Parallel Community Detection For Cross-Document Coreference

Fatemeh Rahimian
Swedish Institute of Computer Science
KTH - Royal Institute of Technology
fatemeh@sics.se

Sarunas Girdzijauskas
KTH - Royal Institute of Technology
sarunasg@kth.se

Seif Haridi
Swedish Institute of Computer Science
KTH - Royal Institute of Technology
seif@sics.se

***Abstract*—This paper presents a highly parallel solution for cross-document coreference resolution, which can deal with billions of documents that exist in the current web. At the core of our solution lies a novel algorithm for community detection in large scale graphs. We operate on graphs which we construct by representing documents' keywords as nodes and the co-location of those keywords in a document as edges. We then exploit the particular nature of such graphs where coreferent words are topologically clustered and can be efficiently discovered by our community detection algorithm. The accuracy of our technique is considerably higher than that of the state of the art, while the convergence time is by far shorter. In particular, we increase the accuracy for a baseline dataset by more than 15% compared to the best reported result so far. Moreover, we outperform the best reported result for a dataset provided for the Word Sense Induction task in SemEval 2010.**

I. Introduction

Resolving entities in a text may not always be a difficult task for humans. When one comes across Mercury in an article about the solar system, they instantly think of Mercury, the planet, and not about Mercury, the chemical element or Freddie Mercury. For a computer though, such a disambiguation requires a considerable amount of processing. This problem, i.e., the task of disambiguating manifestations of real world entities in various records or mentions, is known as *Entity Resolution* or *Coreference Resolution*. Often disambiguation is required across multiple documents. Given a set of such documents with an ambiguous *mention* (Mercury, for example), the *Cross-Document Coreference* problem seeks to group together those documents that talk about the same *entity* in real world (e.g., one group for the planet, one for the chemical element, etc.).

This problem is challenging because: (i) often the number of underlying entities and their identities are not known (e.g., we do not know how many different Mercuries are to be discovered), and (ii) the number of possible classifications grows exponentially with the number of input documents.

A widely used approach to this problem, known as *Mention-Pair model*, is to compute a pair-wise similarity value based on the common keywords that exist in each pair of documents [2]. If two documents are found similar more than a predefined threshold, they are classified together. Finally, a clustering step is required to partition the mentions into coreferent groups. The clustering itself is a challenging task and is known to be NP-hard. In the related work section, we discuss some of the approximate solutions that address this problem. As we will see, the high complexity of the Mention-Pair model renders it impractical for web-scale coreference, where we have to process millions of documents in a reasonable time.

In this paper[1] we propose a novel approach to coreference resolution, which does not require separate classification and clustering steps. Instead, we transform the problem to a node-centric graph processing task. This enables us to take advantage of the recent advances in graph processing frameworks, such as GraphChi [16] or GraphLab [18], and apply our algorithm to extremely large graphs.

To construct the graph, we create two types of nodes. One type represents the ambiguous word, which we assume is given in advance. Another type of nodes represents the unambiguous words that surround the ambiguous word in each document. Since we do not know whether or not different mentions of the ambiguous word are referring to the same real-world entity, we create as many nodes as the number of documents mentioning them. The unambiguous words might as well appear in multiple documents. For them, however, we do not create a new node, if they already exist. Finally, we add an edge between two nodes, if their corresponding words co-occurred in the same document. Consequently, each single document is represented by a full mesh, or *clique*, of all its keywords.

The constructed graph for our Mercury example is depicted in Figure 1(b). As shown, some cliques overlap, which indicates that their corresponding documents have a similar context. In fact, the main insight to our work is that the topological community structure of the constructed graph identifies similar contexts and thus, is an accurate indicator of the coreference classification. Based on this fact, we propose a novel community detection algorithm for coreference resolution. Our algorithm is diffusion based and exploits the fundamentals of flow networks. In such a network each node has a capacity and each edge can transfer a flow, just like a pipe, between two nodes. We envision multiple flows in our graph, one per community. To distinguish these flows, we assign a distinct color to each of them.

Initially each single document constitutes a distinct community, i.e., it will be assigned to a unique color. All the nodes that belong to a document will get a unit of the color of their document. Therefore, those nodes that are shared between documents, will receive multiple units of colors. However, each node always identifies itself with only one single color,

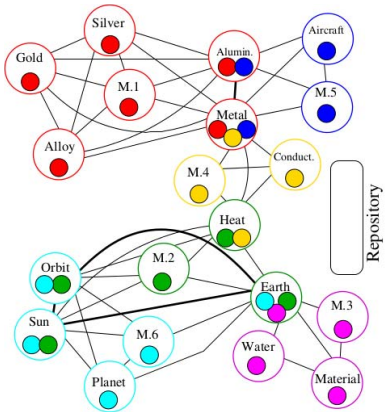
[1]This work was funded (in part) by the European Commission within the Marie Curie ITN "iSocial" (grant PITN-GA-2012-316808).

978-1-4799-4143-8/14 $31.00 © 2014 IEEE
DOI 10.1109/WI-IAT.2014.79

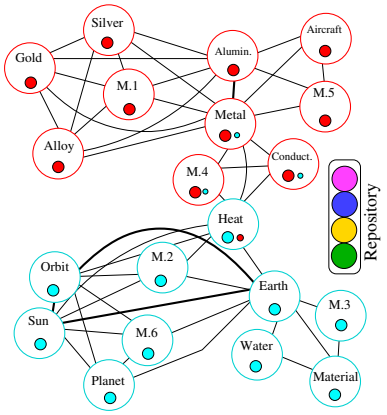
IEEE computer society

1	**Mercury** easily forms alloys with other metals, such as gold, silver, and aluminum.
2	The heat encountered once in **Mercury**'s orbit will be the equivalent of 11 suns beating down on Earth, about 700 degrees.
3	**Mercury** probably acquired much of its water and organic material the same way Earth did, researchers said.
4	**Mercury** is a relatively poor conductor of heat. Most metals are excellent thermal conductors.
5	**Mercury** generally is not allowed on aircraft because it combines so readily with aluminum, a metal that is common on aircraft.
6	**Mercury** orbits the sun every 88 Earth days, zipping around at a faster pace than any other planet.

(a) Fragments of several documents, all with a mention of "Mercury", which is ambiguous. The underlined words, represent the context of the the ambiguous word in each document. Initially each document is assigned to a unique color.



(b) Each context words is represented by a node in the graph and gets one or more color(s) that correspond(s) to the document(s) it occurred in. An edge between two nodes implies the collocation of them in the same document. Node border colors indicate the dominant color in the vicinity of each node.



(c) The final coloring scheme identifies the communities of the graph. Mercury 1, 4, and 5, all belong to the red community, thus, are considered to be coreferent. Likewise, Mercury 2, 3, and 6 are considered to be coreferent because of the aqua community. The unused colors reside in a repository.

Fig. 1. The main steps towards coreference resolution

which has the highest collective volume in its neighborhood (including the node itself), so-called the *dominant color*. The initial coloring scheme of our Mercury graph is shown in Figure 1(b). Nodes continuously exchange parts of their colors with their neighbors by diffusing the colors through their links. Therefore, the available volume of color at nodes, and accordingly the dominant color in their vicinity, changes during the course of algorithm. We will show that with appropriate diffusion policies it is possible to accumulate one distinct color in each of the well connected regions of the graph, e.g., as in Figure 1(c). Finally, the ambiguous nodes that end up having the same dominant color are considered to be coreferent. Since our constructed graph is sparse, the overhead of such computation remains low (for complexity analysis see Section III-D). Moreover, we can produce more accurate results, compared to the state of the art. This twofold gain is owed to the combination of two ideas, that constitute our main contributions:

- a technique for transforming the expensive coreference problem, into a graph problem, in which the coreferent words belong to the same topological community structure. The graph that we construct is sparse, because those documents that have dissimilar contexts, will have very few or even no direct connections. The computations on the graph are performed per edge basis, i.e., only if there is an edge between two nodes, they will communicate some flows. Hence, the irrelevant documents which are weakly connected, if not disconnected, will not impose any computation in the graph. At the same time, a more thorough search of the solution space is possible, as we are not limited to pair-wise similarity discoveries only. Instead, similarity between any number of documents is naturally captured within the community structures that emerge from the inter-linked context words.
- a novel node-centric diffusion-based community detection algorithm that mainly uses local knowledge of the graph at each node. Hence, it allows for highly parallel computations and usage of the existing graph processing frameworks.

We run our algorithm on different datasets, which are transformed to graphs with distinct structural properties. For example, on a baseline dataset for person name disambiguation, we produce a classification with an F-score 15% higher than that of the state of the art algorithm by Singh et al. [34]. Moreover, on a dataset provided in the Word Sense Induction task of SemEval 2010, we achieved as good F-score as the best reported result. However, we considerably outperform the other solutions with respect to a complementary accuracy metric, which measures the average number of detected communities.

II. Terminology

The main terms that we are going to use hereafter are:

- *Entity* is a unique representation of someone/something in the real world, e.g., "Paris Hilton".
- *Mention* (or **target word**) is a literal manifestation of a real world entity. Since multiple entities could share a similar name, mentions can sometimes be ambiguous, e.g., "Paris" in the sentence "Paris is nice.", which could refer to the capital city of France, the city in Texas, Paris Hilton, or many other possibilities. Two mentions are said to be *coreferent*, if they refer to the same real world entity.
- *Context* of an ambiguous mention M is a set of unambiguous mentions surrounding mention M. For example, in Figure 1(a), the underlined words in each sentence constitute the context of the ambiguous word Mercury in that sentence. Note, extracting an appropriate context for a mention is not trivial, and is

an active field of research. However, this task is out of the scope of this paper, and we only use the extracted context as an input to our algorithm.

- *Community* or *cluster* in a graph is a densely connected component. *Community detection* is the task of grouping the vertices of the graph into clusters taking into consideration the edge structure of the graph in such a way that there should be many edges within each cluster and relatively few between the clusters. Note, this problem is different from *graph partitioning*, where the goal is to divide the graph into a predefined number of roughly equal size components, such that the number of edges crossing different component is minimum (a.k.a., the *min-cut* problem). In Community detection we neither know the number of existing communities nor the size of them.

III. Solution

Given an ambiguous word [2] and a set of documents with preprocessed context words, we classify the ambiguous words into coreferent groups in two main steps: (i) Graph Construction, and (ii) Community Detection.

A. Graph Construction

We create a weighted undirected graph using the ambiguous and unambiguous words as nodes. For the ambiguous word we always create a new node per document. We call such nodes the *target nodes*, as those are the nodes we aim to classify. For the non-ambiguous words, if a matching node already exists, we reuse it; otherwise, we create a new node. Moreover, if two words have appeared together in a document, we add an edge between the nodes representing them. The resulting graph is weighted, as some words may frequently appear together in multiple documents. More precisely, the weight of each edge is proportional to the number of documents that contain both endpoint nodes of that edge.

The graph corresponding to our Mercury example is depicted in Figure 1(b). Each mention of Mercury has a distinct node in the graph, tagged with its document identifier. Edge thickness (weight) between pair of nodes is relative to the number of times the two words have appeared together across different sentences. Now our task is to classify these Mercury nodes into several (ideally two, in this case) clusters, where all the Mercuries in a cluster refer to the same entity. Note, we do not know, in advance, how many different Mercuries are expected to be resolved, i.e., we do not know the number of expected clusters/communities.

B. Community Detection

We propose a massively parallel diffusion based algorithm for community detection. Without lack of generality, we assume that the algorithm proceeds in rounds.

1) Initialization: Initially, each document is associated with a distinct color. Every node is given a unit of color corresponding to the color of the document that holds it. Accordingly, if a node belongs to multiple documents, it receives multiple colors.

Each node identifies itself with a *dominant color*, which is the color with the highest total volume among the node and its neighbors [3]. The dominant color of a node also indicates the community that the node belongs to.

2) Diffusion: Every node repeatedly runs the diffusion algorithm, until the convergence criteria, defined in section III-C, is satisfied. In each round a node sends out some amount of color to its neighboring nodes, and likewise, receives some amount of color from its neighbors. The key point is to decide which color or colors should be sent out and in what quantity. This decision is made locally at each node and is based on one main objective, that is, each node tries to change its color to the one that is *dominant* in its neighborhood, i.e., the color with the largest collective quantity across its neighbors and the node itself.

The effort that a node makes to change its color to the dominant color (or maintain it, if it already has the dominant color) consists of two main forces: (i) an attraction force that conserves the dominant color, and (ii) a repulsion force that evacuates all the non-dominant colors. The algorithm is further completed with a recycling mechanism that nourishes the diffusion by collecting the colors from the regions where they are non-dominant and putting them back into the regions where they can become influential again.

Note, since the colors only flow through the edges of the graph to the neighboring nodes, disconnected components will never get the same color, as there will be no link connecting them and carrying the flows. This property of the solution is desirable, because disjoint clusters indicate disparate contexts and are not expected to be in the same coreference chain.

The diffusion algorithm at each node is composed of the following three rules:

- *Attract: Keep fraction α of the dominant color, and divide the rest between neighbors.* The attraction force should be applied with a subtlety. If nodes are too greedy, meaning that they do not let any dominant color to leak out, then there will be no chance that their neighboring nodes, which have a different color, will get influenced and change their color. On the other hand, if the leakage is too high, all the colors will freely and rapidly explore the entire graph, and the concentration of a distinct dominant color in each of the community structures will not take place. However, if nodes allow for an appropriate amount of leakage, i.e., α , over time they not only maintain their color, but also might be able to expand their territory and let more nodes into their community. Parameter α determines to what extent the communities are likely to merge, thus, controls the resolution of the detected communities. A bigger α produces more communities with smaller sizes, whereas a smaller α is likely to

[2] Our solution is not limited to a single ambiguous word and we can have multiple of such words. However, for the sake of clarity, we consider the case with one ambiguous word only. To discover which word(s) is(are) ambiguous, is a different problem which is orthogonal to our work and out of the scope of this paper.

[3] We break the ties with some globally known ordering of colors, e.g., least color id.

produce fewer communities of a bigger size. The fraction (1-α) of the dominant color is then shared between all the neighbors, proportional to the the weight of the edge to the neighbors.

- *Repel: Divide all the non-dominant colors between the neighbors.* With this rule, nodes evacuate the non-dominant colors. If a color is not dominant in any region, it will be subject to this repulsion force all over the graph, and thus it will be highly fragmented. Consequently, no node will identify itself with that color, as though the color has disappeared from the graph. On the other hand, there are colors that are recessive in some region, but dominant in another region. This repulsion force allows such colors to flow in the graph and be absorbed in the regions, where there is an attraction force for them. Note, here again the colors are divided between neighbors, proportional to the weight of the edges. That is, if an edge has a weight that is twice as big as another edge, then it will carry twice the amount of color accordingly.
- *Recycle: If surrounded by neighbors of the same dominant color, send all the non-dominant colors (if any) to a repository, and get some dominant color from the repository (if there is any).* The repulsion force works blindly, meaning that the non-dominant colors are sent to all directions, with the hope that they might encounter an attraction force in the neighboring nodes. But a node may be completely surrounded by nodes of the same dominant color as the node itself. We call such a node, an *interior node*. Consider a node with the dominant color `red`, surrounded by neighbors of the same dominant color. If such a node receives some `blue` color, sending it to the neighbors, would only disturb the `red` territory. Instead of blindly repelling non-dominant flows, interior nodes take a more efficient approach, by sending the non-dominant colors directly to a repository, where all such colors are accumulated. The repository is accessible by all the nodes and acts as a container for the colors that are collected from the graph. It also keeps track of the number of interior nodes per color.
  The recycling process is completed by putting the abandoned colors back in the regions where they are dominant. When an interior node send some non-dominant color to the repository, in return, the repository send a share of the the node's dominant color back to it. The amount of this share depends on the available amount of the dominant color in the repository as well as the number of interior nodes that require it. Since the repository knows the number of such nodes, it divides the color equally between them. The augmented amount of the dominant color in the interior nodes will then help taking over the neighboring regions, if they are strongly connected to the reinforced community. Those colors that are not dominant in any region of the graph will remain in the repository for ever, and will never again flow in the graph. Consequently, over time, nodes of the graph have to deal with fewer and fewer number of colors, which makes their computations much faster and more efficient.

C. Convergence

When the coloring scheme of the graph does not change any more, and we remain in a stationary state, we consider the algorithm is converged. At the convergence time, the coloring scheme of the whole graph determines the clustering of the nodes. More precisely, the target nodes that have the same dominant color at the end, will be considered as coreferent. Also, the number of different colors for the target nodes indicate the number of different references for the ambiguous word.

D. Complexity

In this section, we give some bounds on the complexity of our algorithm. First of all, if N is the number of documents, and if we choose on average c words from each document, the number of nodes would be at most $c \times N$, which is $O(N)$. In each round, every node communicates with its neighboring nodes. If d indicates the degree of a node, then the communication complexity of the algorithm in each round is $O(N \times \overline{d})$, where $\overline{d}$ represents the average node degree. Hence, the overall complexity will be $O(N \times \overline{d} \times rounds)$.

The maximum number of rounds before convergence is proportional to the time required for the existing colors in the graph to spread out to all the reachable regions. Therefore, it is proportional to the diameter of the biggest connected component in the constructed graph. It is important to note that the diameter of the graph is inversely proportional to the average node degree, means if the average node degree is higher, the expected number of rounds is lower. Also note that our constructed graph is sparse, i.e., the average node degree is far less than N. Hence, in practice, $O(N \times \overline{d} \times rounds)$ becomes significantly smaller than $O(N^2)$, which makes the large-scale coreference feasible.

IV. Experiments

Our algorithm is vertex-centric and can be deployed over any of the existing graph processing frameworks, such as Graphlab [18] or Graphchi [16], which are proven to be scalable, efficient and fast. At the moment we are using Graphchi framework [16], which is a disk-based system for efficient computations on graphs with billions of edges. We have implemented the color repository as an object that is shared between all nodes, i.e., all the nodes can access this repository. We ran several experiments to tune parameter α that is used in the Attract rule. We concluded that the best result is achieved when α is set to $\frac{2}{3}$.

A. Metrics

We compare our detected communities (of target words) with the true classification of coreferent words, using a metric, called *B3*, which breaks down to a few other metrics. For every target word:

- B^3*Precision* is the fraction of detected coreferents that are actually coreferent with it.
- B^3*Recall* is the fraction of its actual coreferents that are detected as being coreferent with it.

We then calculate the overall *Precision* and *Recall* by taking the average of B^3Precisions and B^3Recalls for all the target words (documents). To clarify these metrics, note that if we put each document in a separate community, we will have 100 percent precision, because no irrelevant words are wrongly grouped together. This naive way of clustering shows that precision alone, can not give us an idea how accurate our clustering is. Hence, we need other complementary metrics. Now, if we put all the documents in one single community, we will have 100 percent recall. This is again an extreme case, where we do not produce meaningful clusters, thus, the precision will be very low. Therefore, we use the harmonic mean of these two metrics, which is widely known as F_1-*Score*. Since we compute the precision and recall based on B^3, our F_1-score is also a B^3 score:

$$B^3F_1 = 2 \times \frac{\overline{B^3 Presicion} \times \overline{B^3 Recall}}{\overline{B^3 Precision} + \overline{B^3 Recall}}$$

Hereafter, whenever we use the terms precision, recall or F_1-score we always refer to the B^3 variants of these metrics. Moreover, to be comparable to other reported results for the SemEval dataset, we use the evaluation script, given for the same task. In addition to precision, recall and F_1-score, this script measures the average number of detected clusters. Therefore, we also measure this additional metric in our SemEval experiments.

B. Results

We have used three different datasets, namely Chris Andreson, John Smith, and SemEval. For the first two datasets we have used OpenCalais [1] to extract the name entity mentions as context. For the SemEval task we have used all the context words. The results for each dataset are reported separately.

1) Chris Anderson Corpus: This corpus, provided by our industrial partner Recorder Future[4], is extracted from 1185 documents from the web. Each document contains a reference to one out of 8 persons, named Chris Anderson.

The graph produced for this dataset is depicted in Figure 2. We monitored the progress of our community detection algorithm and have reported the accuracy metrics in each round in Figure 3. As shown, in the very beginning the precision is 100%, because each document is considered to be a distinct community, thus, no two documents are wrongly classified together. However, the recall is very low, because no two documents are correctly classified together either. Over time, as the communities merge, recall improves considerably, at the cost of a small downgrade in precision. Overall, the F_1-score increases significantly up to a certain level, by then the community detection algorithm has converged. We observe that after only 7 rounds the community detection algorithm has converged and the final communities are detected with more than 85% accuracy.

Note, the fast convergence of the algorithm is due to the topological properties of this specific graph. As shown in Figure 2, the small non-ambiguous nodes constitute only a small fraction of the nodes in the graph. This means, the

[4]www.recordedfuture.com

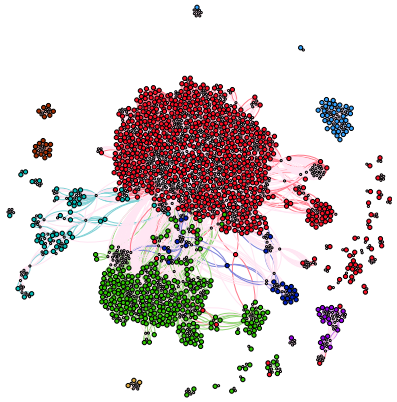


Fig. 2. Chris Anderson Graph. The ambiguous nodes are depicted in larger size. The coloring scheme represent the true clustering.

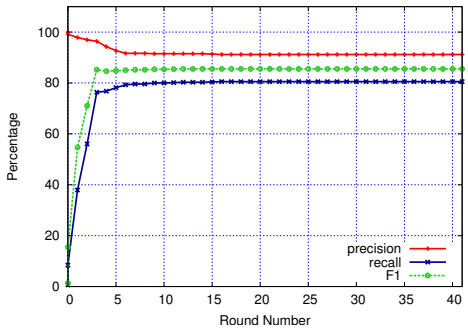


Fig. 3. Chris Anderson Accuracy over time

ambiguous words that are coreferent have extremely similar context. Thus, their cliques are tightly connected, and it takes very few steps for one dominant color to take over the tightly connected regions of the graph. Moreover, some of the communities that have the same color in Figure 2, are not even connected. This means that the vector of the terms extracted from their corresponding document are not informative enough for the two documents to be classified together. In such cases, we do not expect our algorithm to merge the two communities, as there is no way any color will flow from one to another. In fact, no other solution could do any better with only this incomplete information.

2) John Smith Corpus: This corpus, originally introduced by Bagga and Baldwin [2], contains 197 New York Times articles about 35 different people named John Smith. Each article mentions a single John Smith. Twenty four clusters contain a single document, while the rest contain the following numbers of documents: 2, 2, 2, 4, 4, 5, 9, 15, 20, 22, 88. This dataset gives us a different graph type, as shown in figure 4.

The progress of precision, recall and F_1-score over time is depicted in Figure 5. Again we start with 100% precision and a very low recall. As shown, it takes longer for the algorithm to converge. This is expected, because as opposed to Chris Anderson graph, there are so many small clusters connected by very few links (See Figure 4). Here, the fraction of non-ambiguous (small dots) to ambiguous nodes (depicted with a larger size) is not negligible. Therefore, it takes longer for a color to reach out to other regions of the graph and explore any

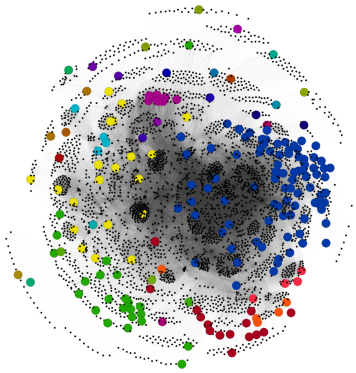


Fig. 4. John Smith Graph. The ambiguous nodes are depicted in larger size. The coloring scheme represent the true clustering.

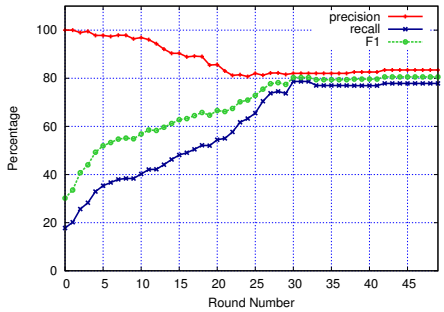


Fig. 5. John Smith Accuracy over time

potential community merge possibilities. Finally, when we do not observe any more change, the F_1-score is over 80%. It is important to compare this result with other existing solutions that have worked with this same dataset. For example, in a nice work by Singh et al. [34] from Google, which has also proved scalable, F_1-score is reported to be 66.4%. Also, the best reported result so far is 69.7%, by Rao et al. [29]. This means we have increased the accuracy more than 15% compared to the state-of-the-art.

3) SemEval 2010, WSI task: This dataset [19] was given as the test data for the Word Sense Induction and Disambiguation task in SemEval 2010. As opposed to the John Smith and Chris Anderson datasets, the work on this dataset is not limited to person name entity resolution. The texts come from various news sources including the Wall Street Journal, CNN, ABC and others. It includes 50 ambiguous nouns and 50 ambiguous verbs. Each test instance consisted of a maximum of three sentences.

Here again we construct our graph for each word separately and apply our community detection algorithm. The evaluation script provided in the task, enables us to report the results for nouns only, verbs only, or all the words. The results are reported in Figure 6. Our solution produces clusterings with F_1-scores 62.2, 70.7, and 56.4 for "all", "verbs", and "nouns", respectively. Among the reported results so far, only *Duluth-WSI-SVD-Gap* [19] has a slightly higher F_1-Score than us (63.3, 72.4, and 57.0 percent accuracy for "all", "verbs" and "nouns", respectively). However, the average number of

System	FS (%) all	FS (%) nouns	FS (%) verbs	#Cl
MFS	63.5	57.0	72.7	1
Duluth-WSI-SVD-Gap	63.3	57.0	72.4	1.02
Our Solution	62.2	56.4	70.7	3.61
KCDC-PT	61.8	56.4	69.7	1.5

Fig. 6. Accuracy result for SemEval dataset

clusters found by this solution ($\#Cl$ in Figure 6) is 1.02, which is far from the actual numbers (4.46 for nouns, 3.12 for verbs, and 3.79 for all). In fact, this result is very close to a naive clustering, known as the most frequent clustering (MFS), in which all the mentions of a word are classified into one single cluster, hence the average number of clusters per word is 1. With our solution, however, the average number of clusters is 3.61, which is much closer to the actual number of clusters in the golden truth. In [14], which bears the closest resemblance to our work, the best reported F1-score is 63.4.

V. Related Work

We organize this section in two main parts. In the first part we summarize the related work on Coreference, which is the problem that we are addressing. Since our solution is a community detection algorithm, in the second part we survey the existing solutions for graph clustering and community detection.

A. Related Work in Cross-Document Coreference

Coreference is one of the complicated NLP problems that has received a lot of attention in the community, and has yielded numerous solutions. Several publications has already surveyed the coreference research in details [35], [23], [25]. Due to the space limitations, we focus on the related work in cross-document coreference resolution, which is the problem addressed in this paper. Since we use a graph-based solution, we also overview some of the existing graph-based approaches.

One of the most well-known approaches to cross-document coreference is presented by Bagga and Baldwin [2]. They used the original vector space model introduced in [31], in which every document is represented by a vector of its terms. These terms are weighted proportional to their frequency in the text. Then a similarity measure between each pair of documents is computed based on the common terms in the two documents, also considering their weights. If the similarity of two documents is above a predefined threshold, then the entity of interest in those two documents are considered to be coreferent. This approach is still widely used and several improvements to it are proposed [36], [30], [10]. These solutions generally fall into the category of *mention-pair models* [25] and require a final clustering step after the initial pair-wise comparisons, which is not always straightforward. This is mainly because the transitivity of coreference can not be always enforced. For example, if the similarity comparisons suggest that A and B are similar, as well as A and C, but B and C cannot be the same, due to gender or size contradictions, then the final clustering becomes challenging. Although multiple approximate solutions to this problem are proposed [27], Ng [25] in his survey paper argues that the mention-pair model

remains fundamentally weak. Various alternative approaches have thusly been proposed to improve on the classical mention-pair model [21], [24], [28].

There have also been a few graph-based approaches to coreference resolution. These approaches do not require separate classification and clustering steps. Instead, they construct a graph in which nodes represent the mentions, and links represent the relation between pairs of mentions. Several interesting ideas have been proposed for how to compute the weight on the links and how to put it in use. In [4], [3], [20], for example, there can be multiple links (of different types) between nodes, thus, a hyper graph is constructed. Weights on the links are then learned through a training phase. Then the hypergraph is partitioned into multiple sub-hypergraphs by means of a spectral graph clustering or greedy partitioning. Finally, all the mentioned of a partitioned component are considered coreferent. In [11], the betweenness centrality of the nodes that may be coreferent is computed. Similarly, [37] investigated graph attributes and links to rank the similar nodes of the graph. Another interesting graph-based approach is recently proposed in [13]. First, they transform an unweighted undirected cyclic entity graph, into an unweighted, directed, acyclic one. Then, they find the maximal quasi-strongly connected components of the transformed graph. Finally, all the mentions that belong to the same component are considered to be coreferent. The main problem with these solutions is that they are computationally very expensive, thus, it is impractical to use them in web scale.

An existing work that can be applied on very large datasets is [34], which transforms the problem to a Markov chain Monte carlo (MCMC) based inference. Mentions and entities are random variables. Each mention takes an entity as its value, and each entity takes a set of mentions as its value. To scale up the MCMC-based inference, initially the entities are distributed among multiple machines. Then, independent MCMC chains are computed on each machine using only the local merge proposals. After a certain number of rounds, the entities are redistributed among machines to enable merge proposals for entities that were previously on different machines. Another scalable solution, proposed in [29], is a streaming algorithm for coreference resolution, which can work on very large datasets. The closest work to ours is perhaps by Jurgen et al. [14], which also applies a community detection algorithm on a collocation graph. As opposed to our node-centric solution, [14] uses an agglomerative algorithm for community detection, which could become expensive for big graphs.

B. Related work in Graph Clustering

Graph clustering problem is to divide nodes of a graph into multiple components, such that the number of edges that cross different components is minimum. This is also known as the *min-cut problem*, because it identifies the minimal set of edges that can be cut in order to split the graph. Two main graph clustering problems are known: (a) *balanced graph partitioning*, and (b) *community detection*. For balanced graph partitioning the number of desired components is given in advance, and there is a constraint that the components should hold roughly equal number of nodes [15], [32], [26]. However, in the community detection problem neither the size nor the number of components are known. Instead the task is to cluster the nodes into groups or *communities*, which are tightly connected together, but very sparsely connected to the rest of the nodes. Despite their differences, the two problems are not conceptually different, as they both strive to find the min-cut of the graph. Thus, many ideas and techniques can be applied to both of them. Here, we give an overview of the main category of graph clustering solutions. For more detailed information please refer to the existing surveys by Fortunato et al. [7] and Schaeffer et al. [33].

1) Spectral Clustering: Spectral clustering consists of a transformation of the initial set of nodes/edges into a set of points in the space, whose coordinates are the element of the eigenvectors of the graph adjacency matrix. The set of points, can then be clustered by using well-known techniques, such as K-means clustering [17]. The very nice property of spectral graph theory is that we can acquire a lot of topological knowledge about the graph just by looking at its eigenvectors. For example, we can discover if the graph is connected, or even how many connected components exist in a graph. Similarly, we can find the number of existing communities in a graph. Nevertheless, the problem is computing the eigenvectors of a large graph is very costly if not impossible. We, therefore, has to look for alternative approaches to clustering, in order to process very large graphs.

2) Hierarchical Clustering: Hierarchical clustering approaches also aim at dividing the graph into tightly connected groups of node, where often the number and size of the groups are unknown. These techniques, therefore, allow for clustering with different resolutions, i.e., with a low resolution clustering the graph is partitioned into few big components, while we can zoom into these coarse components and find components of a finer resolution. As a result, a hierarchy of the clusters will be computed. These solutions can be classified into two main categories: *agglomerative algorithms*, as in [12], and *divisive algorithms*, as in [9]. In the first set of algorithms, we start by singles nodes in each cluster, and merge clusters that have many neighbors in common, whereas in the second set, we take the opposite approach. We start from one cluster that holds all the nodes, and then iteratively remove the edges to break the clusters into smaller pieces. Hierarchical algorithms are nice for they enable us to acquire clusters of any desired resolution. However, not all the graphs have a hierarchical topology, in which case zooming in and out of a cluster does not necessarily identifies meaningful clusters. Also, these algorithms are computationally expensive , thus, are impractical for big graphs.

3) Diffusion Based Clustering: The well-known theorem of Ford and Fulkerson [6] states that the min-cut between any two vertices s and t of a graph, i.e., any minimal set of edges whose deletion would topologically separate s from t, carries the maximum flow that can be transformed from s to t across the graph. The duality of the min-cut/max-flow problem has indeed inspired a class of diffusion based algorithm, including our work in this paper, for graph partitioning and community detection. There are already several solutions to detect max-flow in graphs, including [5], [22], and DiDic [8]. DiDic uses a statistical model to diffuse flows of different colors in a graph, while it biases the flows towards well-shaped regions. The problem is we need to give it the maximum number of communities in advance, and if this number is unnecessarily large, it can slow down the process significantly. Also, since

the initialization is random, each run of the algorithm on the same graph might result in a drastically different result. In effect, the final result is more affected by the initialization scheme rather than the graph topology.

VI. Conclusions

In this paper, we introduced a graph-based approach to coreference resolution. We showed that by using a graph representation of the documents and their context, and applying a community detection algorithm we can speed up the task of coreference resolution by a very large degree. More precisely, the complexity of our algorithm is $O(N \times \overline{d} \times rounds)$, where $\overline{d}$ is the average node degree and $rounds$ is the number of rounds before convergence. Moreover, the convergence time of the algorithm highly depends on the topology of the constructed graph and is proportional to the diameter of the largest connected component. The accuracy of coreference resolution could also be improved at the same time, because we are able to search beyond only pair-wise comparisons. The graph that we construct enables us to discover any existing closeness/similarity between any subset of documents. Thus, we can explore the solution space more freely and more smartly.

References

[1] http://www.opencalais.com/, January 2014.

[2] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *Proc. of conference on Computational linguistics-Volume 1*, pages 79–85. Association for Computational Linguistics, 1998.

[3] Jie Cai, Éva Mújdricza-Maydt, and Michael Strube. Unrestricted coreference resolution via global hypergraph partitioning. In *Proc. of CoNLL'11*, pages 56–60. Association for Computational Linguistics, 2011.

[4] Jie Cai and Michael Strube. End-to-end coreference resolution via hypergraph partitioning. In *Proc. of COLING'10*, pages 143–151. Association for Computational Linguistics, 2010.

[5] Gary William Flake, Steve Lawrence, C Lee Giles, and Frans M Coetzee. Self-organization and identification of web communities. *Computer*, 35(3):66–70, 2002.

[6] DR Ford and Delbert Ray Fulkerson. *Flows in networks*. Princeton university press, 2010.

[7] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[8] Joachim Gehweiler and Henning Meyerhenke. A distributed diffusive heuristic for clustering a virtual p2p supercomputer. In *Proc. of IPDPSW'10*, pages 1–8. IEEE, 2010.

[9] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proc. of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[10] Chung Heong Gooi and James Allan. Cross-document coreference on a large scale corpus. In *HLT-NAACL*, pages 9–16, 2004.

[11] Sherzod Hakimov, Salih Atilay Oto, and Erdogan Dogdu. Named entity recognition and disambiguation using linked data and graph-based centrality scoring. In *Proc. of SWIM'12*, page 4. ACM, 2012.

[12] John Hopcroft, Omar Khan, Brian Kulis, and Bart Selman. Natural communities in large linked networks. In *Proc. of SIGKDD'03*, pages 541–546. ACM, 2003.

[13] David Hope and Bill Keller. Maxmax: a graph-based soft clustering algorithm applied to word sense induction. In *Computational Linguistics and Intelligent Text Processing*, pages 368–381. Springer, 2013.

[14] David Jurgens. Word sense induction by community detection. In *Graph-based Methods for Natural Language Processing*, pages 24–28, 2011.

[15] George Karypis and Vipin Kumar. Multilevelk-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed computing*, 48(1):96–129, 1998.

[16] Aapo Kyrola, Guy Blelloch, and Carlos Guestrin. Graphchi: Large-scale graph computation on just a pc. In *Proc. of OSDI'12*, pages 31–46, 2012.

[17] Stuart Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

[18] Yucheng Low, Danny Bickson, Joseph Gonzalez, Carlos Guestrin, Aapo Kyrola, and Joseph M Hellerstein. Distributed graphlab: A framework for machine learning and data mining in the cloud. *Proc. of VLDB'12*, 5(8):716–727, 2012.

[19] Suresh Manandhar, Ioannis P Klapaftis, Dmitriy Dligach, and Sameer S Pradhan. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proc. of SemEval'10*, pages 63–68. Association for Computational Linguistics, 2010.

[20] Sebastian Martschat, Jie Cai, Samuel Broscheit, Éva Mújdricza-Maydt, and Michael Strube. A multigraph model for coreference resolution. In *Proc. of CoNLL'12*, pages 100–106. Association for Computational Linguistics, 2012.

[21] Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. 2003.

[22] Henning Meyerhenke, Burkhard Monien, and Thomas Sauerwald. A new diffusion-based multilevel algorithm for computing graph partitions of very high quality. In *Proc. of IPDPS'08*, pages 1–13. IEEE, 2008.

[23] Ruslan Mitkov. *Anaphora resolution*, volume 134. Longman London, 2002.

[24] Vincent Ng. Unsupervised models for coreference resolution. In *Proc. of EMNLP'08*, pages 640–649. Association for Computational Linguistics, 2008.

[25] Vincent Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proc. of ACL'10*, pages 1396–1411. Association for Computational Linguistics, 2010.

[26] Fatemeh Rahimian, Amir H. Payberah, Sarunas Girdzijauskas, Mark Jelasity, and Seif Haridi. Ja-Be-Ja: A distributed algorithm for balanced graph partitioning. In *Proc. of SASO'12*. IEEE, Sep. 2013.

[27] Altaf Rahman and Vincent Ng. Supervised models for coreference resolution. In *Proc. of EMNLP'09*, pages 968–977. Association for Computational Linguistics, 2009.

[28] Altaf Rahman and Vincent Ng. Syntactic parsing for ranking-based coreference resolution. In *Proc. of IJCNLP'11*, pages 465–473, 2011.

[29] Delip Rao, Paul McNamee, and Mark Dredze. Streaming cross document entity coreference resolution. In *Proc. of Coling'10*, pages 1050–1058. Association for Computational Linguistics, 2010.

[30] Yael Ravin and Zunaid Kazi. Is hillary rodham clinton the president?: disambiguating names across documents. In *Proc. of the Workshop on Coreference and its Applications*, pages 9–16. Association for Computational Linguistics, 1999.

[31] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[32] P. Sanders and C. Schulz. Engineering Multilevel Graph Partitioning Algorithms. In *Proc. of ESA'11*, volume 6942 of *LNCS*, pages 469–480, 2011.

[33] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[34] Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Proc. of Annual Meeting of the ACL: Human Language Technologies-Volume 1*, pages 793–803. Association for Computational Linguistics, 2011.

[35] Michael Strube. NLP approaches to reference resolution. *Tutorial notes, ACL*, 2, 2002.

[36] Nina Wacholder, Yael Ravin, and Misook Choi. Disambiguation of proper names in text. In *Proc. of ANLP'97*, pages 202–208. Association for Computational Linguistics, 1997.

[37] Yotaro Watanabe, Masayuki Asahara, and Yuji Matsumoto. A graph-based approach to named entity categorization in wikipedia using conditional random fields. In *Proc. of CoNLL'07*, pages 649–657, 2007.