An Approach to Web-scale Named-Entity Disambiguation

Luís Sarmento[1], Alexander Kehlenbeck[2], Eugénio Oliveira[1] and Lyle Ungar[3]

[1] Faculdade de Engenharia da Universidade do Porto - DEI - LIACC
Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal
las@fe.up.pt, eco@fe.up.pt
[2] Google Inc
New York, NY, USA
apk@google.com
[3] University of Pennsylvania - CS
504 Levine, 200 S. 33rdSt, Philadelphia, PA, USA
ungar@cis.upenn.edu

Abstract. We present a multi-pass clustering approach to large scale, wide-scope named-entity disambiguation (NED) on collections of web pages. Our approach uses name co-occurrence information to cluster and hence disambiguate entities, and is designed to handle NED on the entire web. We show that on web collections, NED becomes increasingly difficult as the corpus size increases, not only because of the challenge of scaling the NED algorithm, but also because new and surprising *facets* of entities become visible in the data. This effect limits the potential benefits for data-driven approaches of processing larger data-sets, and suggests that efficient clustering-based disambiguation methods for the web will require extracting more specialized information from documents.

1 Introduction

Realistic named-entity disambiguation (NED) of Web data involves several challenges that have not yet been considered simultaneously. First, when moving NED to the web we need to deal with high levels of ambiguity. Since there are so many documents in the Web, the same name will often refer to hundreds of different entities. This makes the problem much harder as compared with NED approaches for small collections where one needs to disambiguate only among a few possibilities. Second, distributions of mentions on the web are highly skewed. For each ambiguous name, there is usually one or two dominant entities to which the vast majority of mentions refer to, even when many entities share the same name. For example, most mentions of the name "Paris" found on the web refer to the capital of France (and a smaller number to Paris Hilton), while there are dozens of well-known entities with that name [4]. Table 1 shows hit counts for five queries sent to Google containing the word "Paris" and additional (potentially) disambiguating keywords. These values are merely indicative of the orders of magnitude at stake, since hit counts are known to change significantly over time. The real

[4] See the Wikipedia disambiguation page for "Paris": http://en.wikipedia.org/wiki/Paris_(disambiguation)

challenge is to be able to disambiguate between mentions of the less frequently mentioned entities, for which there is proportionally much less information and more noise. Third, most solutions to NED presented so far involve processing relatively small datasets. Realistic NED involves processing web-scale collections (terabyte size), requiring computationally efficient ways of representing and processing data and, sometimes, involving practical decisions that might affect negatively final results for some cases.

# query	# hit count (x10^6)	%
paris	583	100
paris france	457	78.4
paris hilton	58.2	9.99
paris greek troy	4.130	0.71
paris mo	1.430	0.25
paris tx	0.995	0.17
paris sempron	0.299	0.04

Table 1. Number of Google hits obtained for several entities named "Paris"

There are also other fundamental questions that have not yet been investigated. Many of the solutions to NED involve data-driven techniques, such as clustering. Such techniques usually benefit from processing larger amounts of data. Therefore, one would expect to obtain better NED results as the size of the collection to be disambiguated increases. However, as the size of the collection to be disambiguated becomes larger, the variety of different entities and contexts that have to be dealt with also increases. As the contexts in which mentions occur become more diverse, data-driven approaches potentially become harder. The exact balance between these two effects has yet to be quantified.

In this paper we present a clustering-based approach to disambiguating entities on the Web. The algorithm we propose is capable of dealing with an arbitrarily high number of entities types, is scalable to the number of mentions on the web, and can be distributed over a cluster of machines to process large web collections. For evaluating the results of the disambiguation procedure we developed a gold standard based on entity information extracted from Wikipedia. We experimented disambiguating samples of the web with increasingly large sizes to test how well the algorithm scales and whether or not more data leads to better results. Results suggest that as the size of the collection increases, more complex cases of ambiguity emerge, making the definition of the NED task itself less clear. This seems to be an intrinsic characteristic of highly heterogeneous document collections, and suggests the existence of fundamental upper limits on the performance of clustering-based approaches to NED based only on name co-occurrence information.

2 Related Work

There are currently two main lines of research on NED: (i) clustering approaches based on information extracted from the documents (e.g., [1–4]) and (ii) approaches that use external knowledge sources containing information about entities (e.g., the Wikipedia) to perform disambiguation (e.g., [5–7]).

Mann and Yarowsky [3] present a disambiguation procedure for person names based on a multi-pass clustering procedure. First, mentions are compared using an all-against-all strategy, in order to obtain very "pure", yet small, *seed* clusters, which should represent the main entities. Then, the remaining mentions are assigned to these seed clusters using a nearest-neighbor policy. In a third step, clustering proceeds until no more clustering is possible. The authors experimented using several different features to describe mentions. Best results were obtained using biographic features in combination with other statistically obtained features. Another clustering-based disambiguation method is presented in [1]. Mentions are described by a vector composed of tf-idf weighted terms extracted using a 55-word window. The authors compare two methods based on variations of streaming-clustering (which are computational efficient but order dependent and sensitive to outliers) and one agglomerative clustering method (which involves all-against-all comparisons). Results showed that the agglomerative clustering method leads to better precision and recall figures and higher stability to changes in parameters (similarity threshold and data partitioning). Two other methods to disambiguate personal names, based on clustering and graph partitioning, are presented and compared in [2]. Again, information about name co-occurrence is used to disambiguate person entities. The authors conclude that name co-occurrence information provides an advantage over using other features to achieve disambiguation: However, this method considers only situations where only one name at a time is ambiguous. The approach presented in [8] is more sophisticated because it assumes that co-occurring names are themselves ambiguous. Thus, an iterative clustering approach is proposed that aims at *collectively* resolving ambiguity.

In [5], a set of *disambiguation vectors* is built using information extracted from Wikipedia. Assuming each entity has its own Wikipedia page, a vector description of the entity is build using words found inside a 55-word window around mention of the name in the corresponding page. Wikipedia categories are also added to the vectors, using a pre-computed *word to category* index, thus exploiting strong correlations between words in text (sparse features) and categories (e.g: "concert" is more strongly correlated with category "Musicians" than with "Professional Wrestlers"). Disambiguation is made by comparing vectors of mention to be disambiguated with the set of *disambiguation vectors*. In [6] Wikipedia is also used to build vector representations of entities. However, the method does not rely on direct comparison between vector representations of entities and vector of each individual mentions but, instead, it tries to maximize the agreement between *all* the disambiguation hypothesis of *all* mentions in a document. In [7] the authors attempt large-scale taxonomy based disambiguation / resolution, over a collection 264 million documents (although the number of mentions to disambiguate was limited to 550 million). The method involved comparing the 10-word window context around a mention with "typical" contexts that had been previously collected and manually associated with the 24 *reference nodes*, i.e. largest nodes of the

taxonomy (e.g. city, profession, country). Disambiguation (or resolution) is achieved by finding which node in the taxonomy that includes the ambiguous name belongs to the subtree of the reference node with higher similarity with the context of the mention (based on cosine metric and tf-idf feature weighting). The authors report an accuracy of 82%.

3 A Clustering Approach to NED

In this work we focus on the *disambiguation problem*, that is the problem of determining whether occurrences of the *same name* in *different documents* refer to the same entity, or to different ones that share the same lexical representation (following standard practice – [9] – we assume that a name inside a document can only refer to one entity). For example, the name "Amsterdam" can be used refer to many different geographic locations, to a novel, to several songs, to a ship, to a pop music band, and to many other entities[5]. We do not address the related problem of *conflating mentions* that use different names to refer the same entity (e.g., "George W. Bush", "George Bush", "Mr. Bush", "president Bush", "the President", "Dubya"). Solution to the *name conflation* problem can be built on top of the solution provided for the name ambiguity problem (for an interesting approach to large-scale name conflation check [4]).

NED can be formulated as a clustering task. Let m_{ij} represent a *mention*, i.e., the occurrence of name n_i in document d_j, and let $M_{all} = \{m_{11}, m_{21}, ...m_{ik}\}$ be the set of all mentions found in a given document collection $C = \{d_1, d_2, ...d_k\}$. Disambiguation can be achieved by clustering together all mentions in M_{all} that refer to the same entity e_j. The goal is to partition M_{all} in several disjoint clusters of mentions, M_1, M_2, M_3 ... M_n, so that each of them contains mentions that refer to one and only one entity e_j. Also, all mentions of a given entity e_j should end up in a single cluster.

3.1 Feature Vector Generation

We start by assuming that a mention of a given name can be disambiguated using information about the names with which it co-occurs *within the same document*. For example, mentions of "Amsterdam" that refer to the capital of the Netherlands will probably co-occur with mentions of "Netherlands", "Utrecht" or "Rijksmuseum", while those mentions of Amsterdam that refer to the novel, will probably co-occur with "Ian McEwan" or "Amazon". Under this assumption, describing mentions using the set of co-occurring names as features ({"Netherlands", "Utrecht", "Rijksmuseum"...} vs. {"Ian McEwan", "Amazon"...}) should lead clusters that group mentions that refer unambiguously to one specific entity (the capital of the Netherlands vs. the novel).

Let N(d_k) be set of names found in document d_k. The mention of name n_j in document d_k, m_{jk} will be described by a feature vector of tuples name - value, (n_i, v_i):

$$\overline{m}_{jk} = [(n_1, v_1), (n_2, v_2), (n_3, v_3), ...(n_i, v_i)] \quad (1)$$

with $n_i \in N(d_k) \setminus n_j$, and v_i being a value obtained through a generic feature weighing function (for example TF-IDF or Mutual Information).

[5] Check `http://en.wikipedia.org/wiki/Amsterdam_(disambiguation)`

The input for our clustering procedure is an annotated collection of documents, C_{annot}. Therefore, it requires names to be previously *identified* in each document, although *type classification* is not needed.

3.2 Clustering Procedure Overview

The procedure we propose for performing NED over a collection of annotated documents C_{annot} starts by extracting all names from each document d_k to generate mention feature vectors $\overline{m}_{jk}$ (a mention is the occurrence of a name in a document). Feature vectors are then grouped by name, so as to have a set of mention feature vectors *per name*: $M(n_j) = \{\overline{m}_{j1}, \overline{m}_{j2}...\overline{m}_{jx}\}$. Vectors inside each set $M(n_j)$ are then compared according a given comparison strategy and similarity metric $sim(\overline{m}_{nj}, \overline{m}_{nk})$ (e.g: Cosine or Jaccard Distance). Finally a clustering algorithm is applied to each $M(n_j)$, using information about vector similarity computed in previous step.

The algorithm itself is generic in the sense that it does not establish any specific strategy for comparing feature vectors prior to clustering, nor a specific choice for the clustering technique. At this point, we assume only that an efficient algorithm exists for performing vector comparison and clustering. For example, Min-Hash techniques [10] provides a efficient way for computing an approximation to the nearest-neighbor problem, which can be used for computing distances between vectors. Clustering by Committee [11] and variations of streaming clustering techniques [12] might be an option for the clustering stage. In any case, one important advantage of this algorithm is that it provides a natural way for distributing computational load. Since feature vectors are grouped by name, all information that is required to resolve ambiguity for each name is aggregated and can be processed separately: both the calculation of vector similarities and the clustering process can be distributed over a cluster of machines, on a per-name basis, thus helping scalability.

4 Vector Comparison and Clustering

As explained, the size of the clustering problem at stake - millions of names and thousands of millions of mentions - requires distributed algorithms that can be deployed on large computer clusters. Right from the beginning our method was designed to be run on a Map-Reduce [13] platform, a data intensive supercomputing paradigm that simplifies the distribution of data (hundreds of gigabytes) and tasks over thousands of computer nodes (typical commodity computers). Map-reduce provides a generic framework for scaling algorithms to very large data sets but in order to choose an appropriate clustering method for NED, some specific characteristics of the dataset and of the problem should be taken into account. First, the mention distribution is highly skewed, and is dominated by the one or two most popular entities. Thus, the clustering algorithm chosen should be able to handle unbalanced data distributions and still produce correct clusters both from dominant and non-dominant entities. Second, the number of entities in which the set of mentions $M(n_j)$ should be mapped, and thus the final number of clusters, is not known in advance. Therefore, the stopping criteria for the clustering procedure should not depend on a predefined number of final clusters desired, which is

difficult to estimate. Instead, it should depend on parameters related with input data and cluster properties.

We propose using a graph-based clustering approach. For each name n_j, we start by computing pairwise distances between feature vectors to build the link graph $\mathcal{G}(n_j)$. Two mentions are linked in the graph if their similarity is higher than a given threshold s_{min}. Then, find the connected components of the Link Graph $\mathcal{G}(n_j)$. The retrieved connected components represent the clusters we seek. The only parameter of this approach is s_{min}; there is no need to set the target number of clusters to be produced. So far we have not yet found an automatic method for estimating the s_{min} parameter. Values used in our experiments range from 0.2 to 0.4.

When building the link graph for each name $\mathcal{G}(n_j)$ one only needs to perform enough comparisons between mentions to build a graph that is sufficiently connected to allow retrieving the correct components. The fact that the distribution of mentions among the entities is highly skewed turns out to be advantageous for building the link graph $\mathcal{G}(n_j)$. If we pick mentions randomly from the set $M(n_j)$, for any of the mentions belonging to the dominant entities (one or two) it should be possible to quickly find another one that turns out have a higher than threshold similarity (because there are so many of them). Then, for mentions of the dominant entities, we can obtain a significant decrease in the number of comparisons while almost surely keeping enough connectivity to retrieve the connected components. We showed elsewhere [14] that if each mention is compared to other mentions only until k_{pos} above-threshold similar mentions are found, it is possible to build a sufficiently connected link graph in O($|M(n_j)| \cdot C \cdot k_{pos}$), with C being the number of true clusters (i.e., different entities for the name n_j) in $M(n_j)$. Since the number of entities for each name is expected to be orders of magnitude smaller than the number of it mentions, this approach leads to significant savings in computational work as compared to an all-against-all comparison strategy (i.e. O($|M(n_j)|^2$)).

4.1 Additional Scalability Issues

There are still some important scalability problems that we need to solve. First, there are so many mentions on the web for the most frequent names that the corresponding feature vectors cannot be simultaneously fit into the RAM of a single machine to perform comparisons between them. For illustration purposes, we present in Table 2 the number of documents (hence mentions under our definition) found by Google for a few very frequent, and ambiguous, names (we use the number of possible entities found in the corresponding Wikipedia disambiguation page for each name as a rough indicator of its ambiguity). Second, even if they did fit simultaneously in RAM, processing these very frequent names would require much more time than processing less frequent names (which may have only a few hundred mentions), leading to extremely long tails in the overall processing time. Therefore, we need to break the set of mentions for each name into smaller partitions, each with n_{max} mentions, so that they can be distributed more evenly across machines.

However, by splitting the data into multiple partitions and placing them in different machines, we loose the ability to compare all mentions that would be required to find appropriate (i.e. complete) clusters. In fact, for each (frequent) name we are breaking the

name	# Wiki Entities	Google Hits ($\times 10^6$)
Paris	90	583
Amsterdam	35	185
Jaguar	34	73.4
Pluto	25	13.8

Table 2. An illustration on the number of Google hits found on the web for some frequent names (hits may change), and the corresponding number of entities found in Wikipedia.

corresponding clustering problem into several *independent* clustering problems. Many of these partitions will produce clusters that correspond to the same entity, and so they need to be merged afterwards. Since after the first clustering pass we should have much less clusters than mentions, re-clustering these clusters is certainly a more tractable problem. Clusters can be described by the feature vectors generated from the aggregation of feature vectors of the mentions they contain (e.g., their centroid). Comparisons can then be made using any vector distance metric over such vector descriptions, also on a per-name basis.

After the first stage of clustering, the size of the resulting clusters should also follow a highly skewed distribution. There will be several larger clusters corresponding to the few dominant entities, and many smaller clusters corresponding both to non-dominant entities and to (small fragments of) dominant entities. Taking into account this typical distribution (that we systematically found in our experiments), we developed a dedicated re-clustering procedure to merge results from partitions. This procedure is applied independently for each name, and thus it can be trivially run in parallel. For each name, we group all clusters obtained in each partition and divide them in two groups: Big Clusters, C_{big} and Small Clusters, C_{small}. C_{big} is composed of the 10% biggest clusters produced in the first clustering pass, while all others are included in C_{small}. We then use the following re-clustering strategy:

1. **Pre-assign Small Clusters to Big Clusters:** Start by trying to assign each small clusters to one big cluster. This assignment is made using a nearest neighbor strategy (with a minimum similarity threshold), and thus tends not to make many incorrect assignments, while greatly reducing the total number of clusters. Cluster descriptions are updated accordingly.
2. **Merge Small Clusters:** Try to merge all the unassigned small clusters with each other. The main goal here is to make sure that some of the less represented entities grow into medium size clusters, so they get enough "critical mass" to be kept, even if we simply filter out the smaller clusters. Cluster descriptions are updated accordingly.
3. **Merge Big and Medium Clusters:** Try to re-cluster the medium and big clusters based on only a *few top features*. The intuition is that big clusters can usually be "described" by a small number of features (e.g., their top 3), which will be highly discriminative for the entity at stake. We thus achieve cluster consolidation, while reducing the risk of performing incorrect merge operations due to noisy features.
4. Repeat 2 and 3 to reduce fragmentation.

Note that Big clusters and Small Clusters are never compared simultaneously, (i.e. all-against-all), which avoids the problems that might come from comparing elements of with significant size differences.

5 Evaluation Framework

Evaluating the results of clustering algorithms is difficult. When gold standard clusters are available, one can evaluate clusters by comparing clustering results with the existing standard. Several metrics have been proposed for measuring how "close" test clusters are to reference (gold standard) clusters. Simpler metrics are based frequency counts regarding how individual items [15] or pairs of items [16, 17] are distributed among test clusters and gold standard clusters. These measures, however, are sensitive to the number of items being evaluated, so we opted for two information-theoretic metrics, which depend solely on the item distributions.

Given two sets of clusters, the *test clusters*, T with $|T|$ clusters, and the *gold clusters*, G, with $|G|$ clusters, we wish to evaluate how well clusters in T, t_1, t_2,...$t_{|T|}$ represent the clusters in G, g_1, g_2,... $g_{|G|}$. We first obtain the $|I|$ (intersection) matrix with $|T|$ lines and $|G|$ columns. Elements i_{xy} of $|I|$ indicate the number of items in common between the test clusters t_x and gold clusters g_y. Ideally, all the elements in a given test cluster, t_x, should belong to only one of the gold clusters. Such t_x cluster is considered "pure" if it contains only mentions of a unique entity as defined by the gold standard. If, on the other hand, elements from t_x are found to belong to several gold clusters, then the clustering algorithm was unable to correctly delimit the entity, and disambiguation was not totally achieved.

To quantify how elements in test cluster t_x are distributed over the gold standard, we use the entropy of the distribution of the elements in t_x over all the clusters g_y. High quality clusters should be very pure and thus have very low entropy values. Let $I_t(x)$ be the total number of elements of cluster t_x that were found in gold clusters. Then:

$$e_t(t_x) = \sum_{y=0}^{|G|} -\frac{i_{xy}}{I_t(x)} \cdot \ln(\frac{i_{xy}}{I_t(x)}) \tag{2}$$

Therefore, for all test clusters obtained for name n_j we can compute $E_t(n_j)$ as the weighted average of the entropy values e(t_x) obtained for each test cluster, t_x:

$$E_t(n_j) = \frac{\sum_{x=0}^{|T(n_j)|} |t_x| \cdot e(t_x)}{\sum_{x=0}^{|T(n_j)|} |t_x|} \tag{3}$$

with $|t_x|$ being the number of mentions in cluster t_x, including those not found in gold clusters. $|T(n_j)|$ is the number of test clusters obtained for name n_j. We are also interested in measuring how elements from clusters in gold standard are spread throughout the test clusters we produced. Again, we would like to have all elements of gold standard clusters in the least number of test clusters possible, ideally only one. Then, for each gold cluster g_y we can also use entropy $e_g(g_y)$ to measure how the elements of a

gold standard cluster g_y are spread over the clusters we are testing. $e_g(g_y)$ can be computed by a formula similar to that of Equation 2, substituting references to test cluster by reference to gold clusters, and vice-versa. Similarly, a global performance figure, $E_g(n_j)$, can be obtained by performing a weighted average over $e_g(g_y)$ for all gold clusters (similar to Equation 3).

Finally, we need to evaluate recall, i.e., the proportion elements in gold cluster that are in fact found in any test cluster. If $I_g(y)$ is the total of elements in cluster g_y that were found test clusters, we may define the *mention* recall metric for gold cluster g_y as:

$$r_m(g_y) = \frac{I_g(y)}{\sum_{y=0}^{|G(n_j)|} |g_y|} \tag{4}$$

An overall Recall figure for this name, $R_m(n_j)$, could be obtained again by doing a weighted average of $r(cg_k)$ over all gold clusters:

$$R_m(n_j) = \frac{\sum_{k=0}^{|G(n_j)|} |g_y| \cdot r_g(g_y)}{\sum_{j=0}^{|G(n_j)|} |g_y|} \tag{5}$$

Similarly we can compute $R_e(n_j)$ which measures how many of the *entities* included in the gold standard clusters for n_j are found in the corresponding test clusters. This figure is important because mention distribution among entities is expected to be very unbalanced.

The previous figures are calculated for each name, $n_j \in \mathcal{N}$. For assessing the global performance of the clustering-based NED procedure for all names in $\mathcal{N}$, we need to combine the performances obtained for the individual names, n_i. To do so, we use the arithmetic average of the previous metrics over all names: E_t, E_g, R_m and R_e.

5.1 Preparing the Gold Standard

We used the English version of the Wikipedia to develop a gold standard for evaluating NED (although the procedure can be replicated for other languages). We assume that each article in Wikipedia can be related to one unambiguous entity / concept. Let $W_{seed}(n_j)$ be the set of Wikipedia articles found for name n_j (n_j can usually be easily identified by the article title). If the number of articles for n_j is greater than one, then n_j is know to be ambiguous, and each possible entity is unambiguously related to one of the articles.

The set $W_{seed}(n_j)$ can be used as seed for obtaining more documents that unambiguously refer entities mentioned using name n_j. For each page in $W_{seed}(n_j)$, which refers to an unambiguous entity e_k, we find all its immediate neighbors in the web link graph, both inside and outside Wikipedia. These linked pages will probably have mentions of the name n_j, which can be assumed to refer to the same entity e_k described by the Wikipedia article to which they are linked. The output of the expansion procedure is a set of gold clusters for each name, n_j. These gold clusters are a set of pages that mention name n_j and that can be uniquely assigned to one Wikipedia article (which stands for a specific entity). A problem arises when such pages are linked to more than one Wikipedia article that describes entities mentioned by the same name, i.e. to more

than one article from the same seed set $W_{seed}(n_j)$. In those cases, we cannot automatically decide which entity is in fact being mentioned, and thus all occurrences of the corresponding name in that document have to be considered ambiguous. Thus, those documents are excluded from the gold clusters for the name at stake (n_j). Using such expansion and filtering procedures, we obtained a gold standard with around 9.3 million mentions for about 52,000 ambiguous names. In Table 3 we present the distribution the gold names in four classes based on the entropy of the corresponding gold clusters. Low entropy values correspond to names where there is clearly one dominant entity to which the vast majority of the mentions belong, while high entropy values are related with names for which mention distribution among entities is less skewed.

Entropy	# names	% names
0 to 0.1	768	1.5
0.1 to 0.5	7610	14.5
0.5 to 1	29304	56.0
1 or more	14657	28.0

Table 3. Internal entropy of the names in the gold standard

6 Experimental Setup

In order to investigate how scalable our algorithm is and whether or not NED performance improves as the amount of data to be disambiguated grows, we experimented clustering different portions of a large web collection with over a billion documents (in English). The web collection had been previously analyzed by a wide scope named-entity recognition system [18], so we were able to use name annotations in each document to produce feature vectors for the clustering procedure. We first took a 1% sample of the complete web collection (randomly choosing 1% of the documents) and we performed the complete NED procedure several times while slowly increasing the value of the s_{min} parameter, i.e. the minimum similarity for two mention vectors to be considered linked in the Link Graph. This allowed us to obtain several reference points for the values of E_t, E_g, R_m and R_e for a 1% sample. We then proceeded by performing NED over samples of different sizes - 0.5%, 2% and 5% - so that we could compare the results with the ones previously obtained for 1%. To allow a fair comparison, we matched the results obtained for the 0.5%, 2% and 5% samples with those obtained for one of the 1% samples with the *closest value for* E_t, i.e., similar "purity" values. Results were evaluated against the gold standard (see Section 5.1).

All code was implemented in the Map-Reduce [13] paradigm and experiments were run in parallel over 2048 machines. Because of limited RAM and load balancing issues, names were divided in partitions of maximum size 3000. For very frequent names, this may lead to a considerable fragmentation, because there can be hundreds of thousand of mentions for such names. Each mention vector was limited to having, at most, 5000

features (i.e., corresponding to co-occurring names in the same document). We use the Jaccard Metric to compare vectors (we previously perform filtering of less significant features based on minimum tf-idf and frequency values). At the end of first stage of clustering, all clusters with less than 5 elements are filtered out to reduce the total number of clusters to be processed in the second stage. This can have obvious impacts on final recall values, if there are too many such small clusters at the end of the first stage.

7 Results and Analysis

Table 4 contains the values for E_t, E_g, R_m and R_e for the 0.5%, the 2% and the 5% samples, and corresponding values for the 1% samples with the closest E_t obtained. It also presents the value of s_{min} with which each result was obtained, and the *clustering ratio* parameter, C_{rat}, which gives the relation between the number of clusters obtained (after filtering) and the number of clusters in the gold standard.

$\%@s_{min}$	E_t	E_g	$R_m(\%)$	$R_e(\%)$	C_{rat}
0.5@0.3	0.0003	0.0056	0.024	1.16	1.23
1.0@0.4	0.0001	0.0085	0.055	1.74	1.82
1.0@0.25	0.0042	0.0226	0.135	3.70	2.06
2.0@0.3	0.0042	0.0312	0.294	5.43	3.27
1.0@0.2	0.0103	0.0212	0.186	5.00	2.18
5.0@0.3	0.0140	0.0797	0.912	12.4	6.91

Table 4. Performance metrics for three different comparison scenarios.

One first observation is that for keeping the values of E_t comparable, the s_{min} parameter of the larger sample has to be higher than that of the smaller sample. This was expected, because as the number of mentions to be disambiguated increases, the corresponding vector space tends to become more dense. Thus, in order to avoid noisy clusters we need to increase s_{min} to make sure that only mention vectors that are really close in the vector space actually become linked in the Link Graph, and thus generate pure clusters. Increasing s_{min} should, however, lead to higher fragmentation and to producing many small clusters. The C_{rat} parameters increases both when the size of the sample increases, and when s_{min} increases for the same sample size (the 1% sample), which confirms that fragmentation does in fact increase.

Recall values, R_m and R_e, seem very low. However, one has to take into account that the number of gold standard documents in the sample is proportional to the sample size. Thus, for the 1% sample, recall values cannot be higher than 1% (if sampling is unbiased as we expect it to be). We are primarily interested in observing the relative changes of recall with sample size. For that, we computed the ratios between the recall figures (R_m and R_e) obtained for the larger and the smaller samples that are being compared in each pair of rows. Table 5 shows the value of these two parameters $r_m^{+/-}$, $r_e^{+/-}$ for the three comparison situations. For the *0.5% vs 1%* and the *1% vs 2%* scenarios, we can see that even with better (i.e., lower) values for E_t, the mention recall R_m

% vs %	$r_m^{+/-}$	$r_e^{+/-}$
0.5% vs. 1.0%	2.28	1.5
1.0% vs. 2.0%	2.17	1.48
1.0% vs. 5.0%	4.9	2.48

Table 5. Ratio between Recall values R_m and R_e of larger and smaller samples.

increased faster than the data size; in both cases the recall ratio $r_m^{+/-}$ is higher than the data increase ratio (twice as many documents). For the *1% vs 5%*, the 5-fold increase in the number of documents did not lead to a 5-fold increase in R_m, although it almost did. However, if we look at the $r_e^{+/-}$ ratio for the entity recall, we see that it is not increasing as fast as the data size is, meaning that we are losing entities (found in the gold standard) as we process more data. The combination of these two factors indicates that for the entities being kept we are able to cluster more and more mentions, but we are losing all the mentions for some more obscure entities. Additionally, recall ratios are systematically decreasing as we increase the size of the data sets to be disambiguated. We believe that there are two main reasons for this.

The first reason is a consequence of the compromises we had make in our algorithm to allow it to process web-scale collections. As we increase the size of the sample, and thus the number of mentions to be disambiguated, the number of partitions made for each name also increases (each partition has 3,000 mentions). The overall clustering problem is thus divided into a large number of smaller independent clustering problems whose solutions should ideally be merged in the re-clustering stage. However, for less frequent entities, the partitioning procedure will disperse the mentions over too many partitions, which, in combination with high values for s_{min}, will lead to generation of more but much smaller clusters. Chances are that most of these clusters end up being filtered out after the first stage of clustering and do not even get the chance of being merged in the second clustering stage. Since our gold standard contains some quite exotic entities mentioned in Wikipedia that are probably under-represented in the web collection, the corresponding clusters will be relatively small and will eventually be completely filtered out. This progressively affects R_t, and also R_m, as we the sample gets larger, compensating possible positive effects that would result from having more data and a more dense vector space. These positive effects were only visible when partitioning was not too problematic (i.e., for the 0.5%, 1.0% and 2.0% samples).

The second reason has to do with a more fundamental issue for NED, and it only became obvious after manually inspecting the results for very frequent names, such as "Amsterdam". As we increased the size of the data to be disambiguated, and s_{min} accordingly, we noticed that results for such type of names were composed of many clusters concerning the several possible entities, as expected, but for the dominant entities at stake (for example Amsterdam, the Dutch capital) there was a surprisingly high number of medium and large clusters. These clusters should have been merged together into a single very large cluster since they all rather obviously (based on inspection of their features) seemed to refer to the same (dominant) entity.

However, each of these clusters appeared to contain mentions that referred to specific *scopes* to which the entity occurs, or to different *facets* that the entity could as-

sume. For example, some clusters referred to "Amsterdam" as *world capital*, for which the typical features of the clusters (co-occurring names) were other large cities of the world, such as "Paris", "New York" or "London", while others clusters would refer to "Amsterdam", a *city in the Netherlands*, and would have as typical features names of cities in the Netherlands. In other cases, the clusters produced had features that apparently were not related to the entity, but that were in fact associated with specific contexts of the entity at stake. For example, since there are many scientific editors based in Amsterdam, we found relatively large clusters whose typical features are names of editors (such as "Elsevier" or "Elsevier Science"), and other names related to scientific conferences and societies. There are many other similar examples, where the clusters refer to distinct possible facets of the entities, such as different geographic scopes or different times in history ("Paris" nowadays v.s during the French Revolution). Interestingly, most clusters corresponding to different and highly specialized facets of a dominant entity contained many more mentions than the "main" clusters of non-dominant entities (e.g. "Amsterdam" the novel, or "Paris" of Troy from Greek mythology).

From a clustering point of view, the different, yet consistent, name co-occurrence patterns that dominant entities are seen as distinct "sub-entities", leading to smaller clusters in both clustering stages. The resulting fragmentation effect only becomes obvious when one tries to disambiguate very large and heterogeneous data-sets such as the web: as the size of the corpus increases, more facets of the same entity tend to emerge and make this fragmentation effect more visible. The key point is that, even if we had enough RAM and CPU resources to avoid the partitioning of mentions, fragmentation for these dominant entities would probably still occur. The problem arises from the features used to describe each mention, i.e., the set of co-occurring names, which does not carry sufficient information for merging the existing facets.

Conceptually, this situation is close to the *homonymy* vs. *polysemy* problem ([19]), which is often encountered in word-sense disambiguation tasks. While homonyms have no related senses ("river bank" vs. "bank account"), polysemous words do share some relation ("the Stoic school" vs. "the school room"). In our case, different entities with the same name ("Amsterdam" the city vs. "Amsterdam" the novel) should be seen as homonynmy, while the multiple "facets" found for the same entity can be seen as the multiple "senses" of a polysemous name ("Amsterdam" a world capital vs. "Amsterdam" a city in the Netherlands). Recently, some Named-Entity Recognition (NER) evaluation programs, such as ACE [20] and HAREM [21], have recognized the existence of inherently ambiguous situations, specially those that exhibit a more or less systematic pattern. For example, ACE introduced the notion of *geo-political entities* for entities such as countries, that contain a population, a government, a physical location, and a political existence, and that can thus be mentioned by several different facets. However, the large number of possible facets that we observed in our experiments, some quite specialized (e.g. "Amsterdam" as an important city in the field of scientific publishing), does not allow a simple and systematic identification of all relevant cases.

Ideally we would want to merge all facets belonging to the same entity but still keep information about the distinct facets (whose meaning might be understandable at a later stage). What our results show is that name co-occurrence information is not sufficient for merging facets and that more specialized information is required. For in-

stance, e-mail addresses or biographic features might help merging different facets of people entities, as geographic related information (geo-codes) might help in the case of locations. More generally, web link information might provide good clues for merging facets of arbitrary types of entities. Mentions of the same name in highly connected parts of the web graph indicate that we are probably dealing with the same entity, even if the corresponding mentions have been placed in different clusters. All this additional information might be used in a third clustering stage to merge all possible facets (i.e clusters) of the same entity.

8 Conclusion and Future Work

We have presented a wide-scope NED algorithm that is scalable and explicitly handles the power law distribution of entities in the web, allowing us to cluster a billion mentions. We also presented a novel evaluation strategy that uses information extracted from Wikipedia to automatically generate a gold-standard. Our experiments do not provide a complete solution to web-scale NED. Instead, they raise several fundamental questions (both theoretical and practical) that have so far been neglected by most approaches to NED. We showed that NED on the web involves dealing not only with obvious scaling issues, but with less obvious and more fundamental problems related to the intrinsic variety of web data. As the data volume grows, new facets of entities become apparent, making NED a more complex and less clearly defined task. We showed that name co-occurrence information is not sufficient for merging distinct facets of the same entity. Future work will include investigating potential features such a document links, email addresses, and geocodes that can serve to merge different facets of entities.

Acknowledgements

This work was developed while Luís Sarmento was an *engineering intern* and Lyle Ungar was a *visiting researcher* at Google offices in NYC. The authors would like to thank the Google team for all the help and support. Special thanks to Nemanja Petrovic for his work in developing the gold standard set and to Casey Whitelaw for his help in providing NER-annotated data.

References

1. Gooi, C.H., Allan, J.: Cross-document coreference on a large scale corpus. In: HLT-NAACL. (2004) 9–16
2. Malin, B.: Unsupervised name disambiguation via social network similarity. In: Workshop on Link Analysis, Counterterrorism, and Security in conjunction with the SIAM International Conference on Data Mining. (2005) 93–102
3. Mann, G.S., Yarowsky, D.: Unsupervised personal name disambiguation. In: Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003, Morristown, NJ, USA, Association for Computational Linguistics (2003) 33–40
4. Yates, A., Etzioni, O.: Unsupervised resolution of objects and relations on the web. In: Proceedings of NAACL HLT, Rochester, NY (April 2007) 121–130

5. Bunescu, R., Pasca, M.: Using encyclopedic knowledge for named entity disambiguation. In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06). (2006) 9–16
6. Cucerzan, S.: Large scale named entity disambiguation based on wikipedia data. In: The EMNLP-CoNLL Joint Conference. (June 2007) 708–716
7. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J.Y.: Semtag and seeker: bootstrapping the semantic web via automated semantic annotation. In: WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, ACM (2003) 178–186
8. Bhattacharya, I., Getoor, L.: Collective entity resolution in relational data. ACM Trans. Knowl. Discov. Data **1**(1) (2007) 5
9. Gale, W.A., Church, K.W., Yarowsky, D.: One sense per discourse. In: HLT '91: Proceedings of the workshop on Speech and Natural Language, Morristown, NJ, USA, Association for Computational Linguistics (1992) 233–237
10. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proc. of 30th STOC. (1998) 604–613
11. Pantel, P., Lin, D.: Document clustering with committees. In: SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM Press (2002) 199–206
12. Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L.: Clustering Data Streams: Theory and Practice. IEEE Transactions on Knowledge and Data Engineering **15**(3) (2003) 515–528
13. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: OSDI ' 04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, google labs (2004) 137–150
14. Sarmento, L., Kehlenbeck, A., Oliveira, E., Ungar, L.: Efficient clustering of web-derived data sets. In: Proceedings of the International Conference on Machine Learning and Data Mining (MLDM) 2009. LNAI, Leipzig, Germany, Springer Verlag (July 23-25 2009)
15. Zhao, Y., Karypis, G.: Criterion Functions for Document Clustering: Experiments and Analysis. Technical report, University of Minnesota, Minneapolis (2001)
16. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems **17** (2001) 107–145
17. Meilă, M.: Comparing clusterings—an information based distance. J. Multivar. Anal. **98**(5) (2007) 873–895
18. Whitelaw, C., Kehlenbeck, A., Petrovic, N., Ungar, L.: Web-scale named entity recognition. In: ACM 17th Conference on Information and Knowledge Management: CIKM 2008, ACM Press (2008)
19. Krovetz, R.: Homonymy and polysemy in information retrieval. In: In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97. (1997) 72–79
20. Doddington, G., Mitchell, A., Przybocki, M., Ramshaw, L., Strassel, S., Weischedel, R.: The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. Proceedings of LREC 2004 (2004) 837–840
21. Santos, D., Seco, N., Cardoso, N., Vilela, R.: Harem: An advanced ner evaluation contest for portuguese. In Calzolari, N., Choukri, K., Gangemi, A., Maegaard, B., Mariani, J., Odjik, J., Tapias, D., eds.: Proceedings of the 5th International Conference on Language Resources and Evaluation, LREC 2006, Genoa, Italy, ELRA (22–28 May 2006) 1986–1991