Anaphora Resolution in PARE, an Automatic Text Summarizer

Morgan Bates  Sandy Mtandwa  Jason Rush Wray  Scott Thede
DePauw University  DePauw University  Hiram College  DePauw University
Greencastle, IN 46135  Greencastle, IN 46135  Hiram, OH 44234  Greencastle, IN 46135
mbates@depauw.edu  smtandwa@depauw.edu  wrayjm@hiram.edu  sthede@depauw.edu

Abstract

We have developed and tested an anaphora resolution module that is integrated into PARE, an automatic summarizer of English texts. The anaphora resolution module attempts to resolve third person pronouns to their antecedents using a variation of the algorithm used by Kennedy and Boguraev [5], adapted for use with a link grammar parser. Our anaphora resolution accuracy compares favorably with other efforts using a link grammar parser for anaphora resolution in natural language processing applications.

1. PARE Background

PARE is an automatic summarizer of English texts originally developed at DePauw University by Johnson, Vlahov and Thede [1]. It attempts to produce a summary of a text by selecting the most important sentences in the original and concatenating them, a methodology referred to in automatic text summarization literature as “extraction” [2].

Naturally, the quality of summarizations produced by extraction is entirely dependent upon the method used to determine which sentences are the most important. The algorithm that PARE employs is an example of what Mani describes as the “cohesion graph topology” approach to determining sentence importance [2]. This approach rests upon the assumption (which Mani refers to as the “Graph Connectivity Assumption”) that if a cohesion graph is constructed which portrays the semantic connections between parts of a text as edges connecting vertices, then those vertices which have the most edges—or the most heavily weighted edges—will represent the most important parts of the text. Thus, algorithms of this sort, by one method or another, build a cohesion graph, and then use its topology to identify which parts of the text are the most important. PARE’s method of graph construction is inspired by Google’s PageRank algorithm [1] to base the importance of a vertex on the importance of the vertices adjacent to it, modified by the type of edge connecting the two.

In PARE’s cohesion graphs, vertices represent words which occur in the text while edges represent very simple semantic relationships that exist between these words. PARE builds its cohesion graph by parsing all of the sentences of the document using a link grammar parser. PARE then searches the resulting parse information to identify occurrences of the syntactic patterns it recognizes as signifying an important semantic relationship. When such a semantic relationship is found, PARE examines the two words in the relationship. If either or both words are not in the cohesion graph, a new vertex or vertices are added to the graph to represent the new word or words. Then, an edge representing the semantic relationship is added between the vertices.
PARE, as it was originally designed, determined that a word corefered with an existent vertex if and only if the word within the vertex was lexicographically equivalent to the current word. In other words, PARE assumed that all words spelled the same referred to the same concept, while all words spelled differently referred to different concepts. This is clearly false in certain circumstances, one of the most frequent and devastating to the quality of the cohesion graph is the occurrence of pronouns.

This fact is clearly illustrated by comparing Figure 1a and Figure 1b. Figure 1a shows the fragment of the cohesion graph that PARE would produce if given the sentence “Bob bought the sandwich, and then he ate the sandwich.” A vertex is produced for the word “he.” Since “he” is assumed to be a unique concept, the graph fails to recognize that any relationship exists between “Bill” and “ate.” There is little doubt that the graph in Figure 1b is a better representation of the semantic content of the sentence. Since PARE’s performance as a summarizer depends upon the quality and correctness of the cohesion graph, if PARE’s ability to handle pronouns could be improved it should improve PARE’s overall accuracy.

Therefore, to improve the performance of PARE, we have developed and evaluated a pronominal anaphora resolution module for it. As PARE’s development is currently focused on the summarization of news articles, our resolution module is limited in scope: it only attempts to resolve third-person, non-lexical pronouns, which are the most frequent and seem to be the most important pronouns in this domain.

![Figure 1a. The cohesion graph that PARE would make.](image1.png)

![Figure 1b. The cohesion graph that PARE should make.](image2.png)

### 2. Anaphora Resolution Background

Anaphora is defined by Mitkov as “the [linguistic] act of pointing back to a previously mentioned” linguistic form [3]. Machine anaphora resolution is the automated process of identifying to what linguistic forms the instances of anaphora refer. Though the field thus delineated is very broad, most of the research in this area of natural language processing (NLP) has been on the more specific problem of pronominal anaphora resolution, the resolution of pronouns to the antecedents to which they refer.

Many approaches to pronominal resolution have been attempted, including knowledge-
based solutions in the 1980s, and corpus-based machine learning and probabilistic models in the 1990s. Some of the most influential work over the last ten years has been in what Mitkov refers to as the “knowledge-poor” research program [3]. Research in this area has sought to provide effective and accurate pronoun resolution techniques for NLP applications without relying on costly domain-specific knowledge bases or training corpora.

The Lappin-Leass pronominal resolution algorithm, RAP [4], has arguably been the most influential of these approaches. Figure 2 illustrates the overall design of the algorithm. First, RAP parses every sentence in the text using a slot grammar parser, a special kind of grammar that belongs to the class of grammars called *dependency grammars* [6]. Then, starting from the beginning of the text, it proceeds through the text, sentence by sentence. Every time a non-pronominal noun phrase is encountered, a *discourse referent* is produced to represent it, and it is either added to one of the already existing *coreference classes* – sets of discourse referents which all refer to the same thing – or a new coreference class is created for it. Every discourse referent that is produced has an integer value called its *salience* representing the likelihood that a pronoun will refer to it. Salience is determined by a number of factors: part of speech, whether the pronoun is in the current sentence, whether the word is in a subordinate clause, etc. Each coreference class also has a salience, which is the sum of the salience of all of its discourse referents.

Resolution occurs each time RAP encounters a pronoun\(^2\). The first step is to eliminate as many resolution candidates as possible so that the correct resolution is made more probable: to this end, two filters are used to eliminate impossible candidates from consideration. The agreement filter eliminates otherwise possible resolutions based upon agreement features, such as gender, plurality and person. The syntactic filter eliminates resolution candidates in the same sentence which, by the

---

\(^2\) The process described here does not apply to so-called lexical (reciprocal and reflexive) pronouns in RAP or the Kennedy-Borguraev algorithm. As our algorithm does not attempt to resolve lexical pronouns, the description of their resolution has been omitted. See [4, 5].
rules of English grammar, can not corefer.

Once impossible resolution candidates are eliminated, heuristics in the form of two bonuses and one penalty are applied: the proximity bonus is applied to discourse referents in the current sentence, the parallelism bonus is applied to discourse referents that share the same part of speech, and the cataphora penalty is applied to discourse referents that are in the same sentence but follow the pronoun. The saliences of the coreference classes are tallied, and the pronoun is resolved and added to the coreference class with the greatest salience that was not eliminated by filters.

One other aspect of RAP bears mentioning: its modeling of human attentional state. After completing each sentence, the salience of each discourse referent is halved. This is modeling the fact that as the distance between the pronoun and its potential referent increases, their likelihood of matching decreases. As RAP assumes that a human can identify the antecedents of the pronouns it encounters, it attempts to replicate this behavior.

The developers of RAP reported an excellent 86% precision in the resolving of pronouns to antecedents [4]. There are weaknesses to their approach, however: first, it requires a high-quality slot grammar parser as it was originally presented and such parsers are not widely available [5]. Second, its syntactic filter, which is important to its performance [4], is quite complex, consisting of six conditions that assume a dependency grammar parse, and translating them into other grammars, even closely related ones, is non-trivial [6].

The Kennedy-Boguraev algorithm was developed in response to these criticisms of RAP [5]. It was designed to function without a parser, working instead from the output of a part-of-speech tagger and a text segmenter, using the relative position, for instance, of subjects and verbs to roughly estimate where clauses, phrases and other constituents begin and end. Figure 3 illustrates its design.

For the most part, the Kennedy-Boguraev algorithm functions in the same manner as RAP. It proceeds through the text, building discourse referents for each noun phrase it
encounters, assigning these discourse referents a salience value, and adding these discourse referents to coreference classes, which themselves have saliences, which are the sum of the saliences of the discourse referents they contain.

Aside from the fact that no parse is performed by the algorithm, the two primary ways in which the Kennedy-Boguraev approach differs from RAP are in its syntactic filter and its heuristic bonuses and penalties. As these are both important to our anaphora resolution algorithm, they shall be described in detail.

Kennedy-Boguraev’s syntactic filter is simpler than the RAP filter not only due to the fact that it is not expressed in terms of the slot grammar, but also due to the fact that it has three conditions rather than RAP’s six. If any of these three conditions are true, then the pronoun and the candidate discourse referent cannot corefer. These conditions are:

- A pronoun cannot corefer with a coargument. (“Him” can not corefer with “Bob” in “Bob killed him.”)
- A pronoun cannot corefer with a nonpronoun constituent which it both commands and proceeds. (“It” can not corefer to “the bus stop” in either “It was by the bus stop.” or “It was here because the bus stop was closed.”)
- A pronoun cannot corefer with a constituent that contains it. (“Her” can not corefer with “Her dog” in “Her dog ate the artichoke.”)

It is worthy of reiteration that the Kennedy-Boguraev algorithm tests these conditions using nothing more than a part-of-speech tagger. Constituency information, such as the fact that “because the bus stop was closed off” is a subordinate clause is entirely inferred from the parts of speech involved and word order.

The other way in which Kennedy-Boguraev differs significantly from RAP is in its heuristic bonuses. While the Kennedy-Boguraev algorithm retains the cataphora penalty in its entirety, it changes the parallelism heuristic significantly, and replaces the proximity heuristic with what it calls the “locality heuristic” [5]. In the Kennedy-Boguraev algorithm, the parallelism heuristic rewards candidate discourse referents which have parts of speech the same as other discourse referents that had been previously been the resolution of pronouns with the same part of speech as the current pronoun being resolved. The locality heuristic rewards embedded discourse referents when the current pronoun being resolved is at the same level in order “to temporarily treat the discourse referent as if it were “not in the subordinate context.” [5].

In spite of the vast simplification of the syntactic filter, and the fact that this algorithm functions without a parser, Kennedy and Boguraev reported 75% accuracy – only an eleven percent drop from the reported accuracy of RAP.

3. Our Anaphora Resolution Module

We decided to adapt the Kennedy-Boguraev algorithm to use it with a link grammar parser. The algorithm that we implemented follows the same basic procedure as the Kennedy-Boguraev algorithm, with the differences falling into two categories: those required to allow it function using the link grammar parser rather than a part-of-speech tagger, and those involving the heuristic bonuses and penalties. These two categories will be discussed in turn.
3.1. Using the Link Grammar Parser

PARE was originally designed to use the link grammar parser to parse the sentences of the original text. The parser is capable of producing two kinds of output: the parse itself, which consists of a set of “links” that join two words and identify the grammatical relationship that exists between them, and constituency information, which uses a LISP-style parenthesized tree to denote where phrases and clauses begin and end [1, 7]. These two types of output are illustrated in Figure 4.

![Figure 4. An example of link grammar parse output and constituency information from the Link Grammar Parser](image)

The original Kennedy-Boguraev algorithm attempted to deduce constituency information from parts-of-speech and word order. The fact that our parser provides relatively detailed constituency information means that our algorithm can forgo this step, instead relying upon the parser to determine when noun phrases are embedded in subordinate clauses, or adjacent in prepositional phrases.

The original Kennedy-Boguraev algorithm required a part-of-speech tagger so that the parts of speech can be identified for use in the syntactic filter, and that plurality can be determined. While part-of-speech information and plurality are not explicit in the link grammar parse information, it can be extracted from the directed links that connect words, eliminating the need for a separate part-of-speech tagger. In some cases, translating this information into standard parts of speech is easy: links whose names begin with “S,” for instance, all link the subject of a clause to a verb. In other cases, however, translating this information is more difficult. For example, links whose names start with “O” and end in “n” link a verb to a direct object, if and only if the verb does not have another link that begins with “O”, in which case “O...n” links to the indirect object, and “O...” links to the direct object.

Extracting plurality from the link grammar parse is more difficult. The parser only produces plurality information (represented as lower-case “p”s and “s”s in links) when there are grammatical constraints on number, as in the case of subject-verb agreement. Still, it is more accurate than attempting to determine number purely from morphological criteria. In one special case, we do just this, however: if the word “and” is found in a noun phrase, we define the resulting discourse referent as plural. While this is not a perfect rule (“War and Peace is a long book.”), it still seems to be a good guideline.

3.2. Heuristic Bonuses and Penalties

The original Kennedy-Boguraev algorithm implemented three heuristics that were applied to the salience of specific discourse referents immediately before coreference classes were ranked to resolve a specific pronoun. Our set of heuristics differs substantially from theirs. While we implement the cataphora penalty in the same manner, implementing their parallelism and locality heuristics would require retaining a great deal of information during the execution of the algorithm that is otherwise unnecessary. Instead, we replaced their parallelism heuristic with the parallelism heuristic implemented in RAP (possible
resolutions are rewarded for having the same part of speech as the pronoun) and simply omitted the locality heuristic.

Moreover, we added our own penalty, called the Coreferential Disagreement penalty. Like Kennedy and Boguraev, we have no gender information for nonpronominal noun phrases. During preliminary testing, however, we made the observation that frequently one pronoun would be correctly resolved, but then pronouns immediately following the original pronoun that disagree with it would be resolved to the same noun phrase, partially because of the salience that the original pronoun grants the coreference class. The Coreferential Disagreement penalty penalizes resolving subsequent disagreeing pronouns to the same discourse referent as the previous pronouns, in order to limit the degree to which discourse referents that disagree with the pronoun currently being resolved contribute to the possibility that they corefer.

4. Integration Into PARE

Due to the need for future experimental evaluation, as well as the on-going nature of the PARE project, and the object-oriented spirit of Java, we designed the integration of anaphora resolution into PARE with modularity in mind. The PARE engine deals only with the abstract superclass of anaphora resolvers; it only assumes that by giving an anaphora resolver access to its internal representation of the Link Grammar parse, the constituency information, and the file that correlates the content of the two, the anaphora resolver will edit the representations of the Link Grammar parse and the constituency information to show resolved anaphora.

Most of the time this editing amounts to merely replacing the strings representing pronouns with strings representing the canonical discourse referent of the coreference class to which they have been resolved. In two situations, this is not the case; in the cases of compound noun phrases and possessive pronouns, internal structure considerations make this straightforward approach too costly to the adequacy of the cohesion graph.

In both of these cases we attempt to imitate the output of the parser in order to retain as much information as possible without executing another lengthy parse. In the case of resolving anaphora to compound noun phrases, for every link that involves the pronoun being resolved, a link of the same type is made for every noun phrase in the series. In the case of resolving possessive pronouns, in the links involving the pronoun, the pronoun is replaced with the string “'s”. Then, a new link is made of type “YS” is added, connecting the noun phrase that was possessed to the noun phrase replacing the possessive pronoun.

5. Evaluation

Our anaphora resolution module was tested on five randomly chosen newswire articles from the corpus. These articles contained a total of 87 third person, non-lexical pronouns. Of these, PARE was only able to identify 66, as a result of the link grammar parser failing to parse the sentences which contained the data flow, even in simple cases. The internal structure of the noun phrases are lost, and the coherence graph fails to demonstrate a relationship between the non-pronominal references and the pronominal references in the same coreference class. See section 7 for some discussion of possible new directions on this topic.
others, or as a result of our tokenization procedures failing to correctly partition the newswire articles into sentences for the parser. Of these 66 pronouns for which resolution was attempted, 30 were correctly resolved. Put in terms of the standard metrics of recall and precision, this is a recall of 75.9%, and a precision of 45.5%

While these numbers may seem low when compared to the numbers reported by the algorithm designers, they are comparable to other similar efforts. For example, Dowdall, et al., recently used the link grammar parser to emulate a dependency grammar in order to implement RAP for ExtrAns, a question answering program [6]. They hand-selected sentences containing, in total, 60 pronouns in examples of intrasentential anaphora, and reported only 43% accuracy. They suggested that poor performance (namely incorrect and failed parses) on the part of the link grammar parser was to blame; we agree that it seems that in many cases incorrect resolutions result from the parser failing to parse sentences that contain the correct antecedents. Many of these failures may be the result of the parser’s included lexicon being insufficient for actual application use.

It is important to note that neither our sample nor the ExtrAns sample is large enough to draw definitive conclusions. However, when one considers that the ExtrAns project only sought to resolve third-person, neuter pronouns while we sought to resolve all third-person pronouns despite the likelihood of gender errors, and that the ExtrAns included reflexive pronouns in their results, which tend to resolve more accurately due to having more constraints placed upon their resolution, these results at least suggest the possibility that the Kennedy-Boguraev algorithm, because it is less dependent upon precise parses, is a better choice for many applications than the RAP, even when a parser is available.

6. Conclusion

While the Kennedy-Boguraev algorithm was designed for use without a parser, its insights remain valuable tools even when working with a parser that provides constituency information, as its syntactic filter provides an alternative to attempting to implement the RAP syntactic filter without a dependency grammar parser. Still, better, more freely available parsing technology seems to be a prerequisite for long term improvement in the utilization of anaphora resolution in NLP applications.

7. Future Work

We are currently performing an extrinsic evaluation of PARE’s performance in terms of speed and accuracy with regards to the task of document sorting. The results will allow us to determine how anaphora resolution affects the quality of PARE’s summaries. Avenues of research which would likely lead to improvement of PARE and its anaphora resolution are also readily apparent.

First of all, a more thorough integration of the results of anaphora resolution into PARE’s coherence graph would likely lead to the coherence graph being an even better representation of the semantic content of the original text. There are two paths that could be taken toward this end: either the anaphora resolution module could resolve pronouns at the level of link grammar phrases, or the vertices of the coherence graph which represent nouns could be replaced with vertices which represent coherence classes.

Also, the performance of the anaphora resolution module could likely be improved
by several means. First, implementing a lexicon to identify the likely genders of noun phrases would greatly decrease the number of gender mismatches the algorithm produces, as is noted in the original Kennedy-Boguraev paper [5]. Secondly, steps could be taken to make the parser more robust, in order to decrease the number of sentences it fails to parse. This would likely be to the benefit of not only the anaphora resolution module, but to PARE as a whole. Finally, if coreference could in some manner be determined among non-pronominal discourse referents, the precision of the algorithm could likely be improved.

Acknowledgments

We are indebted to the DePauw Computer Science faculty for their guidance and support. This work was supported by National Science Foundation Grant EIA-0242293 and the DePauw University Science Research Fellows Program.

References


