Developing a re-usable web-demonstrator for automatic anaphora resolution with support for manual editing of coreference chains

Anders Nøklestad [*], **Øystein Reigem** [⋆], **Christer Johansson** [†]

[*]University of Oslo, Norway
anders.noklestad@ilf.uio.no
[⋆]Aksis Unifob AS, Bergen, Norway
oystein.reigem@aksis.uib.no
[†] University of Bergen, Norway
christer.johansson@uib.no

Abstract

Automatic markup and editing of anaphora and coreference is performed within one system. The processing is trained using memory based learning, and representations derive from various lexical resources. The current model reaches an expected combined precision and recall of F=62. The further improvement of the coreference detection is work in progress. Editing of coreference is separated into a module working on an xml-file. The editing mechanism can thus be reused in other projects. The editor is designed to store a copy on the server of all files that are edited over the internet using our demonstrator. This might help us to expand our database of texts annotated for anaphora and coreference. Further research includes creating high coverage lexical resources, and modules for other languages. The current system is trained on Norwegian bokmål, but we hope to extend this to other languages with available tools (e.g. POS-taggers).

1. Introduction

Our demonstrator consists of several smaller programs that work together via a web interface. The task is to be able to enter an arbitrary text, in a language for which we have some linguistic resources available. Those resources include a part-of-speech tagger, a functional role finder, and a chunking mechanism. From these resources, a representation of each markable word (mainly common nouns, proper nouns, and pronouns) is created, and compared to each new markable, which may be a possible anaphor (i.e., a word that has a coreference relation with the current markable). We have plans to extend the resources to accommodate more languages.

The goal of this presentation is to demonstrate the generation of an interactive webpage that presents the result in a graphical format, and allows for easy manipulation of reference chains, thus allowing fast and intuitive editing of coreference chains. There are various autonomous steps that will be fairly easy to replace with new modules for each added language.

The first step is to format the text for further processing. After this step, a program that implements a Constraint Grammar of Norwegian (Hagen et al., 2000) is run to label the words in the text with part-of-speech tags, functional roles and information relating to finding named entities (Johannessen et al., 2005). This information is the basis for generating a description vector for each word that may enter an anaphora-relation. Each potential anaphor-antecedent pair has a match vector calculated, which represents how well the words in the pair fit together (Aone and Bennett, 1995; McCarthy and Lehnert, 1995; Soon et al., 2001b; Johansson and Nøklestad, 2005).

Before processing new material, match vectors were calculated for a fairly large portion of texts, and stored for all candidates of the training set, together with the information if the items of the pair were referring to the same entity or not. The training set is laboriously tagged by hand for anaphora relations, which include also other types of reference than identity of reference, although we only automatically identify this type of relation. Additionally, we represented the pleonastic pronouns found in the material. The training set is used as the basis for automatic decisions regarding the status of unknown pairs of words. The computational tool that allows us to do this is called TiMBL (the Tilburg Memory Based Learner). TiMBL implements many versions of memory-based learning.

The system accepts a Norwegian text, analyses the text looking for anaphora relations, and shows the text with markables (e.g. noun phrases) and automatically found relations highlighted, using colored boxes and lines. The highlighting is dynamic, triggered by the user moving the mouse cursor over the relevant and available phrases.

The relations the machine has found may easily be edited. After the editor is satisfied with the result, the annotated text can be stored in a format that includes the efforts of the mechanisms used (e.g. part-of-speech tagging, and functional roles) as well as the final annotation of coreference. The final result is presented as an xml-tagged file.

2. Detection of Coreference Chains

We use a memory-based mechanism for anaphora resolution and coreference. A program labels the words in the text with part-of-speech tags, functional roles, and lemma forms. This information is used for generating a representation of each anaphor and antecedent candidate. Each potential anaphor-antecedent pair has a match vector calculated, which is used to select similar cases from a database of labeled cases. We are currently testing many feature combinations to find an optimal set for the task. The most recent results show an overall F-measure (combined precision and recall) of 62, with an F-measure of 40 for those cases where anaphor and antecedent are non-identical, and 81 for identical ones. The coreference chains are restricted so that an anaphor is only allowed to link to the last item in a chain.

2.1. Previous research

One early algorithm (Hobbs, 1978) performs searches for antecedents in parsed syntactic trees. Preferences in the model are present in the order of the tree search. Another approach is to model the salience of possible antecedents (Lappin and Leass, 1994). Salience factors are weighted somewhat ad hoc, but with a rank-order inspired by observed ratings of salience in psycho-linguistic experiments. A third influential theory is Centering Theory (Grosz et al., 1995). Centering also assumes a salience rating, but has an added constraint that there is a single item which is in focus at any given time and therefore most likely to be pronominalized. None of these models can be regarded as robust models, as they rely on fixed methods, which are not changed by experience. Hobbs' method relies heavily on parsing, and depends on having the correct parse trees available. Lappin and Leass' approach relies on a saliency rating, as well as parsing and finding functional roles. Their algorithm also has a heuristics that takes distance into account. Centering depends on maintaining forward and backward looking centers, and selecting the most likely candidate to be in focus.
Recently, anaphora resolution has been performed by machine learning techniques, using resources that are less demanding. For example, Lappin and Leass' algorithm has been modified so that it can operate on the result of a statistical part-of-speech tagger (Kennedy and Boguraev, 1996). Cardie and Wagstaff (1999) introduced an unsupervised clustering technique that performs better than many handcrafted systems on the related task of noun phrase coreference. Because the algorithm is unsupervised, finding coreference does not rely on preclassified exemplars. Other benefits of using a clustering technique is that it can more easily be adapted to different domains. Coreference can also be viewed as a classification task. A comparison between decision tree learning (classification) and the clustering algorithm, shows, not surprisingly, that training on pre-classified examples can provide better results (Soon et al., 2001a). An F-ratio of 62.6 was obtained for decision tree learning, whereas the clustering algorithm produced a measure of 53.6 on the same dataset (MUC-6). There is, however, a slight gap to the best system, which produced a measure of 65, according to Cardie and Wagstaff (1999, p.82).

3. Memory-based learning

We decided to work with memory-based learning, as implemented in the TiMBL software package (Daelemans et al., 2004). This decision was inspired by anaphora resolution results from decision tree learning (Soon et al., 2001a), which is a type of exemplar based machine learning mechanism.
Memory-based learning is a kind of k-nearest neighbor approach, which operates on exemplars, or *instances*, in the form of feature vectors. It is a supervised machine learning method that requires an annotated training corpus. During the training phase, training instances are stored in a database, or *instance base*, along with their annotated categories. Classifying a new instance amounts to comparing it to the instances in the instance base, finding the ones that are most similar to the new instance (i.e., its *nearest neighbors*), and selecting the category that is shared by the majority of those neighbors.
The classification process is illustrated in general terms in table 1, which shows three training instances and a test instance. The features can take the values plus or minus. The reader is encouraged to look for feature values that are the same for the test instance and the training instance.
The test instance matches Train1 on Feature1 and Feature2, and it matches Train2 on Feature2 and Feature4, while Train3 only matches on Feature2. Thus, Train1 and Train2 constitute the set of nearest neighbors for Test. Since the majority (in this case all) of the nearest neighbors share the C1 category, this category is also selected for the Test instance.

	Train1	Train2	Train3	Test
Feature1	+	–	–	+
Feature2	–	–	–	–
Feature3	–	–	–	+
Feature4	+	–	+	–
Category	C1	C1	C2	*C1*

Table 1: Classification based on nearest neighbors.

4. Anaphora Resolution

We use a corpus which has been automatically annotated for part-of-speech tags and syntactic categories, and hand annotated for coreference. The algorithm was trained on a subset of 41970 tokens, of which 35780 were words and the rest typographical annotations. From these 35780 words, we have 11831 markables (potential members of reference chains).
In order to experiment with a more limited material, we selected a set of pronouns which had all been marked either with or without an antecedent. Pronouns without antecedents refer to something not present in the (preceding) text. The chosen pronouns were *han* "he", *hun*, "she", *seg* "himself/herself/itself/themselves", *den* "it" masc./fem., and *de* "they". The set of pronouns consisted of 2199 anaphor candidates, of which 2129 were marked with a coreferential antecedent in the text. Cataphors were excluded from the training set.

4.1. Salience

The classification of an anaphor–antecedent pair is affected by a number of features, such as whether they agree in number, person, case and gender. All these features create subdivisions of the representational space. Since the model is not probabilistic, it is not correct to associate probabilities to this selection. The method selects from the nearest neighbors, regardless of their likelihood of occurrence, or likelihood of predicting the correct class. Additional constraints may stem from syntactic restrictions (for example, whether a reflexive interpretation is possible), and selectional restrictions (e.g., which are the typical objects of specific verbs). Real knowledge of what is possible, may help. Recency, repeated mention and parallelism are other important factors in deciding coreference (Jurafsky and Martin, 2000, pp.678–684). We use information from a machine annotated corpus, so not all of these factors necessarily have correct values. The factors are as follows (see also table 3):

- Do the lemmas of the anaphor candidate and the antecedent candidate match?
- Do the anaphor candidate and the antecedent candidate have the same
  - syntactic function?
  - grammatical gender? (We also want to add natural gender in the future.)
  - number?
- Is the anaphor candidate a distinctly human pronoun (*han* "he" or *hun* "she") and the antecedent candidate a proper name?
  - If so, do they have the same (natural) gender?
- Is the anaphor candidate a reflexive and the antecedent candidate its closest subject?
- Is the antecedent candidate a subject?
- Is the number of sentence boundaries, as detected by the tagger (e.g. some conjunctions, and punctuation), between anaphor and antecedent candidates
  - less than 2?
  - less than 3?
- The lemmas of the anaphor and the antecedent candidates concatenated

Note that we have no information about natural gender for lexical noun phrases, which is information we would want to have, in addition to grammatical gender. Various ways of finding the natural gender (Hale and Charniak, 1998, inter al., for English) will be tried out later in our project.
Since we are using memory-based learning for this task, the full impact of each feature is determined after the nearest neighbors have been found. A feature that is highly valuable in some contexts may simply not add anything in other contexts, for example if all the selected nearest neighbors contain the same feature. This is an advantage of memory-based learning, since it does not rely heavily on global statistics. TiMBL may also weigh features by their informativeness, by calculating the information gain of each feature (i.e. how much we gain in classification accuracy by knowing the value of the feature). Still, TiMBL is essentially a selection mechanism and not a probabilistic mechanism.

4.2. Percolation

A strategy to percolate most of the matching features to the next referring expression in a chain was adapted. This means that the most recent antecedent candidate matches the union of its own matching features and those of the preceding antecedent candidates of the same chain. Sometimes information is not available immediately, but will be known after we have established coreference (see table 2) .

3		3.1		3.2
Calvin	$\rightarrow$	who	$\rightarrow$	he
—		Calvin		—
—		....		—

Table 2: Percolation of *Calvin* to *who*

Take for example the following discourse, adapted and translated from a folk tale: *"[Three brothers lived in a forest.] The oldest was called $Adam_1$, the second $Bert_2$, and the youngest $Calvin_3$, $who_{3.1}$ used to tend to the $ashes_4$. The $Sunday_5$ when the $notice_6$ from the $King_7$ about the $ship_8$ was posted, $he_{3.2}$ happened to be there. When $he_{3.3}$ came home and told about $it_{6.1}$, $Adam_{1.1}$ wanted to set out immediately, and prepared some $food_9$ for the $journey_{10}$. For $he_?$ wanted to find out ..."*
The second time that *he* refers to Calvin at point 3.2, the information from the first mention of Calvin at point 3 has been percolated to the *who* at 3.1. After linking up, the information in 3.2 contains the (lemma)/name "Calvin", the functional roles of position 3 (predicate filler), 3.1 (subject), and 3.2 (subject), as well as the number (singular), and the gender (masculine).

feature	3	3.1	$3 \cup 3.1$
match on			
lemma	–	–	–
syn.func.	–	+	+
gender	–	–	–
number	+	+	+
human pro.			
and prop.	+	–	+
and gen	–	–	–
refl+subject	–	–	–
subject-ant	–	+	+
$dist. < 2$	+	+	+
$dist. < 3$	+	+	+

Table 3: Match between *he* at 3.2. and antecedents *Calvin*(3), or *who* (3.1), or $3 \cup 3.1$) with percolation

This strategy is thought to be important for full anaphora resolution. From table 3, we see that we get a match vector with six matching features with percolation, instead of four features for match with Calvin, and five features matching *who*. It is an open question whether there should be a lemma match between a pronoun and the same pronoun only, or if a pronoun should be able to unify with all kinds of strings for surface match. We have decided to allow a lemma match between the same form of pronouns only, but we will try using an unknown value for this type of match. Notice that it would be a good idea to have three values for gender matches: $+$, $-$, and *unknown*. If *Calvin* was found to be a male name, for example from a list of male names, we would be able to access a masculine gender for both *Calvin* and *who*. (This is not to say that "who" is a word with inherent gender.) An unknown value would be good to have when we cannot disprove a match. In addition, we would create and search for the concatenated lemmas $he/Calvin$, and he/who respectively. These items are not percolated, but contain the value of the candidate antecedent and the anaphor.

Table 4 shows the match vectors, after percolation, for $he_?$ matching with either *Adam* or *Calvin*. As can be seen, *Calvin* matches on more features than *Adam*. Still, Adam might be selected as antecedent because it is closer to the anaphor. This is due to the fact that the search for an antecedent moves backwards in the text, starting at the anaphor and stopping as soon as there is a positive classification. Hence, if the match vector for Adam has more positive than negative nearest neighbors in the instance base, *Adam* will be selected as antecedent, and *Calvin* will never be considered.

feature	1.1	3.3
match on		
lemma	–	+
syn.func.	+	+
gender	–	+
number	+	+
human pro.		
and prop.	+	+
and gen	–	–
refl+subject	–	–
subject-ant	+	+
$dist. < 2$	–	–
$dist. < 3$	+	–

Table 4: Match vectors for $he_?$, with $Adam1.1$, and $he_{3.3}$/*Calvin*

Two important points are illustrated. First, that a closer antecedent will have an advantage, because it will be selected before its competitors if TiMBL decides for it, since the search for an antecedent is stopped when there is a positive classification. Second, that the final categorization does not depend on how many matches are recorded, but on how the full vector and its neighbors have been classified previously. A last point is that proper names are assumed to be in the singular; however, for some types of proper names (e.g. organizations) the *number* feature may have a different default than singular, or may come from a knowledge base. This is crucially an issue of what our lexical resources will deliver.

5. Training and Testing

Several different combinations of the available features were tried, and the previously presented 9 features were those that gave the highest scores. They are most likely not the optimal features, but they are the features that are available in our machine tagged example collection. We have only scored hits on the closest antecedent in a chain, whereas it could be argued that a hit on any antecedent in a chain would suffice.

Feature percolation allows previously found antecedents of a chain to influence the decision on anaphoric status. Features percolate and accumulate towards the most recent member of a chain.

Training consists of providing postive and negative examples to the memory-based learner. Positive examples are pairs of anaphor and antecedent candidates, described by how they match on the features we have decided to include. Negative examples are pairs consisting of an anaphor and any markable that is closer to the anaphor than the actual antecedent. Typically, there are many more negative examples than positive examples.

In testing, we start searching from the pronoun (or potential anaphor, in the more general case of coreference detection) backwards in the text until the algorithm finds a markable that is classified as an antecedent for the pronoun. The classification decision is binary, so we assign the first found candidate marked by the mechanism as an antecedent. We have experimented with using the strength of the classification decision (recalculated into z-scores, for general comparison), but this did not improve the results and was abandoned.

The classification decision is simplistic: The memory-based learner is consulted for each markable that is encountered. If the positive nearest neighbor examples in the database outvote the negative examples, the classification is a yes, otherwise a no. When the mechanism says yes, there is no need to search further.

The results of our cross-validation experiments are shown in table 5. The overall F-measure is 62.14. It should be noted, however, that the system performs much better in those cases where the anaphor and the antecedent are identical than in cases where they are not identical. Closely related to this observation is the fact that in 78% of the test cases, the classifier selects an antecedent that is identical to the anaphor. This strong tendency to select identical pairs is likely due to the fact that 67% of all manually annotated anaphor–antecedent pairs in the training data fit this characteristic. This is particularly noticeable for *han* "he" and *hun* "she", which also account for 78% of the relevant pronouns. The problem with this pattern is that we often miss many of the more interesting pairs, where there is a non-pronoun as antecedent. The examples favor a negative classification, simply because this is correct most of the time. A simple bias towards positive answers will hurt our precision, but this could be worth it if we could quantify how much more such a pair is worth. This is a highly task dependent decision. In many tasks, chains containing only pronouns would not make much difference, whereas a correct chain that links with keywords would be highly valuable (for example in estimating the topic of a paragraph).

We performed our 10-fold cross-validation experiments on a corpus of novels, which is a complex type of text with typically elaborate and long reference chains. Newspaper texts, in contrast, select for shorter, coherent paragraphs since space is a more limited resource in this domain.

We have pinpointed where the automatic coreference finding could be improved, namely the cases where antecedents and anaphors are different. Improvements could consist of providing more semantic information, such as finding default pronouns for names, finding occupations (journalist, writer, police) that are likely to take a human pronoun, finding things and part-of relations and more. One problem with this is getting enough coverage. The information can often be found directly on the web using some smart search patterns (such as "his name is X", "her name is X"). Recent

Selection	Proportion	Recall	Precision	F-measure
Identical anaphor-antecedent	67%	83.95	78.88	81.31
Non-identical anaphor-antecedent	33%	40.75	40.32	40.51
All cases	100%	62.12	62.18	62.14

Table 5: Results from 10-fold cross-validation experiments.

research (Markert and Nissim, 2005; Markert et al., 2003, inter al.) has shown that the web provides very good coverage, even compared to very large corpora with extensive mark-up, such as the British National Corpus. We will look into how we can use the web to gain more detailed and useful information with a high degree of coverage. Another source of information are lists, which are available on the web via various government organizations.

6. Integrating Automated Coreference with easy Editing

The process of finding coreference chains starts up several subprocesses: format the text, analyze the text to find the needed features, create representations, and match vectors for each anaphor-antecedent candidate, and then start a TiMBL server that will provide the decisions as to which pairs could be considered coreferent.

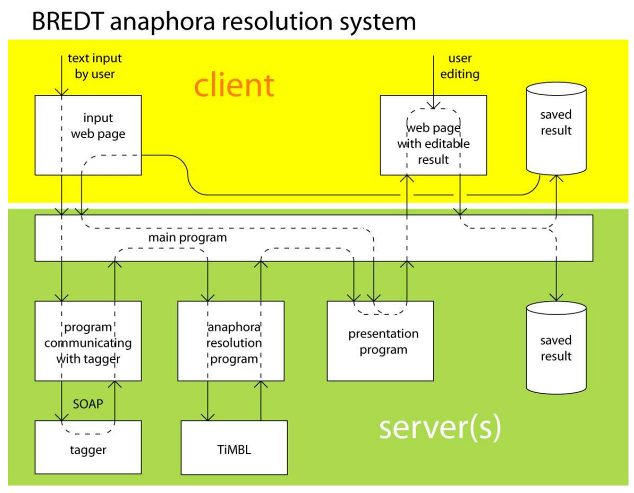

Figure 1: System architecture

The result of the analysis is an interactive web page where the user may point at a candidate noun phrase (indicated by a grey box) and see the requested information. For example, the user may request to look at referents before and after the selected candidate, but only the nearest referents. The available choices for direction are to 1) show only antecedents (preceding the selected), and 2) anaphors (succeeding the selected candidate) or 3) to choose both antecedents and anaphors for display. The depth of the analyses may either be 1) only the nearest available candidates, or 2) the whole reference chain.

At the moment, the success rate for automatically finding complete reference chains is low, but we hope to improve this using a variety of techniques.

6.1. Editing the automatic suggestion for coreference

The relations the machine has found may easily be edited. Selecting two markables with a link between them unlinks them, and any intermediate candidates between them keep their internal links. For example, consider a coreference chain $A \leftarrow B \leftarrow C \leftarrow D$ suggested by the automatic coreference finder. Selecting D and then A creates a new link between them. Then D cannot continue to be linked to C, and B cannot continue to be linked to A, because of the constraint that chains should have no branching. This results in $A \leftarrow D$ and $B \leftarrow C$ as two separate chains . If we instead had decided to unlink D from C (by first selecting D and then C) we would be left with $A \leftarrow B \leftarrow C$, and D.

Just select the two words you wish to link, or unlink. The action of linking or unlinking is based on whether there exists a relation or not. If the user selects (*clicks on*) two noun phrases (grey rectangles) that are already related, the existing relation is removed. If the user clicks on two noun phrases that are *not* related, a relation between the noun phrases is added. When a relation is added any conflicting relations are automatically removed. Conflicts arise when relations form branches. The system allows relations to form chains, but not branches, i.e, a noun phrase cannot have more than one antecedent and not more than one anaphor. Another constraint is that relations can only run in one direction, i.e, anaphor follows antecedent (and clicking the two noun phrases in different order will not change that).

Enforcing the structure of coreference chains facilitates editing the chains. For example, our current rules do not allow chains to branch, which then implies unchaining intermediate candidates when needed.

The user can save the result at any point. The result is saved as a file to the user's computer, with a copy stored at the server, thus helping us to extend our database. The user can later upload her/his file for further editing.

After the editor is satisfied with the result, the annotated text can be stored in a format that includes the efforts of the mechanisms used (e.g. part-of-speech tagging, and functional roles) as well as the final annotation of coreference. The final result is presented as an xml-tagged file, which is available for both the editor, and the research group that provided the mechanism over the net. We hope this will lead to increased co-operation in creating larger databases of coreference in multiple languages. We are tentatively investigating the possibility to bootstrap models for newly added languages, by utilizing examples from other analyzed languages. This seems feasible, since most of the features we are currently using are based on the concept of match, and not on storing the feature value itself.

6.2. The demonstrator as an editor for other projects

It is possible to use the editing capabilities of our demonstrator as a front end for other projects. Editing is alternatively made possible from a xml-file of the correct format, and thus any project that needs a graphical editor for chain structures might use our demonstrator to do the work. The user who wants to use the editor for other projects should beware that ”difficult” attribute values (characters like ’<’, ’>’ and ’&’, ’) are encoded as html entities (’&lt;’, ’&gt;’ and ’&amp;’). The system has not been tested with editing on data without ’feature’ and ’lemma’ attributes.

Example xml input: *Grunnen er et kjemisk middel* The reason is a chemical substance

```
<?xml version=”1.0” encoding=”UTF-8”?>
<data>
<br/>

<np id=”np0”>
  <word features=”subst mask be ent” id=”w0” lemma=”grunn”>
      Grunnen
   < /word>
< /np>

   <word features=”verb” id=”w1” lemma=”være”>
      er
   < /word>

<np id=”np1”>
      <word features=”det nøyt ent kvant @det&gt;” id=”w2”
lemma=”en”>
      et
   < /word>
     <word features=”adj nøyt ub ent pos @adj&gt;” id=”w3”
lemma=”kjemisk”>
      kjemisk
   < /word>
    <word features=”subst appell nøyt ub ent @s-pred” id=”w4”
lemma=”middel”>
      middel
   < /word>
< /np>

< /data>
```

7. Conclusion

We have shown a demonstrator for handling coreference and anaphora. The demonstrator is mainly targeted at Norwegian bokmål. The automatic annotation of anaphora is work in progress. Improvement of this subtask will need lexical and semantic resources with a high coverage, and we have pointed out the web as a likely source of such information. The demonstrator contains a graphical editor that can be used to quickly edit reference chains. The results are stored at the server so that we can use the results to improve our database, and it is also sent to the user as a community service. There is also a possibility to use our demonstrator to edit any chain structure which is encoded in the correct format in an xml-file.

Acknowledgements The BREDT project is supported by a grant from the Norwegian Research Council under the KUNSTI programme to the last author. The first author is supported by a Ph.D. grant from the University of Oslo. A demonstrator of early results for anaphora resolution is available from http://bredt.uib.no

8. References

Chinatsu Aone and S. Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 122–129.

C. Cardie and K. Wagstaff. 1999. Noun phrase coreference as clustering. In *Proc. of the joint SIGDAT Conf. on Empirical Methods in NLP and Very Large Corpora*, pages 82–89.

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van der Bosch. 2004. TiMBL: Tilburg Memory-Based Learner, Version 5.1, Reference Guide. Technical Report ILK 04–02, the ILK Group, Tilburg, the Netherlands.

B.J. Grosz, A. Joshi, and S. Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

K. Hagen, J. Bondi Johannessen, and A. Nøklestad. 2000. A Constraint-Based Tagger for Norwegian. In C.-E. Lindberg and S. Nordahl Lund, editors, *17th Scandinavian Conference of Linguistics*, volume 19(I) of *Odense Working Papers in Language and Communication*, Odense.

J. Hale and E. Charniak. 1998. Getting Useful Gender Statistics from English Text. Technical Report CS-98-06, Comp. Sci. Dept. at Brown University, Providence, Rhode Island.

J.R. Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:311–338.

J. Bondi Johannessen, K. Hagen, Å. Haaland, A. Björk Jónsdottir, A. Nøklestad, D. Kokkinakis, P. Meurer, E. Bick, and D. Haltrup. 2005. Named entity recognition for the mainland scandinavian languages. *Literary and Linguistic Computing*, 20(1):91–102.

C. Johansson and A. Nøklestad. 2005. Detecting Reference Chains in Norwegian. In *Proceedings of the 15th NoDaLiDa Conference*, pages 1–10, Joensuu, Finland. University of Joensuu electronic publications in linguistics and language technology.

D. Jurafsky and J.H. Martin. 2000. *Speech and Language Processing*. Prentice Hall, New Jersey.

C. Kennedy and B. Boguraev. 1996. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics.*, Copenhagen, Denmark.

S. Lappin and H. J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

K. Markert and M. Nissim. 2005. Comparing knowledge sources for nominal anaphora resolution. *Computational Linguistics*, 31.

K. Markert, N. Modjeska, and M. Nissim. 2003. Using the web for nominal anaphora resolution. In *EACL Workshop on the Computational Treatment of Anaphora*, citeseer.ifi.unizh.ch/markert03using.html.

J. F. McCarthy and W. G. Lehnert. 1995. Using decision trees for coreference resolution. In C. Mellish, editor, *Proceedings of the Fourteenth International Conference on Artificial Intelligence*, pages 1050–1055.

Wee Meng Soon, Hwee Tou Ng, and D. Chung Yong Lim. 2001a. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

W.M. Soon, H.T. Ng, and D. Lim. 2001b. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.