Entity Set Expansion from Twitter

He Zhao
Beijing Institute of Technology, School of Computer Science
Beijing, China
zhaohe1995@outlook.com

Chong Feng
Beijing Institute of Technology, School of Computer Science
Beijing, China
fengchong@bit.edu.cn

Zhunchen Luo
Information Research Center of Military Science, PLA Academy of Military Science
Beijing, China
zhunchenluo@gmail.com

Changhai Tian
Information Research Center of Military Science, PLA Academy of Military Science
Beijing, China
jamestch@163.com

ABSTRACT

Online social media yields a large-scale corpora which is fairly informative and sometimes includes many up-to-date entities. The challenging task of expanding entity sets on social media text is to extract more uncommon entities only using several seeds already in hand. In this paper, we present an approach which is able to find novel entities by expanding a small initial seed set on Twitter text. Our method first generates candidate sets on the basis of the semantic similarity feature. Then it jointly utilizes 2 text-based features and other 12 ones which carry social media specific information. With the scores on those features, a ranking model is learned by a supervised algorithm to synthetically score each candidate terms and then the final ranked list is taken as the target expanded set. We do experiments with 24 entity classes on the Twitter corpus and in the expanded sets there come many novel entities which have not been completely detected in previous researches. And the experimental results on the datasets of different years can perfectly consist with the objective law that fresh entities change as time goes on.

KEYWORDS

Social media mining; information extraction; entity set expansion

ACM Reference Format:
He Zhao, Chong Feng, Zhunchen Luo, and Changhai Tian. 2018. Entity Set Expansion from Twitter. In *2018 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '18), September 14–17, 2018, Tianjin, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3234944.3234966

1 INTRODUCTION

Have you ever expected to find out a TV series as hot as the one that you are obsessed with? Or have you ever wanted to know the newly released product by the brand of which you have been a great fan? As is well known, the information in social media, where all these topics are included, always keeps abreast of the times. In order to extract entity instances of a latent semantic class, a set expansion method takes an initial set of seeds as input, expands the set and then outputs more entity instances of that class. For example, one expecting to find all fast-food restaurants may give two or three well-known names like "KFC" and "Taco Bell" as seeds. Based on the seed set and the text corpus it can discover other more restaurant names such as "Mcdonald's", "Pizza hut" and "Dairy Queen". The task has been well-studied on the news corpus and the web-based text in the past decades. However, in recent years the booming user quantity of social media platforms such as Twitter and Facebook provides tremendous amount of noisy and informal short text, simultaneously presents a new challenge of expanding entity sets by common methods and technologies. Furthermore, due to the low-barrier of posting and the proliferation with mobile devices, the open-domain social media text is often more up-to-date and inclusive than the news articles. Conceivably, many newly presented entities are often expressed in abbreviated form or have various aliases, being widely mentioned in the informally written text. For example, "LOL" usually refers to "League of Legends" which is a popular online game and "FB" usually refers to "FaceBook" which is a social media website or an Internet company name. It is profoundly significant to extract these entities from the social media text, which provides convenience for the downstream applications such as constructing entity repositories for social media and delivering precise searching results in social media query suggestion systems.

Most semantic classes studied in previous work can be explicitly defined by several entity instances, like given "LALakers" and "Chicago Bulls" as seeds, the semantic class name can be easily concluded and the intention can be wisely inferred as to get other entities which are also "NBA teams".

Whereas different from common entities, plenty of newly-presented entities have more than one semantic layers, leading to that it is hard to give an absolute definition to the entity set. A more specific example is, for "Temple Run" and "Subway Surfers", we can conclude both of them are mobile games, iOS Applications or parkour games. There is no denying that the two instances should be contained into one set in some sense but the semantic class description may be diverse from different point of view. Unlike previous work, our research aggregates the entities into one set just considering their implicit semantic concepts, while not imposing restrictions on the possible class names. By this means, our method benefits the users who want to find more instances but are not sure about how to describe query intention, and no need to give the exact entity class names.

Corpus-based method has been applied to solve entity set expansion problem on general-purpose web source data and promising results have been reported in the past years. Wang and Cohen [11, 12] developed the SEAL system on semi-structured documecnts from web. He and Xin [4] proposed SEISA system using lists from HTML web pages and search query logs. A more latest work by Dalvi et al. [1] relied on clustering terms found in HTML tables and mainly focused on tabular data in web pages. Other than the web corpus, limited in 140 characters, in most cases the text from social media does not comprise distinct structured tables or lists. And due to the short text length, there's often not enough maximal context to build reliable wrappers for each seed mentions. Hence obviously most off-the-shelf web-based methods show ineffective when applied to solve the entity set expansion task on social media corpus. Another prominent line of work using bootstrapping approaches like that by Thelen and Riloff [10], the ASIA system [13] and KnowItAll [2]. They extracted context patterns to capture information about the behavior of a term or utilized hyponym patterns (e.g.,"fruits such as apples, bananas and oranges") to find instances based on the *"is-a"* relation. But with informal writing style and unconventional grammar, social media text often lacks signal words like "such as" to make up hyponym patterns. And since the oral expression characteristics differ depending on individual users, context around an entity mention is rarely to be coincident to construct patterns.

In this paper, we present a novel entity set expansion method which especially orients to the Twitter text and attempts to find some newly-presented entities that have not been sufficiently concerned before. Our method uses a two-phase architecture of extracting and ranking. In the first phase, to extract and generate candidate lists we adopt an embedding-based strategy to get scores on the feature of *semantic similarity*. Through observing plenty of tweets, we additionally propose some features that are specific in the social media text. Then for each term in a candidate list, we respectively calculate the score on all the features. In the second phase, we present a ranking model using a supervised learning method to combine all those features and provide the optimized coordination coefficient vector. We do experiments on the Twitter corpus and show that our approach can achieve high precision on 24 entity classes. And among the expanded lists, there are some newly-presented entities different from the ones in the old days.

2 RELATED WORK

Most previous work use two-phase methods to expand sets, so as we do, which divides the task into two steps, extracting and ranking. In the extracting phase, some corpus-based, search-based, pattern-based and bootstrapping approaches are adopted extensively. And in the ranking phase, most base on probability statistics, distributional similarity and some graph-based structures.

The SEAL system proposed by Wang and Cohen[11, 12] gets data source from web pages. In the extracting phase they construct maximal context wrappers using the given seeds and gather the candidate terms enclosed by these wrappers. Then in the ranking phase they use a random walk process and rank by similarity metric on the basis of a graph containing seeds, wrappers and candidate terms.

The KnowItAll system proposed by Etzioni et al.[2] also targets to web-scale data source. In the extracting phase they use textual patterns like "colors such as pink, blue" as generic templates to generate candidate lists. And in the ranking phase they adopt a bootstrapping method with the PMI measure computed by web search engines to estimate the likelihood of each candidate term.

Similar to the KnowItAll, the ASIA system of Wang and Cohen[13] is another pattern-based method. In the extracting phase they also use the hyponym patterns to get candidate terms but different from SEAL, they utilize a bootstrapper to make the process iterative. Then in the ranking phase supported by a random walk with restart on their graph it can determine the final ranking.

Talukdar et al.[9] propose a context pattern method that firstly find "trigger words" indicating the beginning of a pattern to extract candidate terms. And then they set evaluation mechanism to rank both the patterns and candidates.

Other existing work specially focuses on ranking the candidate sets. Bayesian Sets by Ghahramani and Heller[3], which is based on Bayesian inference, aims to estimate the probability that a candidate term belongs to a set by learning from a positive set P and an unlabeled candidate set U.

Sarmento et al.[8] propose a distributional similarity method according to a co-occurrence assumption that two elements consistently co-occur tent to be in similar semantic class. They find entity pairs by the coordination structures and represent entities in a vector space by encoding the co-occurrence frequency to compute the similarity with distance measures.

A PU-Learning method by Li et al.[5] transforms set expansion into a two-class classification problem. The seed set is regarded as a set P of positive examples and the candidate set is a set U containing hidden positive and negative cases. The task of filtering the candidate set turns to building a classifier to determine if each candidate term is positive or not.

Beyond the work listed above, we want to mention another more recent work closed to set expansion on social media text. Qadir et al.[7] propose a novel semantic lexicon induction approach to learn new vocabulary from Twitter. Their method roots on the ability to extract and prioritize N-gram context patterns that are semantically related to the categories and is able to deal with multi-word phrases of flexible term length. Starting with a few seed terms of a semantic category, they first explore the context around seeds in the corpus, and identify context patterns relevant to that category, which are then used to extract candidate terms. They experiment with three commonly discussed semantic categories, which are Food&Drinks, Games&Sports and Vehicles, and show good performance in these topics.

3 ENTITY SET EXPANSION

Our task can have a following definition: given a text corpus T and a set S of seed entities of a hidden class C, we expand S by finding new terms t (we use the word "term" interchangeably with "entity" since textual terms are taken as candidate entities in the expanded sets) from T, such that t is also semantically in the class C.

3.1 Entity Class

Class Num	*Entity Seed Set*
1	{Sociology, Chemistry, Biology}
2	{Avatar, Transformers, Iron Man}
3	{lawyer, doctor, manager}
4	{iPhone, iPad, MacBook}
5	{hat, shirt, sweater}
6	{Microsoft, IBM, Apple}
7	{Canada, France, Morocco}
8	{keyboard, print, mouse}
9	{steak, pork, sausage}
10	{KFC, Taco Bell, Starbucks}
11	{fever, headache, migraine}
12	{Temple Run, Warfare, Dota}
13	{milk, juice, coffee}
14	{Amazon, eBay, Google}
15	{brother, sister, dad}
16	{Super Junior, BIGBANG, CNBlue}
17	{head, ear, mouth}
18	{mcflurry, frappe, smoothie}
19	{football, baseball, tennis}
20	{Twitter, Facebook, Instagram}
21	{piano, guitar, drum}
22	{Justin Bieber, Taylor Swift, Katy Perry}
23	{gym, store, library}
24	{Skype, FaceTime, kik}

Table 1: 24 Entity classes and 3 representative seed examples for each set.

In this paper, we do experiments on 24 entity classes, far more than the ones used by Qadir et al.[7] of which the entity class number is only 3. We mainly choose the classes that are more concerned and widely discussed on the social media platforms. Among the classes, there contains both some common ones that are often taken as test data in previous entity set expansion works (e.g., {tea, coffee, beer}), and other novel ones that have got little attention before (e.g., {Avatar, Transformers, Iron Man}). For each class, we manually select some most representative entities by the help of *Wikipedia Lists* to build the initial candidate seed sets. With 29 as maximum and 5 as minimum, there contains 9 entities in our initial candidate sets on average. To expressly illustrate the classes, we give 3 entities as seed examples for each set, as is shown in Table 1. In consideration of the ambiguity caused by multilayer in semantic, we give neither straightforward description nor entity class name for all those classes. And in the actual experiment, we construct seed sets with 3 entities selected from the initial candidate seed sets, about which the strategies will be described in the following sections.

3.2 Corpus Description & Pre-processing

To build our corpus dataset, we use Twitter in this research. Crawled by online Twitter API, each tweet is wrapped in a JSON format, with text context, time information, user profiles and other informative fields included. The tweet text is in a short message with a maximum length of 140 characters, in the nature of informal grammar, abbreviated expressions and misspellings. We collect 9,582,314 pieces of English tweets published in January, 2013, and 8,453,895 ones published in March, 2017. To restrain the corpus scale, we remove other redundant information but only keep the fields and values left that are helpful in our method. For pre-processing, we utilize the spaCy, an industrial-strength NLP API with higher speed than the Stranford coreNLP to tokenize and chunk each tweet text.

3.3 Generating Candidate Sets

We first get the word embeddings for the corpus T using Word2Vec proposed by Mikolov et al. [6]. Then to construct seed sets, we select 3 entities from each initial candidate seed sets adopting three different manners:

- *Random Manner:* Randomly select 3 entities from the candidate seed set.
- *Maximum Similarity:* Select the most 3 semantically similar entities as seeds.
- *Maximum Frequency:* Select the most 3 frequent entities occurred in corpus as seeds.

To extract candidate terms for each seed set, with the word embeddings, the semantic similarity between two terms turns to be computed by the distance in the vector space. For each seed term $s \in S$, we select the top 500 semantically similar terms to add to the candidate set D and remove the duplicates. Then for each candidate term $item \in D$, we calculate the mean similarity with all seed terms in S as its score on the feature of *Semantic Similarity*.

$$SC_{sim}(item) = \frac{\sum_{s \in S} Similar(s, item)}{|S|} \qquad (1)$$

Finally we rank all those candidate terms. And the top 500 ones with the highest *Semantic Similarity* scores constitute the candidate set.

3.4 Scoring Candidate Terms on Features

Besides the feature of *Semantic Similarity*, we propose 14 more features, including 2 text-based ones (*Connective Patterns* and *Prefix Rules*) and 12 social media specific ones.

Through the observation of vast tweet text, we find that many homogeneous entities are in appositive construction, connected by some conjunction words or symbols (e.g., ",", "and", "or", et al.). The intuition has been verified in the research of Widdows and Dorow [14] and been also depended on in the entity set expansion algorithm of Sarmento et al. [8]. Another observation is that in one single piece of tweet, the entities, which synchronously have the prefixes "@" (to remind someone to notice) or "#" (to be a hashtag), are very likely in the same class. We show some examples from Twitter platform in Example 1. So here we propose the following hypotheses:

(1) If a term is often connected with seed entities by the conjunction symbols that frequently occur between two seed entities in the text corpus, the term is more likely to be an instance of that class set.

(2) If a term often occurs in the same piece of message with seed entities, both having "@" or "#" as prefix, the term is more likely to be an instance of that class set.

Example 1:

- *LA Lakers:* ***Los Angeles Lakers*** *vs* ***Detroit Pistons*** *— Mr. Throwback Showdown. Welcome to THROWBACK THROWDOWN, wher...*
- *John Wood: So, lives in a city and goes to* ***Starbucks***, ***Costa*** *or* ***Cafe Nero****? That is probably 75% of the population*
- *Vala Afshar: % of users on each site who get news there:* ***@Twitter*** *74%* ***@Raddit*** *68%* ***@Facebook*** *68%* ***@Tumblr*** *39%* ***@Instagram*** *27% pewrsr.ch/2gKsCf4*
- *Pop Predictions reply @2016_PopVisions: #EMAs2017 prediction for: Biggest Fans* ***#LadyGaga #JustinBieber #BTS #ArianaGrande #FifthHarmony***

Based on the two hypotheses, for each term in the candidate set, we respectively get scores on the feature of *Connective Patterns* and *Prefix Rules*.

Connective Patterns: By observing tweet text we conclude a set CS of 24 conjunction symbols that are generally used to connect two homogeneous terms as is showed in Table 2. For each conjunction symbol $c \in CS$, and a pair of seed terms $(s_1, s_2) \in S$, we respectively construct two reversed

and	or	,	&	+	-
x	X	/	$\star$	>	<
$\vert$	vs	VS	.	:	//
$\cdot$	$\backslash$	$\backslash\backslash$	=	$\ll$	$\gg$

Table 2: Conjunction symbols set CS

strings "s_1 c s_2" and "s_2 c s_1" which we called *searching patterns* (e.g., "KFC & Starbucks"). We count frequencies of these two *searching patterns* in corpus and sum them to get the frequency value $f_{(c,pair)}$ of c in regard to pair (s_1, s_2). The sum of the values computed by all possible seed pairs in S indicates the weight of that symbol c.

$$Weight(c) = \frac{\sum_{pair \in S} f_{(c,pair)}}{\sum_{c \in CS} \sum_{pair \in S} f_{(c,pair)}} \qquad (2)$$

Then for each candidate term $item \in D$ and each seed term $s \in S$, we use symbol c to construct two reversed strings "s c $item$" and "$item$ c s" which we called *matching patterns* (e.g. "KFC & Mcdonald's", "Mcdonald's & KFC"). Counting and summing the frequencies of the *matching patterns*, and multiplying the weight of c, we get the frequency value $f_{(item,c,s)}$of $item$ weighted by c. So the sum of the values computed by all symbols in CS indicates the score of $item$ in feature of *Connective Patterns*.

$$SC_{con}(item) = \sum_{c \in CS} \sum_{s \in S} Weight(c) \times f_{(item,c,s)} \qquad (3)$$

Prefix Rules: To get the score in feature of *Prefix Rules*, we respectively count the co-occurrence in *Hashtags("#")* and *User Mentions("@")* for each candidate term and seed sets.

For each candidate term $item \in D$ and each seed term $s \in S$, we count and sum the frequencies of that "#$item$" and "#s" co-occur in a same tweet as $f_{Hashtags}(item, s)$, and the frequencies of that "@$item$" and "@s" co-occur in a same tweet as $f_{UserMentions}(item, s)$. So that summing up the two values we can get the score $SC_{pre}(item)$ in feature of *Prefix Rules*.

$$SC_{pre}(item) = f_{Hashtags}(item, s) + f_{UserMentions}(item, s) \qquad (4)$$

Timestamp: Through the "created_at" field and its value, we can get the posted time of each tweet. Based on the fact that people often tweet about the entities under some hot topics or new events aggregating in a period of time, we take the timestamp as a feature and calculate similarity score in time for each candidate term.

First we collect timestamps of all the tweets where there is any seed entity mentioned. Then we get a centroid of the timestamps as a center time point $T_{center}(seeds)$. For each candidate term $item \in D$, we can also get its center time point $T_{center}(item)$ by averaging all the timestamps of its mentions. So that its time similarity can be calculated as delta-T between $T_{center}(item)$ and $T_{center}(seeds)$. By this way we can get the score in feature of *Timestamp* for each candidate term.

$$SC_{time}(item) = \frac{1}{|T_{center}(item) - T_{center}(seeds)| + 1} \quad (5)$$

Status ID: Each tweet has its unique identification value in the "id" field, which we can use to calculate the score in the feature of *Status ID*. It is obvious that the candidate terms more frequently co-occur with seeds in same tweets ought to get higher possibility of being homogeneous. So we put all "id" values of the tweets having seed mentioned into a candidate pool P. Then we remove the ids only occur once from the pool, and keep the ones of which the frequency is greater than or equal to 2. By this way, we capture the tweets having at least 2 seed mentions included. For each $id \in P$, we use f_{id} to represent its frequency. For each candidate term $item \in D$, we count the times $t_{id}(item)$ of it mentioned by each id. So the score in feature of *Status ID* can be presented as:

$$SC_{id}(item) = \sum_{id \in P} f_{id} \times t_{id}(item) \quad (6)$$

For most following features (except the *Coordinates*), we use similar process to calculate the scores. First for seed set, the candidate pool P of each field is generated by cutting the members of which frequency is less than 2. Then for candidate term, we count the times it is mentioned synchronously with each member of P. At last calculating the product of these two values and summing it up for all members in P, we can finally get the score in each feature:

$$SC_{field}(item) = \sum_{id \in P} f_{field} \times t_{field}(item) \quad (7)$$

So in what follows, we will only illustrate the motivation and reason why we choose these features, but omit the detailed description of how to calculate the scores.

User ID: Unique identification of tweeter is stored in "user_id" field, which we can use to calculate the score SC_{user_id}. It is because that if a user have once tweeted about one entity, there is a great possibility the user has interests in related topics and may tweet more entities in the same class. For example, someone is keen on soccer may usually tweet "Real Madrid", "Man Utd" or some other club names.

In-reply-to Status ID & In-reply-to User ID: These two features are about comment and reply information. If a tweet is replying to another one, the id of that target tweet is stored in "in_reply_to_status_id" field and the $user\_id$ is stored in "in_reply_ to_user_id" field. Just as the following example shows, the replies to the same tweet or user are likely to mention homogeneous entities. Here the entities in bold type are all children's literature names.

Example 2:
- *The British Library: On #InternationalLiteracyDay, we look at the origins of children's literature (A Little Pretty Pocket-Book, 1770) bit.ly/2xiJtAt*
- *Debbie Mancuso reply@britishlibrary: Can't name just 1! All **Harry Potter** @jk_rowling books, **Unicorn Chronicles** @brucecoville, **Breakfast of Champions**, **The Outsiders**, & more.*
- *Dusty(Bos10lot) reply@britishlibrary: **Hamilton** by Ron Chernow and **Lord of the rings**.*

For these two features, we respectively get the scores $SC_{in\_reply\_to\_status\_id}$ and $SC_{in\_reply\_to\_user\_id}$.

User Location & Country Code: These two features are about geographic information. The "user_location" field indicates the location name (e.g."Parana", "California") and the "country_code" field is more formal (e.g. "US", "CH", "DE"). It is based on the assumption that people always discuss about some local events so the tweets from the same location may contain homogeneous entities. Hence, we get two scores $SC_{user\_location}$ and $SC_{country\_code}$.

Coordinates: The "coordinates" field is also about geographic information. But different from the above two, this feature can calculate the similarity of position more directly. Each "coordinates" value is a point in two dimensional space. So we can easily measure similarity by calculating the distance between every two position points. We first get the coordinate for each seed entity mentions and put them into a position pool P. Then for each candidate term $item \in D$, we calculate the average coordinate distance between all its mentions $M(item)$ and the members of P. So we get the score of feature in *Coordinates* as:

$$SC_{coordinates}(item) = \frac{|M(item)|}{\sum_{[X_s,Y_s] \in P} \sum_{[X_m,Y_m] \in M(item)} Dis([X_s,Y_s],[X_m,Y_m])} \quad (8)$$

Retweet Status ID: That is pretty intuitive the field "retweet_id" is about retweet information, which indicates the unique identification of its original tweet. Due to that social function of Twitter, people often talk about the same topic in their retweet status as the original one. So similar entities often occur in those tweets retweeting from the same source. We can get the score $SC_{retweet\_id}$ in this feature.

Hashtags & Urls & User Mentions: These three features are about additional entity information. On most social media platforms, users take part in the discussion of certain topic through "hashtags" ("#"), link to other pages through "urls" (often transformed into a short form) and remind someone to have a look through "user_mentions" ("@"). As is shown in example 1, the tweets with hashtag "#InternationalLiteracyDay", with url "bit.ly/2xi JtAt" or with user mention "@britishlibrary" may also have some book names mentioned. So it is reasonable for us to assume that the tweets having same values in these additional entity fields may have homogeneous entities involved in text. Therefore, for these three features, we respectively calculate the score $SC_{hashtags}$, SC_{urls} and $SC_{user\_mentions}$.

Candidate Terms Ranking: With the scores of each candidate terms in *Semantic Similarity* feature, text-based features (*Connective Patterns* and *Prefix Rules*), and other social media information features, now we can respectively rank the candidate set according to the 15 features, and get 15 ranking lists. We present a synthetic ranking model R by learning a set of coordination coefficient vector $\vec{W} = (\alpha_1, \alpha_2, ..., \alpha_{15})$ to combine the each item's positions, which are $R_1(item)$, $R_2(item)$ ... $R_{15}(item)$ in the 15 ranking lists:

$$R(\vec{W}) = \sum_{i=1}^{15} \alpha_i R_i(item) \tag{9}$$

First we give some notations and explanations using those defined in AdaRank by Xu and Li [15] as reference, as is showed in Table 3. Here we use MAP(Mean Average Precision) as the performance measure. For a given semantic class c_i, rank of candidate lists Y_i and a permutation π_i on candidate set D_i, average precision for c_i is defined as:

Notations	Explanations
$c_i \in C$	i^{th} entity class
$D_i = \{item_{i1}, \cdots, item_{i,n(c_i)}\}$	Candidate Set of Class c_i
$y_{ij} \in \{r_1, r_2, \cdots, r_l\}$	Rank of $item_{ij}$ w.r.t. c_i
$Y_i = \{y_{i1}, y_{i2}, \cdots, y_{i,n(c_i)}\}$	List of ranks for c_i
$S = \{(c_i, D_i, Y_i)\}_{i=1}^m$	Training Set
$\vec{W} = (\alpha_1, \alpha_2, ..., \alpha_{15})$	The coordination coefficient vector
$f, R(\vec{W}), R_1, ..., R_{15} \in \mathcal{R}$	Ranking models
$\pi(c_i, D_i, f)$	Permutation for c_i, D_i, and f
$E(\pi(c_i, D_i, f), Y_i) \in [-1, +1]$	Performance measure function

Table 3: Notations and explanations

$$AvgP_i = \frac{\sum_{j=1}^{n(c_i)} P_i(j) \cdot y_{ij}}{\sum_{j=1}^{n(c_i)} y_{ij}}, \tag{10}$$

where y_{ij} takes on 1 and 0 as values, which presents $item_{ij}$ is positive or negative instance and $P_i(j)$ is defined as precision at the position of d_{ij}:

$$P_i(j) = \frac{\sum_{k:\pi_i(k) \le \pi_i(j) y_{ij}}}{\pi_i(j)}, \tag{11}$$

where $\pi_i(j)$ presents the position of $item_{ij}$. To learn coordinate coefficient for each rank model, first we assign the same value to each coefficient α_i. With a training set $S = \{(c_i, D_i, Y_i)\}_{i=1}^m$ as input, the algorithm takes the performance measure function E and runs at most T rounds. At each round, it creates a new rank model f by linearly combining the 15 feature rankers with $\vec{W}$. Each coefficient α_i will be updated to increase if the corresponding feature rank model outperforms model f, but decrease the opposite. The update operation can be expressed as:

$$\alpha_k = \alpha_k \frac{exp\{\frac{\sum_{i=1}^m E(\pi(c_i, D_i, R_k), Y_i) - \sum_{i=1}^m E(\pi(c_i, D_i, f_t), Y_i)}{m}\}}{Z_m} \tag{12}$$

Algorithm 1 The rank model algorithm

```
Input: Training set $S = \{(c_i, D_i, Y_i)\}_{i=1}^m$
       Performance measure function $E(\pi(c_i, D_i, f), Y_i)$
       Max iterations $T$
Initialize: for $\alpha_k$ in $\vec{W}$: $\alpha_k = \frac{1}{|\vec{W}|}$
Output: $R(\vec{W}) = f_t$
1: for $t = 1$ to $T$ do
2:     $f_t = \sum_{k=1}^{15} \alpha_k \cdot R_k$
3:     for $\alpha_k$ in $\vec{W}$ do
4:         Update $\alpha_k$
5:     $R_t = \sum_{k=1}^{15} \alpha_k R_k$
6:     if $MAP(R_t) < MAP(f_t)$ then
7:         return $f_t$
```

Then we test if the rank model combined with updated $\vec{W}$ is better than f. If not, the algorithm reaches the maximum point at f, the loop ends and f is returned. The pseudocode of the whole process is shown in Algorithm 1, where Z_m is a standardization factor, to make $\sum \vec{W} = 1$:

$$\Delta = \sum_{i=1}^m E(\pi(c_i, D_i, R_k), Y_i) - \sum_{i=1}^m E(\pi(c_i, D_i, f_t), Y_i) \tag{13}$$

$$Z_m = \sum_{k=1}^{15} \alpha_k \cdot exp\frac{\Delta}{m} \tag{14}$$

4 EXPERIMENTS

4.1 Baselines

To compare the performance of our set expansion algorithm with previous work, we use two pattern-based approaches. One is a **C**ontext **P**attern **I**nduction **M**ethod [9], which we briefly name as **CPIM**, and another is a **T**witter-oriented **S**emantic **L**exicon **I**nduction **M**ethod [7], briefly named as **TSLIM**. Due to that many previous works target to structured text data source from web and utilize tables, lists and markup tags to solve the problem, obviously they are not fit for our tweet text. Hence we choose approaches based on context patterns that do not impose restrictions on specific corpus. Besides, to the best of our knowledge very few works study entity set expansion problem on social media, so we unavoidably take Qadir's lexicon induction method as a baseline, which is more or less distinguishing compared with our task. Furthermore, to demonstrate our synthetic ranking model is effective, we also compare the ranking results with that only using semantic similarity from Word2Vec, and show the increase in precision.

4.2 Evaluation

Since the output of all these approaches are ranked lists, we adopt *Rank Precision@N*, which is commonly used for

Method	*Dataset 2013 Jan*					*Dataset 2017 Mar*				
	5	**10**	**20**	**50**	**100**	**5**	**10**	**20**	**50**	**100**
CPIM	0.45	0.40	0.37	0.32	0.30	0.48	0.44	0.42	0.31	0.32
TSLIM	0.72	0.60	0.54	0.42	0.39	0.74	0.63	0.50	0.39	0.37
Word2Vec	1.00	0.99	0.94	0.83	0.70	1.00	0.98	0.92	0.83	0.68
Our method	**1.00**	**0.99**	**0.96**	**0.87**	**0.75**	**1.00**	**0.99**	**0.95**	**0.88**	**0.74**

Table 4: Precision @ top N for four methods (with 3 seeds).

Seed Select	Corpus size	5	10	20	50	100
Random	**1,398,511**	0.91	0.85	0.77	0.64	0.51
MaxSim	**1,398,511**	0.99	0.95	0.86	0.73	0.58
MaxFreq	**1,398,511**	1.00	0.96	0.86	0.73	0.59
MaxFreq	**4,080,031**	1.00	0.97	0.93	0.85	0.73
MaxFreq	**9,582,314**	**1.00**	**0.99**	**0.96**	**0.87**	**0.75**

Table 5: Precision @ top N when varying seed sets and corpus size (with 3 seeds).

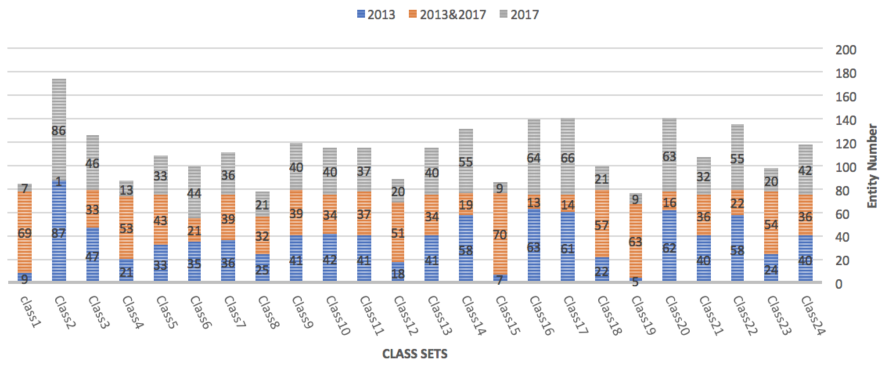

Figure 1: The difference between the expanded entity set of Dataset 2013 and Dataset 2017.

the evaluation of entity set expansion techniques [5] with following definition:

Precision@N : The percentage of correct entities among the top N entities in the ranked list.

For each entity class, we collect the top 100 terms in the ranked lists generated by each of the approaches, and provide the sets to three individual annotators. The annotators were given only the initial seed examples as guidelines, without any redundant definitions or descriptions of the class, and were asked to determine if each term is a positive or negative instance of that set. At last we assign class membership to each term follow the majority voting rule, that is, a term will be judged as positive only if more than two annotators deem it is.

4.3 Results

In our experiment, we use the word embeddings of 200 dimensions, and set the window size to 5, minimum word count to 5. To train our synthetical rank model and test the performance, we utilize 6-fold cross validation on our 24 entity classes. The number of initial seed entities is fixed to 3. And for four methods, we present the precision at the top 5-, 10-, 20-, 50- and 100-ranked positions (i.e., precision@5, 10, 20, 50 and 100). The detailed experimental results are shown in Table 4 for both the *Dataset 2013 Jan* and the *Dataset 2017 Mar*. It indicates that our method is more effective when dealing with those 24 entity classes on both two datasets. To test the sensitivity of the initial seeds, we show results when varying the seed selecting strategies. We also use different subsets of Dataset 2013 to analyze the effects of the corpus size. All the results are shown in Table 5.

From Table 5 we can see that using MaxFreq to select seeds performs better. We speculate that the reason is, whether the seed entities are common in corpus has a great influence on the precision result. A seed more frequent in corpus is able to bring more information when constructing patterns and rules. For example, “BurgerKing” and “KFC” as seeds are better than “WhataBurger”. Hence uncommon and infrequent entities should be avoided when generating initial seed sets. In addition, we speculate that the relevancy in semantic among seed entities also affects much. It is probably because more similar seeds are often more representative for their class, and may prevent the semantic center drift. Such as the seed set {“ukulele”, “violin”, “guitar”} is better than {“piano”, “violin”, “guitar”}, where the former is more likely to have the center in “string instruments”. Not surprisingly, enlarging the corpus size has a significant impact on set expansion performance. One side the word embeddings work better on larger corpus, and on the other side, most terms occur more times with the corpus growing, which makes improvement in dealing with the entity sparseness problem.

Figure 1 shows the differences between the expanded entity sets of Dataset 2013 and Dataset 2017. It can be seen that both datasets have particular entities, which can perfectly conform to the fact that novel entities will arise as time goes on. For example, “Zootopia” and “Deadpool” do appear in the expanded set of 2017 whereas do not in that of 2013.

5 CONCLUSION

We present a set expansion method which can help to find novel entities from Twitter. We first adopt word embeddings to generate candidate sets and get scores of semantic similarity. And the other two text-based feature scores and 14 social feature scores are acquired by using connective patterns, prefix rules and some fields of the tweet. Then we propose a synthetical ranking model to integrate the feature scores and show evident improvement promotion in more entity classes than previous work.

6 ACKNOWLEDGEMENTS

This work was supported by the National High-tech Research and Development Program (863 Program) (No. 2014AA015105) and National Natural Science Foundation of China (No. 61602490).

REFERENCES

[1] Bhavana Bharat Dalvi, William W. Cohen, and Jamie Callan. 2012. WebSets: extracting sets of entities from the web using unsupervised information extraction. In *ACM International Conference on Web Search and Data Mining*. 243–252.

[2] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2004. Web-Scale Information Extraction in KnowItAll. *Proc Www* 27, 2 (2004), 146–52.

[3] Zoubin Ghahramani and Katherine A. Heller. 2005. Bayesian Sets. *In Proc. Adv. in Neural Inform. Processing Systems (NIPS-05)* (2005), 435–442.

[4] Yeye He and Dong Xin. 2011. SEISA:set expansion by iterative similarity aggregation. In *International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April*. 427–436.

[5] Xiao Li Li, Lei Zhang, Bing Liu, and See Kiong Ng. 2010. Distributional similarity vs. PU learning for entity set expansion. In *ACL 2010 Conference Short Papers*. 359–364.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Computer Science* (2013).

[7] Ashequl Qadir, Pablo N. Mendes, Daniel Gruhl, and Neal Lewis. 2015. Semantic lexicon induction from twitter with pattern relatedness and flexible term length. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2432–2439.

[8] Luis Sarmento, Valentin Jijkuon, Maarten De Rijke, and Eugenio Oliveira. 2007. ”More like these”: growing entity classes from seeds. In *Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November*. 959–962.

[9] Partha Pratim Talukdar, Thorsten Brants, Mark Liberman, and Fernando Pereira. 2006. A Context Pattern Induction Method for Named Entity Extraction. *in Computational Natural Language Learning (CoNLL-X)* (2006), 141–148.

[10] Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Acl-02 Conference on Empirical Methods in Natural Language Processing*. 214–221.

[11] Richard C. Wang and William W. Cohen. 2007. Language-Independent Set Expansion of Named Entities Using the Web. In *IEEE International Conference on Data Mining*. 342–350.

[12] Richard C. Wang and William W. Cohen. 2009. Iterative Set Expansion of Named Entities Using the Web. In *Eighth IEEE International Conference on Data Mining*. 1091–1096.

[13] Richard C. Wang and William W. Cohen. 2010. Automatic Set Instance Extraction using the Web. In *ACL 2009, Proceedings of the Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing of the Afnlp, 2-7 August 2009, Singapore*. 441–449.

[14] Dominic Widdows and Beate Dorow. 2002. A graph model for unsupervised lexical acquisition. In *International Conference on Computational Linguistics*. 1093–1099.

[15] Jun Xu and Hang Li. 2007. AdaRank:a boosting algorithm for information retrieval. (2007), 391–398.