RT-NED: Real-time named entity disambiguation on Twitter streams

Alexandre Davis[1], **Walter Santos**[1], **Adriano Veloso**[1], **Wagner Meira Jr.**[1], **Alberto Laender**[1], **Altigran Soares da Silva**[2]

[1]Universidade Federal de Minas Gerais
[2]Universidade Federal do Amazonas

{agdavis, waltersf, adrianov, meira, laender}@dcc.ufmg.br
alti@dcc.ufam.edu.br

Abstract. *Mining data from social midia channels, such as Twitter, is an increasingly important challenge. Messages continuously flow through the Web in high volumes, and the lack of a syntactical structure makes it hard to extract information from them. For instance, identifying entities in such messages is a requirement for applications such as context extraction and sentiment analysis. In this paper, we present a novel technique to solve the many-to-many correspondence between ambiguous names and a unique real-world entity, in real-time, using only a stream of messages as data source. This novel technique is implemented using a three-stage pipeline. In the first stage, previously defined filtering rules (colocations, users, hash tags) are used to identify clearly positive examples of messages truly mentioning the real-world entity(ies). These messages are given as input to a Expectation-Maximization method on the second stage, which produces training information to be used during the last stage. Finally, on the last stage we use the training set produced by the previous stage to classify unlabeled messages in real time. We validate this technique by disambiguating tweets that might refer to teams in the Brazilian Soccer League. This technique is used at the project "Observatório da Web"*[1].

1. Introduction

Entity[2] disambiguation is a classical problem in Information Retrieval. For this matter, there are several algorithms to disambiguate polysemous[3] and homonymous[4] entity names in static data sets. However, due to the emergence of new communication technologies, such as micro-blog platforms (i.e., Twitter), we now face the challenge of disambiguating references to entities on a real-time basis. Given that micro-bloggers do not obbey syntax rules, and have a rapidly changing vocabulary, the developement of algorithms that replicate the human ability to disambiguate entity names is a very difficult task. An urgent issue that crucially impacts Information Retrieval and Natural Language Processing applications is to find new ways of treating all those problems using only 140 characters messages (a.k.a. tweets). Use of external sources of information is not possible because they are too expensive to be retrieved in real-time. But, we argue that the lack of that information can be compensated by learning from a large sample of the target corpus. This

[1]www.observatorio.inweb.org.br
[2]An entity is anything that has a distinct, separate (materialized or not) existence.
[3]Multiple names refer to one entity
[4]Same name may refer to different entities depending on the surrounding context

is a reasonable strategy, considering the abundance of text available in the Twitter stream as whole. However, this would require a costly preparation of training data.

Consider a stream of tweets S composed of triples $< e, m, l >$ where e is the real world entity, m is a message that contains at least one keyword that might refer to e (in a potentialy ambiguous way), and $l \in \{\oplus, \ominus\}$ is a binary variable which indicates whether or not the keyword in m refers to the desired real-world entity e. In order to predict the value of l, we could use a classifier. However, as any supervised algorithm, it would require previously defined positive and negative training examples. Although traditional approaches tend to create training examples through human effort, this is not possible in a streamming application-scenario, because most of the relevant training messages have very short lifespam[Silva et al. 2011]. To overcome this problem, an expert picks rules that give high certainty over time whether a message refers to the entity in question, such as colocations, users and hash tags. Messages that fall into those rules are automatically labeled as positive examples, while the remaining messages (unlabeled examples) might still refer to the entity. Note that negative examples are still needed for creating training information to the classifier.

In order to produce the training set (containing positive and negative examples), we used the Expectation-Maximization (EM) method. This algorithm is an iterative two-step procedure which methods employ a classifier that assigns to each example a probability $\alpha(x, \ominus)$ of being negative, and they iterate performing a series of label-transition operations, so that, in the end of the process, it is expected that the assigned labels converge to the combination for which the data is most likely. Typically, a transition threshold α_{min} is employed, so that a label-transition operation $x^{\ominus \rightarrow \oplus}$ is always performed if x is negative and $\alpha(x, \ominus) \leqslant \alpha_{min}$. Similarly, label-transition operation $x^{\oplus \rightarrow \ominus}$ is always performed if x is positive and $\alpha(x, \ominus) > \alpha_{min}$.

As the vocabulary in S keeps changing over time, we need to maintain the training set as fresh as possible. For that we developed a three-stage pipeline. The first stage (data gathering) works with data that is being collected on-the-fly. Given an input S, this stage sorts positive messages according to expert picked rules. All of those messages have their l variable set to positve, while the remaining ones (unlabeled) are directed to the disambiguation stage. Also, a random sample of all messages - positive and unlabeled - is stored to create the input for the EM stage. When EM converges, the new training set is loaded into the classificaton stage. Finally, the disambiguation stage runs a disambiguation server in order to classify the remaning unlabeled messages that have just been collected. The objective of this last stage is to label all remaining messages as positive or negative according to the classifier output. We show a graphical representation of the entire process in Figure 1. This process and EM^2 method are the contributions of this paper. In the following section, we will discuss previous work on entity disambiguation.

2. Related Work

In the context of databases, traditional methods for entity disambiguation rely on using approximate similarity functions over attributes associated with the entities [de Carvalho et al. 2011]. Obviously, such approach is not feasible for the scenario we consider here. Still on databases, [Bhattacharya and Getoor 2007] and [Dong et al. 2005] propose graph-based methods for entity disambiguation that generate clusters of co-referent entities in an unsupervised way using known relationships between entities of several types (e.g., marriage, co-authorship).

In the case of textual corpora, traditional entity disambiguation methods represent entity names and their context, i.e., words, phrases and other names occurring near them, as weighted vectors [Bagga and Baldwin 1998, Pedersen et al. 2005]. To evaluate if two names refer to the same entity, these methods propose computing the similarity between these vectors. Clusters of co-referent names are then built based on such similarity measure in an unsupervised way. Although effective for the tasks considered in these papers, the over-simplified BOW-based approaches they adopt are not suitable for cases in which the context is harder to capture due to the small number of terms available or to informal writing style. This is the case in micro-blog messages, but also in other types of textual corpora. To address these problems, many authors argue that contextual information can be enriched with knowledge from external sources, such as search results and the Wikipedia [Cucerzan 2007, Bunescu and Pasca 2006, Han and Zhao 2009]. While such a strategy is feasible in an off-line setting, two problems arise when monitoring streams of micro-blog messages. First, gathering information from external sources through the Internet can be costly and, second, informal mentions to named entities make it hard to find related information in such sources.

3. Disambiguation Pipeline

In this section, we give details about the three stages that compose the proposed disambiguation pipeline, that can be visualized in Figure 1.

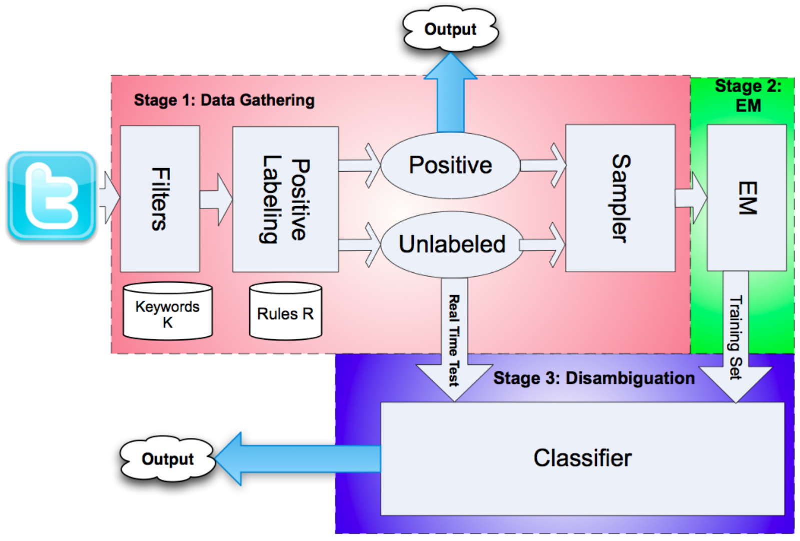

Figure 1. A graphical representation of the proposed pipeline

3.1. Data Gathering (Stage 1)

Given a set of keywords K that might refer to an entity e, we collect all messages containing at least one of these keywords. For instance, if we want messages that make reference to the soccer team “Clube Atlético Mineiro”, we may use keywords such as “galo”, “atlético”, “atletico-mg”, “alvinegro”, that are usually used for referencing this soccer team. We may also use adjectives as keywords, like “atleticano” and “atleticana”.

Once we collected these messages, we need to sort out those that we have high certainty of their positiveness. For that, our expert picks filtering rules R, such as ”o São Paulo”, which can only refer to the soccer team because if a user wants to make a reference to the city or the state he would rather say “em São Paulo”. We may also use the presence of hash tags, like “#galodoido”, or influential users that are related to an entity, such as soccer players and team managers. The definition of K and R is the only supervision required in the entire process. Note that it is required only once, since both sets will not change much over time.

All messages falling into one rule in R have their l variable set to $\oplus$, while the remaining ones are directed to disambiguation stage, so that our disambiguation algorithm determines whether it is positive or not. We also randomly sample all messages to create an input file to the EM stage. We wait until that input file have a previously defined number of tweets to use it on the following stage.

3.2. Expectation-Maximization (Step 2)

At this stage, the pipeline receives a random sample of the messages accumulated on the first stage. Specifically, in this sample, there are positive and unlabeled messages. Initially, all unlabeled messages are arbitrarily set as negative ones. Our goal at this point is to extract as much positives messages as possible from the negative set. By minimizing the number of false negatives, we improve the quality of the training set for the last stage.

We extract positive messages from the negative set using the Expectation-Maximization method. As EM do not support including new examples during the iterative process, this step is the only one which is not executed in real-time. Given a set of positive examples, we use association rules to score the correlation of negative examples with the positive examples (Expectation Step). Once we have that score for all examples, we annotate those which have score above the threshold α_{min} as positives (Maximization Step). These steps are repeated until convergence. Knowing that the quality of the training set produced depends on the α_{min} chosen, we need an efficient algorithm to determine this threshold. For that matter, we propose the EM^2 method (Expectation Maximization using Entropy Minimization) which calculates an individual α^x_{min} for each example x on the input sample.

In order to properly choose that threshold, EM^2 uses demand-driven rule extraction [Veloso et al. 2006, Menezes et al. 2010], which is a recent strategy used to avoid the huge search space for rules, by projecting the training data according to the example being processed, x. Terms in x are used as a filter which configures the training data $\mathcal{D}$ in a way that only rules that are applicable to x can be extracted. This filtering process produces a projected training data, denoted as $\mathcal{D}_x$, which contains only terms that are in x. As shown in [Menezes et al. 2010], the number of rules extracted using this strategy grows polynomially with the size of the vocabulary.

The proposed EM^2 method searches for a threshold α^x_{min} which provides the best entropy cut in the probability space induced by $\mathcal{D}_x$. Specifically, given the examples $\{y_1, y_2, \ldots, y_k\}$ in the projected training data $\mathcal{D}_x$, the EM^2 method first calculates the probability of y_i be a negative example, denoted as $\alpha(y_i, \ominus)$, for all $y_i \in \mathcal{D}_x$. Then, the values for $\alpha(y_i, \ominus)$ are sorted in ascending order. In an ideal case, there is a cut α^x_{min} :

- if $\alpha(x, \ominus) \leq \alpha^x_{min}$, then l=$\oplus$

- if $\alpha(x, \ominus) > \alpha^x_{min}$, then l=$\ominus$

Once α^x_{min} is calculated, it can be used to activate a label-transition operation, that is, if x is a negative example and $\alpha(x, \ominus) \leq \alpha^x_{min}$ then x becomes positive. However, more difficult cases exist, for which it is not possible to obtain a perfect separation in the probability space. Thus, we propose a more general approach to calculate a cut in the probability space. The basic idea is that any value for α^x_{min} induces two partitions over the space of values for $\alpha(x, \ominus)$ (i.e., one partition with values that are lower than α^x_{min}, and another partition with values that are higher than α^x_{min}). The EM^2 method is to set α^x_{min} with the value which minimizes the average entropy of these two partitions.

3.3. Disambiguation (Stage 3)

After loading the training set, the disambiguation stage applies a classifier on each unlabeled message as they come in the stream. We chose associative rule classification [Veloso et al. 2006] as classifier for this stage because it scales for high volumes of data [Menezes et al. 2010] and have good results on micro-blog data [Silva et al. 2011]. This classifier works as a server once it loads all the training set only once, and waits for each instance on real-time.

4. Results and Application

For validation, we employ standard evaluation measures, such as F-score and ROC-curves (AUC). We present the accuracy of EM^2 method on a real-world Brazilian Soccer League entities. This collection is composed of 6 large-scale corpora. Each corpus contains messages in Portuguese mentioning the name/nickname of a Brazilian soccer team. All messages were collected from 11/21/2010 to 12/21/2010. In addition, for each corpus we built a separate labeled dataset, each one containing 1,000 messages that were manually labeled as $\oplus$ or $\ominus$ by three annotators. These collections were used as input in the last stage of the pipeline, and Table 1 shows the disambiguation performance achieved for each collection.

Table 1. Results for the proposed EM^2 method in comparison to the baseline Biased SVM method [Liu et al. 2003], which is representative of the state-of-the-art, on the Brazilian Soccer team Collection

	EM^2				Biased-SVM		
	AUC	F-score	time(s)	#iterations	AUC	F-score	time(s)
Fluminense Football Club	0.76	0.59	4.2	3	0.74	0.57	425.2
Sport Club Internacional	0.78	0.76	3.2	3	0.74	0.70	301.7
Sociedade Esportiva Palmeiras	0.79	0.67	2.4	5	0.74	0.57	253.1
São Paulo Football Club	0.87	0.84	1.3	6	0.84	0.81	122.8
Clube Atlético Mineiro	0.68	0.66	2.0	5	0.62	0.59	181.3
Cruzeiro Esporte Clube	0.80	0.74	2.1	3	0.68	0.65	208.1

The ability of disambiguating entities on data streams is an important first step to deal with this kind of data. The proposed pipeline is effective and is currently being used in the project "Observatório da Web".

5. Conclusion

The method proposed in this work achieved over 70% accuracy on average, which is an impressive result considering that no supervision will be required for a long time after the start because the rules defined manually are reliable over time. Given the dinamicity of the vocabulary in micro-blogs, we consider regenerating the training set manually would be impractical. Even more important than renewing training examples with no aditional supervision, is to develop an efficient and fast algoritm that enables us to disambiguate on real-time.

Acknowledgements

This research was sponsored by UOL (www.uol.com.br) through its UOL Bolsa Pesquisa program (grant number 20110215172500), and partially supported by CNPq, Capes, Finep, Fapemig, and by project INCTWeb (CNPq grant number 573871/2008-6).

References

Bagga, A. and Baldwin, B. (1998). Entity-based cross-document coreferencing using the vector space model. In *ACL*, pages 79–85. Association for Computational Linguistics.

Bhattacharya, I. and Getoor, L. (2007). Collective entity resolution in relational data. *ACM Trans. Knowl. Discov. Data*, 1.

Bunescu, R. and Pasca, M. (2006). Using encyclopedic knowledge for named entity disambiguation. In *ACL*. Association for Computational Linguistics.

Cucerzan, S. (2007). Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, pages 708–716.

de Carvalho, M., Laender, A., Gonçalves, M., and Soares, A. (2011). A genetic programming approach to record deduplication. *IEEE Trans. on Knowledge and Data Engineering*.

Dong, X., Halevy, A., and Madhavan, J. (2005). Reference reconciliation in complex information spaces. In *SIGMOD*, pages 85–96. ACM.

Han, A. and Zhao, J. (2009). Named entity disambiguation by leveraging wikipedia semantic knowledge. In *CIKM*, pages 215–224. ACM.

Liu, B., Dai, Y., Li, X., Lee, W., and Yu, P. (2003). Building text classifiers using positive and unlabeled examples. In *ICDM*, pages 179–188.

Menezes, G., Almeida, J., Belém, F., Gonçalves, M., Lacerda, A., de Moura, E., Pappa, G., Veloso, A., and Ziviani, N. (2010). Demand-driven tag recommendation. In *ECML/PKDD*, pages 402–417.

Pedersen, T., Purandare, A., and Kulkarni, A. (2005). Name discrimination by clustering similar contexts. In *Intl Conf. on Computational Linguistics and Intelligent Text Processing*, pages 226–237.

Silva, I. S., Gomide, J., Veloso, A., Jr., W. M., and Ferreira, R. (2011). Effective sentiment stream analysis with self-augmenting training and demand-driven projection. In *SIGIR*, page to be revealed.

Veloso, A., Jr., W. M., Gonçalves, M. A., de Almeida, H. M., and Zaki, M. J. (2011). Calibrated lazy associative classification. In *Inf. Sci.*, pages 2656–2670.

Veloso, A., Meira Jr., W., and Zaki, M. (2006). Lazy associative classification. In *ICDM*, pages 645–654. IEEE Computer Society.