Anaphora Resolution

Katalin E. Lejtovicz, Zsolt T. Kardkovács

Department of Telecommunications and Mediainformatics
Budapest University of Technology and Economics
Magyar tudósok krt. 2, H-1117 Budapest, Hungary
{lejtovicz, kardkovacs}@tmit.bme.hu

Abstract: This paper describes an anaphora resolution system that identifies and resolves anaphors in Hungarian texts. The system works on syntactically parsed texts. It employs anaphora identifying techniques in conjunction with syntax and morphology based resolution techniques. The most novel aspect of the system lies in its anaphora resolution technique that combines syntactic, morphologic and discourse information.

Keywords: anaphora resolution, BFP algorithm, Centering Theory

1 Introduction

The amount of data available on the Internet is vast. Majority of the information is stored in text files. Therefore, it is a well-founded statement, that one of the most important and most complex problem of today's computer science, is natural language processing. A lot of applications exist already, dealing with the English language. For example, there are morphologic, syntactic and semantic parsers for English; in addition, there are content summarizing, content retrieval and machine translation programs. The techniques and methods used for processing the English language cannot be easily adapted – or even it is impossible to adapt them – to Hungarian because of the different structural properties of the two languages.

In this paper we focus on the problem of algorithmic anaphora resolution. Anaphora resolution means the process of determining the referent of the anaphors in a discourse. We process Hungarian texts with a discourse based concept, and concentrated on the problem of resolving anaphors belonging to the next five categories: pronouns, demonstrative, relative pronoun, personal pronoun, and interrogative pronoun type adverbs.

Anaphora resolution consists of three main tasks. The first task is to decide which elements of a sentence are anaphors. The second task is to allocate the possible antecedent candidates of a given anaphora. The third and at the same time final task is to determine which phrase or simple word of the possible candidate list is the antecedent of the given anaphora.

Anaphora detecting and resolving systems already exist for the English language. The two main types are the syntactic tree based methods and the discourse based solutions. The main difference between these two types is, that while the tree based solution operates on the sentences' syntactic parse tree (using only syntactic structural information), the discourse based methods use only discourse information (exp. focus, comment).

Anaphora resolution systems haven't been developed for Hungarian yet. The main idea presented in this paper is that Hungarian texts should be handled using both syntactic and discourse information. This way of solving the problem ensures that the agglutinative type of languages (exp.: Hungarian) can be handled precisely in point of the anaphora resolution problems. We adapted the so called BFP algorithm [1] (Brennan, Friedman, Pollard algorithm) to Hungarian, and also made some changes according to the characteristics and features of the Hungarian language. BFP is a discourse type algorithm. This algorithm is the most suitable for resolving anaphors in Hungarian texts, due to the fact, that BFP was developed for languages with a topic-comment structure, and Hungarian is a language, which has topic-comment structure. Using BFP algorithm for resolving anaphors in English texts, the following results were achieved [2]: it resolved 59% of the total number of anaphors correctly in the actual text, meanwhile this rate is 20% less, then the results of algorithms for English language using grammatical information (exp.: subject, predicate, object). As it will be outlined in the Conclusions part of this article, the BFP based algorithm developed by the authors achieved 40% in resolving anaphors correctly.

2 Anaphora

Anaphors are elements of a language – either a simple word or a whole phrase – which refer back to previously mentioned words or phrases in the text. The following chart (Figure 1) presents the most common anaphora types of Hungarian. The anaphors are indicated in the table according to their frequency of occurrance. On the top of the table the most frequent anaphors are present, meanwhile on the bottom of the table the least frequent ones appear. In the exaples below the first words/phrases highlited in blue in a sentence are the antecedents, and the second words/phrases highlited in blue are the anaphors.

The anaphors that the algorithm introduced in this paper resolves, are the following: pronominal and adverbial.

TYPE	EXAMPLE	Meaning in English
Pronominal	Péter azért nem jött el a moziba, mert ő már látta a filmet.	Peter didn't come to the movie, because he has already seen the film.
Zero	A lány elment a boltba, de (ő) nem vitt magával pénzt.	The girl went to the shop, but (she) didn't take money.
Adverbial	Mi elmegyünk a vendéglőbe, és veled majd ott találkozunk.	We will go to the restaurant and we'll meet you there.
NP (noun phrase)	Marilyn Monroe amerikai színésznő volt. A díva valószínűleg öngyilkos lett.	Marilyn Monroe was an american actress. The diva probably committed suicide.
Verbal	A kislány énekelt, és a testvére is így tett.	The girl sang, and her sister did too.
Rész-egész	Holott a szupermarket csak most nyitott, a salátás részleg már tömve volt.	Although the supermarket has just opened, the salad section was already crowded.

Figure 1
Anaphora types

3 BFP Algorithm

3.1 The Input

The input of the program were the grammatically parsed sentences of the Szeged Treebank 2.0 [3] CD. The sentences on this CD are syntactically and morphologically analysed and are stored in XML structure. The parsed sentences contained information that were not relevant to the problem of resolving anaphors. Such irrelevant information were left out of consideration, because these are not used in the process of anaphora resolving. To reduce the size of the files, and to eliminate irrelevant data, we wrote a Java code, which cuts out unnecessary tags (i.e.: tags not used in the anaphora resolution) from the input.

3.2 The Process

The algorithm always resolves anaphors between two clauses, this means that it is able to resolve inter-, and intrasentential anaphors as well. Intersentential anaphors are anaphors where the antecedent is not in the same senetence as the anaphora (it occures in previous sentences), while intrasentential anaphors are the ones, where the antecedent is in the same senetence as the anaphora. The algorithm works on clauses that follow each other in the input text. If a sentence is not complex (i.e.: it does not contain more than one clauses), then the clause will be the whole sentence, and the resolution will work on the actual and the preceding sentence. So the algorithm can handle both complex and not-complex sentences. Let U_n denote the n^{th} clause in the text, and U_{n-1} denote the previous (i.e.: n-1) clause.

3.2.1 The First Step

The first step of the algorithm is to assign values to the following three variables (the values originate from the first two sentences):

C_f (forward looking centers): C_f is a list, which contains those words/phrases of the given sentence, which can be referred to by an anaphora. Those words/phrases which cannot be referred to by an anaphora will not appear in the C_f list (exp.: articles, prefixes, postpositions, etc.) Our application does not deal with verbal anaphors – because they are very rare – and so C_f does not contain the verbs/verb phrases of the given sentence.

C_b (backward looking centres): C_b contains a phrase. In the first sentence C_b contains the topic (i.e.: the phrase of which we predicate something in the sentence). In the latter sentences C_b contains the previous sentences' topics.

C_p (preferred element): C_p contains the C_f list's first element.

3.2.2 The Second Step

The second step of the algorithm is to generate all the possible anaphora-antecedent pairs. This should be done the following way: if the n^{th} senetence's C_f list (lets denote from now on as C_f^n) contains a possible anaphora (possible anaphors are: pronouns, and certain adverbs) then we pair that possible anaphora with every element of the C_f^{n-1} list. The pairs that we gained this way will be given to the functions *Szures (i.e.: filtering)* and *RagSzures (i.e.: filtering by inflexion).* These functions will rule out anaphora-antecedent pairs that are certainly incorrect pairs (i.e.: the anaphora does not refer to the antecedent int the given text).

The *Szures* function rules out anaphora-antecedent pairs, that do not match in their value of grammatical number. This means that if an anaphora-antecedent pair does not match in number, then we no longer consider them as a possible pair, because an anaphora cannot refer to an antecedent that is not in the same grammatical number.

Those pairs, that return with a true value from the *Szures* function will be passed on to the *RagSzures* function.

In the *RagSzures* function both the antecedent and anaphora will be categorised into four groups depending on the inflexion they carry. The four categories are: individuum, place, time, place-time. Every inflexion belongs to one or more of these categories. An anaphora-antecedent candidate is certainly not a pair, if their inflexions belong to two diferrent categories. In this case the candidates will not be used in the further resolution.

The two filtering methods were developed by the authors, taking into account the features of the Hungarian language.

3.2.3 The Third Step

The pairs that were not ruled out in the two filtering functions will take part in a method, which determines the transition between two clauses.

As you can see on Figure 2 there are four possible transitions between two clauses: continuing, retaining, smooth shift and rough shift.

	CbUn = CbUn-1	CbUn != CbUn-1
CbUn = CpUn	continuing	smooth shift
CbUn = CpUn-1	retaining	rough shift

Figure 2
Transitions

The meanings of the transitions are:

Continuing: There is a continuing transition between the U_{n-1}^{th} and the U_n^{th} sentences, if the U_{n-1}^{th} sentence continues the theme of the U_{n-2}^{th} sentence and this theme occures in the U_n^{th} sentence too. In other words, the theme is carried on since at least three sentences.

Retaining: There is a continuing transition between the U_{n-1}^{th} and the U_n^{th} sentences, if the U_{n-1}^{th} sentence continues the theme of the U_{n-2}^{th} sentence but we introduce a new theme in the U_n^{th} sentence. This means that the actual sentence is about a new topic than the previous ones, there has been a shift in the theme of the discourse.

Smooth shift: There is a smooth shift transition between the U_{n-1}^{th} and the U_n^{th} sentences, if the U_n^{th} sentence continues the theme of the U_{n-1}^{th} sentence but the U_{n-2}^{th} sentence is not about this topic. So two sentences earlier there has been a change in the theme of the discourse, but in the actual sentence we continue the new topic.

Rough shift: There is a rough shift transition between the U_{n-1}^{th} and the U_n^{th} sentences, if the U_n^{th} sentence does not continue the theme of the U_{n-1}^{th} sentence and the U_{n-1}^{th} sentence also does not continue the previous, U_{n-2}^{th} sentence's topic. This means that three successive sentences are all about different themes.

According to the Centering Teory (CT) [4] [5], the most probable is that there is a continuing transition between two sentences, less probable is that there is a retaining transition, even less probable is there is a smooth shift transition, and finally the least probable is that there is a continuing transition between two sentences. The algorithm uses the Centering Theory to decide which anaphora-antecedent candidate is correct.

The algorithm tests whether $C_bU_{n-1} = C_bU_n$ is true. If it is true, then there is continuing or retaining transition between the two sentences/clauses.

1 If the antecedent is in the first position of the sentence (i.e.: topic position), and the anaphora is in the topic position of the second sentence, than this anaphora-antecedent pair will be the correct candidate, because there is continuing transition between the two sentences/clauses.

2 If there was not a continuing transition between the two sentences/clauses, then, if the antecedent is not in the topic position of the first sentence, but the anaphora is in the topic position in the following sentence, then this anaphora-antecedent pair will be the winner. This is because after the continuing transition the retaining transition is the most probable. There is also retaining transition between the two sentences, if the antecedent is in the topic position of the first sentence, but the anaphora is not in the topic position of the second sentence. In this case we return this anaphora-antecedent pair as the winner, because there is a retaining transition between the two sentences, and after the continuing transition the retaining transition is the most probable.

If $C_bU_{n-1} = C_bU_n$ is false, then there is either smooth or rough shift between the two sentences.

This means that there was no continuing and no retaining transition between the sentences, so the steps in 1 should be repeated, and if the conditions in 1 are holding, then this anaphora-antecedent pair will be chosen as the winner.

If there was no no continuing, retaining and no smooth shift transition between the two sentences, then the steps in 2 should be repeated, and if the conditions in 2 are holding, then this anaphora-antecedent pair will be chosen as the winner.

4 An Example

Péter azért nem jött el velünk a moziba, mert ő már látta a filmet. (1)

Peter *conj*. not came *prefix* us the movie, because he already saw the film.

Peter didn't come with us to the movie, because he has already seen the film.

Step 1: Set the values

C_{f1} = {Péter, velünk, a moziba}

C_{b1} = {Péter}

C_{p1} = {Péter}

C_{f2}= {Ő, a filmet}

C_{b2} = C_{p1}= {Péter}

C_{p2} = {Ő}

Step 2: Matching and filtering

C_{f1} = {Péter, velünk, a moziba} and C_{f2}= {Ő, a filmet}. C_{f2}'s those elements will be paired with C_{f1}'s every element, which are pronouns or adverbs. The following pairs will be created in this example:

{ő – Péter}; {ő – velünk}; {ő – a moziba}

Every pair will go through the *Szures* function: {ő – velünk} will be ruled out, because „ő" is a single form, while „velünk" is a plural form word. All teh other candidates will go through *RagSzures* function. In *RagSzures* the {ő – a moziba} pair will be ruled out, because the *-ba* suffix belongs to the place category, while the suffix of the nominative case (i.e.: zero suffix) belongs to the individuum category.

Step 3: Determine the transition

If C_bU_n = C_bU_{n-1} is true, then there is either continuing or retaining transition between the two clauses. „Péter" is the first clause's topic, and „ő" is the second clause's topic. According to the Centering Theory the msot likely event is, that there is a continuing relation between the two clauses. If in this example „ő" refers to „Péter" – as it is the case- , then there is continuing between the clauses. As continuing has the highest probability, the algorithm choses „ő" to be the anaphora and „Péter" to be the antecedent. The result is correct.

5 Results

We used the sentences of Szeged Treebank 2.0 [3] CD for testing. The text was the nv.xml XML document, which consists of 1500 sentences, outof which we tested on the first 500 sentences.

The results for the 500 sentences:

The number of anaphors, that have been correctly resolved, is 15.

The total number of anaphors in the text is 71. This includes anaphors that the program is not prepared to resolve, because we defined to deal only with pronominal, and adverbial anaphors.

The number of pronominal and adverbial anaphors is 63.

The number of anaphors recognised in the text is 25.

So the program recognises the 37% of the anaphors out of the total number of anaphors. The program correctly resolves 39,6% of the anaphors that the algorithm is trained for. The program correctly resolves 21% of the whole number of anaphors. The program correctly resolves 21% of the anaphors that the algorithm is trained for.

The results in percentages can be seen on Figure 3.

	Number of anaphors (71)	Number of pronominal and adverbial anaphors (63)
Anaphors resolved correctly (15)	21%	23,8%
Anaphors recognised (25)	37%	39,6%

Figure 3
Results

Conclusions

The conclusions drawn from these tests show that the program can find and resolve anaphors correctly in numerous cases, but there are still a number of cases when we get incorrect results. The advantage of this program is that it mainly needs linguistic and not algorithmic improvements. On the other hand a typical error is, that the program cannot make difference between anaphors and cataphors (cataphors are elements that refer forward in the text). A solution for this problem

could be to train the algorithm to resolve cataphors as well. We would also like to achieve better results in anaphora resolution, by improving the linguistic part of the program. These improvements are: more crucial filters on the C_f lists, distinguishing new and old information, making difference between the different types of noun phrases. We have further ideas of improvements that affect the core of the algorithm. These are the following: resolve anaphors between two distant sentences and resolve zero anaphors. In the future we plan to develop the program, by adding a module that recognises and resolves cataphors, and we will use the rankings especially for free word ordered languages (these rankings are specified in the Functional Centering [6] article).

References

[1] Brennan, Susan E., Marilyn W. Friedman, Carl J. Pollard: A Centering Approach to Pronouns, in Proceedings of the 25th Meeting of the Association for Computional Linguistics 1987

[2] Tetrault, Joel R.: A Corpus-based Evaluation of Centering and Pronoun Resolution, in Computional Linguistics, 2001, p. 515

[3] Szeged University of Sciences Department of Computer Science, Morphologic Ltd., Hungarian Academy of Sciences Research Institute for Linguistics: Szeged Treebank 2.0, 2004, http://www.inf.u-szeged.hu/hlt/szeged_treebank_2.0

[4] Roger, Kibble: A Reformulation of Rule 2 of Centering Theory, in Computional Linguistics, Volume 27, Number 1, March 2001

[5] Grosz, Barbara J., Aravind K. Joshi, Scott Weinstein: Centering: A Framework for Modelling the Local Coherence of Discourse, in Computional Linguistics, 1995

[6] Strube, M., U. Hahn: Functional Centering, in Proceedings of the 34th Annual Meeting of the Association for Computional Linguistics, 1996, pp. 270-277