A Graph Partitioning Approach to Coreference Resolution

Emili Sapena, Lluís Padró, Jordi Turmo
TALP Research Center
Universitat Politècnica de Catalunya
Barcelona, Spain
{esapena, padro, turmo}@lsi.upc.edu

Abstract

This report presents a graph partitioning approach given a set of constraints to resolve coreferences. Coreference resolution is the task of determining which referring expressions in a discourse refer to the same entity. Coreference resolution is a natural language processing task which has a direct effect on the field of Text Mining and its related areas such as Information Extraction, Question Answering, Summarization, Machine Translation.

This report summarizes the research done in coreference resolution and presents our machine learning graph-based system and our baseline with preliminary results in comparison with other machine learning systems in the state of the art.

1 Introduction

Coreference resolution is a natural language processing (NLP) task which has a direct effect on the field of Text Mining and its related areas that need a discourse interpretation such as Information Extraction, Question Answering, Summarization, Machine Translation and so on. Furthermore, in order to *understand* a text document or even a speech, it is mandatory to resolve its coreferences.

Coreference resolution is the task of determining which mentions in a discourse refer to the same entity. A mention, normally a noun phrase (NP), is a referring expression having an entity as a referent. Coreference chains are groups of referring expressions having the same referent. The goal of a coreference resolution system is to find coreference chains given an arbitrary text as input.

First machine learning systems developed for coreference resolution were based on pairwise classifiers using decision trees (DT) (normally C4.5 or C5: (Quinlan, 1993)) (McCarthy and Lehnert, 1995; Soon et al., 2001; Ng and Cardie, 2002). Each pair of mentions found in the document (following an arbitrary order) is considered as possible coreferential pair. Then the DT classifies each pair as coreferential or not. Once all pairs are classified, implicitly, a single-link clustering is done producing final coreferential chains.

Groupwise approaches such as graph partitioning and clustering are a natural evolution of pairwise classifiers in order to resolve corefences (Nicolae and Nicolae, 2006; Culotta et al., 2007; Klenner and Ailloud, 2008). A set of advantages are easily incorporated when resolving coreferences as groups. Viewing it from groups point of view one can avoid contradictions in the results and lacks of information found when classifying by pairs.

We propose a graph representation of the problem reducing coreference resolution to a graph partitioning problem given a set of constraints. In this report we introduce our graph partitioning approach and the experiments done.

The report is organized as follows. In Section 2 we explain our implementation of a pairwise approach similar to some references in the state of the art. That system will be used as baseline. Section 3 introduces the graph partitioning approach and the algorithms proposed for resolution. Next, Section 4 describes our proposed system. Finally, last sections explain our experiments, results and conclusions.

2 Pairwise Approach

In order to evaluate the expected improvement of graph partitioning against pairwise approach, we have developed a baseline based on the later.

Pairwise approach for coreference resolution considers each pair of mentions as a possible coreferential pair. A set of feature functions evaluates pair compatibility, each one according to its own criterion. Then, taking account of feature functions returned values, the classifier assigns each pair as coreferential or not. Most of the published systems for coreference resolution based on pairwise approach used decision trees for classification (McCarthy and Lehnert, 1995; Soon et al., 2001; Ng and Cardie, 2002). Many other works have trained a pairwise classifier different of decision trees such as RIPPER (Ng and Cardie, 2002), maximum entropy (Denis and Baldridge, 2007; Ji et al., 2005) or Support Vector Machines (Yang et al., 2006). The baseline developed in our work is based in Soon et al. (2001) and Ng and Cardie (2002), which use decision trees. Table 1 show the feature functions used in our systems.

Training set creation is done as follows. Each pair of mentions annotated as coreferential in training corpus generates several training instances. The number depends on the number of candidate mentions obtained by the preprocess. Concretely, if $mention_a$ and $mention_b$ are annotated as coreferential, the pair $mention_a - mention_b$ is a positive example and each $mention_i$ between $mention_a$ and $mention_b$ in the document generates a pair $mention_i - mention_b$ which is a negative example for training. This training set is used in order to learn a DT.

The resolution is done evaluating in order each pair of mentions using the learnt DT. Each mention is evaluated with all the previous mentions found in the document. The process starts at the beginning of the document, concretely with the second found mention. Pairs are classified with a confidence value and not with straightforward binary decisions. It means that even when a positive match is found the system keep seeking for another one with higher confidence value. Once all pairs are classified, implicitly, a single-link clustering is done producing final coreferential chains.

3 Graph Partitioning Approach

Coreference resolution problem consists of a set of references to entities (mentions) that have to be mapped to the minimal collection of individual entities. Representing the problem in a graph we are reducing coreference resolution to a graph partitioning problem given a set of constraints. At the end of the process, every partition will be a group of mentions refererring to the same entity, i.e. a coreference chain. Viewing it from groups point of view one can avoid contradictions in the results and lacks of information found when classifying by pairs.

In this section we first define the graph partitioning problem and then the algorithms proposed for resolution.

3.1 Problem Definition and Representation

Let $G = G(V, E)$ be an undirected graph where V is a set of vertices and E a set of edges. Each mention in our data is represented as a vertex $v \in V$ in the graph and an edge $e \in E$ is added to the graph for every pair of vertices representing mentions which can potentially be the same entity.

The set of constraints between two mentions is used to compute a weight value in each edge, which indicates how sure we are that the mentions represented by the two adjacent vertices may be referring to the same entity. Negative weights indicates that the involved mentions should not be in the same partition.

Let $\mathbf{x} = (x_1, ...x_n)$ be the set of mentions to resolve. For each x_i, a vertex v_i is added to the graph. The mentions may have some attributes and we write them as $\mathbf{x}_i = (x_i.a_1, x_i.a_2, x_i.a_3, ...)$ where, for instance, when x_i is a pronoun, $x_i.a_1$ is the attribute *gender* and $x_i.a_2$ is the attribute *number*.

Edges weight for the graph partitioning is obtained before resolution like follows:

$$e_{ij}.weight = \sum_k \lambda_k f_k(\mathbf{x}_i, \mathbf{x}_j) \qquad (1)$$

where $f_k(\cdot)$ is a function that evaluates constraint k. It may use the information of the mentions x_i and x_j. And λ_k is the weight applied to the function.

Feature Function	Description
DIST	Distance between $mention_i$ and $mention_j$ in sentences: number
NUMBER	The number of both mentions match: y,n,u
SEMCLASS	Semantic class of both mentions match: y,n,u (the same as Soon et al. (2001))
GENDER	The gender of both mentions match: y,n,u
PROPER_NAME	Both mentions are proper names: y,n,u
ALIAS	One mention is an alias of the other (only entities, else uknown): y,n,u
APPOSITIVE	One mention is in apposition with the other: y,n
PRO_STR	Both are pronouns and their strings match: y,n
PN_STR	Both are proper names and their strings match: y,n
SOON_STR_NONPRO	String matching like in Soon et al. (2001) and mentions are not pronouns
AGREEMENT	Gender and number of both mentions match: y,n,u
NESTED	One mention is included in the other: y,n
I_PERSON	$mention_i$ is a person (pronoun or proper name in a list): y,n
J_PERSON	$mention_j$ is a person (pronoun or proper name in a list): y,n
ANIMACY	Animacy of both mentions match (persons, objects): y,n
HEAD_MATCH	String matching of NP heads: y,n
MAXIMALNP	Both mentions have the same NP parent or they are nested: y,n
J_INDEF_NP	$mention_j$ is an indefinite NP: y,n
I_EMBEDDED	$mention_i$ is a noun and is not a maximal NP: y,n
BINDING	Conditions B and C of binding theory: y,n
I_TYPE	$mention_i$ is a pronoun (p), entity (e) or nominal (n)
J_TYPE	$mention_j$ is a pronoun (p), entity (e) or nominal (n)

Table 1: Feature functions used in our systems.

3.2 Algorithms

We propose the use of two algorithms for coreference resolution reduced to a graph partitioning problem given a set of constraints. The reason to compare a deterministic algorithm (Relax) with a probabilistic one (Ant) is the scalability. While a deterministic algorithm can ensure that the result is the best possible, for larger datasets it needs more resources and might be intractable. On the contrary, a probabilistic algorithm like Ants can achieve good performance (not the optimal) besides computational cost issues.

3.2.1 Relaxation Labeling Algorithm

Relaxation is a generic name for a family of iterative algorithms which perform function optimization, based on local information. They are closely related to neural nets and gradient step.

Although relaxation operations had been long used in engineering fields to solve systems of equations, they didn't got their biggest success until (Rosenfeld et al., 1976) applied their extension to symbolic domain –relaxation labeling– to constraint propagation field, specially in low-level vision problems.

In the Artificial Intelligence field, relaxation has been mainly used in computer vision –since it is where it was first used– to address problems such as corner and edge recognition or line and image smoothing. Nevertheless, many traditional AI problems can be stated as a labeling problem: the traveling salesman problem, n-queens, or any other combinatorial problem. The algorithm also has been widely used to solve NLP problems such as from PoS-tagging (Màrquez et al., 2000), chunking, knowledge integration, and Semantic Parsing (Atserias, 2006).

Relaxation labeling (Relax) solves our weighted constraint satisfaction problem dealing with edge weights as *compatibility coefficients*. Each vertex is assigned to a partition satisfying as many constraints as possible.

3.2.2 Ants Algorithm

The ants algorithm is a multiagent system based on the idea of parallel search. A generic version of the algorithm was proposed in (Comellas and Ozon, 1998). The algorithm faces the problem as a graph coloring problem, optimizing a global fitness function. In theoretical computer science, *"graph coloring"* usually refers to a very specific constraint satisfaction problem: assigning colors to vertices such that no two adjacent vertices have the same color. However, this algorithm is more general and optimizes a global fitness function using colors as a vertex state, and using local fitness function to decide the color of each vertex. Playing with local and global fitness functions one can adapt the algorithm to solve almost any problem of constraint satisfaction.

The algorithm works as follows. Initially, all vertices are randomly colored and a given number of agents (ants) is placed on the vertices, also at random. Then the ants move around the graph and change the coloring according to a local optimization criterion. The local and global fitness functions depend on the problem to solve and are the only part that normally needs adaption.

Each movement or decision taken by an ant has a probability of error, which prevents the algorithm falling in local minima.

The adaption of the algorithm to our coreference resolution task is done by finding correct global and local fitness functions. Local *fitness* function is defined as:

$$Fit(v) = \frac{\sum_{i=0}^{m-1} e_i.weight - \sum_{j=m}^{l-1} e_j.weight}{\sum_{i=0}^{l-1} |e_i.weight|} \quad (2)$$

where vertex v has l adjacent vertices and $e.weight$ are the values of edge weights. From 0 to $m-1$ are the edge weights corresponding to the adjacent vertices with the same color and from m to $l-1$ are the ones corresponding to adjacent vertices with different color. Note that the values of the edge weights can be negative.

The global fitness function is then the sum of all the vertices fitness:

$$GlobalFitness = \frac{\sum_{i=0}^{n-1} Fit(v_i)}{n} \quad (3)$$

where n is the total number of vertices. At the end of execution, vertices sharing a color are elements that refer to the same entity.

4 Proposed System

We propose a coreference resolution system based on graph partitioning given a set of constraints as explained in Section 3. In this section, we describe in detail our system preprocess, how the constraints are generated, the training process and the resolution.

4.1 Preprocessing

In order to develop our own coreference resolution system it is mandatory to have a text processing pipeline. In our system we use Freeling (Atserias et al., 2006) for sentence splitting and tokenization, SVMTool (Gimenez and Marquez, 2004) for Part of Speech tagging, BIO (Surdeanu et al., 2005) for Named Entity Recognition and Classification, and an in-house NP-Chunker for detection of mentions. The information obtained in each step is stored using MMAX2 format, which is an XML-based multi-level annotation format (Müller and Strube, 2006).

4.2 Models of Constraints

We propose three different models of constraints, all three using the same feature functions used in the pairwise system. First, Model 1 directly uses each possible value of the feature functions as a constraint. Second, Model 2 learns a decision tree and uses the confidence value returned for each pair of mentions as the edge weight. And third, Model 3 extracts rules from a learnt decision tree and uses each rule as a constraint.

4.2.1 Model 1: Feature functions

Model 1 uses each feature function that applies to two mentions to generate constraints. For instance, *StringMatch* or *GenderMatch* are included, but functions that evaluate some feature of only one mention such as *IsPronoun* or *IsProperName* are not included. For each possible value of a feature function a new constraint is added. The contribution of each constraint for the final edge weight will be determined in the training process. The process to find the optimal constraint weight combination is explained in Section 4.3.2.

4.2.2 Model 2: Confidence value

Model 2 first learns a probabilistic decision tree using all the training mention pairs. The confidence value returned for each pair of mentions is used as the edge weight in the graph for the corresponding pair of adjacent vertices.

4.2.3 Model 3: Extracted rules

Model 3 is divided in two steps. First, a decision tree is generated with a half of the training pairs of mentions. Given that tree, a set of rules are extracted. Each one of these rules is used as a constraint. Second, similar to Model 1, the contribution of each constraint for the final edge weight will be determined in the training process. However, in this model only the unseen part of training pairs (the other half) is used here.

4.3 Training Method

Depending on the chosen constraint model, the training set is used for learning constraint weights, to induce a decision tree or both. The training set consists of a set of instances with the returned values of all the feature functions. Each instance corresponds to a pair of mentions found in the training documents. The following two subsections explains how the training set is generated and the learning process used to find the optimal constraint weight combination.

4.3.1 Generating Training Examples

The training set generation process is done applying all the feature functions for a set of mention pairs of the training documents. Each mention in the training documents annotated as coreferential (true mentions) forms a pair with all the previous mentions found in the document (system mentions). When both mentions corefer they form a positive sample, while the others are negative ones. For experiments using true mentions only true mentions are used also for training.

4.3.2 Finding Optimal Weights

The performance of the graph partitioning algorithms used here (relax and ants) depends on the edge weights which, at the same time, depend on the constraint weights. In order to achieve good performance, it is mandatory to find a good constraint weight combination. Searching the space of weight combinations is intractable here by an exhaustive search. Therefore, we use Genetic Algorithms for this task (Goldberg, 1989). Genetic Algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination). Other works have also used evolutionary algorithms to train similar processes successfully (Pelillo et al., 1995; Sapena et al., 2008).

First, an initial random population of individuals (weight combinations) is generated. Then, each weight combination is evaluated using the selected algorithm (relax or ants) over the training data. The best individuals of the population are selected for the next generation, where new individuals are generated using the previous survivors as parents. After an arbitrary number of generations the best individual is selected as our best weight combination.

4.4 Resolution

The input document is processed using the preprocess pipeline (Section 4.1) and a graph is generated with all the mentions found, where every mention is adjacent with all the others. Depending on the model, the corresponding constraints are applied for each pair of mentions. The weight of each edge is determined using the constraint weight combination found in the training process. Finally, the graph partitioning problem is solved using one of the algorithms of Section 3.2 (relax or ants).

5 Evaluation framework

We evaluate our approach to coreference resolution using ACE-phase02 corpus, which is composed of three sections: Broadcast News (BNEWS), Newswire (NWIRE) and Newspaper (NPAPER). Each section is in turn composed of a training set and a test set. To score the output of our system we use Constrained Entity-Alignment F-Measure (CEAF) (Luo, 2005). CEAF is computed based on the best one-to-one map between key coreference chains and response ones. We use two versions of the metric: entity-based and mention-based. The difference between these two versions is the similarity metric. Mention-based metric simply counts the number of common mentions shared by key coreference chains and response ones, while entity-based metric is the mention F-measure between coreference chains in the key and the response.

6 Experiments and Results

In order to evaluate our proposed system we have done some experiments with the following goals. First, our processing system (preprocess, and mention detection) and baseline should achieve performances comparable with the state-of-the-art ones. Second, we expect that our graph-based proposed system achieve similar or even better performances than the baseline. Finally, we want to study the learning process of our system.

In the first experiment we evaluate our system developing a pairwise solution as is explained in Section 2. As we are using the same corpus and metric, we compare our results with Ng (2008), which uses as supervised learning system an improved version of our reference for the baseline

True mentions mention-based CEAF	BNEWS R	P	F_1	NWIRE R	P	F_1	NPAPER R	P	F_1
Ng08 supervised	61,30	61,30	61,30	64,20	64,20	64,20	-	-	-
Baseline	60,86	60,86	60,86	57,20	57,20	57,20	51,45	51,45	51,45

Table 2: Results of the first experiment using true mentions.

System mentions mention-based CEAF	BNEWS R	P	F_1	NWIRE R	P	F_1	NPAPER R	P	F_1
Ng08 supervised	65,80	63,20	64,50	63,40	60,30	61,80	-	-	-
Baseline	44,61	33,43	38,22	46,71	34,86	39,92	43,07	28,32	34,17

Table 3: Results of the first experiment using system mentions.

True mentions mention-based CEAF	BNEWS R	P	F_1	NWIRE R	P	F_1	NPAPER R	P	F_1
Ng08 supervised	61,30	61,30	61,30	64,20	64,20	64,20	-	-	-
Baseline	60,86	60,86	60,86	57,20	57,20	57,20	51,45	51,45	51,45
Model 3 (No Train)	54,79	69,44	61,25	52,22	70,33	59,94	49,38	61,80	54,89
Model 3 (Trained)	56,82	69,49	62,52	54,60	70,86	61,68	49,84	62,47	55,45

Table 4: Results of the second experiment using mention-based CEAF metric.

True mentions entity-based CEAF	BNEWS R	P	F_1	NWIRE R	P	F_1	NPAPER R	P	F_1
Baseline	68,12	43,35	52,98	63,24	36,56	46,33	55,33	33,49	41,73
Model 3 (No Train)	57,75	64,33	60,86	51,04	59,70	55,03	48,02	53,71	50,70
Model 3 (Trained)	59,52	64,61	61,96	50,03	63,15	55,83	48,52	54,77	51,45

Table 5: Results of the second experiment using entity-based CEAF metric.

(Ng and Cardie, 2002), and also evaluates an unsupervised EM-based model. In this experiment we resolve coreferences in two ways. First, using our process system and mention detection (*system mentions*). And second, using *true mentions*, it means, the annotated mentions for train or test proposes.

Table 2 shows how our baseline achieves reasonable results and comparable to state-of-the-art ones when solving coreferences using true mentions, which are exactly the same for our system and for the reference system (Ng08). The differences between Ng's system and ours are not only due to a slightly different set of feature functions and their implementation, but also because Ng's 2008 system does a bell-tree with pair confidence values in spite of our single-link clustering.

Table 3 shows that our baseline performance heavily decreases when using our system mentions. Our mention detection system must be improved. Consequently, following experiments have been done only over true mentions.

In the second experiment we score the output of the Model 3 of our proposed system using Relax algorithm for resolution. We compare the performances with the baseline and the model itself with and without training. The system without training uses the same weight for each constraint, while the trained system uses the best weight combination found.

Table 4 shows that our proposed system significantly outperforms the baseline in NWIRE and NPAPER sections while the improvement in BNEWS is not significant (± 2 corresponds to a confidence level of 98%). Using entity-based CEAF metric, our graph-based system clearly outperforms the pairwise baseline (Table 5). However, the training process is not as helpful as we expected initially. The performances in test set are not significantly better than the ones obtained without training. This might be a consequence of an overfitting in the training set.

7 Conclusions and Future Work

We have developed a machine learning coreference resolution system that reduces the task to a graph partitioning problem given a set of constraints. We have also developed a baseline system based on classification by pairs using decision trees. The experiments show that the performance of the baseline is comparable to the others in the state of the art, and our proposed solution outperforms our baseline using the same features, i.e. the same information.

However, we have to take care about two important matters. First, our mention detection system detects too many mentions and it causes an important decrease in the resolution performance. We should refine our in-house NP-chunker and maybe some processes of the preprocess pipeline. Second, the genetic algorithms training in our proposed system does not achieve significant improvements in test dataset, which implies an overfitting in the training. More research is needed in this step of our proposed system.

References

Atserias, Jordi, Bernardino Casas, Elisabet Comelles, Meritxell Gonzlez, Llus Padr, and Muntsa Padr. 2006. Freeling 1.3: Syntactic and semantic services in an open-source nlp library. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC 2006), ELRA*. Genoa, Italy.

Atserias, Jordi. 2006. *Towards Robustness in Natural Language Understanding.* Ph.D. Thesis, Dept. Lenguajes y Sistemas Informáticos. Euskal Herriko Unibertsitatea. Donosti. Spain.

Comellas, F. and J. Ozon. 1998. An ant algorithm for the graph colouring problem. In *ANTS'98 - From Ant Colonies to Artificial Ants: First international workshop on ant colony optimization, Brussels. 1998.*

Culotta, A., M. Wick, and A. McCallum. 2007. First-Order Probabilistic Models for Coreference Resolution. *Proceedings of NAACL HLT*, pages 81–88.

Denis, P. and J. Baldridge. 2007. Joint Determination of Anaphoricity and Coreference Resolution using Integer Programming. *Proceedings of NAACL HLT*, pages 236–243.

Gimenez, J. and L. Marquez. 2004. Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43–46.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Ji, H., D. Westbrook, and R. Grishman. 2005. Using semantic relations to refine coreference decisions. *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 17–24.

Klenner, M. and É. Ailloud. 2008. Enhancing Coreference Clustering. In *Proceedings of the Second Workshop on Anaphora Resolution*. WAR II.

Luo, X. 2005. On coreference resolution performance metrics. *Proc. of HLT-EMNLP*, pages 25–32.

Màrquez, Lluís, Lluís Padró, and Horacio Rodríguez. 2000. A machine learning approach for pos tagging. *Machine Learning Journal*, 39(1):59–91.

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

Müller, Christoph and Michael Strube. 2006. Multi-level annotation of linguistic data with MMAX2. In Braun, Sabine, Kurt Kohn, and Joybrato Mukherjee, editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany.

Ng, V. and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 104–111.

Ng, Vincent. 2008. Unsupervised models for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. EMNLP-08.

Nicolae, C. and G. Nicolae. 2006. Best Cut: A Graph Algorithm for Coreference Resolution. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 275–283.

Pelillo, Marcello, Fabio Abbattista, and Angelo Maffione. 1995. An evolutionary approach to training relaxation labeling processes. *Pattern Recogn. Lett.*, 16(10):1069–1078.

Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rosenfeld, R., R. A. Hummel, and S. W. Zucker. 1976. Scene labelling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6(6):420–433.

Sapena, E., L. Padró, and J. Turmo. 2008. A Graph Partitioning Approach to Entity Disambiguation Using Uncertain Information. In *Proceedings of the 6th international conference on Advances in Natural Language Processing*, pages 428–439. Springer-Verlag Berlin, Heidelberg.

Soon, W.M., H.T. Ng, and D.C.Y. Lim. 2001. A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics*, 27(4):521–544.

Surdeanu, M., J. Turmo, and E. Comelles. 2005. Named Entity Recognition from Spontaneous Open-Domain Speech. In *Ninth European Conference on Speech Communication and Technology*. ISCA.

Yang, X., J. Su, and C.L. Tan. 2006. Kernel-based pronoun resolution with structured syntactic knowledge. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 41–48.