CCSE Computer and Information Science
www.ccsenet.org/journal.html
Vol. 2, No. 4
November 2009

Unsupervised Coreference Resolution with HyperGraph Partitioning

Jun Lang (Corresponding author)
School of Computer Science and Technology
Harbin Institute of Technology
PO box 321, Harbin 150001, China
Tel: 86-451-8641-3683 E-mail: bill_lang@ir.hit.edu.cn

Bing Qin
School of Computer Science and Technology
Harbin Institute of Technology
PO box 321, Harbin 150001, China
Tel: 86-451-8641-3683 E-mail: qinb@ir.hit.edu.cn

Ting Liu
School of Computer Science and Technology
Harbin Institute of Technology
PO box 321, Harbin 150001, China
Tel: 86-451-8641-3683 E-mail: tliu@ir.hit.edu.cn

Sheng Li
School of Computer Science and Technology
Harbin Institute of Technology
PO box 321, Harbin 150001, China
Tel: 86-451-8641-3683 E-mail: lis@ir.hit.edu.cn

The research is supported by National Natural Science Foundation of China (60675034, 60803093), National High Technology Research and Development Program of China (863 Program) (2008AA01Z144). (Sponsoring information)

Abstract

Unsupervised-learning based coreference resolution obviates the need for annotation of training data. However, unsupervised approaches have traditionally been relying on the use of mention-pair models, which only consider information pertaining to a pair of mentions at a time. In this paper, it is proposed the use of hypergraph partitioning to overcome this limitation. The mentions are modeled as vertices. By allowing a hyperedge to cover multiple mentions that share a common property, the additional information beyond a mention pair can be captured. This paper introduces a hypergraph partitioning algorithm that divides mentions directly into equivalence classes representing individual entities. Evaluation on the ACE dataset shows that our unsupervised hypergraph based approach outperforms previous unsupervised methods.

Keywords: Coreference resolution, HyperGraph partitioning, Unsupervised learning

1. Introduction

Coreference resolution is the process of partitioning mentions into different real world entities. It is a key component of many Natural language processing (NLP) applications. Especially, due to its important role in Information extraction

(IE), coreference resolution was defined as an IE subtask and officially evaluated in the Message Understanding Conference (MUC) and Automatic Content Extraction (ACE) programs. So far, supervised-learning-based approaches have been widely applied to coreference resolution, which requires a set of training data to build a classifier for coreference judgment (Soon et al., 2001; Ng and Cardie, 2002). However, coreference annotation is a difficult task, which involves not only deep linguistic knowledge, but also background knowledge related to the domain. For this reason, the size of existing annotated coreference corpora is quite small (e.g., 599 documents in the ACE2005 corpus) compared with other NLP tasks, and is limited only in some specific domains.

To deal with the lack of the training data, several unsupervised approaches were proposed which require no training data for coreference resolution and are adaptive to different domains. For example, Cardie and Wagstaff (1999) suggested recasting coreference resolution to a clustering problem, which tries to group noun phrases into different coreference clusters. They defined a distance function to measure the incompatibility of two mentions. Given a document, mentions are processed backwards one by one. Two mentions are placed into the same cluster if their distance is below a threshold, and no mentions from their respective clusters are incompatible. Wagstaff (2002) further enhanced this method by adding more linguistic constraints (must-link and cannot-link) during clustering.

However, there are several problems with the previous clustering based unsupervised methods.

(1) As with many other learning based approaches to coreference resolution (e.g., Soon et al. (2001), and Ng and Cardie (2002)), these methods adopt a mention-pair model. The distance function is only based on the information of two given mentions. However, as individual mentions lack adequate information about the entities they refer to, the distance may be not accurate to represent the (in)compatibility of two mentions. For example, the compatibility between mentions "*Powell*" and "*She*" may be different, depending on the gender information of "*Powell*" which cannot be determined from the mention alone.

(2) As the clustering is agglomerative, the wrong linking decision could not be undone and would lead to cascading errors. Suppose we have three mentions "*Mr. Powell*", "*Powell*", "*She*". If "*She*" is wrongly linked to "*Powell*", the cluster cannot be broken and will prevent the subsequent linking of "*Powell*" with "*Mr. Powell*".

To overcome the above problems, this paper proposes an unsupervised coreference resolution approach with hypergraph partitioning. Hypergraph is a special graph in which an edge connects more than two vertices (Berge, 1989). To model coreference resolution, mentions could be viewed as vertices. A set of mentions is covered by a hyperedge if they show a specific common property. As a hyperedge can describe information shared by two or more mentions, it has a more powerful representation capability for knowledge than a traditional mention-pair feature. By using a partitioning algorithm, mentions are divided into equivalence partitions representing individual entities. The partitioning process can avoid the cascading errors in the clustering-driven unsupervised approaches. In our experiments, we evaluated our approach on the ACE data and our experimental results show that our approach is effective for coreference resolution.

The following sections are organized as following: Section 2 describes some related works for coreference resolution and hypergraph with its applications. Section 3 introduces the basic concepts of hypergraph and the partitioning algorithm. Section 4 describes the hypergraph-based model for coreference resolution. Section 5 gives the experimental results with some discussions. Finally, section 6 summarizes the conclusion and presents future works.

2. Related work

Supervised-learning-based approaches are widely adopted in coreference resolution. It was first proposed by using decision tree approach (McCarthy and Lehnert, 1995), and later many other systems follow. A typical one of them is presented by (Soon et al., 2001). In it, coreference resolution is deemed as a classification problem. A training or testing instance is formed by two mentions, with a feature vector describing their properties and relationships, including the information of gender, number, person, semantic, string match, appositive, name alias, and so on. When testing, a mention to be resolved is checked against its preceding mentions, and is linked with the closest one that is classified as positive. The work is further enhanced by expanding the feature set and adopting "best-first" linking strategy (Ng and Cardie, 2002).

Such a mention-pair-based model only considers information related to two mentions in question, and would cause triangular contradiction errors at a testing time. Suppose we have three mentions "Mr. Powell", "Powell", and "she" in a document. The model tends to link "*she*" with "*Powell*" because of their proximity, and link "*Mr. Powell*" with "*Powell*" since head string matching. Merging the two pairs together, nevertheless, would lead to gender disagreement between "*she*" and "*Mr. Powell*".

Several researchers proposed to use graph theory to deal with the triangular contradiction errors in coreference resolution. They converted a document to a graph in which mentions in the document are mapped to vertices in the graph. An edge connecting two vertices represents the coreference relationship between the two corresponding mentions. The weight of an edge accounts for the confidence of the coreference relationship and is derived from coreference classification. Then, some graph partitioning algorithms can be used for global optimization, such as BESTCUT

(Nicolae and Nicolae, 2006). Similarly, Bell-Tree global searching (Luo et al., 2004) and triangular contradiction constraint learning with Conditional Random Field (McCallum and Wellner, 2003) are proposed for such problem. However, they all are supervised learning methods.

Hypergraph has shown many advantages in clustering and classification problems (Zhou et al., 2006). In recent years, it is also employed in NLP applications like sentence parsing (Klein and Manning, 2001; Huang, 2008), word sense disambiguation (Klapaftis and Manandhar, 2007) and document clustering (Shinnou and Sasaki, 2007). However, to our knowledge, our work is the first effort to adopt this technique to the coreference resolution task.

3. Basic concepts of hypergraph

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a finite set, $H = \{E_1, E_2, \ldots, E_m\}$ be a family of subsets of X. The family H is said to be a hypergraph on X if

$$E_i \neq \phi (i = 1,2,\ldots,m) \tag{1}$$

$$\bigcup_{i=1}^{m} E_i = X \tag{2}$$

$H = (X\colon E_1, E_2, \ldots, E_m)$ is called a *hypergraph*. $|X| = n$ is the order of the hypergraph. The elements $x_1, x_2, \ldots, x_n$ are *vertices* and the sets $E_1, E_2, \ldots, E_m$ are called *hyperedges*.

An example hypergraph is shown in Figure 1. An edge E_i with $|E_i| > 2$ is drawn as a curve encircling all of its vertices. An edge E_i with $|E_i| = 2$ is drawn as a line connecting its two vertices. An edge E_i with $|E_i| = 1$ is drawn as a loop as in a graph. If $|E_i| \leq 2$ for all i, a hypergraph is reduced to a common graph. In a hypergraph, two vertices are said to be adjacent if there is a hyperedge E_i that contains both of these vertices. Two hyperedges are said to be adjacent if their intersection is not empty.

The incidence matrix of hypergraph $H = (X\colon E_1, E_2, \ldots, E_m)$ is a matrix $A = (a_{ij})$ with m rows that represent the hyperedges of H and n columns that represent the vertices of H, that is,

$$a_{ij} = \begin{cases} 1, if\ x_j \in E_i \\ 0, if\ x_j \notin E_i \end{cases} \tag{3}$$

Each (0, 1)-matrix is an incidence matrix of a hypergraph if no row or column contains only zeros. For illustration, the hypergraph of Figure 1 can be converted to the following one.

$$A = \begin{array}{c|cccccccc|} & x_1 & x_1 & x_1 & x_1 & x_1 & x_1 & x_1 & x_1 \\ E_1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ E_2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ E_3 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ E_4 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ E_5 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ E_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array} \tag{4}$$

There exist quite a few hypergraph partitioning algorithms that have been proved effective in different practical problems, such as partitioning circuit netlists, clustering categorical data, and segmenting images. In our study, we chose hMETIS (2.0pre1) (Note 1) which is capable of providing high quality partitions with a high speed. The algorithm in hMETIS is based on multilevel hypergraph partitioning algorithm (Selvakkumaran and Karypis, 2006). In our study, we used the direct k-way partitioning scheme of hMETIS. The overall quality of the obtained partitioning can be computed using the following quality measures (Note 2):

3.1 Scaled Cost: defined for *k-way* partitioning as

$$\frac{1}{n(k-1)} \sum_{i=1}^{k} \frac{|E(P_i)|}{|P_i|} \tag{5}$$

where $|E(P_i)|$ is the number of hyperedges that are incident but not fully contained inside the partition P_i.

3.2 Absorption: This is defined as

$$\sum_{i=1}^{k} \sum_{e \in E | e \cap P_i \neq \phi} \frac{|e \cap P_i| - 1}{|e| - 1} \tag{6}$$

where E is the set of hyperedges, $|e \cap P_i|$ is the number of vertices of a hyperedge e in partition P_i, and $|e|$ is the number of vertices of e.

The hMETIS program performs partitioning by minimizing the *Scaled Cost*, while maximizing *Absorption* at the same time.

4. HyperGraph modeling for coreference resolution

To recast coreference resolution to a hypergraph partitioning problem, we view mentions as vertices, and use various kinds of knowledge for coreference resolution to create hyperedges. In this section, we will focus on several important aspects of the hypergraph model: designing hyperedges, choosing proper weights of hyperedges, performing partitioning, and setting the stopping criterion.

4.1 Hyperedges

Traditional learning based coreference resolution systems represent knowledge in terms of features. For coreference resolution with hypergraph partitioning, however, we represent knowledge with hyperedges. As introduced in Section 3, a hyperedge is a special edge that covers more than one vertex. We can convert mentions in a document to vertices in a hypergraph. Several mentions are thought of being covered by a hyperedge if they share a specific common property. In this way, we can capture the information of multiple mentions at the same time, instead of a mention-pair as in tradition learning-based approaches to coreference resolution. Hence, the hypergraph would provide us a more powerful representation capability for knowledge.

In our study, we define the following types of hyperedges. For illustration, we use the text in Table 1 as an example.

1) *FullString*: This type of hyperedge covers the mention vertices that have the same string (excluding the determiners). For example, in Table 1, mentions m_2 and m_7 have the same string, and so do mentions m_6 and m_{10}. Then we will have two *FullString* hyperedges that cover $\{m_2, m_7\}$ and $\{m_6, m_{10}\}$, respectively.

2) *Head*: This type of hyperedge covers mentions with the same head string.

3) *Gender*: This type of hyperedge covers the mentions that have the same gender type. To considering only effective partitioning of mentions, there are only two types of gender (Note 3), *Male* and *Female*. A hypergraph has at most two *Gender* hyperedges. A mention with a neuter gender (such as "it", "the president") is not covered by a hyperedge of type *Male* for *Female*. In Table 1, mentions m_4, m_5 and m_9 are male and thus will be covered by a hyperedge {4, 5, 9}.

4) *Number*: This type of hyperedge covers the mentions that have the same number type. A hypergraph may have two *Number* hyperedges for singular or plural mentions.

5) *Person*: This type of hyperedge covers the mentions that have the same person type. There are only two *Person* hyperedges for mentions that are persons or non-persons.

6) *Semantic*: This type of hyperedge covers the mentions that have the same semantic type. A hypergraph may contain five hyperedges of this type for the semantic types *Organization*, *GPE*, *Person*, *Location*, and *Facility* defined in the ACE annotation scheme. Features for semantic types were obtained from the gold annotations. In Table 1, we can get three *Semantic* hyperedges {1, 6, 8, 10, 11}, {2, 7} and {3, 4, 5, 9}, for *Organization*, *GPE* and *Person*, respectively.

7) *ThreeSentences*: This type of hyperedge covers a pronoun (Note 4) and the preceding mentions in the current sentence and previous two sentences. In Table 1, for pronoun its_2, its_7 and He_9, the mentions in the 3-sentence window are $\{m_1 - m_2\}$, $\{m_1 - m_7\}$, $\{m_1 - m_9\}$, respectively. Thus, we have two hyperedges: {1, 2, 3, 4, 5, 6, 7} and {1, 2, 3, 4, 5, 6, 7, 8, 9}. Note that we do not generate a hyperedge {1, 2} as its vertices are fully contained in the hyperedge {1, 2, 3, 4, 5, 6, 7}.

8) *TwoSentences*: This type of hyperedge is similar to *ThreeSentences*, but it just considers mentions within a two-sentence window.

9) *OneSentence*: This type of hyperedge is similar to *ThreeSentences*, but it just considers mentions within the same sentence.

10) *NameAlias*: This type of hyperedge covers the mentions that are name alias of one another in a document. Consider Table 1, $\{m_4, m_5\}$, $\{m_1, m_6, m_{10}\}$ are name-alias groups and thus we can have two *NameAlias* hyperedges {4, 5}, {1, 6, 10}.

11) *Appositive*: This type of hyperedge covers the mentions that are in the same appositive structure.

12) *CannotLink*: As suggested by Wagstaff (2002), we enforce cannot-link constraints during partitioning. For this purpose, we create hyperedges to cover a pair of mentions $<m_i, m_j>$ that are not likely to corefer. In our study, we consider the following constraints:

a. m_j is an indefinite noun phrase.

b. m_i and m_j are three sentences apart and do not have the same head word.

c. m_i and m_j are pronouns and m_i and m_j do not agree in number or gender.

d. m_j is a pronoun and m_i and m_j are three sentences apart.

In Table 1, mention pairs m_2 - m_9 and m_7 - m_9 violate the third constraint and thus are covered by two *CannotLink* hyperedges {2, 9} and {7, 9}, respectively.

The hyperedges generated for Table 1 are in Table 2.

4.2 Weights for hyperedges

We classify the hyperedges into six categories based on their confidence level for a positive coreference determination, as shown in Table 3.

The hypergraph partitioning algorithm tends to divide vertices covered by a hyperedge with a low-weight, and retain in the same partition vertices covered by a high-weight hyperedge. Thus, we manually assign a higher weight to a hyperedge that covers mentions that are likely to corefer, while a lower weight to a hyperedge that covers mentions that are unlikely to corefer. Table 3 shows the different weights for different levels of hyperedges. The hyperedge *CannotLink* was assigned the lowest weight of zero (Note 5). The hyperedges *FullString*, *Appositive,* and *NameAlias*, which are strong indicators of coreference relationship (Soon et al., 2001), were given the highest weight.

4.3 Coreference resolution with mention partitioning

Given a document, all the mentions are placed into a large cluster initially. As described, we map each mention to a vertex in a hypergraph, and find out all the possible hypergraphs for the vertices. Then we can invoke the hMETIS program to perform mention partitioning. The process is done in an unsupervised way. After the partitioning stops, a generated partition could be deemed as a coreferential cluster for a single entity, with all the mentions in the same cluster being coreferential with each other.

4.4 Stopping Criterion

One problem with mention partitioning is when the process should stop. In other partitioning tasks, the number of target clusters is predefined. However, for our task, it is not possible to give a predefined cluster number as the number of entities in a document is unknown before resolution. Therefore, we need to design a stopping criterion for partitioning.

As described in Section 3, to certain partitioning clusters number k, the hMETIS program performs partitioning by minimizing the *Scaled Cost* (5), while maximizing *Absorption* (6) at the same time. After inner optimization, hMETIS would find the best partitioned clusters with the final Scaled Cost and Absorption values, named as $ScaledCost_{final}(k)$ and $Absorption_{final}(k)$ respectively.

Enlarging the target entity number k would make the former value increase while the latter decrease. Without any prior knowledge, we try to find a k which compromise on the two costs varying trends at the same time. For this consideration, we define a stopping criterion based on the product of final generated $ScaledCost_{final}(k)$ and $Absorption_{final}(k)$ values after optimization:

$$P(k) = ScaledCost_{final}(k) * Absorption_{final}(k) \tag{7}$$

We prefer a partition with high product of *ScaledCost* and *Absorption*. For a given document, we put all the mentions in a cluster (k=1) and perform partitioning repeatedly. The process stops at a round when the value of P reaches the peak, or when each cluster contains only one mention. The generated clusters are output as the coreference resolution result. Actually, in our study such stop criterion achieved good result.

5. Experiments and results

5.1 Experimental Setup

In our study, we did evaluation on the ACE-2 V1.0 corpus (NIST, 2003) that is divided into three domains: broadcast news (BNews), newspaper (NPaper), and newswire (NWire). As we conducted unsupervised learning, we did not use the training data and just ran the system on the test data. However, in the comparing supervised systems, the training and testing data were used together. The number of entities with more than one mention, as well as the number of the contained mentions, is summarized in Table 4.

For both training and resolution, an input raw document was processed by a pipeline of NLP modules of OpenNLP (Note 6), including sentence boundary detector, tokenizer, and part-of-speech tagger. The boundaries of a mention are directly from annotation in the corpus. Our experiment setup just follows the official "diagnose" evaluation of ACE in which coreference resolution is done and evaluated on the perfect mentions, which allows the validation of the utility of the hypergraph method under an environment of accurate mention features. We used the mention's head, boundaries, and the semantic type information from "gold" annotation. Other features, like string-matching, apposition, name-alias, distance and so on, were all computed at a running time. Following tradition, all results are reported using recall, precision and F-measure based on the MUC-6 scoring algorithm (Vilain et al., 1995).

5.2 Results and discussion

Table 5 lists the performance of different coreference resolution systems.

For comparison, we first duplicated the traditional unsupervised learning based system by Cardie and Wagstaff (1999) as baseline. The first line of Table 5 shows the results of such a system, which adopted the same clustering radius threshold (i.e., r = 4) as in Cardie and Wagstaff (1999)'s system. Our duplicated system (denoted by Cardie99r4) achieves a recall of 66.30% and a precision of 50.99%, obtaining an F-measure of 57.64%. The F-measure (57.64%) is higher than their results (52.8%) reported on the MUC-6 data.

Cardie and Wagstaff (1999)'s radius value was fined-tuned for the MUC-6 data, and is not necessarily optimal for the ACE data. In our experiments, we examined the performance of the duplicated system under different radius value from 1 to 10. We found that the system achieves the best result when r = 6 (Cardie99r6), with 67.34% recall, 51.73% precision and 58.5% F-measure. The recall, precision, F-measure results for each domain are consistently higher than those of Cardie99r4. It indicates that the performance of their system is significantly affected by the threshold value.

In the experiments, we were also interested in comparing performance difference between the system with unsupervised learning and the system with supervised learning. For this purpose, we implemented the classical decision trees-based coreference resolution system by Soon et al. (2001) (denoted by Soon01), and the results are shown in the third line of Table 5. Compared with Cardie99r6, the system has a drop in recall (up to 11.83%), but achieves a large improvement in precision (up to 21.23%). Overall, it produces an average 55.51% recall, 72.96% precision, and 63.05% F-measure. The F-measure is 4.54% higher than Cardie99r6. This is in line with Wagstaff (2002)'s report that Cardie and Wagstaff (1999)'s unsupervised approach got an F-measure about 9% lower than the supervised system.

The fourth line of Table 5 summarizes the performance of our system with hypergraph partitioning. From the table, the system produces a higher recall of 79.37%, 12.03% than Cardie99r6, with just only 2.87% loss in precision. Overall, the F-measure is about 2% higher than Cardie99r6. The difference against the supervised based system (Soon01) is reduced to 2.57%, and the results are encouraging considering that our approach did not use any training data.

One interesting finding of the table is that unsupervised approaches tend to produce a lower precision but a higher recall than supervised approaches. This should be the case because our hypergraph method is based on top-down partitioning. Mentions tend to be retained in the same cluster unless they have some inconsistency. By contrast, a supervised approach is based on bottom-up merging, mentions are only merged together if some coreference indicators, like string matching, name alias or appositive can be satisfied. The merge is comparatively conservative and thus leads to a higher precision but a lower recall.

We were also concerned how much each type of hyperedge affected the resolution performance. Table 6 summarizes the performance contribution of each kind of hyperedges to our system of HyperGraph. The last three columns show the gain or loss in recall, precision and F-measure, respectively, because of subtracting a particular hyperedge while keeping the rest in the HyperGraph system.

As our approach is partitioning-driven, the low-weight hyperedges play an important role in dividing mentions. We were also concerned how much the hyperedge *CannotLink* affects the resolution performance. The last line of Table 6 shows the loss of performance by removing the *CannotLink* from the system. From the table, the removal of *CannotLink* results in a drop of by 37.56% in recall and 6.14% in precision. Overall, the F-measure decreases by 18.22%. Similarly, the hyperedge contribution to whole system F-measure decreased like *Semantic*(1.27%), *NameAlias*(0.54%), *Number*(0.35%), *ThreeSentences*(0.24%), *Gender*(0.14%), *HeadString*(0.08%), *Appositive*(0.05%).

Interestingly, when only subtracting *FullString*, *Person*, *TwoSentences*, and *OneSentence*, the final F-measure increased 0.25%, 0.11%, 0.30%, 0.21%, respectively. In other words, the four kinds of hyperedges decreased the whole system performance using all features. After deep analysis, we found that the *FullString* with high weight is little repeated by *HeadString* with middle weight. Moreover, *Person* is just a kind of semantic. The *Person=True* hyperedges are replaced by *Semantic=Person* hyperedges. When replaced, the hyperedges are redundant, and hence decrease the final resolution result. Meanwhile *TwoSentences* is repeated to some extent by *ThreeSentences* and *OneSentence*. Similarly, so does *OneSentence* by *ThreeSentences* and *TwoSentences*.

Our results show that reducing feature redundancy is a practical problem for unsupervised coreference resolution hypergraph partitioning. Actually, we experimented on subtracting any two, three or all of the above four kinds of hyperedges while keeping the rest in the HyperGraph system. The results were all worse than using all features. It was because all the features were intersecting in the hypergraph.

6. Conclusion

This paper presented an unsupervised learning approach for coreference resolution based on hypergraph partitioning. It converts a document to a hypergraph where a vertex corresponds to a mention in the document. It uses a hyperedge to cover mentions that share a specific common property, which can capture information about multiple mentions, instead of only two mentions as in the traditional approaches based on the mention-pair model. Our approach adopts a hypergraph partitioning algorithm to divide mentions into clusters each representing a single entity. The partitioning process can avoid the cascading errors in the previous clustering-based unsupervised approaches.

In the paper, we described the resolution framework, the definition of hyperedges, and the stopping criteria of partitioning. The evaluation on the ACE data set shows that the hypergraph partitioning approach performs better than the previous clustering-based unsupervised approach (with up to 1.97% in F-measure), and the gap between the supervised approach is only 2.57% in F-measure.

Our current work focuses on the framework of coreference resolution with hypergraph partitioning. There are several directions for future work:

(1) For simplicity, we currently just used some common knowledge, represented as hyperedges, for coreference resolution. We would like to explore more effective knowledge, such as grammar roles, context template information, and others proposed in Ng and Cardie (2002).

(2) In the current system, the weights for hyperedges were all heuristically designed. We intend to try some weights learning mechanisms, e.g., the genetic algorithm.

(3) The stopping criterion has a big influence on the final resolution performance. However, our current stop criterion was defined in a heuristic way. We would like to incorporate more prior knowledge related to coreference resolution.

(4) Feature redundancy is another problem for hypergraph partitioning. We will try to process it as a learning problem.

References

Andrew McCallum and BenWellner. (2003). Toward conditional models of identity uncertainty with application to proper noun coreference. *Proceedings of the IJCAI-03 Workshop on IIWeb*, pp.79-84.

C. Berge. (1989). *Hypergraphs*. North-Holland, Amsterdam

C. Cardie and K. Wagstaff. (1999). Noun phrase coreference as clustering. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp.82-89.

C. Nicolae and G. Nicolae. (2006). Best cut: A graph algorithm for coreference resolution. *Proceedings of the EMNLP*, pp.275-283.

D. Klein and C.D. Manning. (2001). Parsing and hypergraphs. *Proceedings of the IWPT*.

D. Zhou, J. Huang, and B. Schölkopf. (2006). Learning with hypergraphs: Clustering, classification, and embedding. *Proceedings of the NIPS*, pp.1601-1608.

Hiroyuki Shinnou and Minoru Sasaki. (2007). Ensemble document clustering using weighted hypergraph generated by nmf. *Proceedings of the ACL*, pp.77-80.

I.P. Klapaftis and S. Manandhar. (2007). UOY: A Hypergraph Model For Word Sense Induction & Disambiguation. *Proceedings of the SemEval*, pp.414-417.

Joseph F. McCarthy and Wendy G. Lehnert. (1995). Using decision trees for coreference resolution. *Proceedings of the IJCAI*, pp.1050-1055.

Kiri Lou Wagstaff. (2002). *Intelligent Clustering with Instance-Level Constraints*. Ph.d. thesis, Cornell University,

Liang Huang. (2008). Forest reranking: Discriminative parsing with non-local features. *Proceedings of the ACL-08:HLT*, pp.586-594.

M. Vilain, J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. (1995). A model-theoretic coreference scoring scheme. *Proceedings of the 6th MUC*, pp.45-52.

N. Selvakkumaran and G. Karypis. (2006). Multiobjective Hypergraph-Partitioning Algorithms for Cut and Maximum Subdomain-Degree Minimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(3):504-517.

V. Ng and C. Cardie. (2002). Improving machine learning approaches to coreference resolution. *Proceedings of the ACL*, pp.104-111.

W.M. Soon, 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.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. (2004). A mention-synchronous coreference resolution algorithm based on the bell tree. *Proceedings of the ACL*, pp.135-142.

Notes

Note 1. http://glaros.dtc.umn.edu/gkhome/fetch/sw/hmetis/hmetis-2.0pre1.tar.gz

Note 2. These definitions can be extended in a straightforward manner for hypergraphs with weighted hyperedges, as described in http://glaros.dtc.umn.edu/gkhome/fetch/sw/hmetis/manual.pdf

Note 3. The gender type of a person name was obtained from a name-gender list provided by the corpora from NLTK package, while the gender of a common noun (e.g., mother, son, president) was got from WordNet (if the gender of a mention, such as ”the president”, is not available in WordNet, we set the gender type as neuter).

Note 4. These sentence hyperedges only aim for pronoun resolution. They do not cover non-pronoun anaphora, as the sentence factor plays little influence on the coreference determination for non-pronoun resolution.

Note 5. Intuitively, negative weight for *CannotLink* is better. However, the hypergraph toolkit hMETIS could not accept negative weights. So here, zero is chosen.

Note 6. http://opennlp.sourceforge.net/

Table 1. This is an example about tables

__[Microsoft Corp.]__$_1$ announced __[[its]__$_2$ __new CEO]__$_3$ __[Steve Ballmer]__$_4$ yesterday. __[Mr. Ballmer]__$_5$ said __[Microsoft]__$_6$ would try __[its]__$_7$ best to compete with __[Google]__$_8$. __[He]__$_9$ also mentioned that __[Microsoft]__$_{10}$ had been challenged by __[some companies]__$_{11}$ in internet area during the late years.

Table 2. An example of generated hyperedges

Type	Hyperedges
FullString	{2,7}, {6,10}
HeadString	{2,7}, {6,10}, {4,5}
Gender	{4,9}
Number	{1,2,3,4,5,6,7,8,9,10}
Person	{3,4,5,9}, {1,2,6,7,8,10,11}
Semantic	{1,6,8,10,11}, {2,7}, {3,4,5,9}
ThreeSentences	{1,2,3,4,5,6,7}, {1,2,3,4,5,6,7,8,9}
TwoSentences	{1,2,3,4,5,6,7}, {5,6,7,8,9}
OneSentence	{1,2}, {5,6,7}
NameAlias	{4,5}, {1,6,10}
Appositive	{3,4}
CannotLink	{2,9}, {7,9}

Table 3. Weights for four kinds of hyperedges

Level	Hyperedges	Weight
Cannot	*CannotLink*	0
Low-1	*ThreeSentences*	5
Low-2	*TwoSentences*	10
Low-3	*OneSentence*	15
Middle	*HeadString*, *Gender*, *Number*,*Person*, *Semantic*	20
High	*FullString*, *Appositive*, *NameAlias*	30

Table 4. Statistics of entities (length > 1) and contained mentions for the test data set in ACE

Domain	#entity	#mention
BNews	468	2493
NPaper	365	2290
NWire	411	2304

Table 5. Results of different systems for coreference resolution

System	BNews			NPaper			NWire			Total		
	R	P	F	R	P	F	R	P	F	R	P	F
Cardie99r4	72.06	61.28	66.23	60.58	44.89	51.57	66.26	48.04	55.70	66.30	50.99	57.64
Cardie99r6	73.22	61.89	67.08	62.18	46.26	53.05	66.60	48.19	55.92	67.34	51.73	58.51
Soon01	60.63	81.03	69.36	51.92	65.53	57.94	53.91	72.77	61.94	55.51	**72.96**	**63.05**
HyperGraph	86.83	55.12	67.43	72.44	45.75	56.08	78.81	45.84	57.97	**79.37**	48.86	60.48

Table 6. Results of different systems for features contribution comparison

System	Total			Gain(+)/Loss(-)		
	R	P	F	R	P	F
All Features	79.37	48.86	60.48			
-ExtentString	79.87	48.99	60.73	**0.50**	**0.13**	**0.25**
-HeadString	79.34	48.76	60.40	-0.03	-0.10	-0.08
-Gender	79.06	48.79	60.34	-0.31	-0.07	-0.14
-Number	78.67	48.66	60.13	-0.70	-0.20	-0.35
-Person	79.41	48.98	60.59	**0.04**	**0.12**	**0.11**
-Semantic	77.34	47.96	59.21	-2.03	-0.90	-1.27
-ThreeSentences	79.02	48.67	60.24	-0.35	-0.19	-0.24
-TwoSentences	79.91	49.05	60.78	**0.54**	**0.19**	**0.30**
-OneSentence	79.58	49.05	60.69	**0.21**	**0.19**	**0.21**
-Appositive	79.30	48.82	60.43	-0.07	-0.04	-0.05
-NameAlias	78.73	48.39	59.94	-0.64	-0.47	-0.54
-CannotLink	41.81	42.72	42.26	-37.56	-6.14	-18.22

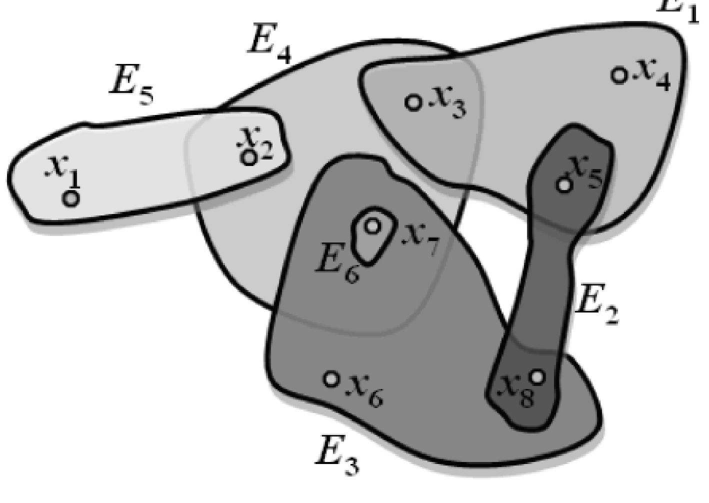

Figure 1. An example of hypergraph