CompanyDepot: Employer Name Normalization in the Online Recruitment Industry

Qiaoling Liu, Faizan Javed, and Matt Mcnair
CareerBuilder LLC
5550-A Peachtree Parkway Suite 200, Norcross, GA 30092
{qiaoling.liu, faizan.javed, matt.mcnair}@careerbuilder.com

ABSTRACT

Entity linking links entity mentions in text to the corresponding entities in a knowledge base (KB) and has many applications in both open domain and specific domains. For example, in the recruitment domain, linking employer names in job postings or resumes to entities in an employer KB is very important to many business applications. In this paper, we focus on this employer name normalization task, which has several unique challenges: handling employer names from both job postings and resumes, leveraging the corresponding location context, and handling name variations, irrelevant input data, and noises in the KB. We present a system called CompanyDepot which contains a machine learning based approach CompanyDepot-ML and a heuristic approach CompanyDepot-H to address these challenges in three steps: (1) searching for candidate entities based on a customized search engine for the KB; (2) ranking the candidate entities using learning-to-rank methods or heuristics; and (3) validating the top-ranked entity via binary classification or heuristics. While CompanyDepot-ML shows better extendability and flexibility, CompanyDepot-H serves as a strong baseline and useful way to collect training data for CompanyDepot-ML. The proposed system achieves 2.5%-21.4% higher coverage at the same precision level compared to an existing system used at CareerBuilder over multiple real-world datasets. Applying the system to a similar task of academic institution name normalization further shows the generalization ability of the method.

Keywords

employer name normalization; entity linking; named entity disambiguation; learning to rank

1. INTRODUCTION

Entity linking links entity mentions in text to the corresponding entities in a knowledge base (KB) and has many applications such as information extraction and content analysis [22], in both open domain and specific domains. For example, in the recruitment domain, linking the company name on a user's professional profile position to a company entity is very important to many business applications [24].

The mission of CareerBuilder[1], a company providing human capital solutions globally, is to empower employment. Its online career site contains over 45 million candidate resumes and 1.6 million jobs. One of its products is Supply & Demand which performs big data analytics on the supply and demand of jobs. For example, on one hand it computes the demand (i.e., the number of open positions) for specific jobs and the top employers with such demand; on the other hand, it also computes the supply (i.e., the number of candidates) for specific jobs and the top places such supply come from. Such analytics are achievable by parsing and analyzing a large number of job postings and candidate resumes related to specific jobs. To enable accurate computation, besides a KB of employers [15] we need a system that can do employer name normalization, i.e., linking the employer names in the job postings and the resumes to the employer entities in the KB. In this paper, we focus on this employer name normalization task, which is illustrated in Figure 1.

Some key challenges of the task are summarized below based on the applications at CareerBuilder. An effective employer name normalization system should be able to:

1. Handle employer names from both job postings and resumes, which are in semi-structured format and could come from different sources. Therefore, the employer name fields could contain varying degrees of noises.

2. Leverage the location context. Jobs are often associated with locations, and so are the employer names in job postings and resumes. The location information associated with an employer name could be useful for normalization, but it could be empty or inaccurate.

3. Handle name variations. An employer entity can have legacy names, nicknames, acronyms (e.g., "Beaver College" vs. "Arcadia University", "Gatech" vs. "Georgia Institute of Technology", "IBM" vs. "International Business Machines"). An employer name could also be wrongly typed (e.g., "macy's" vs. "macys", "wells fargo" vs. "wellsfargo").

4. Handle irrelevant or unlinkable input data. There are two cases: (a) the input does not mean any employer, e.g., "Not specified", "self-employed"; (b) the input means an employer, but it does not refer to any known entity in the KB.

[1]http://www.careerbuilder.com/

KDD '16, August 13-17, 2016, San Francisco, CA, USA

 ISBN 978-1-4503-4232-2/16/08…$15.00

DOI: http://dx.doi.org/10.1145/2939672.2939727

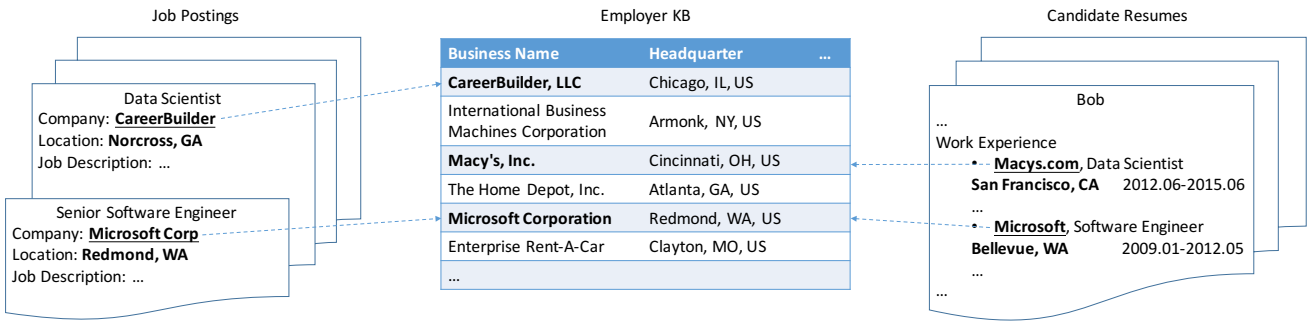

Figure 1: Illustration of employer name normalization in jobs and resumes

5. Handle noises in the KB. The employer KB used in this paper is derived from a database which may contain duplicate records referring to the same employer. For example, it contains different records named "Enterprise Rent A Car", "Enterprise Rentacar", and "Enterprise Rent-A-Car Company" respectively.

Some of these challenges (3 and 4) also exist in the typical entity linking task [8, 22]. Yet, other challenges (1, 2, and 5) are rather unique in the recruitment domain. This paper describes a system called CompanyDepot which aims to address all these challenges. The system takes an employer name and its associated location from job postings or resumes as input and normalizes it into entities in an employer KB. We develop two approaches, one based on machine learning techniques and the other based on heuristics, which address the proposed task in three steps: entity retrieval, reranking, and validation. The retrieval step searches for candidate entities based on a customized search engine for the KB, the reranking step ranks the candidate entities based on learning-to-rank methods or heuristics, and the validation step validates the top-ranked entity based on binary classification or heuristics. We experiment with the proposed system over multiple real-world datasets and compare its performance with an existing employer name normalization system used at CareerBuilder. Our system achieves 2.5%-21.4% higher coverage at the same precision level. We also apply it to a similar task of academic institution name normalization and compare its performance with a state-of-the-art method. The experimental results show the effectiveness, robustness, and generalization ability of our system. Its fast query response time allows the system to be applied to online employer name normalization.

The major contributions of this paper are as follows:

- Proposing a novel employer name normalization task.
- Presenting a system called CompanyDepot with a machine learning based approach and a heuristic approach.
- Comparing its performance on multiple real-world datasets with the existing systems used at CareerBuilder.

The rest of the paper is organized as follows. Section 2 discusses the related work. Next, we provide the problem definition in Section 3 and some preliminaries in Section 4. Then Section 5 describes the system and Section 6 details the experiments. Finally, we conclude the paper in Section 7.

2. RELATED WORK

2.1 Entity Linking with a Knowledge Base

Entity linking [22], also called named entity disambiguation (NED) or named entity normalization (NEN), which links entity mentions in text to the corresponding entities in a KB, has attracted much research effort since the availability of large KBs such as Wikipedia[2] and Freebase [2]. It is a key step to understand and annotate the raw and noisy data in many applications.

A comprehensive survey of the issues and methods about entity linking is provided in [22]. As it summarized, a typical entity linking system has three modules: candidate entity generation, candidate entity ranking, and unlinkable mention prediction. To generate candidate entities, many methods have been proposed, e.g., building a table of mappings from surface forms to the possible entities [26, 28], extracted expanded forms for entity mentions from the local document [25, 27, 28], and finding relevant entity pages via search engines (such as Google API [8] and Wikipedia search engine [26]). To rank the candidate entities, many systems used supervised ranking methods and the most popular two are binary classification [24, 26, 27] and learning to rank [8, 21, 25, 27, 28]. To predict unlinkable mentions, many methods used binary classification to decide whether to output the top-ranked entity or NIL (Not-In-Lexicon) as the final result [21, 25, 27, 28], while some methods integrated NIL prediction into the learning-to-rank process [8].

The employer name normalization task proposed in this paper can be viewed as a general entity linking problem, yet it differs from the traditional entity linking task [22] in three aspects: (1) different data sources: entity linking takes an entity mention recognized in text as input while our task takes an employer name parsed from a semi-structured job posting or resume as input; (2) different contexts: in entity linking the document where an entity mention appears serves as the context, while in our task the location associated with an employer name extracted from the job posting or the resume is used as the context; (3) different KBs: entity linking often focuses on a global KB and multiple types of entities while our task is concerned with a domain-specific KB and a single type of entities.

Because of these differences, the employer name normalization task has unique challenges such as handling the location context and noises in the KB. Our system adapts the three-module framework used in the entity linking systems.

[2]https://www.wikipedia.org/

First, we build our own search engine customized for the KB, which enables us to efficiently retrieve relevant entities by tuning the suitable search algorithms. Then we use learning-to-rank methods to rank the candidate entities and a binary classifier to validate the top-ranked entity. Compared to the entity linking systems, our system has a novel candidate entity generation process. We also design novel features (such as query complexity and location matching) for learning based entity ranking and validation. Moreover, we develop a heuristic approach which serves as a practical and strong baseline for the machine learning based approach.

2.2 Domain-Specific Name Normalization

Our work is also related to a set of domain-specific name normalization applications. For example, within the same recruitment domain, Yan et al. described how to normalize the company name on a LinkedIn member's profile position using social graphs based on binary classification [24]. Although their application is within the same domain as ours, a few differences exist: (1) LinkedIn only considers input from member profiles while we need to normalize company names in both job postings and resumes; (2) LinkedIn has the whole profile as the context which makes the important social features derived from member-to-member links available, while we only have the location corresponding to an employer name as the context; (3) LinkedIn has a Typeahead Assist which allows users to select the correct company entity from a suggested list, thus automatically reducing input noises and creating training data, which is unfortunately not available in our application and many other applications.

The problem of academic institution name normalization discussed by Jacob et al. [11] is very similar to ours. Instead of an employer KB used in our work, they used a KB of academic institutions. Their method, named sCooL, consists of two steps: retrieval and reranking. The retrieval step is also based on building a search engine for the KB. However, sCooL treated each mapping from a surface form to an entity as a document, while our system treats each entity as a document, which leads to very different indexing structures and search processes. The reranking step of sCooL is based purely on heuristics, while we develop a machine learning based approach besides the heuristic approach.

NEMO [13] addressed a related task of extracting and normalizing organization names from PubMed articles. The method first uses multi-layered rule matching to extract entity mentions and then leverages unsupervised clustering to identify entity mentions referring to the same entity.

Besides organization name normalization, there are also many other domain-specific name normalization applications, e.g., product item name normalization [3], gene name normalization [23], disease name normalization [16], and person name normalization [18]. The main differences of these applications with ours lie in different data sources, contexts, and KBs, which often bring some different challenges.

2.3 Deduplicating Domain-Specific Knowledge Bases

The task of employer name normalization depends on an employer KB, and a key step in building such domain-specific KBs is de-duplication [15]. Kardes et al. proposed graph-based blocking and clustering strategies for organization entity resolution [14]. McNeill et al. proposed a dynamic blocking method to efficiently deduplicate around 5 billion people records [19]. The book by Christen summarizes more methods for general duplicate detection [6].

This paper focuses on the task of employer name normalization, for which we performed simple deduplication of the records in an employer database based on their business names. More complex deduplication will be the future work.

3. PROBLEM DEFINITION

Our task is to link the employer names in job postings or resumes to entities in an employer KB, as illustrated in Figure 1. We next describe the problem more formally. The entities in the employer KB are denoted by $E = \{e_1, e_2, ..., e_k\}$. The employer names and the associated location contexts extracted from job postings and resumes[3] are denoted by $Q = \{q_1, q_2, ..., q_c\}$, where $q_i = (n_i, l_i)$ is a pair of employer name and the associated location. In the rest of the paper, we call n_i the query name and l_i the query location. We represent a query location as a triple: $l_i = (City_i, State_i, Country_i)$, where the city, state, country information could be empty. The problem of employer name normalization can then be summarized as inferring a mapping function $f(q_i) \Rightarrow e_j$, where $q_i \in Q$ and $e_j \in E \cup \{NIL\}$. Note that NIL (Not-In-Lexicon) means the input query does not refer to any known entity in the KB.

As discussed in Section 2, our task can be viewed as a general entity linking problem, but is different from the traditional entity linking task for documents [22] and other domain-specific name normalization tasks [24] in terms of different data sources, contexts, and KBs.

4. PRELIMINARIES

4.1 Employer Knowledge Base

The employer KB we used in this paper is based on a third-party employer database, which contains about 21 million employer records. Each record represents an employer with a set of attributes like business name, location, industry code, and company size. Branches of a company are represented as different records, often with the same business name. The database has a good coverage of employers in US, however, it is noisy and may contain duplicate records for the same employer entity. For example, it contains different records named "Enterprise Rent A Car", "Enterprise Rentacar", and "Enterprise Rent-A-Car Company" respectively.

To make an employer KB for the employer name normalization system, we performed simple deduplication of the above employer database. We merged all the records with the same business name into a single entity (by assuming that two records with the same business name refer to the same employer entity) and kept a list of all the original locations. We also computed the number of the original records merged into an entity as the entity popularity. This resulted in a set of around 18 million employer entities that serves as our final employer KB. Note that the noises still exist in the KB, which needs to be considered in the employer name normalization system.

Note that in this KB we treat the business name of an entity as its normalized form and there are no surface forms

[3]The extraction of the employer names and the associated locations is done by in-house job parsers and resume parsers at CareerBuilder, which is out of the scope of this paper.

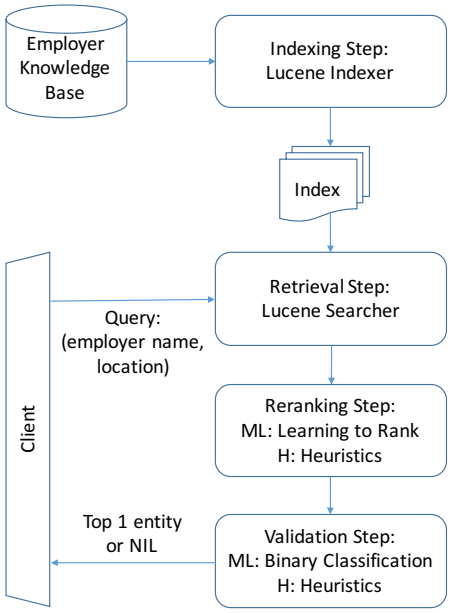

Figure 2: Architecture of the CompanyDepot system.

associated with it. Yet, the proposed system can also make use of the available surface forms in other KBs [11, 15].

4.2 Lucene Search Engine

Our system uses Apache Lucene[4] to index and retrieve the employer entities respectively. Lucene is a high-performance text search engine library. A document in Lucene is a set of fields. Each field has a name and a textual value, with semantics about how it is parsed (e.g., tokenized vs. untokenized, stopwords kept vs. removed). Lucene scoring first works on fields and then combines the results to return documents, allowing giving different weights to different fields.

Lucene supports a wide variety of query implementations, which can be combined to provide complex querying capabilities. For example, it supports the following searches:

- Keyword search which finds documents containing any of the query terms.
- Fuzzy search which finds documents containing terms similar in spelling to a query term.
- Phrase search which finds documents containing all the query terms occurring in the same order as in the query (the number of other terms permitted between terms in the query phrase can be controlled).
- Aggregated search which finds documents satisfying any of the individual searches.

All the above searches are used in our system to retrieve relevant employer entities for a given query.

5. THE COMPANYDEPOT SYSTEM

The architecture of our system is shown in Figure 2. Before taking any queries, the system needs to index the employer KB using a Lucene indexer. Once the index is ready, the system can take normalization requests. Each request consists of an employer name and its location context (part of or whole location information could be empty). The system then uses a Lucene searcher to retrieve a list of N employer entities. We then send these N candidate entities to the reranking step, which first generates a feature vector for each entity and then uses either a machine learning based ranking model or some feature-based heuristics to rank them. Finally, the top-ranked entity is sent to the validation step to decide whether it is a correct result for the query using either a binary classifier or some heuristics. If it says yes, we output this entity to the user; otherwise, we output NIL. We will next describe each step in more detail.

[4]https://lucene.apache.org/

Field	Value
id	15
normalized_form	International Business Machines Corporation
surface_forms[5]	International Business Machines Corporation; IBM; International Business Machines; ...
calibrated_name	internationalbusinessmachines
json	{"id": "15", "normalized_form": "International Business Machines Corporation", "surface_forms": [...], ...}

Table 1: An example document created for an employer entity.

5.1 Indexing Step

The purpose of this step is to index all the entities in the employer KB so that they can be efficiently retrieved in the next step. For each entity, we created a document which contains five fields: id, normalized form, surface forms, calibrated name, and json. An example document is shown in Figure 1. The id field is used to locate an entity. The fields of normalized form and surface forms are used to match the query name in the retrieval step. Distinguishing these two fields allows us to give higher weights to matches in the normalized field. The calibrated name field is used to enable better fuzzy match, derived by compact calibration of the normalized form (details in Section 5.1.1). The json field is a stored field, which is used to get all the detailed information about an entity, e.g., locations and popularity, for computing features used for the machine learning techniques.

Then all these documents are sent to the Lucene indexer to build an index, which is then used by the Lucene searcher in the retrieval step.

5.1.1 Calibration of Employer Names

Proper parsing and pre-processing of text is an important step in building effective search engines [7]. Inspired by this, we propose to calibrate entity names and query names to make them more comparable. The calibration of an employer name works as follows:

1. Convert the name to lowercase, and replace "'s" with "s" (e.g., "Macy's" -> "macys").
2. Convert all the non-alphanumeric characters to space.

[5]We always include the normalized form of an entity as one of its surface forms.

Employer name	After calibration	After compact calibration
International Business Machines Corporation	international business machines	internationalbusinessmachines
Sherman & Howard L.L.C.	sherman howard	shermanhoward
Oxnard Police Dept	oxnard police department	oxnardpolicedepartment
Macy's, Inc.	macys	macys

Table 2: Examples of calibration and compact calibration of employer names.

3. Remove stop-phrases (e.g., "pvt ltd" and "l l c") and stop-words (e.g., "inc", "corporation", "incorporated", and "the").

4. Expand commonly used abbreviations, e.g., "ctr" -> "center", "svc" -> "services", "dept" -> "department".

5. (Optionally) remove all spaces in the name.

Table 2 shows some calibration examples. Considering that a query name is often short while the official name of an entity could be tedious, comparing two calibrated forms often enables more accurate match detection than comparing the two original names.

5.2 Retrieval Step

The purpose of this step is to efficiently generate a pool of N candidate entities so that the correct result is included. To generate this pool, we first use Lucene's powerful querying capabilities to retrieve a large set of entities that are possibly relevant to the query name, and then apply several filters to this entity set and keep only the most likely correct results in the pool.

Specifically, we first search for the query name against all the entity documents via an aggregated search in Lucene which combines (1) keyword searches in the normalized form and surface forms fields; (2) fuzzy searches in the surface forms and calibrated name fields; and (3) phrase searches in the surface forms field. After obtaining top N_0 (e.g., N_0=1000) entities from the Lucene searcher, we then generate the pool of candidate entities as follows:

- From the N_0 results, add to the pool the top N_1 (e.g., N_1=10) entities that have the highest Lucene score.

- From the N_0 results, use Levenshtein Distance to compute the top N_2 (e.g., N_2=2) results whose surface forms have the minimum distance with the query name, and add them to the pool.

- From the N_0 results, use Levenshtein Distance to compute the top N_3 (e.g., N_3=2) results whose compactly calibrated surface forms have the minimum distance with the compactly calibrated query name, and add them to the pool.

- From the N_0 results, add to the pool the N_4 entities obtained by querying a legacy mapping table (details in Section 6.3) which contains mappings from surface forms to the possible linked entities.

The resulted pool contains $N = N_1 + N_2 + N_3 + N_4$ candidate entities, which will be reranked in the next step. Note that the query location is not used in this step because we believe it is more helpful in ranking rather than generating candidate entities.

5.3 Reranking Step: Learning to Rank

The purpose of this step is to effectively rerank the N candidate entities obtained in the previous step so that the correct result is ranked top.

We use supervised learning-to-rank methods to automatically build the ranking model based on training data [17]. A key to the success of such methods is the features used, so next we describe the features used in our system.

5.3.1 Feature Generation

For each (query, candidate entity) pair, we generate a vector of features, which can be grouped into three categories:

- **Query features** indicate the complexity of finding a correct result for this query, which include the length of the query name, whether it indicates irrelevant input, and whether the query location is specified.

- **Query-entity features** indicate the likelihood that the entity is a correct result for the query, which include the score of the Lucene searcher, string comparison of the query name and the normalized form (or the surface forms) of the entity, matching between the query location and the location list of the entity, whether there is a mapping from the query name to the entity in the legacy mapping table.

- **Entity features** indicate the prior knowledge about the entity being a correct result for some query, which include the entity popularity, the number of locations of the entity, whether the normalized form of the entity contains a legal word such as "inc" or "llc".

A full list of the features is shown in Table 3. In Section 6, we will do feature ablation tests to see how each feature group contributes to the learning performance.

5.3.2 Learning Algorithm

Based on the above features, our system uses a listwise learning-to-rank method, coordinate ascent [20], to rerank the candidate employer entities. Coordinate ascent can directly optimize any user specified ranking measure, by updating one parameter at a time while holding other parameters fixed. It has been shown to provide superior performance compared to some other learning-to-rank models [4]. Specifically, we use the coordinate ascent implementation provided in the RankLib[6] library for our experiments.

Before training and testing, we normalize each feature by its mean and standard deviation. Since only the top-ranked entity matters in our task, we choose P@1 to optimize on the training data. To increase the chances of finding a global solution, we use 10 random restarts. For other parameters, the default values provided in RankLib are used.

[6]http://sourceforge.net/p/lemur/wiki/RankLib/

Query features (8 total)
• length of query name in characters. • length of query name in words. • whether query name contains specific phrases or words (e.g., "self employed", "freelancer"). • whether query location (city, state, country) is specified.
Query-entity features (32 total)
• score by Lucene searcher. • whether query name equals entity normalized form by ignoring case. • whether query name equals entity normalized form by considering only alphanumeric characters. • whether query name equals any entity surface form by ignoring case. • whether (compactly) calibrated query name is prefix (suffix) of (compactly) calibrated entity normalized form, or vice versa. • number (percentage) of common words between calibrated query name and calibrated entity normalized name. • Levenshtein (and Jaro-Winkler) distance between (compactly) (calibrated) query name and (compactly) (calibrated) entity normalized form (surface forms). • Jaccard similarity between the two 4-gram sets of (compactly) (calibrated) query name and (compactly) (calibrated) entity normalized form (surface forms). • whether query location (city, state, country) matches entity location (city, state, country). • whether a mapping from query name to entity exists in the legacy mapping table.
Entity features (5 total)
• how many surface forms this entity has. • entity popularity. • length of entity normalized form in characters. • whether entity normalized form contains a legal word like "inc" or "llc". • how many locations this entity has.

Table 3: Features used for learning to rank.

The output of the reranking step is a ranked list of the N candidate entities. Yet, only the top-ranked candidate entity will be sent to the next validation step.

5.4 Validation Step: Binary Classification

The purpose of this step is to validate the top-ranked result so that either a correct result or NIL is sent to the user.

We use a binary classifier to do the validation. The features used for this classification include the features derived in Section 5.3.1 as well as the score output of the learning-to-rank method. We used LibSVM [5] as our binary classifier. In the future, we could add more features indicating NIL such as how similar the top N candidate entities are.

5.5 Heuristic Method

The supervised learning methods above cannot be applied without training data. For the employer name normalization task discussed in this paper, there is no off-the-shelf ground-truth data available. Therefore, we also develop a heuristic method within the system framework, which does not require training data.

The heuristic method shares the same indexing and retrieval steps as the machine learning based approach while having a different process in the reranking and validation steps. In the reranking step, the heuristic method ranks the candidate entities based on the following three feature values (which are also listed in Table 3):

- s_1: Jaccard similarity between the 4-gram sets of the compactly calibrated query name and the compactly calibrated entity normalized form.
- s_2: Entity popularity.
- s_3: Jaccard similarity between the 4-gram sets of the calibrated query name and the calibrated entity normalized form.

In the validation step, the heuristic method simply outputs the top-ranked entity and outputs s_1 as its score.

As will be shown in Section 6, this heuristic method is used to collect manual labels as our training data for the machine learning based approach and serves as a strong baseline compared to other methods.

5.6 Discussion

The Lucene search engine is also used in sCooL [11] to retrieve the candidate entities for academic institution name normalization. However, the index structure in our system is different from the one used in sCooL. While sCooL treats each mapping from a surface form to a normalized form as a document, our system treats each entity as a document. This leads to several benefits. First, fewer documents need to be indexed, which often results in more efficient retrieval. Second, the documents returned by the Lucene searcher directly correspond to a set of entities, which does not need an extra step to compute a list of entities from the returned mappings. In addition, more information of an entity (e.g., the normalized form together with all the surface forms) can be used for computing the matching between the entity and the query in the retrieval process.

Ranking candidate entities based on learning to rank and validating the top-ranked entity based on binary classification have been shown to be effective choices in previous research [21, 25, 27, 28]. We derived similar features (such as name string comparison and entity popularity) that have been shown to be effective, as well as novel features (such as query complexity and location matching) designed for the employer name normalization task. For learning to rank, we choose the coordinate ascent algorithm [20] which can directly optimize the P@1 metric we care most in our task, instead of the more popularly used ranking SVM algorithm [12] in previous work [21, 25, 27, 28]. Our preliminary experiments suggest that coordinate ascent gives better performance than ranking SVM for our task.

6. EXPERIMENTS

In this section, we conduct experiments to answer the following questions: (1) How do the proposed methods compare to the existing employer name normalization systems used at CareerBuilder? (2) How do the proposed methods compare to the state-of-the-art methods on similar tasks? (3) What are the effects of different feature groups on the performance of the machine learning based approach?

6.1 Test Suite

6.1.1 Job and Resume Datasets

To evaluate the performance of the proposed employer name normalization system CompanyDepot, we sampled two resume datasets and two job datasets from real applications at CareerBuilder. The two resume datasets (RDB, EDGE)

Dataset	#Queries	%Country	%US	%State
RDB	1098	58.5%	96.4%	50.9%
EDGE	1093	97.3%	45.3%	20.8%
JOB1	1100	100%	100%	99.7%
JOB2	500	100%	98.4%	100%

Table 4: Statistics about the job and resume datasets (%Country means the percentage of queries with country specified, %US means the percentage of queries with country=US when country is specified, %State means the percentage of queries with state specified).

are sampled from two resume databases respectively. The two job datasets are sampled from a database of job postings based on a legacy normalization system: JOB1 (JOB2) was sampled from cases for which the legacy system returned some (no) result at some time. The datasets were sampled using weighted random sampling techniques [9] so that queries that are more frequent are more likely to be chosen. We believe this sampling strategy would enable more realistic evaluation, better reflecting the overall impact on user experience. The statistics of the datasets are shown in Table 4. We can see that different datasets have different characteristics in terms of the percentage of queries with location specified and the percentage of international queries.

After the query sets were ready, we generated results using three systems: CompanyDepot-H, Legacy, and WService, which will be described in Section 6.3 in more detail. Each system generated a single result or NIL for each query. Each result has information such as business name, location, company size, and industry.

We then asked a group of raters at CareerBuilder to judge the correctness of the results. We showed each (query, result) pair in a random order to the rater, and asked the rater to choose from the following scoring options:

- 3 = Result name correct, location is headquarter.
- 2 = Result name correct, location is not headquarter.
- 1 = Result name acceptable although not perfect.
- 0 = Result name wrong.
- -1 = Query ambiguous (hard to understand the need).

Based on the above scoring options, we consider a result as a correct result if it gets score>=1.

6.1.2 Academic Institution Datasets

To understand the generalization ability of CompanyDepot, we apply it to a similar task of academic institution name normalization [11]. We used the same five datasets that are used in the previous work:

- 4icu-uk: a list of 145 popular universities and colleges in UK from the website of 4ICU [1].
- guardian: a list of 135 universities in UK from the website of TheGuardian [10].
- g1-6: randomly sampled 780 academic institution names that occurred 1-6 times in a resume database for UK.
- g7-39: randomly sampled 736 academic institution names that occurred 7-39 times in a resume database for UK.
- g40-max: randomly sampled 653 academic institution names that occurred 40 or more times in a resume database for UK.

.

6.2 Metrics

To evaluate the performance of different employer name normalization systems, we used the same precision and coverage metrics as used in [11, 24]. For a set of queries, suppose a normalization system returns I_c correct results, I_w wrong results, and I_n null results (i.e., NIL), then the following metrics are computed:

- Precision is the percentage of correct results out of all non-null results generated by the system.

$$Precision = I_c/(I_c + I_w)$$

- Coverage[7] is the percentage of queries that the system returns a non-null result.

$$Coverage = (I_c + I_w)/(I_c + I_w + I_n)$$

Besides the above two metrics, we also add the following two metrics that serve as a combination of precision and coverage to show the overall performance of the system:

$$SuccessRate = Precision * Coverage$$

$$F1 = 2 * Precision * Coverage/(Precision + Coverage)$$

The success rate of a system indicates how likely the system succeeds to generate a correct result, which is important for many applications at CareerBuilder.

6.3 Systems in Comparison

The following systems are compared in the experiments:

- **CompanyDepot-ML**: the machine learning based approach within CompanyDepot that is described in Section 5. For each dataset, we run a model trained on another dataset to generate its results, which are then evaluated as the performance on this dataset. Detailed assignment is shown in Table 5.
- **CompanyDepot-H**: the heuristic method within CompanyDepot that is described in Section 5.5.
- **Legacy**: a system used at CareerBuilder for employer name normalization. It is based on a mapping table which contains a large number of mappings from surface forms to their linked entities, generated via some black-box algorithms with manual corrections.
- **WService**: a third-party web service that supports employer name normalization, with two special characteristics: (1) It requires an employer name and a specified country to return any result; (2) It uses its

[7]Note that coverage here is not the same as recall, because the computation of recall requires that all the true answers of a query (which are very hard to obtain) are known, while the computation of coverage only checks whether a non-null result is returned by the system.

Train	Test
EDGE	RDB
RDB	EDGE
RDB	JOB1
RDB	JOB2

Table 5: Training and test datasets for CompanyDepot-ML.

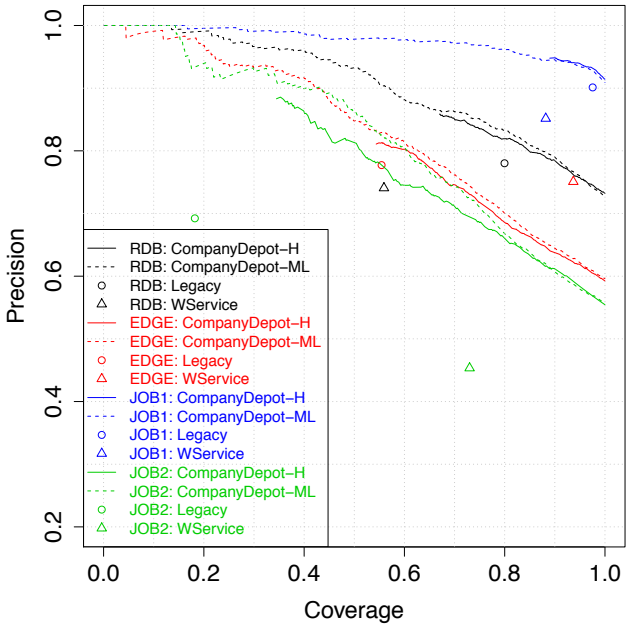

Figure 3: Performance comparison on the job and resume datasets.

own KB, which is a superset of the KB used by other systems, with a much better coverage of international employers.

- **sCooL** [11]: a system used at CareerBuilder for academic institution name normalization.

6.4 Results

6.4.1 *Results on Job and Resume Datasets*

We compared the results of four methods, CompanyDepot-ML, CompanyDepot-H, Legacy, and WService, on the job and resume datasets. For the two CompanyDepot methods (CompanyDepot-H and CompanyDepot-ML), the output contains a confidence score along with the result. By varying the threshold on the confidence score, we can plot a precision-coverage curve. For Legacy and WService, however, the confidence score is not available, so we can only derive a single precision and coverage value.

Figure 3 shows the precision-coverage values of the four methods. We can see that CompanyDepot-ML has the best performance over three datasets (RDB, JOB1, and JOB2), while WService performs best on the EDGE dataset. Moreover, CompanyDepot-ML achieves 2.5%, 14.2%, and 21.4% higher coverage at the same precision level compared to Legacy over JOB1, RDB, and EDGE respectively.

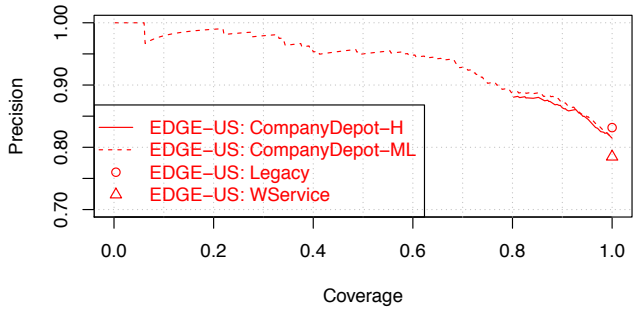

Figure 4: Performance comparison on the EDGE-US dataset (which contains all the US queries in the EDGE dataset).

As shown in Table 4, the EDGE dataset contains a larger percentage of international queries (smaller percentage of US queries, 45.3%) compared to the other datasets. Recall that compared to the KB used by WService, the KB used by CompanyDepot-ML, CompanyDepot-H, and Legacy has a rather poor coverage on international employers. This could be the main reason why these three systems have worse performance than WService on the EDGE dataset. To verify this conjecture, we derived a subset of EDGE called EDGE-US which contains only its US queries. On this subset, we observed that these three systems have better performance than WService, as shown in Figure 4.

The RDB dataset contains a larger percentage of queries with empty country information (smaller percentage of queries with country, 58.5%). Recall that WService requires a country parameter to return any result. This is why WService has a low coverage (55.9%) compared to other systems.

The JOB1 dataset contains the cleanest queries, all about US employers that proactively published job postings. Therefore, all the four systems show the best performance on this dataset compared to on other datasets.

In contrast, the JOB2 dataset contains a set of very hard queries. We can see that both Legacy and WService have very bad performance. Legacy has a low coverage (18%) because the dataset was sampled from the cases that Legacy failed to normalize at some time (which may have been corrected). WService has a low precision (46%), mostly because it puts too much weight on location match as we found many wrong results returned by WService have the same location as the query location (note that all the queries in this dataset have the state specified).

By comparing the two CompanyDepot methods closely, we can see that CompanyDepot-ML performs better on three datasets: RDB, EDGE, and JOB2. While CompanyDepot-H cannot achieve a precision of 90% on these three datasets, CompanyDepot-ML can achieve a 90% precision at a 40% or higher coverage. The main reason is that CompanyDepot-H mostly depends on a single feature while CompanyDepot-ML is able to use all the available features for entity ranking and validation. As shown in Section 5.5, the final score of CompanyDepot-H is the Jaccard similarity between the 4-gram sets of the compactly calibrated versions of the query name and the entity normalized form. However, even with perfect similarity (i.e., score=1), a result can be wrong. In contrast, CompanyDepot-ML can learn these situations based on many other features, e.g., the popularity of the en-

Dataset	Method	Cov	Pre	SR	F1
4icu-uk	sCooL	95.2	92.8	88.3	94.0
	CompanyDepot-H	100	96.6	**96.6**	**98.3**
guardian	sCooL	93.3	92.8	86.6	93.0
	CompanyDepot-H	100	91.8	**91.8**	**95.7**
s40-max	sCooL	99.1	91.7	90.9	95.3
	CompanyDepot-H	100	96.1	**96.1**	**98.0**
s7-39	sCooL	99.3	96.2	95.5	97.7
	CompanyDepot-H	100	97.5	**97.5**	**98.7**
s1-6	sCooL	97.7	94.0	91.8	95.8
	CompanyDepot-H	100	93.5	**93.5**	**96.6**

Table 6: Performance comparison on the academic institution datasets (Cov means coverage; Pre means precision; SR means success rate).

tity or whether a mapping from the query name to the entity exists in the mapping table. This shows that the machine learning based approach is quite promising. It is also easier to extend and improve, because more features can be easily added and automatically integrated, while the optimization of the heuristic approach depends on pure manual effort.

6.4.2 *Results on Academic Institution Datasets*

We compared the results of two systems, CompanyDepot-H and sCooL, on the academic institution datasets. The two systems used the same KB derived in [11] for normalization and the same classifier to detect K-12 schools. A new set of stop-words (e.g., "college", "university", "school") is used in CompanyDepot-H for calibration of academic institution names.

Table 6 shows the performance of these two systems. We can see that CompanyDepot-H achieves better success rate and F1 score over all the five datasets. Specifically, it achieves comparable or higher precisions at better coverages. Note that CompanyDepot-ML was not deployed here as our error analysis revealed that the main room to improve is the quality of the underlying KB.

6.4.3 *Effects of Feature Groups*

We would like to understand how each of the following feature groups contributes to the learning performance: (1) query features; (2) query-entity features; and (3) entity features. Figure 5 shows the performance of CompanyDepot-ML over the EDGE dataset based on different feature groups. We can see that the best performance is achieved using all feature groups, which shows that all the feature groups are contributing to the learning process. Considering a single feature group, query-entity features is the most effective one. Without this feature group, the performance is very bad. Next, we compare the effects of the query feature group and the entity feature group. We see that removing the group of entity features (green curve) from all feature groups leads to a higher decrease in performance than removing the group of query features (red curve). This indicates that the entity feature group is more effective than the query feature group for our task, especially considering the noisy KB which contains entities with varying degrees of quality.

6.5 Discussion

The job and resume datasets used in this paper were originally built for the purpose of creating benchmark datasets for comparing the performance of CompanyDepot-H, Legacy, and WService. For each query, we received up to three results in total, and collected manual judgments about them. With such limited labeled data, the correctness of many results returned by CompanyDepot-ML are unknown, which however were counted as wrong results in our evaluation. Although this setting brings limitations on the training and test data for CompanyDepot-ML, the experimental results still show that the machine learning based approach is quite promising. The performance numbers we computed for it are actually the lower bounds of the performance given better training and test data. While the machine learning based approach has advantages such as extendability and flexibility, the heuristic approach has its own advantage of being a strong baseline and eliciting ground-truth data for CompanyDepot-ML. Both the two methods have query response time within a second, showing the feasibility of applying the system to online employer name normalization.

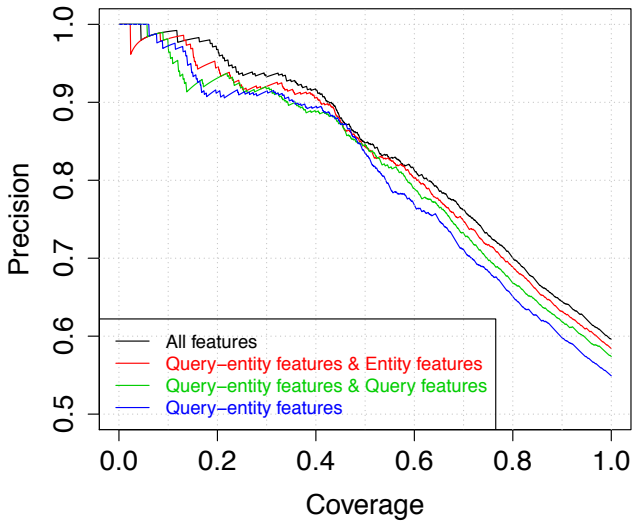

Figure 5: Effects of feature groups on performance of CompanyDepot-ML over the EDGE dataset.

7. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel employer name normalization task which takes an employer name and its associated location from job postings or resumes as input and normalizes it into entities in an employer KB. We then presented a system called CompanyDepot which contains a machine learning based approach CompanyDepot-ML and a heuristic approach CompanyDepot-H to address the task in three steps: entity retrieval, reranking, and validation. We compared its performance with existing employer name normalization systems used at CareerBuilder, applied it to a similar task of academic institution name normalization, and compared its performance with a state-of-the-art method. The experimental results over multiple real-world datasets showed the effectiveness, robustness, and generalization ability of our system.

In our future work, we plan to use more contextual information (e.g., employer address and website) if available. We also intend to develop more features to handle entity quality and query segmentation. Another branch of potential work is to improve the quality and coverage of the employer KB.

8. ACKNOWLEDGMENTS

We would like to thank Kyle Diemer for his help with building the benchmark datasets. We would also like to thank Yun Zhu for his helpful discussions about the project.

9. REFERENCES

[1] 4ICU. Top universities in the united kingdom. http://www.4icu.org/gb/. accessed 2014-01.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[3] A. Borkovsky. Item name normalization, Apr. 29 2003. US Patent 6,556,991.

[4] R. Busa-Fekete, G. Szarvas, T. Éltető, and B. Kégl. An apple-to-apple comparison of learning-to-rank algorithms in terms of normalized discounted cumulative gain. In *B. Proceedings of ECAI-12 Workshop, Preference Learning: Problems and Applications in AI*, 2012.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

[6] P. Christen. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Springer Publishing Company, Incorporated, 2012.

[7] B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 2009.

[8] M. Dredze, P. McNamee, D. Rao, A. Gerber, and T. Finin. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 277–285, 2010.

[9] P. S. Efraimidis and P. G. Spirakis. Weighted random sampling with a reservoir. *Inf. Process. Lett.*, 97(5):181–185, Mar. 2006.

[10] Guardian. Universities. http://www.theguardian.com/education/list/educationinstitution. accessed 2014-01.

[11] F. Jacob, F. Javed, M. Zhao, and M. Mcnair. sCooL: A system for academic institution name normalization. In *Collaboration Technologies and Systems (CTS), 2014 International Conference on*, pages 86–93, 2014.

[12] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM.

[13] S. Jonnalagadda and P. Topham. NEMO: Extraction and normalization of organization names from PubMed affiliation strings. *Journal of Biomedical Discovery and Collaboration*, 5:50, 2010.

[14] H. Kardes, D. Konidena, S. Agrawal, M. Huff, and A. Sun. Graph-based approaches for organization entity resolution in MapReduce. In *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing Workshop at EMNLP*, 2013.

[15] M. Kejriwal, Q. Liu, F. Jacob, and F. Javed. A pipeline for extracting and deduplicating domain-specific knowledge bases. In *Proceedings of 2015 IEEE International Conference on Big Data*, 2015.

[16] R. Leaman, R. I. Dogan, and Z. Lu. Dnorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, 29(22):2909–2917, 2013.

[17] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, Mar. 2009.

[18] W. Magdy, K. Darwish, O. Emam, and H. Hassan. Arabic cross-document person name normalization. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*, Semitic '07, pages 25–32, 2007.

[19] W. P. McNeill, H. Kardes, and A. Borthwick. Dynamic record blocking: Efficient linking of massive databases in mapreduce. In *Proceedings of 10th International Workshop on Quality in Databases (QDB) at VLDB*, 2012.

[20] D. Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Inf. Retr.*, 10(3):257–274, June 2007.

[21] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1375–1384, 2011.

[22] W. Shen, J. Wang, and J. Han. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Trans. Knowl. Data Eng.*, 27(2):443–460, 2015.

[23] J. Wermter, K. Tomanek, and U. Hahn. High-performance gene name normalization with GENO. *Bioinformatics*, 25(6):815–821, 2009.

[24] B. Yan, L. Bajaj, and A. Bhasin. Entity resolution using social graphs for business applications. In *International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2011*, pages 220–227, 2011.

[25] W. Zhang, Y. C. Sim, J. Su, and C. L. Tan. Entity linking with effective acronym expansion, instance selection and topic modeling. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three*, IJCAI'11, pages 1909–1914. AAAI Press, 2011.

[26] W. Zhang, J. Su, C. L. Tan, and W. T. Wang. Entity linking leveraging: Automatically generated annotation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1290–1298, 2010.

[27] W. Zhang, C. L. Tan, Y. C. Sim, and J. Su. NUS-I2R: learning a combined system for entity linking. In *Proceedings of the Third Text Analysis Conference, TAC 2010*, 2010.

[28] Z. Zheng, F. Li, M. Huang, and X. Zhu. Learning to link entities with knowledge base. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 483–491, 2010.