Named Entity Disambiguation in Streaming Data

Alexandre Davis[1], Adriano Veloso[1], Altigran S. da Silva[2]
Wagner Meira Jr.[1], Alberto H. F. Laender[1]
[1]Computer Science Dept. − Federal University of Minas Gerais
[2]Computer Science Dept. − Federal University of Amazonas
{agdavis,adrianov,meira,laender}@dcc.ufmg.br
alti@dcc.ufam.edu.br

Abstract

The named entity disambiguation task is to resolve the many-to-many correspondence between ambiguous names and the unique real-world entity. This task can be modeled as a classification problem, provided that positive and negative examples are available for learning binary classifiers. High-quality sense-annotated data, however, are hard to be obtained in streaming environments, since the training corpus would have to be constantly updated in order to accomodate the fresh data coming on the stream. On the other hand, few positive examples plus large amounts of unlabeled data may be easily acquired. Producing binary classifiers directly from this data, however, leads to poor disambiguation performance. Thus, we propose to enhance the quality of the classifiers using finer-grained variations of the well-known Expectation-Maximization (EM) algorithm. We conducted a systematic evaluation using Twitter streaming data and the results show that our classifiers are extremely effective, providing improvements ranging from 1% to 20%, when compared to the current state-of-the-art biased SVMs, being more than 120 times faster.

1 Introduction

Human language is not exact. For instance, an entity[1] may be referred by multiple names (i.e., polysemy), and also the same name may refer to different entities depending on the surrounding context (i.e., homonymy). The task of named entity disambiguation is to identify which names refer to the same entity in a textual collection (Sarmento et al., 2009; Yosef et al., 2011; Hoffart et al., 2011). The emergence of new communication technologies, such as micro-blog platforms, brought a humongous amount of textual mentions with ambiguous entity names, raising an urgent need for novel disambiguation approaches and algorithms.

In this paper we address the named entity disambiguation task under a particularly challenging scenario. We are given a stream of messages from a micro-blog channel such as Twitter[2] and a list of names $n_1, n_2, \ldots, n_N$ used for mentioning a specific entity e. Our problem is to monitor the stream and predict whether an incoming message containing n_i indeed refers to e (positive example) or not (negative example). This scenario brings challenges that must be overcome. First, micro-blog messages are composed of a small amount of words and they are written in informal, sometimes cryptic style. These characteristics make hard the identification of entities and the semantics of their relationships (Liu et al., 2011). Further, the scarcity of text in the messages makes it even harder to properly characterize a common context for the entities. Second, as we need to monitor messages that keep coming at a fast pace, we cannot afford to gather information from external sources on-the-fly. Finally, fresh data coming in the stream introduces new patterns, quickly invalidating static disambiguation models.

[1]The term entity refers to anything that has a distinct, separate (materialized or not) existence.

[2]Twitter is one of the fastest-growing micro-blog channels, and an authoritative source for breaking news (Jansen et al., 2009).

We hypothesize that the lack of information in each individual message and from external sources can be compensated by using information obtained from the large and diverse amount of text in a stream of messages taken as a whole, that is, thousands of messages per second coming from distinct sources.

The information embedded in such a stream of messages may be exploited for entity disambiguation through the application of supervised learning methods, for instance, with the application of binary classifiers. Such methods, however, suffer from a data acquisition bottleneck, since they are based on training datasets that are built by skilled human annotators who manually inspect the messages. This annotation process is usually lengthy and laborious, being clearly unfeasible to be adopted in data streaming scenarios. As an alternative to such manual process, a large amount of unlabeled data, augmented with a small amount of (likely) positive examples, can be collected automatically from the message stream (Liu et al., 2003; Denis, 1998; Comité et al., 1999; Letouzey et al., 2000).

Binary classifiers may be learned from such data by considering unlabeled data as negative examples. This strategy, however, leads to classifiers with poor disambiguation performance, due to a potentially large number of false-negative examples. In this paper we propose to refine binary classifiers iteratively, by performing Expectation-Maximization (EM) approaches (Dempster et al., 1977). Basically, a partial classifier is used to evaluate the likelihood of an unlabeled example being a positive example or a negative example, thus automatically and (continuously) creating a labeled training corpus. This process continues iteratively by changing the label of some examples (an operation we call label-transition), so that, after some iterations, the combination of labels is expected to converge to the one for which the observed data is most likely. Based on such an approach, we introduce novel disambiguation algorithms that differ among themselves on the granularity in which the classifier is updated, and on the label-transition operations that are allowed.

An important feature of the proposed approach is that, at each iteration of the EM-process, a new classifier (an improved one) is produced in order to account for the current set of labeled examples. We introduce a novel strategy to maintain the classifiers up-to-date incrementally after each iteration, or even after each label-transition operation. Indeed, we theoretically show that our classifier needs to be updated just partially and we are able to determine exactly which parts must be updated, making our disambiguation methods extremely fast.

To evaluate the effectiveness of the proposed algorithms, we performed a systematic set of experiments using large-scale Twitter data containing messages with ambiguous entity names. In order to validate our claims, disambiguation performance is investigated by varying the proportion of false-negative examples in the unlabeled dataset. Our algorithms are compared against a state-of-the-art technique for named entity disambiguation based on classifiers, providing performance gains ranging from 1% to 20% and being roughly 120 times faster.

2 Related Work

In the context of databases, traditional entity disambiguation methods rely on similarity functions over attributes associated to the entities (de Carvalho et al., 2012). Obviously, such an approach is unfeasible for the scenario we consider here. Still on databases, Bhattacharya and Getoor (2007) and Dong et. al (2005) propose graph-based disambiguation methods that generate clusters of coreferent entities using known relationships between entities of several types. Methods to disambiguate person names in e-mail (Minkov et al., 2006) and Web pages (Bekkerman and McCallum, 2005; Wan et al., 2005) have employed similar ideas. In e-mails, information taken from the header of the messages leads to establish relationships between users and building a co-reference graph. In Web pages, reference information come naturally from links. Such graph-based approach could hardly be applied to the context we consider, in which the implied relationships between entities mentioned in a given micro-blog message are not clearly defined.

In the case of textual corpora, traditional disambiguation methods represent entity names and their context (Hasegawa et al., 2004) (i.e., words, phrases and other names occurring near them) as weighted vectors (Bagga and Baldwin, 1998; Pedersen et al., 2005). To evaluate whether two names refer to the same entity, these methods compute the similarity between these vectors. Clusters of co-referent names are then built based on such similarity measure. Although effective for the tasks considered in these papers, the simplistic 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. To address these problems, some authors argue that contextual information may 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 look for related information in such sources.

The disambiguation methods we propose fall into a learning scenario known as PU (positive and unlabeled) learning (Liu et al., 2003; Denis, 1998; Comité et al., 1999; Letouzey et al., 2000), in which a classifier is built from a set of positive examples plus unlabeled data. Most of the approaches for PU learning, such as the biased-SVM approach (Li and Liu, 2003), are based on extracting negative examples from unlabeled data. We notice that existing approaches for PU learning are not likely to scale given the restrictions imposed by streaming data. Thus, we propose highly incremental approaches, which are able to process large-scale streaming data.

3 Disambiguation in Streaming Data

Consider a stream of messages from a micro-blog channel such as Twitter and let $n_1, n_2, \ldots, n_N$ be names used for mentioning a specific entity e in these messages. Our problem is to continually monitor the stream and predict whether an incoming message containing n_i indeed refers to e or not.

This task may be accomplished through the application of classification techniques. In this case, we are given an input data set called the training corpus (denoted as $\mathcal{D}$) which consists of examples of the form $<e, m, c>$, where e is the entity, m is a message containing the entity name (i.e., any n_i), and $c \in \{\ominus, \oplus\}$ is a binary variable that specifies whether or not the entity name in m refers to the desired real-world entity e. The training corpus is used to produce a classifier that relates textual patterns (i.e., terms and sets of terms) in m to the value of c. The test set (denoted as $\mathcal{T}$) consists of a set of records of the form $<e, m, ?>$, and the classifier is used to indicate which messages in $\mathcal{T}$ refer to (or not) the desired entity.

Supervised classifiers, however, are subject to a data acquisition bottleneck, since the creation of a training corpus requires skilled human annotators to manually inspect the messages. The cost associated with this annotation process may render vast amounts of examples unfeasible. In many cases, however, the acquisition of some positive examples is relatively inexpensive. For instance, as we are dealing with messages collected from micro-blog channels, we may exploit profiles (or hashtags) that are known to be strongly associated with the desired entity. Let us consider, as an illustrative example, the profile associated with a company (i.e., @bayer). Although the entity name is ambiguous, the sense of messages that are posted in this profile is biased towards the entity as being a company. Clearly, other tricks like this one can be used, but, unfortunately, they do not guarantee the absence of false-positives, and they are not complete since the majority of messages mentioning the entity name may appear outside its profile. Thus, the collected examples are not totally reliable, and disambiguation performance would be seriously compromised if classifiers were built upon these uncertain examples directly.

3.1 Expectation-Maximization Approach

In this paper we hypothesize that it is worthwhile to enhance the reliability of unlabeled examples, provided that this type of data is inexpensive and the enhancement effort will be then rewarded with an improvement in disambiguation performance. Thus, we propose a new approach based on the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). We assume two scenarios:

- the training corpus $\mathcal{D}$ is composed of a small set of *truly* positive examples plus a large amount of unlabeled examples.
- the training corpus $\mathcal{D}$ is composed of a small set of *potentially* positive examples plus a large amount of unlabeled examples.

In both scenarios, unlabeled examples are initially treated as negative ones, so that classifiers can be built from $\mathcal{D}$. Therefore, in both scenarios, $\mathcal{D}$ may contain false-negatives. In the second scenario, however, $\mathcal{D}$ may also contain false-positives.

Definition 3.1: *The label-transition operation $x^{\ominus \to \oplus}$ turns a negative example $x^{\ominus} \in \mathcal{D}$ into a positive one $x^{\oplus}$. The training corpus $\mathcal{D}$ becomes $\{(\mathcal{D} - x^{\ominus}) \cup x^{\oplus}\}$. Similarly, the label-transition operation $x^{\oplus \to \ominus}$, turns a positive example $x^{\oplus} \in \mathcal{D}$ into a negative one $x^{\ominus}$. The training corpus $\mathcal{D}$ becomes $\{(\mathcal{D} - x^{\oplus}) \cup x^{\ominus}\}$.*

Our Expectation Maximization (EM) methods employ a classifier which assigns to each example $x \in \mathcal{D}$ a probability $\alpha(x, \ominus)$ of being negative. Then, as illustrated in Algorithm 1, label-transition operations are performed, 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. In the first scenario only operations $x^{\ominus \to \oplus}$ are allowed, while in the second scenario operations $x^{\oplus \to \ominus}$ are also allowed. In both cases, a crucial issue that affects the effectiveness of our EM-based methods concerns the decision of whether or not performing the label-transition operation. Typically, a transition threshold α_{min} is employed, so that a label-transition operation $x^{\ominus \to \oplus}$ is always performed if x is a negative example and $\alpha(x, \ominus) \leq \alpha_{min}$. Similarly, operation $x^{\oplus \to \ominus}$ is always performed if x is a positive example and $\alpha(x, \ominus) > \alpha_{min}$.

Algorithm 1 Expectation-Maximization Approach.

```
Given:
    $\mathcal{D}$: training corpus
    $\mathcal{R}$: a binary classifier learned from $\mathcal{D}$
Expectation step:
    perform transition operations on examples in $\mathcal{D}$
Maximization step:
    update $\mathcal{R}$ and $\alpha(x, \ominus)\ \forall x \in \mathcal{D}$
```

The optimal value for α_{min} is not known in advance. Fortunately, data distribution may provide hints about proper values for α_{min}. In our approach, instead of using a single value for α_{min}, which would be applied to all examples indistinctly, we use a specific α^{x}_{min} threshold for each example $x \in \mathcal{D}$. Based on such an approach, we introduce fine-grained EM-based methods for named entity disambiguation under streaming data. A specific challenge is that the proposed methods perform several transition operations during each EM iteration, and each transition operation may invalidate parts of the current classifier, which must be properly updated. We take into consideration two possible update granularities:

- the classifier is updated after each EM iteration.
- the classifier is updated after each label-transition operation.

Incremental Classifier: As already discussed, the classifier must be constantly updated during the EM process. In this case, well-established classifiers, such as SVMs (Joachims, 2006), have to be learned entirely from scratch, replicating work by large. Thus, we propose as an alternative the use of *Lazy Associative Classifiers* (Veloso et al., 2006).

Definition 3.2: *A classification rule is a specialized association rule $\{\mathcal{X} \to c\}$ (Agrawal et al., 1993), where the antecedent $\mathcal{X}$ is a set of terms (i.e., a termset), and the consequent c indicates if the prediction is positive or negative (i.e., $c \in \{\oplus, \ominus\}$). The domain for $\mathcal{X}$ is the vocabulary of $\mathcal{D}$. The cardinality of rule $\{\mathcal{X} \to c\}$ is given by the number of terms in the antecedent, that is $|\mathcal{X}|$. The support of $\mathcal{X}$ is denoted as $\sigma(\mathcal{X})$, and is the number of examples in $\mathcal{D}$ having $\mathcal{X}$ as a subset. The confidence of rule $\{\mathcal{X} \to c\}$ is denoted as $\theta(\mathcal{X} \to c)$, and is the conditional probability of c given the termset $\mathcal{X}$, that is, $\theta(\mathcal{X} \to c) = \frac{\sigma(\mathcal{X} \cup c)}{\sigma(\mathcal{X})}$.*

In this context, a classifier is denoted as $\mathcal{R}$, and it is composed of a set of rules $\{\mathcal{X} \to c\}$ extracted from $\mathcal{D}$. Specifically, $\mathcal{R}$ is represented as a pool of entries with the form <*key, properties*>, where *key*=$\{\mathcal{X}, c\}$ and *properties*=$\{\sigma(\mathcal{X}), \sigma(\mathcal{X} \cup c), \theta(X \to c)\}$. Each entry in the pool corresponds to a rule, and the key is used to facilitate fast access to rule properties.

Once the classifier $\mathcal{R}$ is extracted from $\mathcal{D}$, rules are collectively used to approximate the likelihood of an arbitrary example being positive ($\oplus$) or negative ($\ominus$). Basically, $\mathcal{R}$ is interpreted as a poll, in which each rule $\{\mathcal{X} \to c\} \in \mathcal{R}$ is a vote given by $\mathcal{X}$ for $\oplus$ or $\ominus$. Given an example x, a rule $\{\mathcal{X} \to c\}$ is only considered a valid vote if it is applicable to x.

Definition 3.3: *A rule $\{\mathcal{X} \rightarrow c\} \in \mathcal{R}$ is said to be applicable to example x if $\mathcal{X} \subseteq x$, that is, if all terms in $\mathcal{X}$ are present in example x.*

We denote as $\mathcal{R}_x$ the set of rules in $\mathcal{R}$ that are applicable to example x. Thus, only and all the rules in $\mathcal{R}_x$ are considered as valid votes when classifying x. Further, we denote as $\mathcal{R}_x^c$ the subset of $\mathcal{R}_x$ containing only rules predicting c. Votes in $\mathcal{R}_x^c$ have different weights, depending on the confidence of the corresponding rules. Weighted votes for c are averaged, giving the score for c with regard to x (Equation 1). Finally, the likelihood of x being a negative example is given by the normalized score (Equation 2).

$$s(x, c) = \sum \frac{\theta(\mathcal{X} \rightarrow c)}{|\mathcal{R}_x^c|}, \text{with } c \in \{\ominus, \oplus\} \quad (1)$$

$$\alpha(x, \ominus) = \frac{s(x, \ominus)}{s(x, \ominus) + s(x, \oplus)} \quad (2)$$

Training Projection and Demand-Driven Rule Extraction: Demand-driven rule extraction (Veloso et al., 2006) is a recent strategy used to avoid the huge search space for rules, by projecting the training corpus according to the example being processed. More specifically, rule extraction is delayed until an example x is given for classification. Then, terms in x are used as a filter that configures the training corpus $\mathcal{D}$ so that just rules that are applicable to x can be extracted. This filtering process produces a projected training corpus, denoted as $\mathcal{D}_x$, which contains only terms that are present in x. As shown in (Silva et al., 2011), the number of rules extracted using this strategy grows polynomially with the size of the vocabulary.

Extending the Classifier Dynamically: With demand-driven rule extraction, the classifier $\mathcal{R}$ is extended dynamically as examples are given for classification. Initially $\mathcal{R}$ is empty; a subset $\mathcal{R}_{x_i}$ is appended to $\mathcal{R}$ every time an example x_i is processed. Thus, after processing a sequence of m examples $\{x_1, x_2, \ldots, x_m\}$, $\mathcal{R} = \{\mathcal{R}_{x_1} \cup \mathcal{R}_{x_2} \cup \ldots \cup \mathcal{R}_{x_m}\}$.

Before extracting rule $\{\mathcal{X} \rightarrow c\}$, it is checked whether this rule is already in $\mathcal{R}$. In this case, while processing an example x, if an entry is found with a key matching $\{\mathcal{X}, c\}$, then the rule in $\mathcal{R}$ is used instead of extracting it from $\mathcal{D}_x$. Otherwise, the rule is extracted from $\mathcal{D}_x$ and then it is inserted into $\mathcal{R}$.

Incremental Updates: Entries in $\mathcal{R}$ may become invalid when $\mathcal{D}$ is modified due to a label-transition operation. Given that $\mathcal{D}$ has been modified, the classifier $\mathcal{R}$ must be updated properly. We propose to maintain $\mathcal{R}$ up-to-date incrementally, so that the updated classifier is exactly the same one that would be obtained by re-constructing it from scratch.

Lemma 3.1: *Operation $x^{\ominus\rightarrow\oplus}$ (or $x^{\oplus\rightarrow\ominus}$) does not change the value of $\sigma(\mathcal{X})$, for any termset $\mathcal{X}$.*

Proof: The operation $x^{\ominus\rightarrow\oplus}$ changes only the label associated with x, but not its terms. Thus, the number of examples in $\mathcal{D}$ having $\mathcal{X}$ as a subset is essentially the same as in $\{(\mathcal{D} - x^{\ominus}) \cup x^{\oplus}$. The same holds for operation $x^{\oplus\rightarrow\ominus}$. ■

Lemma 3.2: *Operation $x^{\ominus\rightarrow\oplus}$ (or $x^{\oplus\rightarrow\ominus}$) changes the value of $\sigma(\mathcal{X} \cup c)$ iff termset $\mathcal{X} \subset x$.*

Proof: For operation $x^{\ominus\rightarrow\oplus}$, if $\mathcal{X} \subset x$, then $\{\mathcal{X} \cup \ominus\}$ appears once less in $\{(\mathcal{D} - x^{\ominus}) \cup x^{\oplus}\}$ than in $\mathcal{D}$. Similarly, $\{\mathcal{X} \cup \oplus\}$ appears once more in $\{(\mathcal{D} - x^{\ominus}) \cup x^{\oplus}\}$ than in $\mathcal{D}$. Clearly, if $\mathcal{X} \not\subset x$, the number of times $\{\mathcal{X} \cup \ominus\}$ (and $\{\mathcal{X} \cup \oplus\}$) appears in $\{(\mathcal{D} - x^{\ominus}) \cup x^{\oplus}\}$ remains the same as in $\mathcal{D}$. The same holds for operation $x^{\oplus\rightarrow\ominus}$. ■

Lemma 3.3: *Operation $x^{\ominus\rightarrow\oplus}$ (or $x^{\oplus\rightarrow\ominus}$) changes the value of $\theta(\mathcal{X} \rightarrow c)$ iff termset $\mathcal{X} \subset x$.*

Proof: Comes directly from Lemmas 3.1 and 3.2. ■

From Lemmas 3.1 to 3.3, the number of rules that have to be updated due to a label-transition operation is bounded by the number of possible termsets in x. The following theorem states exactly the rules in $\mathcal{R}$ that have to be updated due to a transition operation.

Theorem 3.4: *All rules in $\mathcal{R}$ that must be updated due to $x^{\ominus\rightarrow\oplus}$ (or $x^{\oplus\rightarrow\ominus}$) are those in $\mathcal{R}_x$.*

Proof: From Lemma 3.3, all rules $\{\mathcal{X} \rightarrow c\} \in \mathcal{R}$ that have to be updated due to operation $x^{\ominus\rightarrow\oplus}$ (or $x^{\oplus\rightarrow\ominus}$) are those for which $\mathcal{X} \subseteq x$. By definition, $\mathcal{R}_x$ contains only and all such rules. ■

Updating $\theta(\mathcal{X} \rightarrow \ominus)$ and $\theta(\mathcal{X} \rightarrow \oplus)$ is straightforward. For operation $x^{\ominus\rightarrow\oplus}$, it suffices to iterate on $\mathcal{R}_x$, incrementing $\sigma(\mathcal{X} \cup \oplus)$ and decrementing $\sigma(\mathcal{X} \cup \ominus)$. Similarly, for operation $x^{\oplus\rightarrow\ominus}$, it suffices to iterate on $\mathcal{R}_x$, incrementing $\sigma(\mathcal{X} \cup \ominus)$ and decrementing $\sigma(\mathcal{X} \cup \oplus)$. The corresponding values for $\theta(\mathcal{X} \rightarrow \ominus)$ and $\theta(\mathcal{X} \rightarrow \oplus)$ are simply obtained by computing $\frac{\sigma(\mathcal{X} \cup \ominus)}{\sigma(\mathcal{X})}$ and $\frac{\sigma(\mathcal{X} \cup \oplus)}{\sigma(\mathcal{X})}$, respectively.

3.2 Best Entropy Cut Method

In this section we propose a method for finding the activation threshold, α^x_{min}, which is a fundamental step of our Expectation-Maximization approach.

Definition 3.4: *Let $c^y \in \{\ominus, \oplus\}$ be the label associated with an example $y \in \mathcal{D}_x$. Consider $N_\ominus(\mathcal{D}_x)$ the number of examples in $\mathcal{D}_x$ for which c^y=$\ominus$. Similarly, consider $N_\oplus(\mathcal{D}_x)$ the number of examples in $\mathcal{D}_x$ for which c^y=$\oplus$.*

Entropy Minimization: Our method searches for a threshold α^x_{min} that provides the best entropy cut in the probability space induced by $\mathcal{D}_x$. Specifically, given examples $\{y_1, y_2, \ldots, y_k\}$ in $\mathcal{D}_x$, our method first calculates $\alpha(y_i, \ominus)$ 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} such that:

$$c^{y_i} = \begin{cases} \oplus & \text{if } \alpha(y_i, \ominus) \leq \alpha^x_{min} \\ \ominus & \text{otherwise} \end{cases}$$

However, there are more difficult cases, for which it is not possible to obtain a perfect separation in the probability space. Thus, we propose a more general method to find the best 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 $\alpha(y_i, \ominus)$ (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}). Our method sets α^x_{min} to the value that minimizes the average entropy of these two partitions. Once α^x_{min} is calculated, it can be used to activate a label-transition operation. Next we present the basic definitions in order to detail this method.

Definition 3.5: *Consider a list of pairs $\mathcal{O} = \{\ldots, <c^{y_i}, \alpha(y_i, \ominus)>, <c^{y_j}, \alpha(y_j, \ominus)>, \ldots\}$, such that $\alpha(y_i, \ominus) \leq \alpha(y_j, \ominus)$. Also, consider f a candidate value for α^x_{min}. In this case, $\mathcal{O}_f(\leq)$ is a sub-list of $\mathcal{O}$, that is, $\mathcal{O}_f(\leq)$=$\{\ldots, <c^y, \alpha(y_i, \ominus)>, \ldots\}$, such that for all pairs in $\mathcal{O}_f(\leq)$, $\alpha(y, \ominus) \leq f$. Similarly, $\mathcal{O}_f(>)$=$\{\ldots, <c^y, \alpha(y, \ominus)>, \ldots\}$, such that for all pairs in $\mathcal{O}_f(>)$, $\alpha(y, \ominus) > f$. In other words, $\mathcal{O}_f(\leq)$ and $\mathcal{O}_f(>)$ are partitions of $\mathcal{O}$ induced by f.*

Our method works as follows. Firstly, it calculates the entropy in $\mathcal{O}$, as shown in Equation 3. Then, it calculates the sum of the entropies in each partition induced by f, according to Equation 4. Finally, it sets α^x_{min} to the value of f that minimizes $E(\mathcal{O}) - E(\mathcal{O}_f)$, as illustrated in Figure 1.

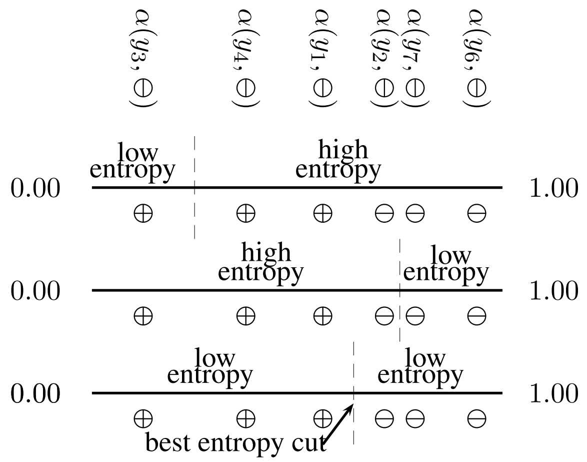

Figure 1: Looking for the minimum entropy cut point.

$$E(\mathcal{O}) = -\left(\frac{\mathrm{N}_\ominus(\mathcal{O})}{|\mathcal{O}|} \times \log \frac{\mathrm{N}_\ominus(\mathcal{O})}{|\mathcal{O}|}\right) - \left(\frac{\mathrm{N}_\oplus(\mathcal{O})}{|\mathcal{O}|} \times \log \frac{\mathrm{N}_\oplus(\mathcal{O})}{|\mathcal{O}|}\right) \quad (3)$$

$$E(\mathcal{O}_f) = \frac{|\mathcal{O}_f(\leq)|}{|\mathcal{O}|} \times E(\mathcal{O}_f(\leq)) + \frac{|\mathcal{O}_f(>)|}{|\mathcal{O}|} \times E(\mathcal{O}_f(>)) \quad (4)$$

3.3 Disambiguation Algorithms

In this section we discuss four algorithms based on our incremental EM approach and following our Best Entropy Cut method. They differ among themselves on the granularity in which the classifier is updated and on the label-transition operations allowed:

- A1: the classifier is updated incrementally after each EM iteration (which may comprise several label-transition operations). Only operation $x^{\ominus\rightarrow\oplus}$ is allowed.
- A2: the classifier is updated incrementally after each EM iteration. Both operations $x^{\ominus\rightarrow\oplus}$ and $x^{\oplus\rightarrow\ominus}$ are allowed.
- A3: the classifier is updated incrementally after each label-transition operation. Only operation $x^{\ominus\rightarrow\oplus}$ is allowed.
- A4: the classifier is updated incrementally after each label-transition operation. Both operations $x^{\ominus\rightarrow\oplus}$ and $x^{\oplus\rightarrow\ominus}$ are allowed.

4 Experimental Evaluation

In this section we analyze our algorithms using standard measures such as AUC values. For each positive+unlabeled (PU) corpus used in our evaluation we randomly selected $x\%$ of the positive examples (P) to become unlabeled ones (U). This procedure enables us to control the uncertainty level of the corpus. For each level we have a different TPR-FPR combination, enabling us to draw ROC curves.We repeated this procedure five times, so that five executions were performed for each uncertainty level. Tables 2–5 show the average for the five runs. Wilcoxon significance tests were performed ($p<0.05$) and best results, including statistical ties, are shown in bold.

4.1 Baselines and Collections

Our baselines include namely SVMs (Joachims, 2006) and Biased SVMs (B-SVM (Liu et al., 2003)). Although the simple SVM algorithm does not adapt itself with unlabeled data, we decided to use it in order to get a sense of the performance achieved by simple baselines (in this case, unlabeled data is simply used as negative examples). The B-SVM algorithm uses a soft-margin SVM as the underlying classifier, which is re-constructed from scratch after each EM iteration. B-SVM employs a single transition threshold α_{min} for the entire corpus, instead of a different threshold α^{x}_{min} for each $x \in \mathcal{D}$. It is representative of the state-of-the-art for learning classifiers from PU data.

We employed two different Twitter collections. The first collection, ORGANIZATIONS, is composed of 10 corpora[3] (O_1 to O_{10}). Each corpus contains messages in English mentioning the name of an organization (Bayer, Renault, among others). All messages were labeled by five annotators. Label $\oplus$ means that the message is associated with the organization, whereas label $\ominus$ means the opposite.

The other collection, SOCCER TEAMS, contains 6 large-scale PU corpora (ST_1 to ST_6), taken from a platform for real time event monitoring (the link to this platform is omitted due to blind review). Each corpus contains messages in Portuguese mentioning the name/mascot of a Brazilian soccer team. Both collections are summarized in Table 1.

[3]http://nlp.uned.es/weps/

Table 1: Characteristics of each collection.

	P	U		P	U
O_1	404	10	ST_1	216,991	251,198
O_2	404	55	ST_2	256,027	504,428
O_3	349	116	ST_3	160,706	509,670
O_4	329	119	ST_4	147,706	633,357
O_5	335	133	ST_5	35,021	168,669
O_6	314	143	ST_6	5,993	351,882
O_7	292	148	–	–	–
O_8	295	172	–	–	–
O_9	273	165	–	–	–
O_{10}	33	425	–	–	–

4.2 Results

All experiments were performed on a Linux PC with an Intel Core 2 Duo 2.20GHz and 4GBytes RAM. Next we discuss the disambiguation performance and the computational efficiency of our algorithms.

ORGANIZATIONS Corpora: Table 2 shows average AUC values for each algorithm. Algorithm A4 was the best performer in all cases, suggesting the benefits of (i) enabling both types of label-transition operations and (ii) keeping the classifier up-to-date after each label-transition operation. Further, algorithm A3 performed better than algorithm A2 in most of the cases, indicating the importance of keeping the classifier always up-to-date. On average A1 provides gains of 4% when compared against B-SVM, while A4 provides gains of more than 20%.

SOCCER TEAMS Corpora: Table 3 shows average AUC values for each algorithm. Again, algorithm A4 was the best performer, providing gains that are up to 13% when compared against the baseline. Also, algorithm A3 performed better than algorithm A2, and the effectiveness of Algorithm A1 is similar to the effectiveness of the baseline.

Since the SOCCER TEAMS collection is composed of large-scale corpora, in addition to high effectiveness, another important issue to be evaluated is computational performance. Table 4 shows the results obtained for the evaluation of our algorithms. As it can be seen, algorithm A1 is the fastest one, since it is the simplest one. Even though being slower than algorithm A1, algorithm A4 runs, on average, 120 times faster than B-SVM.

Table 2: Average AUC values for each algorithm.

	A1	A2	A3	A4	SVM	B-SVM
O_1	0.74 ± 0.02	0.76 ± 0.02	0.74 ± 0.03	**0.79** ± 0.01	0.71 ± 0.03	0.76 ± 0.01
O_2	0.77 ± 0.02	0.78 ± 0.02	0.70 ± 0.03	**0.82** ± 0.02	0.73 ± 0.03	0.75 ± 0.02
O_3	**0.68** ± 0.02	**0.70** ± 0.01	**0.69** ± 0.02	**0.69** ± 0.02	0.64 ± 0.03	0.65 ± 0.02
O_4	0.68 ± 0.02	0.68 ± 0.02	**0.70** ± 0.01	**0.72** ± 0.02	0.63 ± 0.02	0.66 ± 0.02
O_5	**0.71** ± 0.01	**0.72** ± 0.01	**0.71** ± 0.01	**0.72** ± 0.01	0.69 ± 0.01	**0.71** ± 0.01
O_6	0.73 ± 0.01	0.73 ± 0.01	**0.75** ± 0.01	**0.75** ± 0.01	0.68 ± 0.02	0.70 ± 0.01
O_7	0.69 ± 0.01	0.72 ± 0.01	**0.74** ± 0.01	**0.74** ± 0.01	0.66 ± 0.02	0.69 ± 0.02
O_8	0.65 ± 0.02	0.68 ± 0.02	0.69 ± 0.02	**0.72** ± 0.01	0.61 ± 0.03	0.63 ± 0.03
O_9	0.70 ± 0.01	0.70 ± 0.01	**0.72** ± 0.01	**0.72** ± 0.01	0.65 ± 0.01	0.70 ± 0.01
O_{10}	0.70 ± 0.01	**0.74** ± 0.02	0.71 ± 0.02	**0.75** ± 0.02	0.61 ± 0.03	0.66 ± 0.02

Table 3: Average AUC values for each algorithm.

	A1	A2	A3	A4	SVM	B-SVM
ST_1	0.62 ± 0.02	0.63 ± 0.02	0.64 ± 0.01	**0.67** ± 0.02	0.59 ± 0.03	0.61 ± 0.03
ST_2	0.55 ± 0.01	**0.58** ± 0.01	**0.59** ± 0.01	**0.59** ± 0.01	0.54 ± 0.01	0.57 ± 0.01
ST_3	0.65 ± 0.02	0.67 ± 0.01	0.67 ± 0.01	**0.69** ± 0.01	0.61 ± 0.03	0.64 ± 0.03
ST_4	0.57 ± 0.01	**0.59** ± 0.01	**0.59** ± 0.01	**0.59** ± 0.01	0.50 ± 0.04	0.55 ± 0.02
ST_5	0.74 ± 0.01	0.74 ± 0.01	**0.77** ± 0.02	**0.77** ± 0.01	0.67 ± 0.02	0.72 ± 0.03
ST_6	0.68 ± 0.02	0.70 ± 0.01	**0.71** ± 0.01	**0.72** ± 0.01	0.63 ± 0.01	0.68 ± 0.02

Table 4: Average execution time (secs) for each algorithm. The time spent by algorithm A1 is similar to the time spent by algorithm A2. The time spent by algorithm A3 is similar to the time spent by algorithm A4.

	A1(≈A2)	A3(≈ A4)	SVM	B-SVM
ST_1	1,565	2,102	9,172	268,216
ST_2	2,086	2,488	11,284	297,556
ST_3	2,738	3,083	14,917	388,184
ST_4	847	1,199	6,188	139,100
ST_5	1,304	1,604	9,017	192,576
ST_6	1,369	1,658	9,829	196,922

5 Conclusions

In this paper we have introduced a novel EM approach, which employs a highly incremental underlying classifier based on association rules, completely avoiding work replication. Further, two label-transition operations are allowed, enabling the correction of false-negatives and false-positives. We proposed four algorithms based on our EM approach. Our algorithms employ an entropy minimization method, which finds the best transition threshold for each example in $\mathcal{D}$. All these properties make our algorithms appropriate for named entity disambiguation under streaming data scenarios. Our experiments involve Twitter data mentioning ambiguous named entities. These datasets were obtained from real application scenarios and from platforms currently in operation. We have shown that three of our algorithms achieve significantly higher disambiguation performance when compared against a strong baseline (B-SVM), providing gains ranging from 1% to 20%. Also importantly, for large-scale streaming data, our algorithms are more than 120 times faster than the baseline.

6 Acknowledgments

This research is supported by InWeb – The Brazilian National Institute of Science and Technology for the Web (CNPq grant no. 573871/2008-6), by UOL (www.uol.com.br) through its UOL Bolsa Pesquisa program (process number 20110215172500), and by the authors' individual grants from CAPES, CNPq and Fapemig.

References

R. Agrawal, T. Imielinski and A. Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 18th ACM SIGMOD International Conference on Management of Data, Washington, D.C.*, pages 207–216.

A. Bagga and B. Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *Proceedings of the 17th International Conference on Computational Linguistics, Montreal, Canada*, pages 79–85.

R. Bekkerman and A. McCallum. 2005. Disambiguating web appearances of people in a social network. In *Proceedings of the 14th International Conference on the World Wide Web, Chiba, Japan*, pages 463–470.

I. Bhattacharya and L. Getoor. 2007. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1.

R. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceedings of the 11st Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, Trento, Italy*, pages 9–16.

F. De Comité, F. Denis, R. Gilleron and F. Letouzey. 1999. Positive and unlabeled examples help learning. In *Proceedings of the 10th International Conference on Algorithmic Learning Theory, Tokyo, Japan*, pages 219–230.

S. Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 4th Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Prague, Czech Republic*, pages 708–716.

M. G. de Carvalho, A. H. F. Laender, M. A. Gonçalves, and A. S. da Silva. 2006. Learning to deduplicate. *Proceedings of the 6th ACM/IEEE Joint Conference on Digital Libraries, Chapel Hill, NC, USA*. pages 41–50.

A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.

F. Denis. 1998. PAC learning from positive statistical queries. In *Proceedings of the Algorithmic Learning Theory, 9th International Conference, Otzenhausen, Germany*, pages 112–126.

X. Dong, A. Y. Halevy, and J. Madhavan. 2005. Reference reconciliation in complex information spaces. In *Proceedings of the 24th ACM SIGMOD International Conference on Management of Data, Baltimore, USA*, pages 85–96.

X. Han and J. Zhao. 2009. Named entity disambiguation by leveraging wikipedia semantic knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management, Hong Kong, China*, pages 215–224.

T. Hasegawa, S. Sekine and R. Grishman. 2004. Discovering Relations among Named Entities from Large Corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain*, pages 415–422.

J. Hoffart, M. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater and G. Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the 8th Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK*, pages 782–792.

B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury. 2009. Twitter power: Tweets as electronic word of mouth. *JASIST*, 60(11):2169–2188.

T. Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, USA*, pages 217–226.

F. Letouzey, F. Denis, and R. Gilleron. 2000. Learning from positive and unlabeled examples. In *Proceedings of the 11th International Conference on Algorithmic Learning Theory, Sydney, Australia*, pages 71–85.

X. Li and B. Liu. 2003. Learning to classify texts using positive and unlabeled data. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico*, pages 587–592.

B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. 2003. Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd IEEE International Conference on Data Mining, Melbourne, USA*, pages 179–188.

X. Liu, S. Zhang, F. Wei and M. Zhou 2011. Recognizing Named Entities in Tweets. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, Oregon, USA*, pages 359–367.

E. Minkov, W. W. Cohen, and A. Y. Ng. 2006. Contextual search and name disambiguation in email using graphs. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, USA*, pages 27–34.

T. Pedersen, A. Purandare, and A. Kulkarni. 2005. Name discrimination by clustering similar contexts. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing, Mexico City, Mexico*, pages 226–237.

I. S. Silva, J. Gomide, A. Veloso, W. Meira Jr. and R. Ferreira 2011. Effective sentiment stream analysis with self-augmenting training and demand-driven projection. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, Beijing, China*, pages 475–484.

L. Sarmento, A. Kehlenbeck, E. Oliveira, and L. Ungar. 2009. An approach to web-scale named-entity disambiguation. In *Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition, Leipzig, Germany*, pages 689–703.

A. Veloso, W. Meira Jr., M. de Carvalho, B. Pôssas, S. Parthasarathy, and M. J. Zaki. 2002. Mining frequent itemsets in evolving databases. In *Proceedings of the Second SIAM International Conference on Data Mining, Arlington, USA*.

A. Veloso, W. Meira Jr., and M. J. Zaki. 2006. Lazy associative classification. In *Proceedings of the 6th IEEE International Conference on Data Mining, Hong Kong, China*, pages 645–654.

X. Wan, J. Gao, M. Li, and B. Ding. 2005. Person resolution in person search results: Webhawk. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management, Bremen, Germany*, pages 163–170.

M. Yosef, J. Hoffart, I. Bordino, M. Spaniol and G. Weikum 2011. AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables. *PVLDB*, 4(12):1450–1453.