Linking Named Entities to Any Database

Avirup Sil[*]
Temple University
Philadelphia, PA
avi@temple.edu

Ernest Cronin[*]
St. Joseph's University
Philadelphia, PA
ernest.cronin@gmail.com

Penghai Nie
St. Joseph's University
Philadelphia, PA
nph87903@gmail.com

Yinfei Yang
St. Joseph's University
Philadelphia, PA
yangyin7@gmail.com

Ana-Maria Popescu
Yahoo! Labs
Sunnyvale, CA
amp@yahoo-inc.com

Alexander Yates
Temple University
Philadelphia, PA
yates@temple.edu

Abstract

Existing techniques for disambiguating named entities in text mostly focus on Wikipedia as a target catalog of entities. Yet for many types of entities, such as restaurants and cult movies, relational databases exist that contain far more extensive information than Wikipedia. This paper introduces a new task, called Open-Database Named-Entity Disambiguation (Open-DB NED), in which a system must be able to resolve named entities to symbols in an arbitrary database, without requiring labeled data for each new database. We introduce two techniques for Open-DB NED, one based on distant supervision and the other based on domain adaptation. In experiments on two domains, one with poor coverage by Wikipedia and the other with near-perfect coverage, our Open-DB NED strategies outperform a state-of-the-art Wikipedia NED system by over 25% in accuracy.

1 Introduction

Named-entity disambiguation (NED) is the task of linking names mentioned in text with an established catalog of entities (Bunescu and Pasca, 2006; Ratinov et al., 2011). It is a vital first step for semantic understanding of text, such as in grounded semantic parsing (Kwiatkowski et al., 2011), as well as for information retrieval tasks like person name search (Chen and Martin, 2007; Mann and Yarowsky, 2003).

NED requires a catalog of symbols, called referents, to which named-entities will be resolved. Most NED systems today use Wikipedia as the catalog of referents, but exclusive focus on Wikipedia as a target for NED systems has significant drawbacks: despite its breadth, Wikipedia still does not contain all or even most real-world entities mentioned in text. As one example, it has poor coverage of entities that are mostly important in a small geographical region, such as hotels and restaurants, which are widely discussed on the Web. 57% of the named-entities in the Text Analysis Conference's (TAC) 2009 entity linking task refer to an entity that does not appear in Wikipedia (McNamee et al., 2009). Wikipedia is clearly a highly valuable resource, but it should not be thought of as the only one.

Instead of relying solely on Wikipedia, we propose a novel approach to NED, which we refer to as *Open-DB* NED: the task is to resolve an entity to Wikipedia or to *any* relational database that meets mild conditions about the format of the data, described below. Leveraging structured, relational data should allow systems to achieve strong accuracy, as with domain-specific or database-specific NED techniques like Hoffart *et al.*'s NED system for YAGO (Hoffart et al., 2011). And because of the availability of huge numbers of databases on the Web, many for specialized domains, a successful system for this task will cover entities that a Wikipedia NED or database-specific system cannot.

We investigate two complementary learning strategies for Open-DB NED, both of which significantly relax the assumptions of traditional NED systems. The first strategy, a distant supervision approach, uses the relational information in a given database and a large corpus of unlabeled text to learn a database-specific model. The second strategy, a domain adaptation approach, assumes a single source database that has accompanying labeled data. Classifiers in this setting must learn a model that transfers from the source database to any new database, without requiring new training data for the new database. Experiments show that both strategies outperform a state-of-the-art Wikipedia NED system by wide margins without requiring any labeled data from the test domain, highlighting the significant advantage of having domain-specific relational data.

The next section contrasts Open-DB NED with previous work. Section 3 formalizes the task. Sections 4 and 5 present our distant supervision strategy and domain-adaptation strategy, respectively. Section 6 introduces a technique that is a hybrid of the two learning strategies. Section 7 describes our experiments, and Section 8 concludes.

2 Previous Work

As mentioned above, restricting the catalog of referents to Wikipedia, as most recent NED systems do (Bunescu and Pasca, 2006; Mihalcea and Csomai, 2007; Fader et al., 2009; Han and Zhao, 2009; Kulkarni et al., 2009; Ratinov et al., 2011), can restrict the coverage of the system. Zhou *et al.* (2010) estimate that 23% of names in Yahoo! news articles have no referent in Wikipedia, and Cucerzan (2007) estimates the rate at 16% in MSNBC news articles. There is reason to suspect that these estimates are on the low side, however, as news tends to cover popular entities, which are most likely to appear in Wikipedia; the mentions in TAC's 2009 entity linking task are drawn from both newswire and blogs, and have a far higher rate (57%) of missing Wikipedia entries. Lin *et al.* (2012) find that 33% of mentions in a corpus of 500 million Web documents cannot be linked to Wikipedia.

NED systems that are focused on specific domains (or verticals) greatly benefit from repositories of domain-specific knowledge, only a subset of which may be found in Wikipedia. For example, Pantel and Fuxman (2011) use a query-click graph to resolve names in search engine queries to a large product catalog from a commercial search engine, while Dalvi *et al.* (2009; 2012) focus on movie and restaurant databases. Bellare and McCallum (2009) use the sequence information available in citation text to link author, title, and venue names to a publication database. Open-DB NED systems work on any database, so they can serve as baselines for domain-specific NED tasks, as well as provide disambiguation for domains where no domain-specific NED system exists.

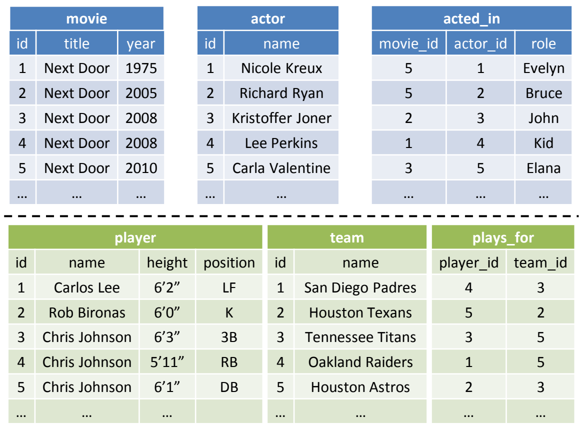
Numerous previous studies have considered distant or weak supervision from a single relational database as an alternative to manual supervision for information extraction (Hoffmann et al., 2011; Weld et al., 2009; Bellare and McCallum, 2007; Bunescu and Mooney, 2007; Mintz et al., 2009; Riedel et al., 2010; Yao et al., 2010). In contrast to these systems, our distant supervision NED system provides a meta-algorithm for generating an NED system for *any* database and any entity type.

Existing domain adaptation or transfer learning approaches are inappropriate for the Open-DB NED task, either because they require labeled data in both the source and target domains (Daumé III et al., 2010; Ben-David et al., 2010), or because they leverage some notion of distributional similarity between words in the source and target domains (Blitzer et al., 2006; Huang and Yates, 2009), which does not apply to the database symbols across the two domains. Instead, our domain adaptation technique uses *domain-independent* features of relational data, which apply regardless of the actual contents of the database, as explained further below.

3 The Open-DB NED Problem and Assumptions

3.1 Problem Formulation

A *mention* is an occurrence of a named-entity in a document. Formally, a mention $m = (d, start, end)$ is a triple consisting of a document d, as well as a start and end position for the mention within the document. We say that d is the *context* of m. A *relational database* is a 2-tuple (S, R). Here, S is a set of symbols for constants, attributes, and relations in the database, and $R = \{r_1, \ldots, r_n\}$ is a set of relation instances of the form $r_i = \{(c_{1,1}, \ldots, c_{1,k_i}), \ldots, (c_{n_i,1}, \ldots, c_{n_i,k_i})\}$, where each c_j is taken from S, k_i is the *arity* of relation r_i and n_i is the number of known instances of r_i. We will write example database symbols in



movie		
id	title	year
1	Next Door	1975
2	Next Door	2005
3	Next Door	2008
4	Next Door	2008
5	Next Door	2010
…	…	…

actor	
id	name
1	Nicole Kreux
2	Richard Ryan
3	Kristoffer Joner
4	Lee Perkins
5	Carla Valentine
…	…

acted_in		
movie_id	actor_id	role
5	1	Evelyn
5	2	Bruce
2	3	John
1	4	Kid
3	5	Elana
…	…	…

player			
id	name	height	position
1	Carlos Lee	6'2"	LF
2	Rob Bironas	6'0"	K
3	Chris Johnson	6'3"	3B
4	Chris Johnson	5'11"	RB
5	Chris Johnson	6'1"	DB
…	…	…	

team	
id	name
1	San Diego Padres
2	Houston Texans
3	Tennessee Titans
4	Oakland Raiders
5	Houston Astros
…	…

plays_for	
player_id	team_id
4	3
5	2
3	5
1	5
2	3
…	…

Figure 1: Example movie database (above) and sports database (below) in BCNF.

`teletype`, and mentions in "quotations." For a particular database DB, we refer to its components as $DB.S$ and $DB.R$. For a set of databases D, define the set of referents as $S_D = (\bigcup_{DB \in D} DB.S) \cup \{OOD\}$, where OOD is a special symbol indicating something that is "out of database", or not found in any of the databases in D.

Given a corpus C, a set of mentions M that occur in C, and a set of databases D, the Open-DB NED task is to produce a function $f : M \rightarrow S_D$, which identifies an appropriate target symbol from one of the databases in D, or determines that the mention is OOD. Note that this problem formulation assumes no labeled data. This is significantly more challenging than traditional NED settings, but allows the system to generalize easily to any new database. In the domain adaptation section below, we relax this condition somewhat, to allow labeled data for a small number of initial databases; the system must then transfer what it learns from the labeled domains to any new database. Also note that the focus for this paper is disambiguation; we assume that the set of mentions are correctly demarcated in the input text. Previous systems, such as Lex (Downey et al., 2007), have investigated the task of finding correct named-entity boundaries in text.

3.2 Assumptions

To allow our systems to handle arbitrary databases, we need to make some assumptions about a standard format for the data. We will assume that databases are provided in a particular form, called *Boyce-Codd Normal Form* (BCNF) (Silberschatz et al., 2010). A relational schema is said to be in BCNF when all redundancy based on functional dependency has been removed, although other types of redundancy may still exist. Formally, a schema R is said to be in BCNF with respect to a set of functional dependencies $\mathcal{F}$ if for every one of the dependencies $(X \rightarrow Y) \in \mathcal{F}$, either

1. $Y \subset X$, meaning this is a trivial functional dependency, or

2. X is a *superkey*, meaning that X is a set of attributes that together define a unique ID for the relation.

In practice, this is a relatively safe assumption as database designers often aim for even stricter normal forms. For databases not in BCNF, such as tables extracted from Web pages, standard algorithms exist for converting them into BCNF, given appropriate functional dependencies, although there are sets of functional dependencies for which BCNF is not achievable. Figure 1 shows two example databases in BCNF. We use these tables as examples throughout the paper.

We will additionally assume that all attributes, including names and nicknames, of entities that are covered by the database are treated as functional dependencies of the entity. Again, in practice, this is a fairly safe assumption as this is part of good database design, but if a database does not conform to this, then there will be some entities in the database that our algorithms cannot resolve to. This assumption implies that it is enough to use the set of superkeys for relations as the set of possible referents; our algorithms make use of this fact.

Finally, we will assume the existence of a function $\mu(s, t)$ which indicates whether the text t is a valid surface form of database symbol s. Our experiments in Section 7.3 explore several possible simple definitions for this function.

4 A Distant Supervision Strategy for Open-DB NED

Our first approach to the Open-DB NED problem relies on the fact that, while many mentions are indeed ambiguous and difficult to resolve correctly, most mentions have only a very small number of possible referents in a given database. "Chris Johnson" is the name of doubtless thousands of people, but for articles that are reasonably well-aligned with our sports database, most of the time the name will refer to just three different people. Most sports names are in fact less ambiguous still. Thus, taking a corpus of unlabeled sports articles, we use the information in the database to provide (uncertain) labels, and then train a log-linear model from this probabilistically-labeled data.

This strategy requires a set of features for the model. Traditionally, such features would be hand-crafted for a particular domain and database. As a first step towards our Open-DB system, we present a log-linear model for disambiguation, as well as a simple feature-generation algorithm that produces a large set of useful features from a BCNF database. We then present a distant-supervision learning procedure for this model.

4.1 Disambiguation Model

Let S_D be the set of possible referents. We construct a vector of feature functions $\mathbf{f}(m, s)$ describing the degree to which m and $s \in S_D$ appear to match one another. The feature functions are described below. The model includes a vector of weights $\mathbf{w}$, one weight per feature function, and sets the probability of entity s given m and $\mathbf{w}$ as:

$$P(s|m, \mathbf{w}) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(m, s))}{\sum_{s' \in S_D} \exp(\mathbf{w} \cdot \mathbf{f}(m, s'))} \quad (1)$$

4.2 Database-driven Feature Generation

Figure 2 shows our algorithm for automatically generating feature functions $f_i(m, s)$ from a BCNF database. As mentioned above, we only need to consider resolving to database symbols s that are keys, or unique IDs, for some tuple in a database. For an entity in the database with key id, the feature generation algorithm generates two types of feature functions: attribute counts and similar entity counts. Each of these features measures the similarity between the information stored in the database about the entity id, and the information in the text in d surrounding mention m.

An *attribute count* feature function $f^{att}_{i,j}(m, id)$ for the jth attribute of relation r_i counts how many attributes of the entity id appear near m. For example, if id is `5` in the `movie` relation in Figure 1, the feature function for attribute `year` would count how often `2010` matches the text surrounding mention m. Defining precisely whether a database symbol "matches" a word or phrase is a subtle issue; we explore several possibilities in Section 7.3. In addition

```
Algorithm: Feature Generation
Input: $DB$, a database in BCNF
Output: $\mathbf{F}$, a set of feature functions
Initialization: $\mathbf{F} \leftarrow \emptyset$
Attribute Count Feature Functions:
For each relation $r_i \in DB.R$
  For each $j$ in $\{1, \ldots, k_i\}$
    Define function $f^{att}_{i,j}(m, id)$:
      $count \leftarrow 0$
      Identify the tuple $t \in r_i$ containing $id$
      $val \leftarrow t_j$
      $count \leftarrow count +$
            $ContextMatches(val, m)$
      return $count$
    $\mathbf{F} \leftarrow \mathbf{F} \cup \{f^{att}_{i,j}\}$
Similar-Entity Count Feature Functions:
For each relation $r_i \in DB.R$
  For each $j$ in $\{1, \ldots, k_i\}$
    Define function $f^{sim}_{i,j}(m, id)$:
      $count \leftarrow 0$
      Identify the tuple $t \in r_i$ containing $id$
      $val \leftarrow t_j$
      Identify the set of similar tuples $T'$:
          $T' = \{t' | t' \in r_i, t'_j = val\}$
      For each tuple $t' \in T'$
        For each $j' \in \{1, \ldots, k_i\}$
          $val' \leftarrow t'_j$
          $count \leftarrow count +$
                $ContextMatches(val', m)$
      return $count$
    $\mathbf{F} \leftarrow \mathbf{F} \cup \{f^{sim}_{i,j}\}$
```

Figure 2: Feature generation algorithm. The $ContextMatches(s, m)$ function counts how many times a string that matches database symbol s appears in the context of m. In our implementation, we use all of $d(m)$ as the context. Matching between strings and database symbols is discussed in Sec. 7.3.

to attribute counts for attributes within a single relation, we also use attributes from relations that have been inner-joined on primary key and foreign key pairs. For example, for movies, we include attributes such as director name, genre, and actor name. High values for these attribute count features indicate that the text around m closely matches the information in the database about entity id, and therefore id is a strong candidate for the referent of m. We use the whole document as the context for finding matches, although other variants are worth future investigation.

A *similar entity count* feature function $f_{i,j}^{sim}(m, id)$ for the jth attribute in relation r_i counts how many entities similar to id are mentioned in the neighborhood of m. As an example, consider a mention of "Chris Johnson", $id = 3$, and the similar entity feature for the `position` attribute of the `players` relation in the sports database. The feature function would first identify that `3B` is the position of the player with $id = 3$. It would then identify all players that had the same position. Finally, it would count how often any attributes of this set of players appear near "Chris Johnson". Likewise, the similar entity feature for the `team id` attribute would count how many teammates of the player with $id = 3$ appear near "Chris Johnson". A high count for this teammate feature is a strong clue that id is the correct referent for m, while a high count for players of the same position is a weak but still valuable clue.

4.3 Parameter Estimation via Distant Supervision

Using string similarity, we can heuristically determine that three IDs with `name` attribute `Chris Johnson` are highly likely to be the correct target for a mention of "Chris Johnson". Our distant supervision parameter estimation strategy is to move as much probability mass as possible onto the set of realistic referents obtained via string similarity. Since our features rely on finding attributes and similar entities, the side effect of this strategy is that most of the probability mass for a particular mention is moved onto the one target ID with high attribute count and similar entity count features, thus disambiguating the entity. Although the string-similarity heuristic is typically noisy, the strong information in the database and the fact that many entity mentions are typically not ambiguous allows the technique to learn effectively from unlabeled text.

Let $\phi(m, DB)$ be a heuristic string-matching function that returns a set of plausible ID values in database DB for mention m. The objective function for this training procedure is a modified marginal log likelihood (MLL) function that encourages probability mass to be placed on the heuristically-matched targets:

$$MLL(M, \mathbf{w}) = \sum_{m \in M} \log \sum_{id \in \phi(m, DB)} P(id|m, \mathbf{w})$$

This objective is smooth but non-convex. We use a gradient-based optimization procedure that finds a local maximum. Our implementation uses an open-source version of the LBFG-S optimization technique (Liu and Nocedal, 1989). The gradient of our objective is given by

$$\frac{\partial LL(M, \mathbf{w})}{\partial w_i} = \sum_{m \in M} \mathbf{E}_{id \in \phi(m, DB)} \left[f_i(m, id) \right] - \mathbf{E}_{id \in DB.S} \left[f_i(m, id) \right]$$

where the expectations are taken according to $P(id|m, \mathbf{w})$.

5 A Domain-Adaptation Strategy for Open-DB NED

Our domain-adaptation strategy builds an Open-DB NED system by training it on labeled examples from an initial database or small set of initial databases. Unlike traditional NED, however, the purpose in Open-DB NED is to resolve to any database. Thus the strategy must take care to build a model that can transfer what it has learned to a new database, without requiring additional labeled data for the new database.

At first, the problem seems intractable — just because a system can disambiguate between "Next Door", the 2005 Norwegian film, and "Next Door", the 1975 short film by director Andrew Silver, that seems to provide little benefit for disambiguating between different athletes named "Andre Smith." The crux of the problem lies in the fact that database-driven features are domain-specific. Counting how many times the director of a movie appears is highly useful in the movie domain, but worthless in the sports domain.

Domain-Independent Feature Functions	
Count Att:	$\sum_{i,j} f^{att}_{i,j}(m,s)$
Count Sim:	$\sum_{i,j} f^{sim}_{i,j}(m,s)$
Count All:	Count Att + Count Sim
Count Unique:	$\sum_{i,j} \begin{cases} 0 & \text{if } f^{att}_{i,j}(m,s) = 0, \\ 1 & \text{if } f^{att}_{i,j}(m,s) > 0. \end{cases}$
Count Num:	$\sum_{i,j \mid j \text{ is a numeric att.}} f^{att}_{i,j}(m,s)$

Table 1: Primary feature functions for a domain adaptation approach to NED. These features made the biggest difference in our experiments, but we also tested variations such as counting unique numeric attribute appearances, counting unique similar entities, counting relation name appearances, counting extended attributed appearances, and others.

Our solution works by re-defining the problem in such a way that we can define domain-independent and database-independent features. For example, rather than counting how often the director of a movie appears in the context around a movie mention, we create a domain-independent Count Att(m, s) feature function that counts how often *any* attribute of s appears in the context of m. For movies, Count Att will add together counts for appearances of a movie's production year and IMDB rating, among other attributes. In the sports domain, Count Att will add together counts for appearances of a player's height, position, salary, *etc.*. But in either domain, the feature is well-defined, and in either domain, larger values of the feature indicate a better match between m and s. Thus there is a hope for training a model with domain-independent features like Count Att on labeled data from one domain, say movies, and producing a model that has high accuracy on the sports domain.

We first formalize the notion of a domain adaptation NED model, and then describe our algorithm for producing such a model. We say that a *domain* consists of a database DB as well as a distribution $\mathcal{D}(\mathcal{M})$, where $\mathcal{M}$ is the space of mentions. For instance, the movie domain might consist of the Internet Movie Database (IMDB) and a distribution that places most probability mass on documents about movies and Hollywood stars. In *domain adaptation*, a system observes a set of training examples $(m, s, g(m, s))$, where instances $m \in \mathcal{M}$ are drawn from a *source* domain's distribution $\mathcal{D}_S$ and referents s are drawn from the source domain's database DB_S. The labels $g(m, s)$ are boolean values indicating a correct or incorrect match between the mention and referent. The system must then learn a hypothesis for classifying examples (m, s) drawn from a *target* domain's distribution $\mathcal{D}_T$ and database DB_T. Note that for domain adaptation, we cannot use the more traditional problem formulation in which the referent s is a label (*i.e.*, $s = g(m)$) for the mention, since the set of possible referents changes from domain to domain, and therefore the output of g would be completely different from one domain to the next.

Table 1 lists the domain-independent features used in our domain adaptation model. These features use the attribute counts and similar entity counts from the distant supervision model as subroutines. By aggregating over those domain-dependent feature functions, the domain adaptation system arrives at feature functions that can be defined for *any* database, rather than for a specific database.

Note that there is a tradeoff between the domain adaptation technique and the distant supervision technique. The domain adaptation model has access to labeled data, unlike the distant supervision model. In addition, the domain adaptation model requires no text whatsoever from the target domain, not even an unlabeled corpus, to set weights for the target domain. Once trained, it is ready for NED over any database that meets our assumptions, out of the box. However, because the model needs to be able to transfer to arbitrary new domains, the domain adaptation model is restricted to domain-independent features, which are "coarser-grained." That is, the distant supervision model has the ability to place more weight on attributes like director rather than genre, or team rather than position, if those attributes are more discriminative. The domain adaptation model cannot place different weights on the different attributes, since those weights would not transfer across databases.

As with distant supervision, the domain adaptation strategy uses a log-linear model over these feature functions. We use standard techniques for training the model using labeled data from the source domain: conditional log likelihood (CLL) as the objective function, and LBFG-S for convex optimization.

$$CLL(L, \mathbf{w}) = \sum_{(m,id,label) \in L} \log P(label|m, id, \mathbf{w})$$

The training algorithm is guaranteed to converge to the globally optimal parameter setting for this objective function over the training data. The manually annotated data contains only positive examples; to generate negative examples, we use the same name-matching heuristic $\phi(m, DB)$ to identify a set of potentially confusing bad matches. On test data, we use the trained model to choose the id for a given m with the highest probability of being correct.

6 A Hybrid Model

The distant supervision and domain adaptation strategies use two very different sources of evidence for training a disambiguation classifier: the string-matching heuristic and unlabeled text from the target domain for the the distant supervision model, and aggregate features over labeled text from a separate domain for domain adaptation. This begs the question, do these sources of evidence complement one another? To address this question, we design a Hybrid model with features and training strategies from both distant supervision and domain adaptation.

The training data consists of a set L_S of labeled mentions from a source domain, a source database DB_S, a set of unlabeled mentions M_T from the target domain, and the target-domain database DB_T. The full feature set of the Hybrid model is the union of the distant supervision feature functions for the target domain and the domain-independent domain adaptation feature functions. Note that the distant supervision feature functions are domain-specific, so they almost always will be uniformly zero on L_S, but the domain adaptation feature functions will be activated on both L_S and M_T. The combined training objective for the Hybrid model is:

$$LL(L_S, M_T, \mathbf{w}) = CLL(L_S, \mathbf{w}) + MLL(M_T, \mathbf{w})$$

7 Experiments

Our experiments compare our strategies for Open-DB NED against one another, as well as against a Wikipedia NED system from previous work, on two domains: sports and movies.

7.1 Data

For the movie domain, we collected a set of 156 cult movie titles from an online movie site (www.olivefilms.com). For each movie title, we executed a Web search using a commercial search engine, and collected the top five documents for each title from the search engine's results. Nearly all top-five results included at least one mention of an entity not found in Wikipedia; overall, only 16% of the mentions could be linked to Wikipedia. After stripping javascript and html annotations, we removed documents with fewer than 50 words, leaving a total of 770 documents. We select one occurrence of any of the 156 movie titles from each document as our set of mentions. Many titles are ambiguous not just among different movies with the same name, but also among novels, plays, geographical entities, and assorted other types of entities. To provide labels for these mentions, we use both a movie database and Wikipedia. We downloaded the complete data dump from the online Internet Movie Database (IMDB, www.imdb.com). For our set of possible referents, we use the set of all key values in IMDB, and the set of all Wikipedia articles. Annotators manually labeled each mention using this set of referents. Table 2 shows summary statistics about this labeled data.

For the sports domain, we downloaded all player data from Yahoo!, Inc.'s sports database for the years 2011-2012 and two American sports leagues, the National Football League (NFL) and Major League Baseball (MLB). From the database, we extracted ambiguous player names and team names, including names like "Philadelphia" which may refer to `Philadelphia Eagles` in the NFL data, `Philadelphia Phillies` in the MLB data, or the city of `Philadelphia` itself (in both types of data). We then collected 1300 Yahoo! news articles which include a mention that partially matches at least one of these database symbols. We manually labeled a random sample of 564 mentions from this data, including 279 player name mentions and 285 city name mentions. Many player name and place name mentions are ambiguous between the two sports leagues, as well as with teams or players from other leagues. In order to focus on the hardest cases, we specifically exclude mentions like "Philadelphia" from the labeled data if any of their

domain	$\|M\|$	$\mathbf{E}\|\phi(m, DB)\|$	OOD	Wiki
movies	770	2.6	13%	16%
sports	549	4.5	0%	100%

Table 2: Number of mentions, average number of referents per mention, % of mentions that are OOD, and % of mentions that are in Wikipedia in our movie and sports data.

System	Accuracy
No-Wikipedia Domain Adapt.	0.61
DocSim-Wikipedia Domain Adapt.	**0.69**

Table 3: Including a simple document-similarity feature for comparing a mention's context with a Wikipedia page provides an 8% improvement over ignoring Wikipedia information.

unambiguous completions appears in the same article (that is, if either of the team names "Philadelphia Eagles" or "Philadelphia Phillies" appears in the same article, we exclude the "Philadelphia" mention). As before, the set of possible referents includes the symbol OOD, key values from the sports database, and Wikipedia articles, and a given mention may be labeled with both a sports entity and a Wikipedia article, if appropriate. All of our data is available from the last author's website.

7.2 Evaluation Metric

We report on a version of exact-match accuracy. The system chooses the most likely label $\hat{s}$ for each m. This is judged correct if $\hat{s}$ matches the correct label s exactly, or (in cases where both a Wikipedia and a database entity are considered correct) if one of the labels matches $\hat{s}$ exactly. This metric allows systems to resolve against either reference, Wikipedia or another database, without requiring it to match both if the same entity appears in both references.

7.3 Exact or Partial Matching?

One important question in the design of our systems is how to determine the "match" between database symbols and text. This question comes into play in two components of our systems: it affects the computation of feature functions that count how often a match of some attribute is found in text, and it affects which set of heuristically-determined database entities are considered to be possible matches for a given mention.

We experiment with two different matching strategies between a symbol s and text t, exact matching and partial matching. Exact matching $\mu_{exact}(s, t)$ requires the sequence of characters in s to appear exactly (modulo character encoding) in t. For instance, the database value `Chris Johnson` would match "Chris Johnson", but not "C. Johnson" or "Johnson" in text. For partial matching, we used different tests for numeric and textual entities. For numeric entities, $\mu_{partial}$ matched s and t if the numeric value of one was within 10% of the other, so that `5312` would match "5,000." We made no attempt to convert numeric phrases, such as "3.6 million", into numeric values. For textual entities, $\mu_{partial}$ matched s and t if at least one token from each matched exactly. Thus `Chris Johnson` matches both "Chris" and "C. Johnson".

We found $\mu_{partial}$ to be consistently superior for computing $\phi(m, DB)$, since it has much better recall for mentions like "Philadelphia". On the other hand, if we use $\mu_{partial}$ for computing our models' feature functions, like the Count Att(m, s) in the domain adaptation model, counts varied widely across domains. A simple version of the domain adaptation classifier (only the Count All and Count Unique features) trained on sports data and tested on movies achieved an accuracy of 24% using $\mu_{partial}$, compared with 61% using μ_{exact}. For all remaining tests, we used μ_{exact} for computing features, and $\mu_{partial}$ for computing $\phi(m, DB)$.

7.4 Incorporating Wikipedia referents

Thus far, all of our features work on relational data, not Wikipedia. In order to allow our systems to link to Wikipedia, we create a single "document similarity" feature describing the similarity between the text around a mention and the text appearing on a Wikipedia page. We build a vector space model of both the document containing the mention and the Wikipedia page, remove stopwords, and use cosine similarity to compute this feature.

To evaluate the effectiveness of this Wikipedia feature, we tested two versions of our domain adaptation system, both trained on sports data and tested

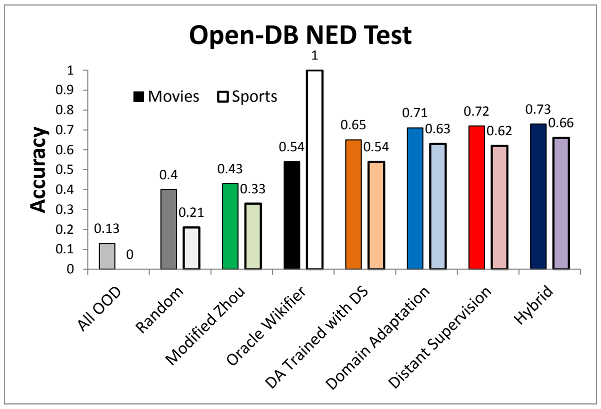


Figure 3: **All three Open-DB NED strategies outperform a state-of-the-art Wikipedia NED system by 25% or more on sports and movies, and outperform a Wikipedia NED system with oracle information by 14% or more on the movie data.** Differences between the Modified Zhou Wikifier and the Open-DB strategies are statistically significant ($p < 0.01$, Fisher's exact test) on both domains.

on the movies domain. The first version involves no Wikipedia information whatsoever, thus it has no reason to select a Wikipedia article over OOD. The second system includes the document similarity feature. Table 3 shows the results of these systems. Encouragingly, our single document similarity feature produces a significant improvement over the model without Wikipedia information, so we use this feature in all of our systems tested below. More sophisticated use of Wikipedia is certainly possible, and an important question for future work is how to combine Open-DB NED more seamlessly with Wikipedia NED.

7.5 Comparing Open-DB NED Strategies

For each domain, we compare our domain-adaptation strategy, distant supervision, and hybrid strategies. The domain-adaptation model is trained on the labeled data for sports when testing on movies, and *vice versa*. We use a movies test set of 180 mentions that is separate from the development data used for the above tests. For the distant supervision strategy, we use the entire collection of texts from each domain as input (1300 articles for sports, 770 articles for movies), with the labels removed during training.

We compare against a state-of-the-art Wikipedia NED system used in production by a major Web company. This system is a modified version of the system described by Zhou et al. (2010), where certain features have been removed for efficiency. We refer to this as the Modified-Zhou Wikifier. This system uses a gradient-boosted decision tree and multiple local and global features for computing the similarity between a mention's context and a Wikipedia article. We also test a hypothetical system, Oracle Wikifier, which is given no information about entities in IMDB, but is assumed to be able to correctly resolve any mention that refers to an entity found in Wikipedia. Thus, this system has perfect accuracy on mentions that can be found in Wikipedia, and accuracy similar to a baseline that predicts randomly on all mentions that fall outside of Wikipedia[1]. Oracle-Wikifier serves as an upper bound on systems that have no access to a domain-specific database. In addition, we compare against two standard baselines: a classifier that always predicts OOD, and a classifier that chooses randomly. Finally, we compare against a system that trains the domain adaptation model using distant supervision ("DA Trained with DS").

Figure 3 shows our results. All three Open-DB approaches outperform the baseline techniques on this test by wide margins, with the Hybrid model increasing by 30% or more over the random baseline. On the movie domain, the Hybrid model outperforms the Oracle Wikifier by nearly 20%. Encouragingly, the Hybrid model consistently outperforms both distant supervision and domain adaptation, suggesting that the two sources of evidence are partially complementary. Distant supervision performs better on the movies test, whereas domain adaptation has the advantage on sports. The differences among all three Open-DB approaches is relatively small, compared with the difference between these approaches and Oracle Wikifier on the movie data.

The domain adaptation system outperforms DA Trained with DS on both domains, suggesting that labeled data from a separate domain is better evidence for parameter estimates than unlabeled data from the same domain. The distant supervision system also outperforms DA Trained with

[1]Alternatively, one could make the oracle system predict OOD on all mentions that fall outside of Wikipedia. Random predictions perform better on our data.

DS on both domains, suggesting that the fine-grained, domain-specific features do in fact provide more helpful information than the coarser-grained, domain-independent features of the domain adaptation model.

All of the Open-DB NED systems outperform the Modified Zhou Wikifier on both data sets by a wide margin. In fact the Modified Zhou Wikifier has similar results on both domains, despite the fact that Wikipedia has far greater coverage on sports than movies. In part, the poor performance of the Modified Zhou Wikifier reflects the difficult nature of the task. In previous experiments on an MSNBC news test set it reached 85% accuracy, but a random classifier there achieved 60% accuracy compared with 21% on our sports data. Another difficulty with the Modified Zhou Wikifier is its strong preference for globally common entities. It consistently classifies mentions that are ambiguous between a city and a team (like "Chicago" in "Chicago sweeps the Red Sox") as cities when they should be resolved to teams, in large part because `Chicago` is a more common referent in general text than either of the baseball teams that play in that city. In sports articles, however, both meanings are common, and only the surrounding context can help determine the correct referent.

Besides wikifiers, NED systems may also be compared with dictionary-based word sense disambiguation techniques like the Lesk algorithm[2] (Lesk, 1986). The Lesk algorithm is "open" in the sense that it works for arbitrary dictionaries, and it defines a vector space model of the dictionary definitions that may be likened to the attribute-value model in our representation of entities in the database. Our approach, however, estimates parameters for a statistical model from data, whereas the Lesk algorithm uses an equal weight for all attributes. To make an empirical comparison, we created a variant of the Lesk algorithm for relational data: we took the disambiguation model from Eqn. 1, supplied all of the features from the distant supervision model, and manually set $\mathbf{w} = \mathbf{1}$. This "relational Lesk" model achieves an accuracy of 0.11 on movies, and 0.15 on sports, significantly below the random baseline. Giving equal weight to noisy attributes like `genre` and more discriminative attributes like `director` significantly hurts the performance.

For both the movie and sports domain, approximately 80% of the Hybrid model's errors are because of predicting database symbols, when the correct referent is a Wikipedia page or OOD. This nearly always occurs because some words in the context of a mention match an attribute of an incorrect database referent. For instance, the `crime` genre is an attribute for several movies, but it also matches in contexts surrounding book titles and numerous other entities. In the movie domain, most of the remaining errors are incorrect OOD predictions for mentions that should resolve to the database, but the article contains no attributes or similar entities to the database entity. In the sports domain, many of the remaining errors were due to predicting incorrect player referents. Quite often, this was because the document discusses a fantasy sports league or team, where players from different professional sports teams are mixed together on a "fantasy team" belonging to a fan of the sport. Since players in the fantasy leagues have different teammates than they do in the database, these articles consistently confuse our methods.

8 Conclusion and Future Work

This paper introduces the task of Open-DB Named Entity Disambiguation, and presents two distinct strategies for solving this task. Experiments indicate that a mixture of the two strategies significantly outperforms a state-of-the-art Wikipedia NED system, on a dataset where Wikipedia has good coverage and on another dataset where Wikipedia has poor coverage. The results indicate that there is a significant benefit to leveraging other sources of knowledge in addition to Wikipedia, and that it is possible to leverage this knowledge without requiring labeled data for each new source. The initial success of these Open-DB NED approaches indicates that this task is a promising area for future research, including exciting extensions that link large numbers of domain-specific databases to text.

Acknowledgments

This work was supported in part by a gift from Yahoo!, Inc.

[2]We thank the reviewers for making this connection.

References

Kedar Bellare and Andrew McCallum. 2007. Learning extractors from unlabeled text using relevant databases. In *Sixth International Workshop on Information Integration on the Web*.

Kedar Bellare and Andrew McCallum. 2009. Generalized Expectation Criteria for Bootstrapping Extractors using Record-Text Alignment. In *Empirical Methods in Natural Language Processing (EMNLP-09)*.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79:151–175.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*.

R. Bunescu and M. Pasca. 2006. 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)*.

Ying Chen and James Martin. 2007. Towards Robust Unsupervised Personal Name Disambiguation. In *EMNLP*, pages 190–198.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716.

Nilesh N. Dalvi, Ravi Kumar, Bo Pang, and Andrew Tomkins. 2009. Matching Reviews to Objects using a Language Model. In *EMNLP*, pages 609–618.

Nilesh N. Dalvi, Ravi Kumar, and Bo Pang. 2012. Object matching in tweets with spatial models. In *WSDM*, pages 43–52.

Hal Daumé III, Abhishek Kumar, and Avishek Saha. 2010. Frustratingly easy semi-supervised domain adaptation. In *Proceedings of the ACL Workshop on Domain Adaptation (DANLP)*.

D. Downey, M. Broadhead, and O. Etzioni. 2007. Locating complex named entities in web text. In *Procs. of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2009. Scaling wikipedia-based named entity disambiguation to arbitrary web text. In *Proceedings of the WikiAI 09 - IJCAI Workshop: User Contributed Knowledge and Artificial Intelligence: An Evolving Synergy*.

Xianpei Han and Jun Zhao. 2009. Named entity disambiguation by leveraging Wikipedia semantic knowledge. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 215–224.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Furstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum1. 2011. Robust Disambiguation of Named Entities in Text. In *EMNLP*, pages 782–792.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-Based Weak Supervision for Information Extraction of Overlapping Relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Fei Huang and Alexander Yates. 2009. Distributional representations for handling sparsity in supervised sequence labeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 457–466.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical Generalization in CCG Grammar Induction for Semantic Parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

M.E. Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the SIGDOC Conference*.

Thomas Lin, Mausam, and Oren Etzioni. 2012. Entity linking at web scale. In *Knowledge Extraction Workshop (AKBC-WEKEX), 2012*.

D.C. Liu and J. Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

G.S. Mann and D. Yarowsky. 2003. Unsupervised personal name disambiguation. In *CoNLL*.

Paul McNamee, Mark Dredze, Adam Gerber, Nikesh Garera, Tim Finin, James Mayfield, Christine Piatko, Delip Rao, David Yarowsky, and Markus Dreyer. 2009. HLTCOE Approaches to Knowledge Base Population at TAC 2009. In *Text Analysis Conference*.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In

Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management (CIKM), pages 233–242.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-2009)*, pages 1003–1011.

Patrick Pantel and Ariel Fuxman. 2011. Jigs and Lures: Associating Web Queries with Structured Entities. In *ACL*.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the Sixteenth European Conference on Machine Learning (ECML-2010)*, pages 148–163.

Avi Silberschatz, Henry F. Korth, and S. Sudarshan. 2010. *Database System Concepts*. McGraw-Hill, sixth edition.

Daniel S. Weld, Raphael Hoffmann, and Fei Wu. 2009. Using Wikipedia to Bootstrap Open Information Extraction. In *ACM SIGMOD Record*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP-2010)*, pages 1013–1023.

Yiping Zhou, Lan Nie, Omid Rouhani-Kalleh, Flavian Vasile, and Scott Gaffney. 2010. Resolving surface forms to wikipedia topics. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling)*, pages 1335–1343.