TURKISH-ENGLISH SENTENCE ALIGNMENT

by

Ahmet Mustafa Güngör

&

Şerafettin Taşcı

Submitted to the Department of Computer Engineering

in the Faculty of Engineering as

CMPE 492

Senior Project

Boğaziçi University

2006

ABSTRACT

TURKISH-ENGLISH SENTENCE ALIGNMENT

Sentence alignment is an important subject in Machine Translation. An aligned parallel corpus provides aid to human translators since it is possible to look up all sentences in which a word or a phrase occurs to see the ways in which that word or phrase has been translated into the other language. Sentence alignment is also a first step towards word alignment, which is used to determine instances where a word in one language consistently appears in sentences aligned with sentences containing the equivalent word in the other language. It is also used in extracting structural and semantic information and deriving statistical parameters from bilingual corpora.

In this project our aim is to design a sentence alignment algorithm for using with bilingual texts in which one of the texts is Turkish. In our method, we use the location information of sentences and paragraphs as well as the lengths of them for aligning the bilingual texts. This method is quite easy to implement and independent of the languages of the bilingual texts. The results of the experiments we did show that our method's success is related with the success in paragraph alignment phase. When the paragraph alignment is successful, if the text is easy (90% 1-1 beads) it has 96.1% accuracy. If the text is difficult (65% 1-1 beads), it has lower (about 70.3%) but still a high accuracy with respect to the difficulty of the text. However if it makes too many errors in paragraph alignment, which is a rare case, it gives continuous blocks of wrong alignment beads.

In this paper, we also propose the difficulties and challenges in Turkish-English sentence alignment and provide a bilingual resource bundle for using in following studies about sentence alignment.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Definition of Problem

Recently, sentence alignment task in machine translation gained more importance. Sentence alignment is the task of finding correspondences of sentences in one language and another. It is a first step before the more ambitious task called word alignment. Basically, alignment aims to succeed the task of extracting structural information and statistical parameters from bilingual corpora.

At first sight, this process might seem very easy but it has some important challenges which make the task difficult:

First of all, most of the time sentences do not align one-to-one. Sometimes a sentence may be translated in 2-3 sentences in the other language or some part of a text may be deleted or some additional sentences may be added to the text which has no matches in the corresponding text. Even the existence of a small amount of such sentences results in remarkable deviations in the matching of sentence beads.

Secondly, there is the problem of robustness. In real life, most of the texts have great inconsistencies with their translation such as the layout of texts, format differences, omission of some part of text and crossovers or inversions in text. The sentence alignment algorithms and programs must be devised in such a way to deal with such diverse situations and problems.

Finally, the problem of accuracy always exists. It is not easy to achieve perfect, % 100 accurate alignments even if the texts are “clean” and easy. Also the accuracies vary largely according to the input text. For example an alignment program may give wonderful results when applied on a scientific text but its success decline dramatically when applied on a novel or philosophy text.

For a sentence alignment program to be called “ideal”, it should be fast, highly accurate and require no special knowledge about the corpus or the two languages. In real world achieving all of these goals is a difficult task because of the following characteristics of real text:

1) There are no strict aligned paragraph boundaries in real bilingual text. Some texts have return character at the end of each line. It makes it difficult to determine if a newline means a new paragraph or not.

2) In many cases it is even difficult to parse the sentences because of the variance in the ending and starting characters of the sentences. In addition, the text may have many punctuation errors or symbols, pictures, etc. which make it impossible to determine the boundaries of sentences without knowing their meanings.

3) Some paragraphs or sentences may be merged into a larger paragraph or sentence because of the translator's individual idea;

4) There are many complex translation patterns in real text. For example, in some texts there is the problem of crossing dependencies in which the order of sentences are changed in the translation. In many algoriths for sentence alignment, this situation is problematic and generally ignored.

5) There exist different styles and themes;

6) Different genres have different inherent characteristics. While an algorithm is perfect in a special kind of text, it may not be successful in another type of text. So, if the researcher wants to devise a method suitable for all kinds of texts, researcher must make several experiments on different kinds of texts to make sure his method is not genre-specific.

1.2 Motivation for the Project

The lack of previous work on texts between Turkish and English is the most prominent motivation for making a research in this field. There is a high similarity between Turkish and English. (Not in the syntax or morphology of the languages but in the number of cognates due to the abundance of borrowed and cognate words, i.e. television-televizyon, tactic-taktik, yoghurt-yoğurt, etc..).

This similarity makes one to guess intuitively that length-based methods or methods using cognates will give good results for alignment of Turkish-English texts. But there has to be done concrete studies to prove or disprove such intuitions. That is first aim of our study: to see the efficiency of proposed methods for other languages in Turkish texts and make modifications such that it will give better results for Turkish.

The second purpose of our study is the collection of reliable and practical bilingual texts to be used in further studies. It is not difficult to find such texts written on paper. But of course they are useless in machine translation studies. Thus we concentrated on finding good quality texts on digital environment and we hope they will save loss of time for collection of data in future studies.

2. BACKGROUND AND PREVIOUS WORK

In the task of sentence alignment there are many papers proposing different methods but as far as the methodology they use is considered, we can group these approaches into 3 classes:

2.1) ***Length-Based Approaches***: In these approaches, content of the text in terms of semantics is not considered. These approaches use statistical methods for the task of alignment. In other words, they only consider the length of sentences while making the decision for alignment. Short sentences match with short sentences, long sentences match with long sentences. Despite their simplicity, these methods have very high accuracy. They are especially useful between texts in similar languages such as German, English and French.

2.2) ***Location-Based Approaches***: These approaches resemble the length-based approaches in respect that location-based approaches are based on statistical information. They use the fact that most of the times, beads of sentences in the two texts have similar positions. For example, if a sentence in source text is in the middle of the text, its conjugate in the target text is probably in the middle of text too.

2.3) ***Lexical Approaches:*** These methods take into account the lexical information about texts. For example, in most of them a bilingual corpus is used to match the content words in one text with their correspondences in the other text and use these matches as anchor points in the sentence alignment process. In some methods, instead of these content word pairs cognates (words in language pairs that resemble each other phonetically, ex. doctor-doktor) are used for determining the beads of sentences.

Below are detailed descriptions of these approaches with some example methods:

2.1) Length-Based Approaches

2.1.1) Goal: Find alignment A with highest probability given the two parallel texts S and T.

$$\max_A P(A, S, T)$$

S: source text, T: target text, A: alignment

- To estimate the probability above, aligned text is decomposed in a sequence of aligned sentence beads where each bead is assumed to be independent of others.
- The question is determining the right formula and parameters for estimating the probability of a certain type of alignment bead such that the sentences in that bead are given.

2.1.2) Proposed Methods:

- **Gale and Church, 1993 :** The algorithm uses sentence length (measured in number of characters) to decide if some sentences in one text is the alignment of some other sentences in the other text. The algorithm also makes use of Dynamic Programming technique which allows the system to consider all possible alignments and finding the minimum cost alignment effectively. The method performs well (at least on related languages). It gets a 4% error rate. It works best on 1:1 alignments [only 2% error rate]. It has a high error rate on more difficult alignments. (about 86%)
- **Brown et al., 1991:** It has the approach as Gale and Church, except that sentence lengths are compared in terms of words rather than characters. In fact, Gale and Church's method is inspired from Brown's method. Other difference exists in the purposes of the methods: Brown didn't want to align entire articles but just a subset of the corpus suitable for further research.

- **Wu, 1994**: Wu applies Gale and Church's method to a corpus of parallel English and Cantonese (a version of Chinese) Text. The results are not much worse than Gale and Church's method which shows that the method can also be used on unrelated languages. To improve accuracy, Wu uses lexical cues.

2.2) Location-Based Approaches

2.2.1) Proposed Methods:

- **Church, 1993:** Church argues that length-based methods work well on clean text but may break down in real-world situations (noisy OCR or unknown markup conventions). Church's method is to induce an alignment by using cognates (words that are similar phonetically across languages) at the level of character sequences.

The method consists of building a dot-plot, i.e., the source and translated text are concatenated and then a square graph is made with this text on both axes. A dot is placed at (x, y) when there is a match. Signal processing methods are then used to compress the resulting plot.

The interesting part in a dot-plot is called the *bitext maps*. These maps show the correspondence between the two languages. In the bitext maps, undetermined, roughly straight diagonals corresponding to cognates can be found. A heuristic search along this diagonal provides an alignment in terms of offsets in the two texts.

- **Fung & McKeown, 1994 :** Fung and McKeown's algorithm works:
    - Without having found sentence boundaries.
    - In only roughly parallel text (with certain sections missing in one language)
    - With unrelated language pairs.

The technique is to infer a small bilingual dictionary that will give points of alignment. For each word, a signal is produced, as an arrival vector of integer numbers giving the number of words between each occurrence of the word at hand.

2.3) Lexical Approaches

2.3.1) Proposed Methods:

- **Kay & Roscheisen, 1993 :** They start their iterations by the assumption that the first and last sentences of the texts align. These are the initial *anchors*. Then, until most sentences are aligned:

  1. Form an envelope of possible alignments.
  2. Choose pairs of words that tend to co-occur in these potential partial alignments.
  3. Find pairs of source and target sentences which contain many possible lexical correspondences. The most reliable of these pairs are used to induce a set of *partial* alignments which will be part of the final result.

- **Chen, 1993 :** Chen does sentence alignment by constructing a simple word-to-word translation model as he goes along. The best alignment is the one that maximizes the likelihood of generating the corpus given the translation model. This best alignment is found by using dynamic programming.
- **Haruno & Yamazaki, 1996 :** Their method is a variant of Kay & Roscheisen (1993) with the following differences:

  - For structurally very different languages, function words impede alignment. They eliminate function words using a POS Tagger.
  - If trying to align short texts, there are not enough repeated words for reliable alignment using Kay & Roscheisen (1993). So they use an online dictionary to find matching word pairs.

3. LITERATURE SURVEY

In this part of the document we will mention some of these methods and their approaches for aligning bilingual corpora.

For having a background for the sentence alignment subject we examined many journal papers related to the subject, most of them proposing a new method for the sentence alignment task. They all share many common properties in the methods they use but suggest a small modifications to the earlier approaches.

3.1 Gale&Church: A Program for Aligning Sentences in Bilingual Corpora

First method we have examined is the method of the Gale and Church[1]. This is maybe the most popular, well-known alignment algorithm in this literature. It has 3 main properties which cause this reputation:

* First of all, it is very simple. It simply counts the number of characters in the sentences and uses the Dynamic Programming Model to find the correct pairs of alignment. Because of this simplicity, many later researchers integrate this method to their methods. Moreover, it is very easy to find the source code of this method on Internet.

* Since it does not use any lexical information for the alignment task, it can be used between any pairs of languages. However, in distant languages in which also letters differ, it is not so efficient.

* As a result of its simplicity, its time cost is very low. In other words, it is one of the fastest algorithms for sentence alignment. Thus it is suitable for aligning a very large bilingual corpora.

When we consider Turkish-English, we see that they are close languages in terms of length of sentences. Also there is no widely known bilingual dictionary for using in sentence alignment task. Due to these properties, this method can be applied to Turkish-English alignment effectively.

3.2 Wu: Aligning a Parallel English-Chinese Corpus Statistically with Lexical Criteria

After Gale and Church's method, we examined the method of Wu[2]. This method is important in two respects: Firstly, by applying the Gale's method to Chinese and English it shows that length-based methods give satisfactory results even between unrelated languages which is a surprising result. Next, it shows the effect of adding lexical cues to a length-based method. According to his results, using lexical information increases accuracy of alignment from %86 to %92.

In our case (Turkish-English), using lexical cues can have a similar positive effect since there are many words in Turkish whose root is an English word. But extracting such an information base is a time consuming operation for which we did not have enough time.

3.3 Moore: Fast and Accurate Sentence Alignment of Bilingual Corpora

An example of a more complex sentence alignment algorithm is Moore's algorithm[3]. In this one Moore tries to solve the problem that using lexical information limits the use of algorithm only between a pair of languages. He tries to overcome this problem via using a method similar to IBM translation model for extracting a bilingual corpus with the texts at hand.

This is a very promising method since it is a language-independent and highly accurate algorithm. The only problem of this method is the slowness of the algorithm. Extracting a bilingual corpus from the texts at hand is not a straightforward and cheap operation.

It is also possible to use this method for Turkish. Despite the structure of sentences in Turkish and English are different, existence of cognates (a word and its translation which have similar sounds) makes the task of extracting bilingual corpora between Turkish and English easier.

3.4 Melamed: A Geometric Approach to Mapping Bitext Correspondence

A different approach in sentence alignment is the approach of Melamed[4]. In his method he uses a bitext map of words for marking the points of correspondances between these words in a two-dimensional graph. When all posible points are marked he finds the true correspondance points in graph by following some rules, sentences location information and boundary of sentences. The weakness of his method is that it needs a good bitext map to have

satisfactory accuracy. The power of the method is that if a good bitext map can be formed, it can give almost perfect results in alignment.

It is best to use this method between popular languages in which many researchers study for alignment task and therefore acquiring a good bitext map is possible. For the present, this method is difficult to use in Turkish.

3.5 Sheng: Aligning Bilingual Corpora Using Sentences Location Information

The last method I will mention about is the method of Sheng[5]. This method is also in the group of methods that do not use any lexical information for the main text but only for the sake of getting higher accuracy. The property of this method is that it uses not only the length of sentences but also the length of texts, the length of upper and lower part of the candidate sentences, and some information like that to reinforce the effect of location of sentences in the text. In this sense it can be said that it is a further step of pure length-based method.

This method is also important in the sense that our method is derived from this method. Since it is a simple algorithm to implement and since it does not require an additional tool or data, it is a good candidate for setting up our method on. We have some difference however: While this method does not use paragraph alignment, we use paragraph alignment too, which we think will enable us to get more accurate results. Also we made some differences in the thresholds and formulas it uses for determining the best parameters to be used with Turkish texts.

3.5.1 Detailed Explanation

Below is a detailed explanation of the method of Sheng:

> "In this paper, authors describe a method for aligning bilingual corpora mainly based on the observation that the location of sentence pairs in two languages are distributed in the texts similarly. Authors also omit the paragraph boundary informaion which sometimes when not aligned correctly, bring a problem for alignment process.
>
> First of all, authors divide the alignment process to two steps. In first step, they combine all paragraphs into a large one paragraph.After that, they consider the alignment process as a matching problem in bipartite graph. Then, they do the alignment process by finding and matching alignment anchors. In their model, all 1-1 sentence beads are candidate anchors.

For solving bipartite graph problem and matching 1-1 anchors correctly they follow 3 rules:

1) vertexes in bipartite graph are ordered.
2) Weight of any edges is smaller than a threshold D, and no cross-match can occur.
3) Last sentences of each text is accepted as an anchor.

For a sentence pair the alignment value, P[i,j], is calculated by using the lengths of sentences, total length of texts, the length of the text fragments below and upper part of the sentences. Aim is to minimize the P[i,j] calculated by using the parameters above. The sentences whose alignment function values smaller than a selected threshold are used as alignment anchors.

For improving the accuracy, also a bilingual dictionary is used to calculate the similarity of sentence pairs. Also a method for handling partial alignment errors, (location is correct, but half of the sentence is missing) is used: Again a function is used to check similarity of context-adjacent sentence pairs.

There are two main assumptions algorithm does: It does not consider the cases 1-3, 3-1, 3-2,etc. Since they occur too rarely. In addition, it assumes no sentence in the upper part of an anchor sentence can match with a sentence in the lower portion of the corresponding anchor sentence."

4. DATA COLLECTION

4.1 Resources

In this phase of our senior project, we concentrated on the task of finding bilingual texts. Since our algorithm takes a bilingual text pair and then makes an alignment between these texts, we have to provide many input texts to the program so as to calculate its accuracy and make a detailed comparison in different types of texts with varying properties.

The sources for collecting blingual texts can be summarized as follows:

Internet: In the contemporary world for every purpose, internet offers abundant opportunities. Especially in our case, it is an invaluable resource. We found nearly 90% of our data via Internet. There are 3 kinds of data that we found via Internet:

- *e-books of popular books (novels, stories, politics, etc..):* Especially "Project Gutenberg" which transfers old, popular and classical books to digital environment with the purpose of free access of book readers and some forum sites in which we found the Turkish translations of these e-books were our main e-book resources.
- *articles in some news sites:* Some of the newspapers in Turkiye who wants to be read worldwide, such as Hürriyet and Zaman, keeps an English version of their websites. In these websites they periodically translate the articles of some authors in newspaper to English. These texts are very good sources since they are smaller and thus easier to trace the translation pattern used in the text. In addition, they are also interesting in the sense that they are Turkish-to-English translations (most of remaining bilingual data we found is translation from English-to-Turkish)
- *small passages and abstracts of master thesis*: The passages are some stories or information texts, advertisements that are generally not more than one pages long. We expected master thesis to be a great data source but we could find only some master thesis on webpage of Bogazici University. Still, they are important because they are in the group of technical data sources and it is difficult to find such data.
- *Some religious books:* This group consists of books of a Turkish author who publishes religious books. The main property of this group is that despite they are religious they

cover a wide range of scientific areas such as biology, astronomy, archeology, psychology. Another property of these books is that they are Turkish-to-English translations. We did not use them in the testing phase of the project but they are suitable for such a purpose.

Bureaus of Translation: Translation bureaus are the formal resources for bilingual texts. We searched for a good bureau and they agreed to give documents to us. However, these documents were generally private or official documents containing business contracts. Therefore, they warned us to use them only after cleaning all private information (company names, person names, amount of money, etc..) in the documents. Since this is a time-consuming and hard operation and since it can cause problems we did not use these documents.

Department of Translation and Interpreting Studies in Bogazici University: In this department there are many translations which are made by professors, assistants and students. However, there is not a common database for these translation texts in digital medium. Because of this, this resource was no beneficial for our project.

4.2 Organization of Resources

Having a big resource pool requires a good classification and ordering of the data for finding what you are looking for easily. It also makes easier the use of resources by others who have no relation with the project. For realizing these goals we kept the identities of the bilingual texts in a table. In Fig. 1, there is a sample entry in the table which reveals the structure of an entry and below it we explain the purpose of each column. At the end of the document we give the full table (Table 1) that contains records for all the resources we have found so far.

ID	NAME	CATEGORY	TYPE	SUB-TYPE	# OF WORDS	LENGTH	FORMAT	QUALITY	SOURCE	ADDITIONAL INFORMATION

Fig 1. Names of the fields in the table of found resources.

B022	Mark Twain - Tom Sawyer	kitap	roman	macera	71.000	139 sayfa	doc, lit	very good	www.gutenberg.org	1 lit EN, 1 doc TR

Fig 2. A sample text description entry in the "Resources" table.

- **ID**: each bilingual pair has a unique ID. The first character shows the type In the name of the files these IDs are used to make the ordering of files easier. Ex: B022
- **Name**: This field is the official name of the resource. For example if it is a book, it contains the name of the book + the name of author of the book.
  Ex: Mark Twain - Tom Sawyer
- **Category:** This field indicates the type of the resource. It can be a book, a short text, a scientific paper or a news article, etc.. Category of a resource is also indicated in the first letter of the ID field.
- **Type & Sub-type :** These 2 fields are used to categorize the resource further. They are essential fields because "book"is a very wide concept. It matters for the alignment task if it is a novel, a story, a course book, etc..
  Ex: category -> book, type -> novel, sub_type ->fiction
- **# of Words:** For determining the length of a resource we need a criterion. Number of pages does not reveal the real length of a document since the font and format may differ in different texts. Moreover, some texts have figures and plentiful blank lines which makes the text seem longer in terms of number of pages. Therefore, we show the length of a document in terms of the # of words it contains.
- **Length:** This field shows the length of a document in terms of pages. Initially, we only used this field to determine the length of a document. But due to its ambiguity, we added the word count as main criterion for determining the length of a document.
- **Format:** This field shows the available text formats of a document we have found. It is essential since some resources have more than 1 available format such as .doc, .pdf, .txt, etc..
- **Quality:** In fact it is diificult to determine the translation quality of a text without fully reading and examining the bilingual document pair. But at least some differentiation is

needed. Thus, we shortly examined the resources and made a basic grading on their quality.

- **Source:** This field is used to show where we have found the resource from. If it is found from internet, we give the address of the website. If it is not from internet, we describe the agent that helped us to find this document.
- **Additional Information:** In this field we give a detailed information about the formats of the resources and state if there is a problem with the text that makes its usage more difficult.

5. PROGRAM DESIGN

5.1 Overview

In this project our aim is to design a sentence alignment algorithm for using with bilingual texts in which one of the texts will be Turkish. We used Java programming language to code this project. We used Java because:

- It is platform independent
- It has efficient and enough String manipulations in it.
- It is in worldwide use so we can take help from internet easily.
- The components (external resources) we used are usually implemented in Java.

5.1.1 Input

In this project our program will align the sentences of two different languages.That means the program will get two texts. Our program is taking these inputs from files of texts. There are two files; source and target. Source and target words are used here to distinguish the textes. It has to be known that there is no data transportation or any other transportation between the files.

The files are read and then the context of the file is splitted to paragraphs, in paragraph splitting class. The result of the splitting is assigned to an ArrayList. Then the resulting ArrayList is returned to our main part of the project. The process in the main part will be explained in a more focus manner later().

Our only input is source and target files(Turkish and English bilingual text).

5.1.2 Output

In the project our aim is to show the aligning results. For this purpose there is a class *align_record* which holds the data of an aligning. In program there is an array of *alignings* that holds all alignings as a result. Program writes the aligning pairs by using this array in the *show_result()* function.

The sample output is as follows:

Source	Target
1-1	1-1
2-2	2-2
3-3	3-4
---	5-5
4-6	6-7
...	....

This output means:

- The range in the source from 1st sentence to 1st sentence (that means only first sentence) is aligned to 1st sentence in target.(1-1)
- The 2nd sentence in source is aligned to the 2nd sentence in target.(1-1)
- The third sentence in source is aligned to the range from 3rd sentence to 4th sentence in target text. That means 3rd and 4th sentences in target.(1-2)
- There is no sentence in source text to align the 5th sentence of target text.(0-1)
- The range in source from 4th to 6th sentences (4th,5th and 6th sentences) is aligned to the range from 6th to 7th sentences(6th and 7th sentences) in the target text. (3-2)

Also as a result text , there will be an output text.This file contains the sentences in it.Its inside will be in the following manner.

<*pair1's ranges in the source and target*><enter>
<*Pair1's source sentence*><*enter*>
<*tab*><*tab*><*tab*><*Pair1's target sentence*><*enter*><*enter*><*enter*>

<*pair2's ranges in the source and target*><enter>
<*Pair2's source sentence*><*enter*>
<*tab*><*tab*><*tab*><*Pair2's target sentence*><*enter*><*enter*><*enter*>
.....

5.1.3 External Resources

In the project the main problem was to split the sentences in the texts properly. Also splititng to paragraphs was a problem but not as big as sentence splitting. In order to achieve the sentence splitting problem we made research via the Internet.We gained few open source programs for sentence splitting[9][10][11][12].Also we gained a good performanced program but we could not arrive the source of it. For the sake of simplicity we choosed the LingPipe's algorithm[9].Also in this algorithm there are some problems with splitting. But there is no perfect algorithm in this area. So we used the algrithm in our program. We modified the class of *SentenceBoundaryDemo_*. In it we code a *split(String)* function. Which tkes a String as whole text to split and return an ArrayList of the result.

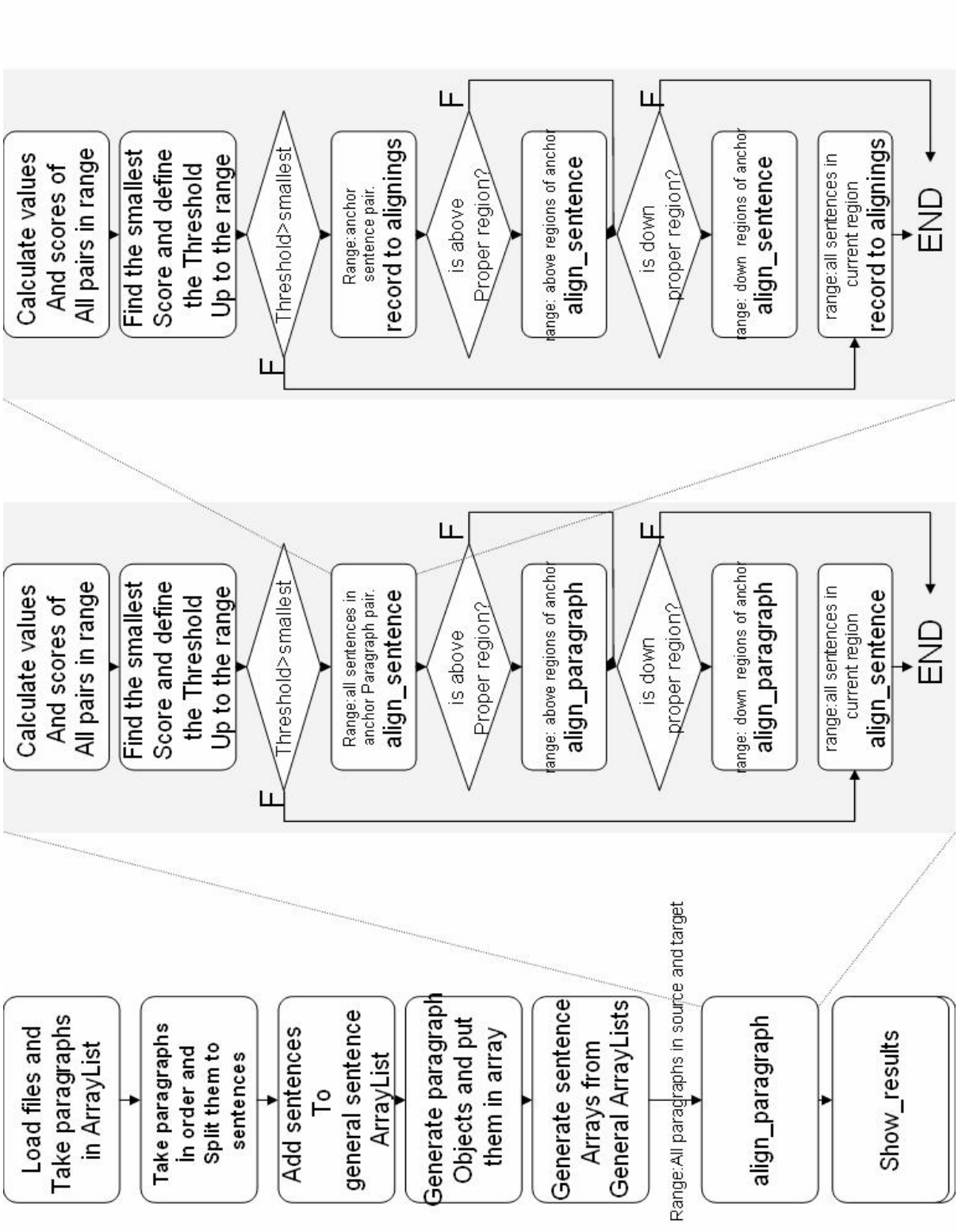
As well as sentence splitting, we decided to use an external resource for paragraph splitting. But we could not find a ready-to-use tool with source code for paragraph splitting. Thus we wrote our own algorithm for this process.

To determine the paragraph boundaries, best criterion is the existence of an "Enter" character. However, in some texts there are "enter" characters at the end of each line. Thus, we also checked if the last character before the "Enter" is a stop punctuation mark like '!', '.' or '?'. We also checked whether the letter following the "Enter" is a capital letter (A-Z). When we did so, new problems arise. There were various paragraph endings such as incomplete sentences, punctuation errors, etc...

As a result, we simply omitted these texts and splitted paragraphs according to only "Enter" character which gave best correctness in correctly-written texts.

5.2 Algorithm

5.2.1 Flow-Chart



5.2.2 Summary of General Principle

1) Load the bilingual text

2) Take paragraphs in arraylist (for source and target)

3) Split the source paragraphs into sentences in order and create paragraphs array

4) Add the splitted sentences to general source sentence arraylist

5) Do the 3 and 4 for target text.

6) Generate *alignings* array in type *align_record*.

7) Generate the sentence array of source text by using general arraylist

8) Do the 7 for target text.

9) Call align paragraph –whole texts at first call-

10) Calculate scores, Find smallest, compare with Threshold

11) If smaller than Threshold 12 else 15

12) Call 16 with anchor point paragraphs' sentence ranges

13) 9 with above of the anchor(if above is proper region)

14) 9 with down of the anchor(if down is proper region) -return called step-

15) Call 16 with the sentence range of all current working region.-return called step-

16) Sentence alignment starts

17) Calculate scores, Find smallest, compare with Threshold

18) If smaller than Threshold 19 else 22

19) Record the alignment to *alignings* array

20) 16 with above of the anchor(if above is proper region)

21) 16 with down of the anchor(if down is proper region) -return called step-

22) Record all regions as alignment. –return to called step-

23) Show results.

5.2.3 General Principle

In our method, we first make the paragraph alignment by using the same algorithm as the paper. We used the scoring technique. We calculate a score between all combinations of paragraph pairs.

5.2.3.1 Scoring:

While calculating the scores between paragraph pairs we use the length values as character count since the number of words between two texts does not give the precise results especially between Turkish-English texts. So we selected the character numbers as length parameter.

The following parameters are used while calculating the scores between the paragraph pairs:

- Whole text lengths:(L_s (source text) ,L_t (target text))
- Length of sentences:(Ls_i the i-th sentence of source , Lt_i i-th sentence of target)
- Upper context lengths of sentences: (Us_i ,Ut_i)
- Nether context lengths of sentences: (Ds_i,Dt_i)
- Whole text length ratio:(P0=Ls/Lt)
- Upper context length ratio:($Pu[i,j]=Us_i/Ut_j$)
- Nether context length ratio: ($Pd[i,j]=Ds_i/Dt_j$)
- Sentence length ratio: ($Pl[i,j]=Ls_i/Lt_j$)
- Weight coefficient : ($Alpha =(Ls/Ls_i+Lt/Lt_j)/2$)

By using these parameters score is calculated in the following manner:

à $P[i,j]=Alpha*(Pu[i,j]-P0)^2+(Pl[i,j]-P0)^2+Alpha*(Pd[i,j]-P0)^2$ ß

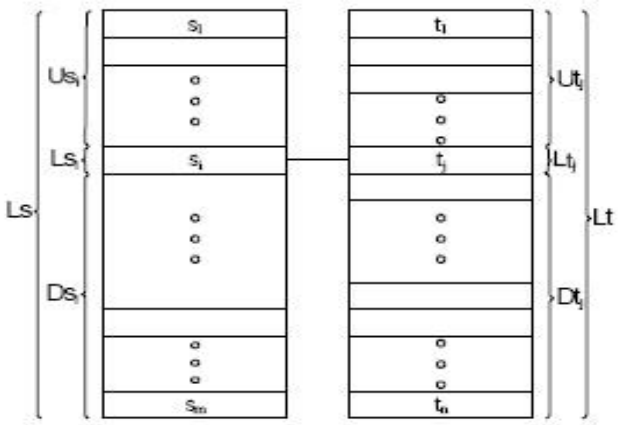


Figure 3.

If the score is less than a threshold defined by us then by selecting the smallest pair, aligning will be done. After aligning, the upper and the nether part of the aligned paragraphs will be considered as whole texts and the same procedure above will be executed for them. If none of the scores between pairs in the text is smaller than threshold then whole texts will be aligned.

5.2.3.2 Initializing Arrays:

In our method program holds the sentence and paragraph attributes in objects of *sentence* and *paragraph*. Four arrays are generated from the objects of these classes: *sentences of source text, sentences of target text, paragraphs of source text* and *paragraphs of target text*.

In a paragraph object program holds the length of the paragraph. Object holds this value by calculating it once. Also the number of sentences in the paragraph is hold in the object. This attribute is not used for now but it may be needed. The most important values that the paragraph object holds are *start_index* and *end_index* values. These are the indexes that hold the place of the paragraph in the sentence array. For example if these values are *start_index=5* and *end_index=11* , that means the paragraphs first sentence is fifth sentence of the text. And the last sentence of the paragraph is 11^{th} sentence of the text.

In a sentence object, the value of the sentence is hold. Also the lengths of the sentence and the upper and down lengths of sentence attributes are present.

At first our program reads the files of source and target texts, then the *Paragraph_splitter.split* function returns the ArrayList of the splitted paragraphs for both source and target texts. I added one more free string in front of (0^{th} index) these ArrayLists to study on

the real order of paragraphs. Then program generates new *paragraph* type arrays (for source and target) in the size of ArrayLists.

After that respectively all paragraphs in ArrayList are (in a *for* loop) called. Before the loop we define a start and end index values to know the total number of sentences. First we initialize the start and end index values to 0.

In the loop, first, program sends the value of paragraph to sentence splitter function which returns an ArrayList of splitted sentences. Then by taking the size of the splitted sentence ArrayList , program defines the number of sentences in it. And by using number of sentences and the start and end index values defined before the loop program defines the *start_index* and *end_index* of the paragraph. Then program generates an object of *paragraph* for this paragraph by placing it into the paragraph array. Then program adds all sentences captured from the paragraph to general ArrayList of sentences respectively.

After this loop, program generated an array of paragraph in *paragraph* type. And program adds all sentences of the text to a general ArrayList. The same loop procedure runs for both source and target texts.

After the loops, program generates array of sentences in *sentence* type in the size of the general ArrayLists of the sentences. After the arrays are sized , in loops all sentences in the ArrayList are used to generate objects of *sentence* and add them to sentence arrays.

After all these procedure now our program has for arrays:

- *source_sentence* à type :*sentence* sentences of source text
- *source_paragraph* à type : *paragraph* paragraphs of source text
- *target_sentence* à type :*sentence* sentences of target text
- *target_paragraph* à type : *paragraph* paragraphs of target text

5.2.3.3 Paragraph and Sentence Aligning Procedure:

Program can start to aligning. The aligning functions take four parameters:

- starting point in the source
- end point in the source
- start point in target
- end point in the target

These four points defines the working region of the function. There are two aligning functions (one for sentence and one for paragraph) and both has same principle. Both work on recursive strategy.

First of all, function calculates the total lengths of source and target paragraphs in the range. Then it calculates the upper and down lengths af all paragraphs in the region. For every paragraph object, the *upper_length* and *down_length* values are changes up to the working region. The calculation of these values for every paragraph will be explained latter in a more focus manner.

In paragraph alignment function takes the range in source text and the range in target text (paragraph numbers). Then calculates score between all pairs of source and target paragraphs (In the double for loop). In this loop also holds the smallest score and the correspondent pair in the smallest score (holds the indexes of paragraphs which gave the smallest score).

Then function calculates the average number of paragraphs in the working region. Up to the value of working region program defines a Threshold value. This is made because in our method smaller the score more precise the aligning of that pair. If working region is very big we assign the Threshold value very big and we do not want any required exactness. We want the the most exact even if it is not very exact. So we can escape from some type of aligning (5-5, 4-6 etc…). But in small working regions such as smaller than 4 paragraphs we want an exact aligning , if it could not find such an exact score so program aligns whole region(maximum 3-3,4-2,1-5 etc…).

After defining Threshold up to the average number of paragraphs in region (*working_region*) program looks whether the smallest value is smaller than the Threshold. If it is smaller than the threshold so the correspondent pair (for smallest score) is an anchor point. Then program calls the *sentence_alignment* function by defining the range of sentences to be aligned as:

- *source_start* à the start index of the source paragraph of the anchor point
- *source_end*à the end index of the source paragraph of the anchor point
- *target_start* à the start index of the target paragraph of the anchor point
- *target_end*à the end index of the target paragraph of the anchor point

Sentence alignment of the aligned paragraphs is made immediately.

After calling sentence alignment function calls itself twice but by condition. If there is one or more paragraphs above the anchor point in both source and target calls itself by new range:

- *source_start*à old source_start
- source_endà index of the *source_paragraph* of anchor point MINUS 1.
- *target_start*à old target_start
- target_endà index of the *target_paragraph* of anchor point MINUS 1.

If there is one or more paragraphs under the anchor point in both source and target calls itself by new range:

- *source_start*à index of the *source_paragraph* of anchor point PLUS 1.
- *source_end*à old source_end
- *target_start*à index of the *target_paragraph* of anchor point PLUS 1.
- *target_end*à old target_end

These explained procedure was if we gained a proper smallest value. But if the smallest value is bigger than *threshold*, then function calls sentence alignment by these sentence ranges:

- source_start à the start index of the source paragraph of the anchor point
- source_endà the end index of the source paragraph of the anchor point
- target_start à the start index of the target paragraph of the anchor point
- target_endà the end index of the target paragraph of the anchor point

That means , all working region is sent to sentence alignment.

5.2.3.4 Differences in Sentence Aligning Procedure:

The same procedure is run by the program for sentence alignment. But there are small changes in the aligning part. In paragraph alignment paragraphs are being aligned (ranges indexes –source_start, source_end etc...- are related to paragraph array) but in sentence alignment the sentences are aligned and also the range indexes are related to sentence array indexes. One more difference between sentence alignment and the paragraph alignment functions is , the discovered anchor points in paragraph alignment were not recorded. The sentence range

of those paragraphs was sent to sentence_alignment immediately. But in sentence alignment when an anchor point is discovered or the whole region is decided to be aligned it will be recorded in the *alignings* array.

5.2.3.5 alignings Array

The alignings array is the array of objects of align_rcord class. In one of these objects an alignment is recorded. It is hold in the following way(with for integers):

alignment x à x (the index of first source sentence of alignment x),
the index of last source sentence of alignment x,
the index of first target sentence of alignment x,
the index of last target sentence of alignment x.
where x= the index of first source sentence of alignment x.

That means program places the objects in the array up to the index of first source sentence of alignment. By doing this we loss some space but we gain a sorted alignings array. Since we run the procedure in recursive way the order of the discovering of alignings is not in order of text.

5.2.3.6 Calculation of Upper and Down Lengths of All Paragraphs

- Upper Length

Also the calculation of upper and down lengths of every paragraph in the region for every recursion is very important. To calculate the upper length of all paragraphs in the region there is a loop. Before the loop the upper length of the first paragraph is assigned 1.(In fact it must be 0 but in calculations there were some divide by 0 problems so we overcome this problem by assigning 1 to upper length of firs paragraph.) . Then in a loop a paragraph's upper length is assigned as the sum of the length of previous paragraph and the upper length of the previous paragraph. So after loop every paragraph's upper length is assigned.

- Down Length

Same principle is used to calculate the down lengths of paragraphs. Initially the downlength of the last paragraph is assigned to 1. By going back in the array calculate the down lengths of paragraphs. The down length of a paragraph is sum of the length of next paragraph and the down length of the next paragraph.

Program does these procedures for the part of both source and target arrays which is in our range.

In sentence alignings same principle is used to calculate the upper and down lengths of sentences.

5.2.3.7 Showing the Results

After alignings the next step is to show the results which are recorded in the alignings array. Also we record array in sorted way. In a loop we look all the aligning records. But first there are two variables: source_next , target_next. These values show us the expected sentence to see next. Since we will start to write from first sentences these variables are initialized to 1 before the loop.

In the loop, program takes all aligning records respectively. It looks first whether the present index is null or not. If it is not null then looks at the source and target start indexes of the record. If both are same as we expected, writes the related sentences to file and writes the aligning as mentioned above in 'OUTPUT' part. Then assigning is made to variables source_next and target_next. Source_next become the source_end of aligning record plus 1. Target_next become target_end +1.

If expected sentence number is not arrived yet in any of source or target part, then program writes 0-1 sentence beads until arriving the expected sentence. Then by decrementing the overall index of the loop, program looks the present align_record again.

After loop, program writes the remaining sentences of source or target if there is sentence beads like 0-1 or 1-0 .

5.2.4 Functions, Variables, Classes

5.2.4.1 Functions

Initializing arrays(): Splits the text to paragraphs and sentences and record them in arrays to use in program.

align_paragraph(int int int int): Takes four integers which define the region of activation of function and calculates score between all pairs in region. Also defining the smallest score and the correspondent pair that generated smallest score sends this pair to sentence alignment and sends upper and down parts of the anchor point to paragraph alignment (itself).

align_sentence(int int int int): It works in the same principle with align_paragraph but on sentences. There is a difference that the discovered anchor points are recorded at alignings array and sends the upper and down part of the region to itself.

Calculate_paragraph_values(int int int int): Calculates the upper and down lengths of paragraphs in the range.

Calculate_sentence_values(int int int int): Calculates the upper and down lengths of sentences in the range.

Show_results(): Regulates the results and prints them to console and writes to resulting file.

ParagraphSplitter.split(String String): Splits the text in the directory given by using String, to paragraphs and writes them to other file whose directory is also given as a String. Also returns the splitting results as ArrrayList.

SentenceBoundaryDemo.split(String): Takes the text as input and returns the result of splitting as an ArrayList.

5.2.4.2 Classes

Sentence:

It is for sentences of text to hold them in an efficient manner. It holds the length (without spaces) of sentence, sentence as String, and upper and down lengths of sentence.

Paragraph:

It is for paragraphs of text to hold them in an efficient manner. It holds the length of paragraph, the upper and down lengths of paragraph, the start and end indexes of paragraph and the number of sentences in the paragraph.

The start_index and end_index attributes of the paragraph object are the orders of the starting and ending sentence of paragrap in the whole text.

Align_record:

It holds the alignment records. It holds data with four integers which determine the ranges in two texts (source and target).

5.2.4.3 Variables

Source_sentence: It is an array of whole sentences in the source text. Its type is *sentence*.

***Target_sentence*:** It is an array of whole sentences in the target text. Its type is *sentence*.

Source_paragraph: It is an array of whole paragraphs in the source text. Its type is *paragraph*.

Target_paragraph: It is an array of whole paragraphs in the target text. Its type is *paragraph*.

Alignings: It is an array of alignings which are discovered. Its type is *align_record*.

5.2.5 Pseudocode

```
main()
begin
        initialize_Arrays()
        align_paragraph(1,length_of_source_paragraph_array
                        1,length_of_target_paragraph_array )
        //here the align paragraph function is ignitor of the recursion
        show_results();
end

initialize_Arrays()
begin
        source_paragraph_arraylist=split_paragraph of source ;
        target_paragraph_arraylist=split_paragraph of target ;
        create arrays of paragraphs in paragraph type

        next_start=1,next_end=1.
        for(i=1;i<number_of_paragraphs;i++)
        begin
                temp_list=sentencesplitter(source_paragraph(i))
                next_end=next_start+number_of_sentences(size_of_temp_lisst)
        source_parapgrph[i]=new paragrph(next_start,next_end,source_pargra(i))

                a=0
                for(i=next_start to next_start+number_sentences)
                        general_sentence_source(i)=templist(a++)
        end

        next_start=1,next_end=1.
        for(i=1;i<number_of_paragraphs;i++)
        begin
                temp_list=sentencesplitter(target_paragraph(i))
                next_end=next_start+number_of_sentences(size_of_temp_lisst)
        target_parapgrph[i]=new paragrph(next_start,next_end,source_pargra(i))

                a=0
                for(i=next_start to next_start+number_sentences)
                        general_sentence_ target(i)=templist(a++)
        end
        for(i=0 to general_sentence_source.length)
                source_sentence[i]=new sentence(general_sentence_source(i))

        for(i=0 to general_sentence_target.length)
                target_sentence[i]=new sentence(general_sentence_target(i))
end
```

```
align_paragraph(source_start,source_end,target_start,target_end)
begin
        for(i=source_start to source_end)
                Ls=Ls+ source_paragraph[i].length
        for(i=target_start to target_end)
                Ls=Ls+ target_paragraph[i].length

        calculate_values_paragraphs(source_start, source_end,target_start,target_end)
        for(i=source_start to source_end)
           for(j=target_start to target_end)
        begin
                P[i,j]=Alpha*(Pu[i,j]-P0)^2+( Pl[i,j]-P0)^2+Alpha*( Pd[i,j]-P0)^2
                if(P[i,j] < smallest)
                        smallest=P[i,j].   smallest_i=i.  smallest_j=j.
        end
        working_region=((source_end-source_start)+(target_end-target_start))/2
        if(working_region > 10)
                Threshold=Infinite.
        else if(working_region >4)
                Threshold=3
        else
                Threshold=1

        if(smallest < Threshold)
        begin
                align_sentences(source_paragraph[small_i].start_index,
                                source_paragraph[small_i].end_index,
                                target_paragraph[small_i].start_index,
                                target_paragraph[small_i].end_index)
                if(there is one or more paragraph above the small i'th paragraph at source
                and there is one or more pargraph above the small j'th paragraph at target)
                begin
                        align_paragraph(source_start,small_i-1,target_start,small_j-1)
                end
                if(there is one or more paragraph under the small i'th paragraph at source
                and there is one or more pargraph under the small j'th paragraph at target)
                begin
                        align_paragraph(small_i+1, source_end,small_j+1,target_end)
                end
        end
        else
        begin
                align_sentence(source_paragraph[source_start].start_index,
                                source_paragraph[source_end].end_index,
                              target_paragraph[target_start].start_index,
                                target_paragraph[target_end].end_index)
        end
end
```

```
align_sentence(source_start,source_end,target_start,target_end)
begin
        for(i=source_start to source_end)
                Ls=Ls+ source_sentence[i].length
        for(i=target_start to target_end)
                Ls=Ls+ target_sentence[i].length

        calculate_values_sentence(source_start, source_end,target_start,target_end)
        for(i=source_start to source_end)
           for(j=target_start to target_end)
        begin
                P[i,j]=Alpha*(Pu[i,j]-P0)^2+( Pl[i,j]-P0)^2+Alpha*( Pd[i,j]-P0)^2
                if(P[i,j] < smallest)
                        smallest=P[i,j].   smallest_i=i.  smallest_j=j.
        end
        working_region=((source_end-source_start)+(target_end-target_start))/2
        if(working_region > 10)
                Threshold=Infinite.
        else if(working_region >4)
                Threshold=3
        else
                Threshold=1

        if(smallest < Threshold)
        begin
                alignings[small_i]=new align_record(small_i,small_i,small_j,small_j)
                if(there is one or more paragraph above the small i'th paragraph at source
                and there is one or more pargraph above the small j'th paragraph at target)
                begin
                        align_sentence(source_start,small_i-1,target_start,small_j-1)
                end
                if(there is one or more paragraph under the small i'th paragraph at source
                and there is one or more pargraph under the small j'th paragraph at target)
                begin
                        align_sentence(small_i+1, source_end,small_j+1,target_end)
                end
        end
        else
        begin
                 alignings[source_start]=new align_record(source_start,source_end,
                                                          target_start,target_end)
        end
end

calculate_values_paragraphs(source_start, source_end,target_start,target_end)
begin
        source_paragraphs[source_start].upper_length = 1
        for(i=source_start+1 to source_end)
                source_paragraphs[i].upper = source_paragraph[i-1].upper+
                                                        source_paragraph[i-1].length
        target_paragraphs[target_start].upper_length = 1
        for(i=target_start+1 to target_end)
                target_paragraphs[i].upper = target_paragraph[i-1].upper+
                                                        target_paragraph[i-1].length
```

```
        source_paragraphs[source_start].down_length = 1
        for(i=source_end-1 to source_start)
                source_paragraphs[i].down = source_paragraph[i+1].down+
                                                        source_paragraph[i+1].length
        target_paragraphs[target_start].down_length = 1
        for(i=target_start+1 to target_end)
                target_paragraphs[i].down = target_paragraph[i-1].down+
                                                        target_paragraph[i-1].length
end
calculate_values_sentences(source_start, source_end,target_start,target_end)
begin
        source_sentences[source_start].upper_length = 1
        for(i=source_start+1 to source_end)
                source_sentences[i].upper = source_sentences[i-1].upper+
                                                        source_ sentences [i-1].length
        target_ sentences [target_start].upper_length = 1
        for(i=target_start+1 to target_end)
                target_ sentences [i].upper = target_ sentences [i-1].upper+
                                                        target_ sentences [i-1].length

        source_ sentences [source_start].down_length = 1
        for(i=source_end-1 to source_start)
                source_ sentences [i].down = source_ sentences [i+1].down+
                                                        source_ sentences [i+1].length
        target_ sentences [target_start].down_length = 1
        for(i=target_start+1 to target_end)
                target_ sentences [i].down = target_ sentences [i-1].down+
                                                        target_ sentences [i-1].length
end
show_results()
begin
        source_next=1,target_next=1 //the expected order of sentence we will have
        for(i=1 to number of sentences)
        begin
            if (alignings[i]!= null)
            begin
                if(starts of aligning in source and target are as we want)
                        write the results to console and to file
                        set new expected number of sentences as (..._end + 1)
                else if (source is expected but target is not)
                        write 0-1 result to console and file until the target order is expected
                        set i to i-1 because we want program to scan this aligning
                else if (target is expected but source is not)
                        write 1-0 result to console and file until the source order is expected
                        set i to i-1 because we want program to scan this aligning
           end
        end
        if (source_next< number of sentences in source)
                write left alignings as 1-0
        if (target_next<number of )sentences in target
                write left alignings as 0-1
end
```

6. EXPERIMENTS

In literature survey we have seen that most researchers on sentence alignment, especially if bilingual texts are French, German, English or Chinese, use hansards of these countries for a reliable common bilingual database. But no such hansard exist in Turkish-English bilingual texts. Thus we used other data sources for experiment. This situation makes the comparison of the accuracy of our method with other alignment methods.

The proposed method described in Section 5 is tested on 3 different data:

1) The philosophy book of Henry Thoreau called "On the Duty of Civil Disobedience"

2) Combination of some articles from a news site (high paragraph fragmentation)

3) Combination of some articles from a news site (less paragraph fragmentation)

1) Data 1 was a text containing large paragraphs in both languages and having somewhat similar paragraph counts. But it was a hard text when we consider the sentence alignment beads. The percentage of 1-1 beads was only 65.2% and the parcentage of 1-2 or 2-1 beads was 22.3%. The remaining 12.5% alignment pairs consisted of more complex beads even containing 1-6, 1-5 or 2-5 sentence beads. It also contained a deleted region of 18 sentences long in English text which is hard to handle.

Under these situations it did 63% of alignments correctly and 24% were complete errors. The remaining 13% was partial errors in which alignment is partially correct. For example, the real bead is a 1-2 bead but our program splits it into two beads: a 1-1 and a 0-1. These are called partial errors. By changing parameters we can avoid these errors up to some extent.

Another important point is the question of how much the deleted block affected overall performance. The 18-sentence long deleted segment was towards the end of the text. For a short period it caused program to give continuous wrong alignments. But it managed to overcome this situation after some paragraphs. If we exclude this continuous segment, the accuracy increases to 73.7% which is very good for such a difficult text.

2) In the experiment on data 2, we had very bad results. Because, the paragraph alignment phase made many errors since there were a lot of 1-6, 1-5, etc.. paragraph beads. When the program failed in paragraph alignment, it inevitably made errors in sentence alignment in large blocks. Due to this problem, it had an accuracy lower than 45% for data 2.

3) Finally, in the experiment on data 3, again we used a data similar to data 2. But this time the paragraphs aligned mostly 1-1 and also they were long paragraphs. In the sentence level, again 1-1 bead percentage was high (about 90%). Under these values, it gave a very good accuracy. The percentage of true alignments was 96.1% and 2.2% was partial alignment errors. Only 1.7% of all alignments was completely wrong.

7. EVALUATION OF RESULTS AND FUTURE WORK

The results of the experiments reveal some deficiencies and advantages of our program:

First of all, the results reveal the importance and effect of paragraph alignment. If paragraphs are well-arranged in both bilingual texts, paragraph alignment is advantageous and increase the accuracy of the alignment remarkably. So it is better to use this program for texts having well-arranged paragraphs. In the future, it can be studied on paragraph alignment to increase its robustness for using in any text.

Secondly, there is the problem of deleted blocks. Since our program works location-based, it takes some period to recover after a deleted segment. If we manage to shorten the length of this recovery period, deleted segments will not be a problem for us anymore. Lexical information may be used for solving this problem. In fact, using lexical information is the most important improvement on our algorithm which can increase the accuracy rates notably.

Finally, values of the parameters may be modified for determining the best values. This is the simplest improvement but it requires too much time to check the effects of variations in parameter values since checking has to be done manually. We could have done this, but lack of time for doing this prevented us to calculate the best values of parameters.

8. REFERENCES

Reference Papers

1. Gale, W.A. and Church, K.W.: A program for aligning sentences in bilingual corpora, In Proc. of the 29th Annual Meeting of the ACL (1991) 177-184.
2. Wu, D.: Aligning a Parallel English-Chinese Corpus Statistically with Lexical Criteria. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, Las Cruces, New Mexico (1994) 80–87
3. Robert C. Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In S. Richardson (ed.), *Machine Translation: From Research to Real Users* (Proceedings, 5th Conference of the Association for Machine Translation in the Americas, Tiburon, California), pp. 135–244, Springer-Verlag, Heidelberg, Germany.
4. Melamed, I.D.: A Geometric Approach to Mapping Bitext Correspondence. IRCS Technical Report 96-22, University of Pennsylvania (1996)
5. *Weigang Li, Ting Liu, Zhen Wang and Sheng Li*: Aligning Bilingual Corpora Using Sentences Location Information, Proceedings of 3rd ACL SIGHAN Workshop, 141-147, (1994)
6. Chen, S.F.: Aligning Sentences in Bilingual Corpora Using Lexical Information, In Proc. of 30th Annual Meeting of ACL (1993) 9-16.
7. Kay, M. and Röscheisen, M: Text-Translation Alignment, Computational Linguistics 19:1 (1994) 121-142.
8. Simard, M., Plamondon, P.: Bilingual Sentence Alignment: Balancing Robustness and Accuracy. Machine Translation **13(1)** (1998) 59–80

Online References

9) LingPipe Sentence Splitter
http://www.alias-i.com/lingpipe/demos/tutorial/sentences/read-me.html
10) GATE Framework
http://gate.ac.uk/
11) BALIE splitter
http://balie.sourceforge.net/
12) LTG software
http://www.ltg.ed.ac.uk/software/pos/
13) Project Gutenberg
http://www.gutenberg.org/
14) Scientific Paper Search Engine
http://www.scirus.com/srsapp/

APPENDIX A: RESOURCES TABLE

ID	NAME	CATEGORY	TYPE	SUB-TYPE	# OF WORDS	LENGTH	FORMAT	QUALITY	SOURCE	ADDITIONAL INFORMATION
B001	harry potter-felsefe taşı	kitap	roman	fantastik	56,000	170 sayfa	txt,doc,pdf	good	http://www.altkitap.com/arsiv.asp	1 pdf EN, pdf+txt+doc TR
B002	harry potter-sırlar odası	kitap	roman	fantastik	67,000	189 sayfa	txt,doc,pdf	good	http://www.altkitap.com/arsiv.asp	1 pdf EN, pdf+txt+doc TR
B003	harry potter-azkaban tutsağı	kitap	roman	fantastik	84,600	178 sayfa	txt,doc,pdf	good	http://www.altkitap.com/arsiv.asp	1 pdf EN, pdf+txt+doc TR
B004	harry potter-ateş kadehi	kitap	roman	fantastik	150,000	302 sayfa	txt,doc,pdf	good	http://www.altkitap.com/arsiv.asp	1 pdf EN, pdf+txt+doc TR
B005	harry potter-zümrüdüanka yoldaşlığı	kitap	roman	fantastik	200,000	418 sayfa	doc, pdf	good	http://www.altkitap.com/arsiv.asp	1 pdf EN, pdf+doc TR--sayfa nolari var
B006	yüzüklerin efendisi-yüzük kardeşliği	kitap	roman	fantastik	142,000	450 sayfa	txt,doc,pdf , lit	good	http://www.e-kutuphane.net/	1 pdf EN, txt+doc+lit TR
B007	yüzüklerin efendisi-iki kule	kitap	roman	fantastik	119,000	380 sayfa	txt,doc,pdf , lit	good	http://www.e-kutuphane.net/	1 pdf EN, txt+doc+lit TR--bazı yerde bölümler yarim
B008	yüzüklerin efendisi-kralın dönüşü	kitap	roman	fantastik	106,000	310 sayfa	txt,doc,pdf , lit	good	http://www.e-kutuphane.net/	1 pdf EN, txt+doc+lit TR
B009	1984 by George Orwell	kitap	roman	bilimkurgu	65,000	220 sayfa	txt	good	http://ekitap.kolayweb.com/	1 txt EN, 1 txt TR
B010	Macbeth by Shakespeare	kitap	oyun	trajedi	18,200	32 sayfa	txt, html, pdf	adequate	http://ekitap.kolayweb.com/	1 pdf EN, html+txt TR ----DİYALOG SEKLİNDE..
B011	Pet sematary by Stephen King	kitap	roman	korku	87,000	142 sayfa	txt, pdf	good	http://www.kitap.perisi.com/	1 txt EN, 1 txt+pdf TR
B012	Da vinci Şifresi	kitap	roman	polisiye	77,200	295 sayfa	pdf	good	http://www.kitap.perisi.com/	1 pdf EN, 1 pdf TR
B013	Descartes- Discourse on method	kitap	felsefe		24,700	47 sayfa	lit, txt	good	http://www.e-kutuphane.net/	1 lit EN, 2 txt TR
B014	Bacon - New Atlantis	kitap	felsefe	politik	13,000	33 sayfa	lit, txt	good	http://www.e-kutuphane.net/	1 lit EN, 2 txt TR
B015	Plato - Statesman	kitap	felsefe	politik	18,700	108 sayfa	lit, txt	good	http://www.kitap.perisi.com/	1 lit EN, 2 txt TR -------diyalog seklinde...
B016	Tommaso Campanells - City of Sun	kitap	felsefe	politik	23,700	40 sayfa	lit, txt	good	http://www.kitap.perisi.com/	1 lit EN, 2 txt TR felsefe
B017	Dostoyevski - Yeraltindan Notlar	kitap	roman	felsefik	30,000	98 sayfa	lit, txt	good	www.freeELiterature.com	1 lit EN, 1 txt TR
B018	Thoreau- Haksız Yonetime Karsi	kitap	felsefe	politik	8,300	21 sayfa	lit, doc	good	www.freeELiterature.com	1 lit EN, 1 doc TR
B019	Tolstoy - Anna Karenina	kitap	roman	dram	351,000	883 sayfa	lit	good	www.freeELiterature.com	1 lit EN, 2 lit TR(cilt1-cilt2 seklinde)
B020	Aristoteles - Atinalilarin Devleti	kitap	felsefe	politik	24,400	43 sayfa	doc, txt	good	http://www.kitap.perisi.com/	1 doc EN, 1 txt TR
B021	Plato - Republic	kitap	felsefe	politik	43,400	349 sayfa	html, lit, txt	good	www.freeELiterature.com	1 lit EN, 1 txt 1 doc TR
B022	Mark Twain - Tom Sawyer	kitap	roman	macera	71,000	139 sayfa	doc, lit	very good	www.insanizm.com	1 lit EN, 1 doc TR
B023	Voltaire - Candide	kitap	roman	dram	36,600	80 sayfa	doc, txt	good	http://www.kitap.perisi.com/	1 doc EN, 1 txt TR
B024	Clausewitz - Savas	kitap	inceleme	savas	98,000	105 sayfa	doc, html	adequate	http://www.kitap.perisi.com/	1 html EN, 4 cilt olarak doc TR
B025	Lenin - Devlet ve İhtilal	kitap	inceleme		28,900	90 sayfa	lit, doc	good	www.insanizm.com	1 lit TR, 7 doc TR(herbiri 1 chapter)
B026	Plato - Apology	kitap	sosyal	politik	11,600	42 sayfa	lit, html, txt	good	http://www.kitap.perisi.com/	1 lit TR, 1 txt+ 1 html EN
B027	Cicero -Yaslilik ve Dostluk	kitap	felsefe		22,000	65 sayfa	lit, html	good	http://www.kitap.perisi.com/	1 lit EN, 1 html TR
B028	Stephen King - Green Mile	kitap	roman	duygusal	134,000	443 sayfa	lit, pdf, txt	good	www.freeELiterature.com	1 lit TR, 1 txt+ 1 pdf EN
B029	Carus - On the nature of Things	kitap	felsefe		74,000	175 sayfa	doc,txt	adequate	www.freeELiterature.com	1 doc EN, 1 txt TR ----siir seklinde yazilmis
B030	Tolstoy - Master and Man	kitap	roman	dram	19,200	64 sayfa	doc, txt	good	http://www.kitap.perisi.com/	1 pdf TR, 1 txt EN
B031	Tolstoy - Ivan Ilic	kitap	roman	dram	15,800	32 sayfa	doc, txt	good	http://www.kitap.perisi.com/	1 doc TR, 1 txt EN
B032	belgariad-1kehanetin oyuncagi	kitap	roman	fantastik	79,540	157 sayfa	htm,pdf	adequate	http://www.gutenberg.org/	1 pdf TR. 1 htm EN
B033	belgariad-2buyuculer kralicesi	kitap	roman	fantastik	106,000	195 sayfa	htm,pdf	adequate	http://www.gutenberg.org/	1 pdf TR. 1 htm EN
B034	belgariad-3sihirbazin tuzagi	kitap	roman	fantastik	97,000	180 sayfa	htm,pdf	adequate	http://www.gutenberg.org/	1 pdf TR. 1 htm EN
B035	belgariad-4buyulu sato	kitap	roman	fantastik	120,000	206 sayfa	htm,pdf	adequate	http://www.gutenberg.org/	1 pdf TR. 1 htm EN
B036	belgariad-5efsuncunun son oyunu	kitap	roman	fantastik	116,580	197 sayfa	htm,pdf	adequate	http://www.gutenberg.org/	1 pdf TR. 1 htm EN
B037	Arthur Clarke-2001A Space Odyssey	kitap	roman	bilimkurgu	61,850	138 sayfa	doc,pdf	good	http://www.gutenberg.org/	1 doc TR. 1 pdf+1 doc EN
B038	Arthur_C_Clarke-Rama_2	kitap	roman	bilimkurgu	114,470	245 sayfa	txt,txt	good	http://www.gutenberg.org/	1 txt TR, 1 txt EN
B039	Arthur Clarke - rendezvous with rama	kitap	roman	bilimkurgu	72,000	193 sayfa	doc,doc	good	http://www.gutenberg.org/	1 doc TR, 1 doc EN
B040	bernard shaw - Sezar ve Kleopatra	kitap	oyun	drama	39,000	102 sayfa	html, txt	good	http://www.kitap.perisi.com/	1 txt EN, 1 html TR
B041	kafka - Metamorphosis	kitap	hikaye		15,700	28 sayfa	doc,txt	adequate	www.insanizm.com	1 doc EN, 1 txt TR
B042	goethe - faust	kitap	şiir		12,700	40 sayfa	doc,txt	adequate	www.insanizm.com	1 doc EN, 1 txt TR
B043	gogol- Taras Bulba	kitap	roman	kurgu	51,760	94 sayfa	pdf,txt	good	http://www.kitap.perisi.com/	1 pdf TR, 1 txt EN
B044	Eleanor_H_Porter-Pollyanna	kitap	roman	iyimserlik	95,000	301 sayfa	txt,txt	very good	http://www.gutenberg.org/	1 txt TR, 1 txt EN
B045	Anatole France - Thais	kitap	roman	ask macera	36,600	69 sayfa	txt,doc	good	http://www.kitap.perisi.com/	1 txt EN, 1 doc TR
B046	Dostoevsky - Karamazov Kardeşler	kitap	roman	dram	350,000	562 sayfa	txt,doc	good	http://www.kitap.perisi.com/	1 txt EN, 2 doc TR(iki cilt seklinde)
B047	Turgenev - rudin	kitap	roman		53,460	118 sayfa	txt,lit	good	http://www.gutenberg.org/	1 txt EN, 1 lit TR
B048	Stevenson - Markheim	kitap	hikaye		5,600	11 sayfa	txt,doc	good	http://www.gutenberg.org/	1 txt EN, 1 doc TR
B049	Dostoyevski-KUMARBAZ	kitap	roman	dram	62,850	126 sayfa	txt,lit	good	http://www.gutenberg.org/	1 txt EN, 1 lit TR
B050	Goethe - Iphigenia in Tauris	kitap	oyun	drama	19,630	45 sayfa	txt,lit	very good	http://www.gutenberg.org/	1 txt EN, 1 lit TR
B051	Lermontov - A Hero of Our Time	kitap	roman	dram, macera	37,000	68 sayfa	txt,doc	good	http://www.gutenberg.org/	1 txt EN, 1 doc TR
B052	Moliere - The Imaginary Invalid	kitap	oyun	eleştiri	14,900	61 sayfa	txt,doc	adequate	http://www.gutenberg.org/	1 txt EN, 1 doc TR
B053	G. Leroux -Mystery of Yellow Room	kitap	roman	polisiye	47,250	85 sayfa	txt,doc	good	www.freeELiterature.com	1 txt EN, 1 doc TR
B054	Jack London - The Call of the Wild	kitap	roman		33,600	63 sayfa	txt,lit	good	www.freeELiterature.com	1 txt EN, 1 lit TR

B055	Dostoyevski - Devils	kitap	roman	siyasal	260,000	440 sayfa	html,lit	adequate	http://www.kitap.perisi.com/	1 html EN, 1 lit TR(turkce karakterler bozuk...)
B056	Balzac - Eugenie Grandet	kitap	roman		55,750	93 sayfa	txt,doc	good	www.freeELiterature.com	1 txt EN, 1 doc TR
B057	Balzac - Hidden Masterpiece	kitap	hikaye		13,300	27 sayfa	txt,lit	good	www.insanizm.com	1 txt EN, 1 lit TR
B058	Anatole France - Penguin Island	kitap	roman		52,800	91 sayfa	txt,doc	very good	http://www.kitap.perisi.com/	1 txt EN, 1 doc TR
B059	Chamisso - Peter Schlemihl	kitap	roman	psikoloji	38,360	75 sayfa	txt,lit	good	http://www.kitap.perisi.com/	1 txt EN, 1 lit TR
B060	Oscar Wilde-The Happy Prince,Tales	kitap	hikaye	cocuk	10,700	18 sayfa	txt,doc	very good	www.insanizm.com	1 txt EN, 1 doc TR
B061	Dostoevsky - Crime and Punishment	kitap	roman	psikoloji	203,000	330 sayfa	txt,doc,lit,pdf	good	http://www.kitap.perisi.com/	1 txt+pdf+doc EN, 2 lit+2 pdf TR(2 cilt halinde..)

SHORT TEXTS

ID	NAME			# OF WORDS	LENGTH	FORMAT	QUALITY	SOURCE	ADDITIONAL INFORMATION
T001	bilkent yonetmelik	kısa metin	yonetmelik	2,800	7 sayfa	.doc	adequate	www.bilkent.edu.tr	atlamalar, farkli cumle yapilari var.
T002	erhan sigorta	kısa metin	poliçe	3,300	9 sayfa	.doc	very good	ceviribilim bolumu	başı doldurulacak form sonrası normal text
T003	ileri eng boun	kısa metin	tanıtım	2,440	7 sayfa	.doc	adequate	www.boun.edu.tr	atlamalar, farkli cumle yapilari
T004	record 2006	kısa metin	mail	345	1 sayfa	.doc	good	mail	informal, kısa
T005	web ornek	kısa metin		499	2 sayfa	.doc	very good	tercuman burosu	formal, kalitesi yuksek
T006	working caapital	kısa metin	rehber	3,200	10 sayfa	.doc	very good	tercuman burosu	işletme sermayesi rehber el kitabı
T007	yorum	kısa metin	kose yazisi	61,880	125 sayfa	.doc	adequate	hurriyet.com, zaman.com	91 adet yorum, daha artacak..
T008	hotelybakkal	kısa metin	reklam	432	2 sayfa	.doc	adequate	http://www.kitap.perisi.com/	otel reklam afişi
T009	marş	kısa metin	şiir	101	1 sayfa	.doc	adequate		istiklal marşı cevirisi
T010	ninni	kısa metin	hikaye	485	2 sayfa	.doc	good	http://www.kitap.perisi.com/	hikaye
T011	şeftali	kısa metin	hikaye	358	2 sayfa	.doc	good	http://www.kitap.perisi.com/	hikaye
T012	yazılıkaya	kısa metin	tanitim		1 sayfa	.doc	good	http://www.kitap.perisi.com/	düzyazi ceviri
T013	savaş dansları	kısa metin	siir		1 sayfa	.doc	adequate		siir
T014	sen suclu degilsin	kısa metin	siir		1 sayfa	.doc	adequate		siir
T015	sevin birbirinizi	kısa metin	siir		1 sayfa	.doc	adequate		siir
T016	unutma	kısa metin	siir		1 sayfa	.doc	adequate		siir
T017	varolus ucgeni	kısa metin	siir		1 sayfa	.doc	adequate		siir

OTHER RESOURCES

ID	NAME	QUANTITY	FORMAT	QUALITY	SOURCE	ADDITIONAL INFORMATION
XOO1	altyazilar	6 adet	srt, txt, sub	adequate	www.divxaltyazi.com	süre satırlarını silmek gerrekiyor,bol atlamalı,zor ceviri
X002	tezler	13 adet	doc, pdf	very good	www.cmpe.boun.edu.tr	kısa özetler sadece, PDF handing sorunu..