five distinct words.

So, the score of a chain is calculated as follows:

$$score = length * homogeneity \tag{4.1}$$

where

$$homogeneity = 1 - \frac{numberOfDistinctOccurrences}{length} \tag{4.2}$$

4.4. Selecting the Strong Chains

In this work, strong lexical chains are assumed to be the ones whose score exceeds the average of the chain scores by two standard deviations. That is, a strong chain must satisfy the criterion;

$$score(chain) > average(chainScores) + 2 * standardDeviation(chainScores)$$

Moreover, chains that contain only one word are not accepted as strong chains. Therefore, a strong chain must also satisfy the criterion;

$$wordCount(chain) > 1$$

5. FEATURE WEIGHTS

In this thesis we use 12 different text features to score sentences. After each sentence of a document is scored, the sentences of the document are sorted according to their scores and the highest scored sentences are selected to form the summary of that document.

However, not all the feature scores have the same importance while calculating the sentence score. A sentence score is a weighted sum of that sentence's feature scores. Each feature may have a different weight and these weights are learned from the manually summarized documents, using machine learning methods. Thus, a sentence's score is calculated as follows:

$$\begin{aligned} Score(s) = {} & w_1 f_1(s) + w_2 f_2(s) + w_3 f_3(s) + w_4 f_4(s) + w_5 f_5(s) + w_6 f_6(s) \\ & + w_7 f_7(s) + w_8 f_8(s) + w_9 f_9(s) + w_{10} f_{10}(s) + w_{11} f_{11}(s) + w_{12} f_{12}(s) \end{aligned} \tag{5.1}$$

f_i are the feature scores of each sentence and their values can range from 0 to 1. They are computed separately for each sentence s. w_i can range from 0 to 15. They are learned using genetic algorithms.

The system has two modes of operation: Training Mode (where the feature weights are learned from the corpus) and Testing Mode (where new documents are summarized using the weighted feature scores). Figure 5.1 shows these two modes.

5.1. Training Mode

In the training mode, the weights of each feature are learned by the system, using the manually summarized documents.

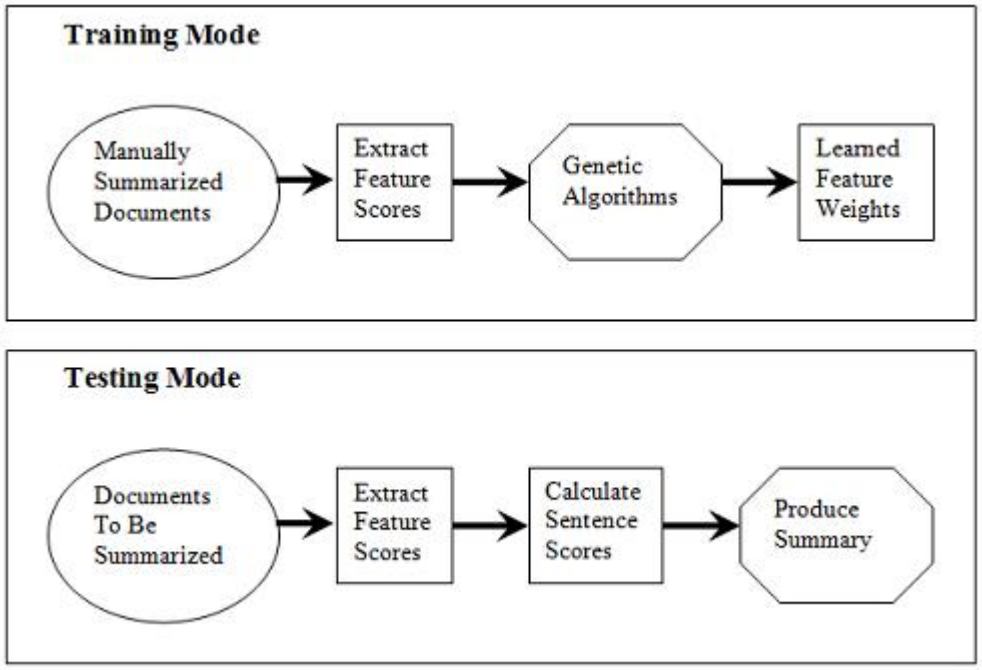

Figure 5.1. Model of the automatic summarization system.

Firstly, the text feature scores are calculated for every sentence. Since these scores are constant for each sentence, they are calculated once before the machine learning procedure starts. The algorithm is shown in Figure 5.2.

Then, these feature scores are integrated by a weighted score function in order to score each sentence. On each iteration of the training routine, random weights are assigned to 12 text features, and thus sentence scores are calculated. According to these sentence scores, a summary is generated for each document in the corpus. The precision of each automatically generated summary when compared to its manually generated summary is calculated using the following formula:

$$P = \frac{|S \cap T|}{|S|} \tag{5.2}$$

```
for each manually summarized document do
   for each feature do
      for each sentence do
         calculate the feature score of the sentence
      end for
   end for
end for
```

Figure 5.2. Feature score calculation.

where T = reference summary and S = machine generated summary.

The average of these precisions gives the performance of that iteration. This performance metric shows how appropriate the random weights of that iteration were for this summarization system. The best of all iterations is selected using genetic algorithms.

5.1.1. Genetic Algorithms

Genetic algorithms (GAs), invented by John Holland in the 1960s, are the most widely used approaches to computational evolution. GAs derive their name from the fact that they are loosely based on models of genetic change in a population of individuals [35]. The algorithms start with a random set of individuals (the population) and evolve by carrying the offspring of the fittest individuals of the population to the next generation. The algorithms consist of three basic parts: a **fitness** notion that indicates how worthy an individual is to be carried to the next generation, a **mating operator** to produce the offspring for the next generation, and **genetic operators** that determine the genetic structure of the offspring according to the genetic structure of the parents.

In this work, each individual of the population is a vector of feature weights. There are 12 features and each feature weight can have a value between 0 and 15. When these weights are represented in binary mode using 4 bits, they form a vector of length 48. This vector is the individual of the GAs.

The fitness of an individual is the performance metric mentioned in section 5.1. Each individual represents a set of feature weights. Using these weights, sentence scores are calculated and summaries are generated for each document in the corpus. The precision of the automatically generated summary when compared to the manually generated summary is calculated for each document and the average of these precision values is the fitness of that individual. There is no need to calculate the recall values because precision is equal to recall for every document. This is because the length of the automatically generated summary is equal to the reference summary, since the summarization ratios are the same.

The algorithm for the fitness function is given in Figure 5.3.

In this study, there are 1000 individuals in the population. At each generation, the mating operator selects the fittest 50 of the population and carries these individuals directly to the next generation. The other 950 are produced by a selected pair of parents. Each individual can be selected to be a parent according to a probability rate calculated from the fitness value of that individual. A child is produced by merging the first n bits of the vector of one parent and the last 48-n bits of the vector of the other parent. n is random for each reproduction.

After a child is produced, it is put through mutation with a predetermined probability. If it goes through mutation, one of its bits will be set to a random value. Finally, after mutation, the produced child is added to the population for the next generation.

The selection algorithm is given in Figure 5.4.

The GAs are run for 100 generations to obtain a steady combination of feature

for each document **do**

for each sentence **do**

using the weights of this individual and precalculated feature scores, calculate the sentence score

end for

sort sentences in a non-ascending score order

pick the top X sentences according to the summarization ratio

calculate the precision of this summary when compared to the reference summary

$$P = \frac{|S \cap T|}{|S|}$$

where T = reference summary and S = machine generated summary

end for

calculate the average precision

Figure 5.3. Fitness function.

```
pass the fittest 50 individuals to the next generation automatically
for the rest 950 individuals do
   calculate a probability rate according to the fitness values
   select a pair of parents for cross over according to these probabilities
   (an individual can be selected to be a parent more than once)
   child ⇐ crossover(parent1, parent2)
   set a constant probability
   mutation(child, probability)
   add child to the population of next generation
end for
```

Figure 5.4. Selection algorithm.

weights. The best individual that is produced after these iterations is selected to be the set of feature weights that will be used in the testing mode.

5.2. Testing Mode

In the testing mode, 20 English documents are summarized automatically by the system. Using the feature weights learned by genetic algorithms as mentioned in section 5.1, weighted scores are calculated for each sentence of these documents. Then the sentences are sorted according to their scores and the ones that have the highest scores are selected to form the summary. The algorithm for the testing mode is given in Figure 5.5.

```
for each document do
   for each feature do
      for each sentence do
         calculate the feature score of the sentence
      end for
   end for
   for each sentence do
      using the learned weights for each feature, calculate the sentence score
   end for
   sort sentences in a non-ascending order
   pick the top X sentences according to the summarization ratio
end for
```

Figure 5.5. Testing mode.

6. EVALUATION

6.1. Training Corpus

The text corpus used in this project consists of 100 manually summarized documents taken from the CAST (Computer-Aided Summarization Tool) corpus [32]. CAST is a project of the Computer Linguistics Research Group of University of Wolverhampton, UK. One of the goals of the CAST project was to develop an annotated corpus for automatic summarization that contains information about the importance of the sentences.

The texts included in the corpus were taken from the Reuters Corpus. Their genre is newswire. The documents consist of 20 sentences in the average. They are manually summarized using a 30% summarization ratio. The annotators were native English speakers. Figure 6.1 shows an example document together with its manual summary.

80 of these documents were used as the training corpus to train the system. Other 20 were used as the testing corpus. For cross-validation, the tests were run 5 times with different sets of training and testing documents. The results show the average of these five runs.

6.2. Evaluation Method

We used the intrinsic evaluation method mentioned in [13]. This method judges the quality of a summary based on the coverage between it and the manual summary. We used precision as the performance measure. Assuming that T is the manual summary and S is the machine generated summary, the measurement of precision P is defined as follows:

Original Text

Hong Kong property glitters like gold.

An advisor to Governor Chris Patten's inner cabinet said on Monday that property in Hong Kong has become like gold that is not merely worn but kept and accumulated giving rise to abnormal demand. Chen Kwan-yiu, a member on the executive council, also hinted that a quick fix to the British colony's property woes was unlikely. "The current supply of flats is actually adequate for demand, but property has become like gold in the 60s, when it was not only used but was kept as an investment," Chen said. "So it has led to an abnormal demand," Chen told reporters. The economist said the solution to the problem was to provide more land for residential use. Hong Kong's property sector has been in the news in recent months with the government and private property developers coming under intense public fire for soaring housing prices. Many young working adults in Hong Kong continue to live with their parents due to notoriously high rents, and the option of owning a home is increasingly seen beyond the reach of many of the territory's 6.4 million population. "If ever we see a revolution in Hong Kong, it'll be over housing. Young people don't have a hope in hell to own their own homes anymore," retiree Alex Lo, 55, told Reuters. "It's now the giant property developers who control Hong Kong," said Lo, who lives in a tiny government flat in Tai Po with his wife and three children in their twenties. Hong Kong has much on its mind as the handover to Chinese rule at midnight on June 30 nears, and in the last few weeks has had to cope with extra stress caused by the housing woes. On March 12, a box thought to contain a bomb planted outside the legislative council building in the territory's Central district froze traffic for several hours. The box turned out to be empty, but its message was no less troubling. Writing on the box complained that the government and the territory's future leader, Tung Chee-hwa, did not care about soaring property prices. And on Thursday, a fist-fight broke out at a flat sale. One property agent was knocked out and two guards were arrested. The government last week acted to curb speculation by requiring developers to put more unfinished flats on the market The move was followed by a 25 basis point rise in the prime lending rate to 8.75 percent, which bankers said could marginally affect the property market. Ever-strong demand led developer Cheung Kong (Holdings) Ltd to release more units on Sunday in the sale of part of its Kingswood Villas development in the New Territories. A total of 600 units, priced mostly at HK$4,793 (US$620) per square foot, was sold in addition to the 264 flats originally put up for sale.

Human Summary

Hong Kong property glitters like gold.

An advisor to Governor Chris Patten's inner cabinet said on Monday that property in Hong Kong has become like gold that is not merely worn but kept and accumulated giving rise to abnormal demand. Chen Kwan-yiu, a member on the executive council, also hinted that a quick fix to the British colony's property woes was unlikely. Hong Kong's property sector has been in the news in recent months with the government and private property developers coming under intense public fire for soaring housing prices. On March 12, a box thought to contain a bomb planted outside the legislative council building in the territory's Central district froze traffic for several hours. Writing on the box complained that the government and the territory's future leader, Tung Chee-hwa, did not care about soaring property prices. The government last week acted to curb speculation by requiring developers to put more unfinished flats on the market The move was followed by a 25 basis point rise in the prime lending rate to 8.75 percent, which bankers said could marginally affect the property market.

Figure 6.1. An example document and its manual summary.

$$P = \frac{|S \cap T|}{|S|}$$

6.3. The Effect of Each Feature on Summarization Performance

We investigated the effects of each text feature used in our model on the summarization performance. In order to calculate these effects, we used the score function in equation (5.1) with all feature weights equal to 0 but one feature weight equal to 1. So, for the effect of the i^{th} text feature, the equation is as follows:

$$Score(s) = f_i(s) \tag{6.1}$$

The aim of this work was to see if lexical chains could be used as a factor while determining the importance of sentences. The other text features were used to score sentences before, however, the effects of lexical chains were not included in these scoring functions. We claim that if we use lexical chains as a representation of the lexical cohesion among the words of a source text, we may get a different level of understanding about the importance of its sentences. For this reason, we wanted to see the performance of each text feature on its own, and compare the performance of lexical chains feature with the performances of other text features.

Table 6.1 shows the average precisions obtained by using each text feature to summarize the documents in the testing corpus.

We can see that using only sentence location feature gives one of the best results. This is predictable since the leading sentences in a document usually give a general understanding about the topic of the document, and they have a high probability to

Table 6.1. The average summarization precision associated with each text feature

Feature	**Average Precision**
P(f1) - Sentence Location	0.43
P(f2) - Sentence Relative Length	0.42
P(f3) - Average TF	0.32
P(f4) - Average TF-IDF	0.30
P(f5) - Sentence Resemblance to Title	0.39
P(f6) - Sentence Centrality	0.43
P(f7) - Sentence Inclusion of Emphasize Words	0.36
P(f8) - Sentence Inclusion of Name Entities	0.43
P(f9) - Sentence Inclusion of Numerical Data	0.29
P(f10) - Number of Synonym Links	0.42
P(f11) - Number of Co-occurrence Links	0.41
P(f12) - Lexical Chain Score	0.40

be included in the summary. Sentence centrality gives also good results. This must be because this feature favors the sentences that mention many of the topics that appear throughout the text. This means these sentences are more important than the sentences that mention only a single topic, and are included in the summary. Moreover, sentence inclusion of name entities feature gives high performance. This may be a sign that sentences that give information about specific people or organizations are likely to be selected for the summary.

The result of lexical chain feature is also among the high performance results. This shows that using lexical chains as an intermediate representation of lexical cohesion that exists throughout the text can be used as a means to determine the importance of sentences in that text. This feature makes better predictions than many of the other text features.

One of the lowest performance results belongs to the sentence inclusion of numerical data feature. This feature considered only the appearance of numerical information in a sentence. If this feature was not successful in predicting the sentences that will be in the summary, then this means that human summarizers do not generally include sentences with numerical information to a summary. Likewise, average TF and average TF-IDF features also gave bad results. Although these features are commonly used in ATS systems, it seems that tying to predict the importance of sentences depending only on word frequencies is not enough to generate high quality summaries.

6.4. The Results of This Model

In this section, we will investigate the performance results of the system when all the 12 text features are used. In order to select the sentences that will be included in the summary, the score function in equation (5.1) is used. The weights of the text features are taken from the results of the training mode.

In the training mode, genetic algorithms were run with the following properties:

- There are 1000 individuals in a population.
- At each iteration, 50 fittest individuals are selected for the next generation.
- 950 more individuals are selected through selection, crossing over and mutation.
- The algorithms are run for 100 iterations.
- Summarization ratio is 30

Table 6.2 shows the weights of each text feature calculated by the training module, and the average precision of documents summarized with these feature weights in the testing mode.

We see that sentence location feature is given the highest weight among all the features. This is predictable and commonly seen in automatic summarization systems. Since the most important content of the text is usually given in the beginning, selecting the leading sentences for summary gives better results than most of the automatic summarization methods.

Following that feature, comes sentence centrality. This is also meaningful, because this feature was the one that gave the highest precision among all features when used alone.

The feature weight for lexical chains was not among the highest ones. However, it supports and reinforces the results of the centrality and co-occurrence link features as it analyses the cohesion in the text from a different perspective.

Finally, when all the feature scores are combined, the system gave higher average precision than any of the single features. When the average precision of the combination of the features (0.46) is compared to the average precision of the best features (0.43 of location, centrality and name entities), the difference is not much. However, this situation probably stems from the genre of the corpus that the experiment was performed on. The precisions of the single features reflect the feature performances on the newswire genre. In another genre, the single feature performances may change. On the other hand, the performance of the combination of features will still be higher.

This shows that using the combination of different features increases the performance of the summary and genetic algorithms were successful to learn a set of weights for those features that would result in a better output.

Table 6.2. The system performance evaluation based on precision

Text Feature	**Feature Weight (over 15)**
P(f1) - Sentence Location	14
P(f2) - Sentence Relative Length	3
P(f3) - Average TF	1
P(f4) - Average TF-IDF	5
P(f5) - Sentence Resemblance to Title	4
P(f6) - Sentence Centrality	13
P(f7) - Sentence Inclusion of Emphasize Words	12
P(f8) - Sentence Inclusion of Name Entities	11
P(f9) - Sentence Inclusion of Numerical Data	2
P(f10) - Number of Synonym Links	10
P(f11) - Number of Co-occurrence Links	12
P(f12) - Lexical Chain Score	5
Average Precision	**0.46**

Figures 6.2, 6.3 and 6.4 show the original, human generated summary and system generated summary of a sample document, respectively. It can be seen that the first sentence of the document is included in both the real summary and the system output. This is predictable, because many features for this sentence will have high scores for this sentence. For example, location feature will have a score value of 1. Relative length score will be high, because this sentence is one of the longest sentences in the text. Resemblance to title score will also be high, because the sentence has common words with the title, like *Serbs*, *party*, and *delegates*. Finally, name entities score will be high, because of the name entities found in the sentence, like *Serbs*, *Croatia* and *U.N.*. Since these are features weighted with high coefficients, the score of the first

sentence will be high and it will be selected to be in the summary.

Serbs throw bricks at Croatian HDZ party delegates.

(1) Hundreds of angry Serbs attacked delegates from Croatia's ruling HDZ party who had to be rescued by U.N. peacekeepers while trying to campaign for upcoming elections in Eastern Slavonia on Monday.

(2) U.N. spokesman Douglas Coffman said the Croatian Democratic party (HDZ) held a press conference in a hotel in the town of Vukovar to present candidates who will run in the Serb enclave during nation-wide local elections on April 13.

(3) The vote for county and municipal bodies will be the first in independent Croatia.

(4) Eastern Slavonia will revert to Croatian rule this summer.

(5) The HDZ members and a dozen Croatian journalists were first confronted by a small group of middle-aged women but the demonstration grew until there were "hundreds of protesters", Coffman said.

(6) "On their way out (of the hotel) HDZ activists and accompanying Croatian journalists were hit by a barrage of eggs and bricks, but nobody sustained any serious injuries, only bruises," he said.

(7) Croatian state radio said one journalist sustained a slight head injury while the bus in which they came was completely destroyed.

(8) "They had to be evacuated by Russian (peacekeepers) in armoured personnel carriers to the U.N. bases and then they were escorted out of the region (to Croatian government territory)," said Coffman.

(9) Eastern Slavonia was one of three regions of Croatia in which local Serbs rebelled against Zagreb's drive for independence from federal Yugoslavia in 1991 and set up their own state.

(10) Croatia recaptured two other enclaves in 1995 and agreed to peacefully "reintegrate" Eastern Slavonia, now monitored by some 5,000 U.N. peacekeepers, whose mandate expires on July 15.

(11) The head of the U.N. mission (UNTAES), U.S. general Jacques Klein, was away during the visit, which was the first time HDZ had campaigned there since 1991.

(12) Western officials are concerned that many Serbs, fearing reprisals or refusing to live under Croatian rule, may leave Eastern Slavonia before Zagreb's control is restored.

(13) Shortly after the incident, Ivica Vrkic, government official in charge of Eastern Slavonia urged the U.N. and international police to step up security measures in the enclave.

(14) "We remind you that it is the duty of UNTAES and the police duty to provide safety for such electoral campaigns.

(15) But today's incident will not postpone the election and the reintegration of Eastern Slavonia," he wrote in a letter.

Figure 6.2. Sample Original Document.

However, this was the only sentence that both the system and the human summarizer included in the summary. The system failed to favor the third sentence, although it was selected by the human summarizer. This may be because the content of the sentence is important to understand the text, but it is independent from the rest of the sentences. The words in the sentence can not be related to other sentences or the title, so the system calculated a low score value for this sentence, and left it out of the

summary. The system also underestimated the sixth sentence, probably because the words egg, brick, injury or bruise did not have any lexical relation to other words in other sentences of the text.

The sentences 9 and 10, which are selected in the human generated summary, scored relatively high values in the system evaluation. However, their scores were not as high as the selected sentences, and they were left out of the summary by the system.

Serbs throw bricks at Croatian HDZ party delegates.

(1) Hundreds of angry Serbs attacked delegates from Croatia's ruling HDZ party who had to be rescued by U.N. peacekeepers while trying to campaign for upcoming elections in Eastern Slavonia on Monday.

(3) The vote for county and municipal bodies will be the first in independent Croatia.

(6) "On their way out (of the hotel) HDZ activists and accompanying Croatian journalists were hit by a barrage of eggs and bricks, but nobody sustained any serious injuries, only bruises," he said.

(9) Eastern Slavonia was one of three regions of Croatia in which local Serbs rebelled against Zagreb's drive for independence from federal Yugoslavia in 1991 and set up their own state.

(10) Croatia recaptured two other enclaves in 1995 and agreed to peacefully "reintegrate" Eastern Slavonia, now monitored by some 5,000 U.N. peacekeepers, whose mandate expires on July 15.

Figure 6.3. Sample Human Generated Summary.

Serbs throw bricks at Croatian HDZ party delegates.

(1) Hundreds of angry Serbs attacked delegates from Croatia's ruling HDZ party who had to be rescued by U.N. peacekeepers while trying to campaign for upcoming elections in Eastern Slavonia on Monday.

(2) U.N. spokesman Douglas Coffman said the Croatian Democratic party (HDZ) held a press conference in a hotel in the town of Vukovar to present candidates who will run in the Serb enclave during nation-wide local elections on April 13.

(4) Eastern Slavonia will revert to Croatian rule this summer.

(8) "They had to be evacuated by Russian (peacekeepers) in armoured personnel carriers to the U.N. bases and then they were escorted out of the region (to Croatian government territory)," said Coffman.

(13) Shortly after the incident, Ivica Vrkic, government official in charge of Eastern Slavonia urged the U.N. and international police to step up security measures in the enclave.

Figure 6.4. Sample System Generated Summary.

6.5. Alternative Models

To be able to compare the results of our model to similar models, we changed some configurations of our system, and ran our experiments with different settings. Like the original runs, 80 of the 100 documents were used as the training corpus and other 20 were used as the testing corpus. For cross-validation, the tests were run 5 times with different sets of training and testing documents. The results show the average of these five runs.

One of the alternative models was using a smaller set of features. Instead of using a single feature, or combining all of the 12 features, we took the first three features that scored best on their own, together with lexical chains feature, and investigated the performance of the system using only these four text features.

Table 6.3. The system performance evaluation using 4 features

Text Feature	**Feature Weight (over 15)**
P(f1) - Sentence Location	14
P(f6) - Sentence Centrality	14
P(f8) - Sentence Inclusion of Name Entities	3
P(f12) - Lexical Chain Score	10
Average Precision	**0.45**

Table 6.3 shows the weights for four of the text features calculated by the training module, and the average precision of documents summarized with these feature weights in the testing mode. We can see that the result is almost the same as the performance of the system when all of the features are considered. This shows that the other eight features did not contribute much to the system performance. Instead, they had a minor effect.

Another alternative model was decreasing the threshold for determining strong

lexical chains. The original criterion for selecting a strong lexical chain was as follows:

$$score(chain) > average(chainScores) + 2 * standardDeviation(chainScores)$$

Selecting more chains as strong chains would increase the lexical chain scores for sentences, since their probability of containing a strong chain word would increase. Assuming that an increase in the lexical chain scores might affect the performance of the system, we ran our tests with the following strong chain criterion:

$$score(chain) > average(chainScores) + standardDeviation(chainScores)$$

Again, only the four text features that were used in the previous alternative model were considered, since it was seen that they had the most influence on the summary performance. Table 6.4 shows the results of the second alternative model.

Table 6.4. The system performance evaluation using different strong chain criterion

Text Feature	**Feature Weight (over 15)**
P(f1) - Sentence Location	15
P(f6) - Sentence Centrality	9
P(f8) - Sentence Inclusion of Name Entities	3
P(f12) - Lexical Chain Score	9
Average Precision	**0.46**

The result did not change when compared to the first alternative. It is even same as the original model. Moreover, we can see that the weight of the lexical chain score did not increase, so we can conclude that this alternative model did not change the system performance much.

7. CONCLUSION

In this study, we have combined two approaches used in automatic text summarization: using Lexical Chains to detect the lexical cohesion that exists throughout the text, and using Genetic Algorithms to efficiently learn the weights to be used in sentence scoring.

We have computed lexical chains in a text depending on the lexical relations among words in the text. These relations were determined using WordNet. All these computed chains were scored in order to select the strongest chains in a given text.

Then we have computed different text features for each sentence in a text. These features tried to analyze the sentence to different levels. We used lexical chains as the basis for one of these feature functions. We gave higher lexical chain feature scores to sentences that contained more strong lexical chain words. After all the feature scores were computed, we used genetic algorithms to determine the appropriate feature weights. These feature weights were then used to score the sentences in the testing mode. The highest scored sentences were selected to be included in the summary.

The results showed that, some features like sentence location, sentence centrality and sentence inclusion of name entities showed better performance in terms of average summary precision than other features. Moreover, some features like sentence inclusion of numerical data, average TF and average TF-IDF had very little effect on the summary performances. However the combination of the features gave better results than any of the features.

The contribution of this study is that it puts the benefits of lexical chain approach and genetic algorithms approach together. It combines information coming from different levels of analysis on text. Different from other work in this area, location features like sentence location, thematic features like sentence centrality and cohesion features like sentence inclusion of strong lexical chain words are all considered together in this

study. It also makes use of machine learning approach to determine the coefficients of this combination.

As a future work, the model can be tested on different text genres. The corpus we used in this study consisted of newswire documents. However, the tests can be run on scientific documents, or some other genre in order to see the change in the text feature performances and in the overall system performance.

Furthermore, the model may be adapted for summarization in Turkish. For this purpose, Turkish WordNet, a Turkish PoS tagger, and a Turkish NE recognizer must be used. Also, the text features must be adapted to Turkish grammar and Turkish prefixes and suffixes must be considered. Finally, some new features that represent Turkish language and summarization habits in a more specific way might be added.

APPENDIX A: STOP WORDS

a	able	about	across	after	all
almost	also	am	among	an	and
any	are	aren't	as	at	be
because	been	but	by	can	cannot
can't	could	couldn't	dear	did	didn't
do	don't	does	doesn't	either	else
ever	every	for	from	get	got
had	hadn't	has	hasn't	have	haven't
he	her	hers	him	his	how
however	i	if	in	into	is
isn't	it	its	just	least	let
like	likely	may	me	might	most
must	mustn't	my	neither	no	nor
not	of	off	often	on	only
or	other	our	own	rather	said
say	says	she	should	shouldn't	since
so	some	than	that	the	their
them	then	there	these	they	this
tis	to	too	twas	us	wants
was	wasn't	we	were	weren't	what
when	where	which	while	who	whom
why	will	with	won't	would	wouldn't
yet	you	your			

APPENDIX B: EMPHASIZE WORDS

very	really	extremely
amazingly	exceptionally	incredibly
remarkably	particularly	absolutely
especially	specifically	quite
certainly	seriously	highly
crucially	importantly	indeed
truly	surely	

REFERENCES

1. Kiani, A. and M. R. Akbarzadeh, "Automatic Text Summarization Using: Hybrid Fuzzy GA-GP", *2006 IEEE International Conference on Fuzzy Systems*, pp. 5465–5471, 2006.

2. Mani, I., *Automatic Summarization*, John Benjamins Publishing Company, Amsterdam/Philadelphia, 2001.

3. Edmundson, H. P., "New Methods in Automatic Abstracting", *Journal of the Association for Computing Machinery, 16, 2*, pp. 264–285, 1969.

4. Fattah, M. A. and F. Ren, "Automatic Text Summarization", *Proceedings of World Academy of Science, Engineering and Technology Volume 27*, pp. 192–195, 2008.

5. Dehkordi, P. K., H. Khosravi and F. Kumarci, "Text Summarization Based on Genetic Programming", *International Journal of Computing and ICT Research Volume 3, No 1*, pp. 57–64, 2009.

6. Jong, G. F. D., "An overview of the FRUMP system", *Strategies for Natural Language Processing*, 1982.

7. Hahn, U. and I. Mani, "Automatic Text Summarization: Methods, Systems, and Evaluations", *Tutorial, International Joint Conference on Artificial Intelligence (IJCAI 99)*, 1998.

8. Hovy, E. and C. Y. Lin, "Automated Text Summarization in SUMMARIST", *Advances in Automatic Text Summarization, I. Mani and M. T. Maybury(eds.)*, pp. 81–94, 1999.

9. Jing, H., R. Barzilay, K. McKeown and M. Elhadad, "Summarization Evaluation Methods: Experiment and Analysis", *Working notes of the AAAI Spring Sympo-*

sium on Intelligent Text Summarization, 1998.

10. Barzilay, R. and M. Elhadad, "Using Lexical Chains for Text Summarization", *ACL/EACL-97 Workshop on Intelligent Scalable Text Summarization*, pp. 10–17, 1997.

11. Silber, H. G. and K. F. McCoy, "Efficient Text Summarization Using Lexical Chains", *Proceedings of the 5th international conference on Intelligent user interfaces*, pp. 252–255, January 2000.

12. Brunn, M., Y. Chali and C. J. Pinchak, "Text Summarization Using Lexical Chains", *Document Understanding Conference (DUC)*, pp. 135–140, 2001.

13. Barzilay, R., *Lexical Chains for Summarization*, M.Sc. thesis, Ben-Gurion University of the Negev, Department of Mathematics & Computer Science, 1997.

14. Haliday, M. and R. Hasan, *Cohesion in English*, Longman, London, 1976.

15. *WORDNET*, `http://wordnet.princeton.edu/`.

16. Li, J., L. Sun, C. Kit and J. Webster, "A Query-Focused Multi-Document Summarizer Based on Lexical Chains", *Proc. of the Document Understanding Conference DUC-2007*, 2007.

17. Fuentes, M. and H. Rodriguez, "Using Cohesive Properties of Text for Automatic Summarization", *JOTRI2002 - Workshop on Processing and Information Retrieval*, 2002.

18. Paice, C. and P. Jones, "The Identification of Important Concepts in Highly Structured Technical Papers", *Proceedings of ACM-SIGIR 93*, 1993.

19. Skorokhodko, E., "Adaptive Method of Automatic Abstracting and Indexing", *IFIP Congress, Ljubljana, Yugoslavia 71*, pp. 1179–1182, 1972.

20. Pollock, J. and A. Zamora, "Automatic Abstracting Research at Chemical Abstracts Service", *Journal of Chemical Information and Computer Sciences, 15, 4*, 1975.

21. Kupiec, J., J. Pedersen and F. Chen, "A Trainable Document Summarizer", *In Proceedings of ACM-SIGIR 95*, 1995.

22. Miike, S., E. Itoh, K. Ono and K. Sumita, "A Full-Text Retrieval System with a Dynamic Abstract Generation Function", *Proceedings of ACM-SIGIR 94*, 1994.

23. Marcu, D., "From Discourse Structures to Text Summaries", *Proceedings of the ACL/EACL 97 Workshop on Intelligent Scalable Text Summarization*, pp. 82–88, 1997.

24. Luhn, H. P., "The Automatic Creation of Literature Abstracts", *IBM Journal of Research and Development, 2*, pp. 159–165, 1959.

25. Brandow, R., K. Mitze and L. Rau, "Automatic Condensation of Electronic Publications by Sentence Selection", *Information Processing and Management, 31, 5*, pp. 675–685, 1994.

26. Maybury, M., "Generating Summaries from Event Data", *Information Processing and Management, 31, 5*, pp. 735–751, 1995.

27. Fum, D., G. Guida and C. Tasso, "Evaluating Importance: A Step Towards Text Summarization", *Proceedings of IJCAI 85*, pp. 840–844, 1985.

28. Chuang, W. and J.Yang, "Extracting Sentences Segments for Text Summarization: A Machine Learning Approach", *Proceedings of the 23th Annual international ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 125–159, 2000.

29. Mani, I. and E. Bloedorn, "Machine Learning of Generic and User-Focused Sum-

marization", *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pp. 821–826, 1998.

30. Alfonseca, E. and P. Rodriguez, "Generating Extracts with Genetic Algorithms", *Information Retrieval, Lecture Notes in Computer Science, 2633*, pp. 511–519, 2003.

31. Fattah, M. A. and F. Ren, "GA, MR, FFNN, PNN and GMM Based Models for Automatic Text Summarization", *Computer Science and Language 23*, pp. 126–144, 2009.

32. *CAST Corpus*, `http://www.clg.wlv.ac.uk/projects/CAST/`.

33. *Illinois Named Entity Tagger*, `http://cogcomp.cs.illinois.edu/page/software/`.

34. *LingPipe*, `http://alias-i.com/lingpipe/`.

35. Jong, K. D., "Learning With Genetic Algorithms: An Overview", *Machine Learning 3*, pp. 121–138, 1988.