EgoSet: Exploiting Word Ego-networks and User-generated Ontology for Multifaceted Set Expansion

Xin Rong[1], Zhe Chen[2], Qiaozhu Mei[1,2], Eytan Adar[1,2]

[1]School of Information, [2]Computer Science and Engineering
University of Michigan Ann Arbor, MI
{ronxin,chenzhe,qmei,eadar}@umich.edu

ABSTRACT

A key challenge of entity set expansion is that multifaceted input seeds can lead to significant incoherence in the result set. In this paper, we present a novel solution to handling multifaceted seeds by combining existing user-generated ontologies with a novel word-similarity metric based on skip-grams. By blending the two resources we are able to produce sparse word ego-networks that are centered on the seed terms and are able to capture semantic equivalence among words. We demonstrate that the resulting networks possess internally-coherent clusters, which can be exploited to provide non-overlapping expansions, in order to reflect different semantic classes of the seeds. Empirical evaluation against state-of-the-art baselines shows that our solution, *EgoSet*, is able to not only capture multiple facets in the input query, but also generate expansions for each facet with higher precision.

CCS Concepts

•Computing methodologies → Information extraction; Semantic networks; •Information systems → *Clustering; Web mining; Information retrieval;*

Keywords

Web Mining; Information Extraction; Entity Set Expansion

1. INTRODUCTION

Entity set expansion is useful for a number of applications, including question answering [27], query suggestion [6], consumer vocabulary construction [29], and knowledge extraction [22]. For example, given *lincoln*, *nixon*, and *obama* as seeds, one might expect a set expander to find all American presidents. Formally, entity set expansion is defined as finding a set of "sibling" entities given one or a few examples, such that the entities and the seeds belong to the same semantic class [35].

The task of set expansion can be challenging if the input seeds fall into multiple semantic classes. If the seeds are multifaceted, the result set will often contain suggestions pulled from multiple senses. For example, given the term *orange* as the seed, a set expansion algorithm may produce a combination of colors (e.g., *red, green, blue*) and fruits (e.g., *peach, grape, lemon*). Some senses may also be significantly less popular than others. For example, *atlanta* may most commonly be of the class *cities in Georgia, US* but it also fits within *Olympic host cities*. Providing multiple input seeds may help orient the algorithm (e.g., *atlanta* AND *barcelona* would make the *Olympic* class more likely). Even if multiple seeds are provided, the results can still be ambiguous, as different classes can have a large number of overlapping members. Having multiple seeds as input may help narrow down the semantics, but multi-seed queries can still possess multifacetedness. For example, *lemon*, *lime*, *olive*, and *orange* all belong to *names of colors* and *names of fruits*, which makes disambiguation potentially challenging.

In reality, users or upstream applications cannot always provide an accurate list of seeds that narrows the search space down to one semantic class. Instead of relying on a univocal set of seeds, we aim to provide an entity set expansion algorithm that automatically organizes the expanded entities into multiple facets, which correspond to the most common senses of the seeds.

User-generated ontologies, such as those found in Wikipedia, represent one possible route forward. For example, a simple algorithm would find all categories for input seeds and then return grouped matches that are siblings within each category. However, such user-generated ontologies, due to its nature of human categorization, often have problems such as mixing multiple concepts in one category, or containing noise in certain categories [10].

An alternative approach is to "learn" sibling relationships among entities directly from a large corpus. Distributional similarity is one way to do this. Distributional similarity measures word relatedness with the assumption that similar words appear in similar contexts [18]. Such similarity metrics can facilitate set expansion by finding relevant terms [33, 30] and optimizing the internal coherence of the result entity set [12]. However, for multifaceted seeds, distributional similarity may not naturally group the related terms into distinct semantic classes, even if the coverage of sibling terms increases with sufficient data.

In order to find desired groupings for multifaceted seeds, we utilize a word-network representation. Specifically, we treat words that are distributionally similar to the seed (the ego) as nodes and use the pairwise similarity between those words to create weighted edges, thereby forming an "ego-network." This representation is useful for our task because ego-networks for multifacted seeds often display modular community structure, where each community contains possible set-expansions for a particular semantic facet of the seed. Figure 1 shows an example ego-network of the term *beijing*, where two distinct clusters exist and can be used for expansions toward different semantic classes. To increase the accuracy of

WSDM'16, February 22–25, 2016, San Francisco, CA, USA.

ISBN 978-1-4503-3716-8/16/02...$15.00

DOI: http://dx.doi.org/10.1145/2835776.2835808

Figure 1: Ego-network of *beijing* with three communities: major cities in China (upper left), provinces in China (lower left), and Olympic host cities (right).

the extracted sibling terms, we use a set of skip-gram features that impose strong positional constraints on the context, and we find that appropriate filtering and sampling on such a feature set can more precisely recover sibling relations and thus result in ego-networks with even better modular structure. However, a limitation of such a refined distributional similarity metric is that it favors popular terms and under-performs for long-tail terms in certain domains, such as musical records and movie titles.

To create a system that performs well in all domains, we fuse the ego-network model with user-created ontologies into a new system: EgoSet. EgoSet uses the ego-network to find the initial clusters for a seed, and aligns those clusters with user-created ontologies. These clusters are "purified" by this process to generate set-expansions that are not only structurally aligned with the seed's facets but also of high precision.

The most significant contributions of this work are: (1) addressing the problem of multifaceted set-expansion through the construction of the ego-networks using novel skip-gram features and (2) the unsupervised integration of the ego-networks and mined ontologies to create comprehensive and internally coherent entity sets. To evaluate the EgoSet system, we created a new evaluation dataset of single- and multi-seed queries (all multifaceted). Empirical evaluation shows that our system outperforms state-of-the-art baselines in terms of the coverage on different semantic classes, as well as the precision in each individual cluster.

2. USER-CREATED ONTOLOGIES

Ontologies are a natural resource for set expansion. Wikipedia, for example, contains various curated lists suitable for ontology construction, including category tags, lists and tables in the articles, and *List-of* pages. Qualitatively, we have found the List-of pages to have the right combination of being prevalent and relatively "clean". To obtain this data, we crawled all English Wikipedia pages titled "List of . . . ", and obtained 187,017 pages (collected on March 12, 2015). If the page contained tables, we treated each table column as a sublist, and obtained 1,124,708 sublists. We kept the 38,078 lists and sublists with at least 5 entities. The remainder were considered as too short and thus discarded. The retained lists became our working ontology.

Table 1 shows the titles of several Wikipedia lists that cover the entity *apple*. Despite their relative high quality, Wikipedia lists

List Title	Size
List of English words of Anglo-Saxon origin	1046
List of record labels	627
List of defunct automobile manufacturers	513
List of multinational corporations	85
List of plants in the Bible	41
List of most valuable crops and live-stocks	29
List of the largest IT companies	9
List of loanwords in Malayalam	5
. . .	. . .

Table 1: Eight example Wikipedia lists (of 99) containing *apple*.

may not be the "ultimate" results of set expansion due to: (1) some groups of lists contain overlapping topics, such as *List of laptop brands and manufacturers* and *List of computer hardware manufacturers*; (2) many lists mix two or more concepts together and thus result in incoherent clusters, such as *List of most valuable crops and live-stocks*, which mixes fruit with vegetables and animals; (3) many lists cover topics that are too rare or obscure, such as *List of loanwords in Malayalam*.

These issues motivate an interesting question: how do we generate internally-coherent clusters that provide a comprehensive coverage of the "common senses" of the input seeds? While there are existing studies that tackle this problem by using topic modeling [1, 38], their performance is usually not satisfactory. An important reason is that the co-occurrence relationship between entities learned from these list ontologies do not correlate well with their similarity relations in real text. While user-created ontologies may combine entities together in various interesting ways, text data most truthfully reflect our common sense about entity relations. Our preference is thus to first identify relevant clusters based on text, and only then refine those with the user-generated ontologies.

3. EGOSET

In this section, we introduce our proposed EgoSet system. As depicted in Figure 2, the system contains both indexing and querying elements.

3.1 Feature Extraction

To extract sibling relations from text, distributional similarity is frequently employed [23, 30, 37]. There are multiple possible feature definitions for distributional similarity. Given a term T_0 in a sentence, one can define a context window W around T_0, and extract features from W. Common features include unigrams (e.g., T_1), n-grams (e.g., T_1T_2), and skip-grams (e.g., "T_{-1} __ T_1", where T_0 is replaced with a placeholder). We use skip-grams instead of unigrams or n-grams because skip-grams impose stronger positional constraints on where contextual words may appear with regard to the target term. Otherwise, we risk finding relevant concepts (e.g., *apple* vs. *ipad*) rather than truly sibling entities (*apple* vs. *microsoft*). In addition to distributional similarity, we could also use patterns (e.g., "w_1, *such as* w_2, w_3, *and* w_4") to find sibling entities, but as Shi et al. [30] indicate, patterns tend to introduce more noise than distributional similarity as the corpus scales up.

In practice, we find that using a combination of skip-grams of varying lengths can lead to even better performance than a single length setting. Figure 2(a) illustrates the skip-gram features we extract from each context window. For target term T_0, six different features are extracted. For example, skip-gram S_2 = "T_{-2} T_{-1} __ T_1".

Note that our skip-gram feature set should not be confused with

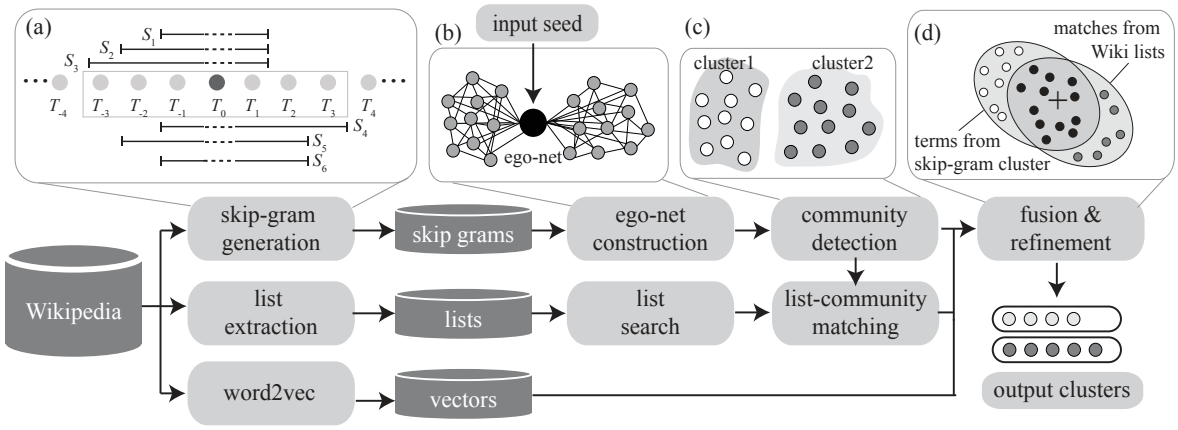

Figure 2: The end-to-end system pipeline showing the indexing of ego-networks and serving of a single-seed query.

the skip-gram model in word2vec, which is a well known word embedding toolkit [20]. In the skip-gram model of word2vec, given an input word, the training objective function measures how well the neural network model predicts every single token in the skip-gram surrounding that input word. The sequential order of the tokens in that skip-gram is ignored as the objective function takes the average of the prediction error for each token [28]. In contrast, we treat each skip-gram S_i as an integral unit, which leads to much sparser feature space but much tighter semantic constraints.

After feature extraction, many existing approaches assign feature weights using point-wise mutual information (PMI) [8] for each skip-gram s and word w:

$$f_{w,s} = \log \frac{X_{w,s}}{\sum_{w'} X_{w',s} \cdot \sum_{s'} X_{w,s'}} \tag{1}$$

where $X_{w,s}$ is the co-occurrence count between w and s. This weighting strategy tends to promote rare skip-grams. Given our large feature space, this strategy is not particularly helpful. We find that many extracted features can be discarded and a sample of the remaining features is sufficient for extracting sibling relations. This can be better understood in the context of a bipartite network.

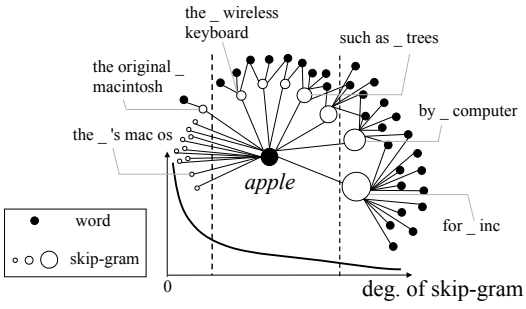

Figure 3: Bipartite network of words and skip-grams in the neighborhood of *apple* and the degree distribution of skip-grams.

Figure 3 illustrates the neighborhood of *apple* in a bipartite network of words and skip-grams. From left to right, the degree of skip-grams increases, and so changes their importance in capturing sibling terms. By inspecting example skip-grams connected to the seed word, it can be seen that skip-grams on the left end of the distribution are not particularly useful in connecting the seed word to the others. For example, "*the __'s mac os*" only captures the uniqueness of *apple* instead of the commonality it shares with its peers. In practice, we find that most sibling relations are effectively recovered by skip-grams with 5–50 neighbors. For example, "*the __ wireless keyboard*" falls in this range, and is useful for linking terms of electronic companies together. In comparison, a more frequent skip-gram, "*for __*", may be less precise in finding siblings, as it mixes a lot of companies together. We have found that we can discard skip-grams with fewer than 5 neighbors, or more than 2,500, without seriously impacting performance.

After this filtering process, we assign weight for each skip-gram s and word w using:

$$f_{w,s} = \log(1 + X_{w,s})[\log |W| - \log(\sum_{w} X_{w,s})] \tag{2}$$

where $X_{w,s}$ is the co-occurrence count between w and s, and $|W|$ is the total number of words in the vocabulary. This weight assignment is equivalent to tf-idf if we consider each w as a "document", and each s as a "term." In practice, we find this weight function to work better than PMI.

Even with strict filtering, each word may still have a large number of skip-grams. To further improve computational efficiency, we reduce the dimensionality of the feature space by sampling 300 skip-grams per word using weighted minhash [14], which is an algorithm that randomly samples elements from sets and still preserve weighted Jaccard similarity between any pair of sets:

$$J(w_1, w_2) = \frac{\sum_k \min(f_{w_1,s_k}, f_{w_2,s_k})}{\sum_k \max(f_{w_1,k}, f_{w_2,s_k})} \tag{3}$$

where f_{w,s_k} is the weight of the k^{th} skip-gram for word w.

3.2 Ego-network Construction

After executing the above process each word has a skip-gram vector. These can be used to construct an ego-network (e.g., Figure 1) which can then be utilized to perform clustering analysis and find semantic classes. For each word w, we find 250 nearest neighbors, which will become the nodes of the ego-network. For each pair of terms we compute a similarity score using weighted Jaccard similarity as defined in Eq. (3). If any pair of words, w_i and w_j, have a similarity score higher than a threshold (e.g., 0.05), then a link is created between them. The reasons for building ego-networks, instead of a giant network of the entire vocabulary are

twofold: (1) ego-networks are more convenient for indexing and serving in run-time; and (2) the filtering and sampling of features in the previous step make the feature space sparse enough for efficiently finding nearest neighbors and constructing ego-networks.

Note that if multiple seeds are given as input, we may still construct an "ego-network" by finding the nearest neighbors that have the closest mean distances to the seeds.

3.3 Ego-community Detection

Community detection is conducted by first removing the "ego" (the seed) from the ego-network (see Figure 2 (c)). The seed, naturally, belongs to every community and the edges from the ego may confuse the algorithm. Experimenting with a number of community detection algorithms [16], we find that Louvain [3] balances efficiency with good performance across many networks. Louvain is a hierarchical clustering algorithm that starts by assigning a different community to each node, followed by greedily aggregating communities to optimize the modularity of the network partition until the modularity cannot be further improved, and the "optimal" number of communities is thus identified.

Figure 1 shows the ego-network of *beijing* and its nearest neighbors. After removing *beijing*, the ego/seed, from the network, we can obtain two distinct clusters (without even having to apply community detection; simply checking connected components suffices for this case). By increasing the number of nearest neighbors, we can identify other clusters, such as Chinese provinces, which *beijing* should also belong to. Since our skip-gram extraction strategy produces much sparser a feature set than existing methods, the resulted ego-networks from our method tend to possess better community structure. We provide an empirical evaluation comparing the quality of the ego-network clusters created using different methods in Section 4.

3.4 Fusing Ego-communities and Ontologies

As explained in Section 1, distributional similarity and ontologies should be used jointly to compensate each other's disadvantages. In order to achieve this, for each cluster in the ego-network, we find Wikipedia lists that "match" the content of the cluster by checking if the Jaccard index between the list and the cluster is above a threshold (e.g., $\theta_1 = 0.75$). Then the elements of both the matched lists and the cluster are all pooled together as candidates, and an ensemble model is employed to decide which candidates should remain (Figure 2(d)).

Assuming there are N candidates, then for each candidate w_0, the ensemble model takes into account the mean distance from w_0 to all other candidates, $\bar{d}_{w_0} = \frac{1}{N}\sum_{w_i} f(w_0, w_i)$, where f is one of three different distance/similarity metrics: (1) Hamming distance of Wikipedia list memberships; (2) weighted Jaccard similarity of the skip-gram vectors; and (3) Cosine similarity of the word embedding vectors learned by word2vec [20] on the same corpus. For each metric f, a binary decision is made as to whether w_0 should be removed from the cluster, based on w_0's percentile in the distribution of $\bar{d}_w$ among all w's (the threshold is empirically determined). Then we take the "majority vote" of those three metrics to finally decide whether to remove w_0. The reason of introducing word embedding vectors is to further improve the reliability of the ensemble model. As evidenced in Section 4, using just Wiki lists and skip-gram vectors can achieve comparable performance.

By employing the ensemble model, we aim to improve precision by pulling outliers from the word cluster as well as to improve recall by adding missing elements from the lists. We call this process "fusion" or cluster purification.

Figure 4 illustrates the distributions of $\bar{d}_w$ under three similarity

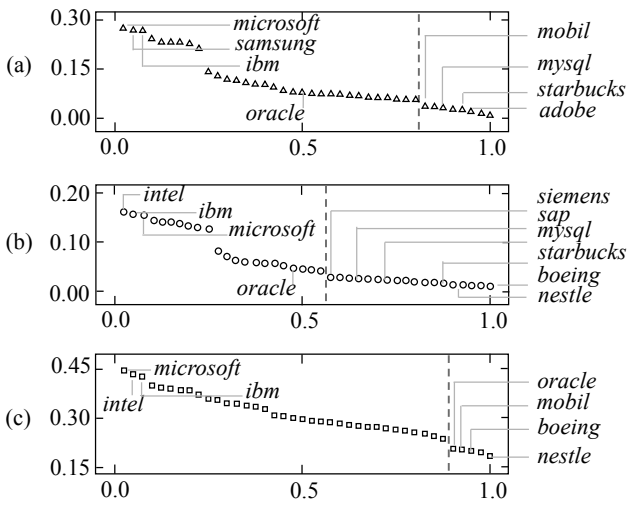

Figure 4: Distribution of $\bar{d}_w$ for candidates in the "company" cluster of *apple* using similarity metrics based on: (a) Wikipedia Lists, (b) skip-gram vectors, (c) word2vec

metrics for the candidate terms corresponding to the "company" cluster of the seed word *apple*. Each dot in the figure corresponds to a candidate term w. The y-axes are the values of $\bar{d}_w$, and the x-axes are the percentiles of w in the distribution of $\bar{d}_w$. In each distribution, from left to right, there can be seen multiple "drops", which separate "good" terms from relatively "worse" terms. The right-most detectable drop in each distribution is marked with a dashed line, which corresponds to empirically determined threshold for distinguishing outliers. While different metrics generally agree on what terms are ranked the highest, they usually differ as to what terms get rejected. In this particular case, several non-IT companies are identified as outliers and removed, including *nestle*, *boeing*, and *starbucks* (*mysql* is also deleted as an outlier). On the contrary, *oracle* is rejected under only one metric, and is thus "saved." Without this step of outlier removal, some of the spurious terms might get a high ranking because of one of the distance metrics and thus may harm the precision of the final result.

After purification, all remaining members in the cluster are ranked by a linear combination of $\bar{d}_w$'s under the three similarity metrics. Therefore, for each cluster, we obtain a ranked list of entities. And overall we obtain a list of ranked lists as result.

3.5 Post-processing

The result of the previous step needs to go through additional post-processing, mainly because of three issues: (1) there may be duplicate clusters, because different clusters may have been matched against the same Wikipedia list and, after purification, there may be pairs of clusters that contain highly similar members; (2) there may exist off-topic clusters, because the matched Wikipedia lists may not capture the topic implied in the seed entities; (3) there may be some clusters that are too small due to data sparsity or the existence of noise in the source data. We adopt the following solutions to solve these problems.

To address the first issue (duplicate clusters), we perform deduplication as follows. For each pair of cluster (c_i, c_j) (note that the elements in the cluster are already ranked by $\bar{d}_w$), define $V_m(c_i, c_j)$ as the number of overlapping elements between the top m elements of c_i and the top m elements of c_j. Then, if any of the conditions

below is satisfied, we drop one of (c_i, c_j) that has a lower coherence score $\Gamma(c)$ than the other:

- $V_5(c_i, c_j) \geqslant 3$;
- $V_{10}(c_i, c_j) \geqslant 5$;
- $V_\infty(c_i, c_j) \geqslant \min(0.3 \times \min(|c_i|, |c_j|), 10)$,

where $|c|$ is the size of the cluster c, and the coherence score of a cluster is defined as the average of pair distance,

$$\Gamma(c) = \frac{1}{|c|^2} \sum_{w_i, w_j \in c} f(w_i, w_j). \quad (4)$$

This deduplication strategy gives high-ranking elements in each cluster higher priorities, which are most important to the perceived quality of the expansion results.

To address the second issue (off-topic clusters), we examine if all of the seed terms are present in the purified cluster.[1] If any seed term is missing, it is a good indication that the matched Wikipedia lists may have led to a semantic drift away from the topics of the seeds, and thus the cluster should be eliminated.

Finally, for the third issue (tiny clusters), we simply remove clusters whose size is smaller than 5. After these post-processing steps, the retained list of ranked lists are returned as the final result.

3.6 Summary

Algorithm 1 summarizes the entire process of serving a set expansion query. Note that the process of creating ego-networks is not included. In the algorithm, θ_1 and θ_2 are parameters used for determining cut-off thresholds. The tuning of these parameters will be discussed in Section 4.4.

Data: $Q = \{q\}$: input seed(s)
$W = \{w\}$: collection of entities (the vocabulary)
$\mathbf{E} = \{E_w\}$: collection of skip-gram-based ego-nets
$L_{\text{wk}} = \{l\}$: collection of Wikipedia lists
$f_{\text{wk}}, f_{\text{ego}}, f_{\text{w2v}}$: three similarity metrics (Section 3.4)
Result: $C = \{c\}$: output clusters
start
$E_Q \leftarrow \cup_{q \in Q} E_q$, obtain "ego-net" of all seeds
$C \leftarrow$ community detection on network E_Q
for *every cluster* $c \in C$ **do**
  $L_c \leftarrow \{l \in L_{\text{wk}}, \text{Jaccard}(l, c) > \theta_1\}$, match lists;
  $W_c \leftarrow \{w \in E_Q\} \cup \{w \in l, l \in L_c\}$, all candidates;
  for $f \in \{f_{wk}, f_{ego}, f_{w2v}\}$ **do**
    for $w \in W_c$ **do**
      $\bar{d}_w = \frac{1}{N} \sum_{w' \in W_c} f(w, w')$;
    end
    $D_f \sim N(\mu, \sigma^2) \leftarrow$ distribution of $\bar{d}_w$ for $w \in W_c$;
    $\xi_f \leftarrow \mu - \sigma\theta_2$, cut-off threshold;
    $\tau_{w,f} \leftarrow \mathbb{1}(\bar{d}_w > \xi_f)$, cut-off decision for $w \in W_c$;
  end
  $c \leftarrow \{w \in c; \text{if} \sum_{f \in \{f_{\text{wk}}, f_{\text{ego}}, f_{\text{w2v}}\}} \tau_{w,f} \geqslant 2\}$;
end
$C \leftarrow$ Dedupe(C), see Section 3.5;
$C \leftarrow$ Drop $c \in C$ if $q \notin c, \forall q \in Q$;
$C \leftarrow$ Drop $c \in C$ if $|c| \leqslant 5$.

Algorithm 1: Entire process of serving a set expansion query.

Multi-seed queries: According to Algorithm 1, it can be seen that multi-seed queries are handled in a very similar fashion to single-seed queries. The differences are: (1) instead of deriving an ego-net of a single seed, we merge the ego-nets of all seeds together into a larger "ego-net"; (2) wherever we need to evaluate the similarity of a candidate entity to the "seed", we replace the "seed" with the centroid of all seeds in the corresponding distance space.

[1] Although the seed terms are removed prior to ego-community detection, they can be added back during list matching.

4. EXPERIMENTS AND RESULTS

We conducted three types of evaluations on EgoSet. First, we use a data-driven approach to create a ground-truth dataset, which includes queries and the corresponding expected classes. This dataset is good for testing how well the system captures the common senses of a query and how precise the returned ranked lists are. Second, we have humans to label all the classes and instances returned by the system given selected queries in order to study its precision. Third, we conduct case studies on selected multiseed multifaceted queries to understand the quality of each individual similarity metric and their interactions.

Data: We retrieved the English Wikipedia 2014 full text (56 million articles, 1.2 billion words). We selected the top 50,000 most frequent n-grams ($n \leqslant 5$) as our core vocabulary (excluding the top-50 as stop words). All text is lower-cased and tokenized using the Stanford Tokenizer.[2]

4.1 Queries and Ground Truth

We created a query set by sampling multifaceted terms from the core vocabulary. To determine the degree to which a term is multifaceted, we look up the term in Wikipedia and examine whether there is a disambiguation page for it and, if so, how many different senses the term possesses. We sampled 50 unigrams with varying degree of multifacetedness. These queries cover a wide variety of topics, such as locations, companies, celebrities, and food. Then we attempted to follow a similar procedure as [38] to manually determine the ground-truth categories for each query. However, we found it to be infeasible to determine a really "comprehensive" or "standard" list of categories per query. Instead, we looked at a minimum of 50 instances of usage in the Wikipedia corpus and Google News[3] search results for each query word, and created a list of *minimally required semantic classes* (MRSC) based on its most popular senses. For example, the MRSCs of *apple* include *Fruit* and *Famous IT Companies*. To avoid bias, we did not look at Wikipedia lists during this process. We found 89 distinct MRSCs, with 5.1 MRSCs per query on average (the median is 4), while allowing different queries to share the same MRSCs.[4]

Note that we did not use existing ground-truth datasets because no existing dataset specifically targets multifaceted entity set expansion, especially for multi-seed multifaceted queries.

Type	Count	Examples
1-seed	50	apple, green, mercury, python, ...
2-seed	50	copper+gold, gnu+squirrel, beaver+elk, ...
3-seed	30	franklin+hamilton+newton, artemis+poseidon+apollo, ...
4-seed	20	orange+lemon+lime+olive, orchid+rose+violet+lavender, ...

Table 2: Summary of queries

Since we also want to test multi-seed multifaceted queries, we looked at the overlap between some pairs or even groups of categories, which enables us to generate multi-seed queries as well.

[2] http://nlp.stanford.edu/software/tokenizer.shtml.
[3] https://news.google.com/
[4] Our dataset is available at http://bit.ly/egoset-data.

Such multi-seed queries are generated by randomly sampling from the common members of all pairs of MRSCs that are overlapping, such as *Colors* and *Fruit*, and then sampling from the overlapping entities. We created 100 multi-seed queries, including 50 two-seed queries, 30 three-seed queries, and 20 four-seed queries. Table 2 shows a summary of the query set with several examples.

4.2 Evaluation of Ego-communities

We compare the quality of the word clusters obtained from the ego-networks based on skip-gram similarity metrics against a state-of-the-art word embedding model, as well as clusters obtained using non-network methods. We use only single-seed queries for this experiment in order to focus on the quality of the ego-communities with the simplest setting. For each query, 150 nearest neighbors are retrieved, and different clustering or community detection methods are used to obtain the clusters, which are then compared to the ground-truth MRSCs. Although MRSCs cannot be considered as having a comprehensive coverage on all possible clusters, they can serve to judge the relative quality of different similarity metrics.

4.2.1 Metrics

For each cluster set $C = c_1, c_2, \ldots, c_K$ and the corresponding MRSC set $R = r_1, r_2, \ldots, r_M$, three metrics from [9] are adopted in evaluating the quality of clusters, including

- Cluster Purity: measures the average accuracy of word assignment in the clusters. With N defined as the total number of nearest neighbors: $\text{purity}(C, R) = \frac{1}{N}\sum_k \max_m |c_k \cap r_m|$
- Normalized Mutual Information (NMI): similar to purity, adding penalization to C if it contains too many clusters.
  $\text{NMI}(C, R) = \frac{I(C,R)}{[H(C)+H(R)]/2}$, where
  $I(C, R) = \sum_k \sum_m \frac{|c_k \cap r_m|}{N}$, and $H(C) = -\sum_k \frac{|c_k|}{N} \log \frac{|c_k|}{N}$.
- Rand Index (RI): defined as the accuracy of predicting whether a pair of words should belong to the same cluster for all of the $\frac{N(N-1)}{2}$ pairs of words.

Note that some of the details are adjusted from [9] to accommodate our experiment setting.

4.2.2 Baselines

The following approaches are compared:

- **word2vec**: the embedding vectors learned using word2vec [20, 21].[5] We used the "skip-gram" model[6], set vector dimension to be 500, window size 8, and used 5 negative samples per update and 5 iterations through the entire Wikipedia corpus. For clustering, k-means with $k = 5$ is used;
- **word2vec-net**: ego-networks built using the above vectors with Cosine similarity;
- **EgoSet-SG**: our skip-gram vectors, used for both k-means and ego-networks.

For both word representations, Louvain [3] is used for community detection on ego-networks. The number of communities is determined by finding the partition with maximum modularity.

4.2.3 Results

Table 3 shows the evaluation results. It can be seen that our proposed similarity metrics consistently outperform word2vec under three different metrics. In addition, for the same set of nearest neighbors, network-based community detection tends to outperform k-means, especially under NMI, although it is not directly comparable as the number of clusters for k-means is predetermined. In short, this result indicates that while both metrics are applied to the same corpus, skip-gram-based methods may better facilitate the recovery of distinct clusters in ego-networks than word embeddings learned by word2vec.

[5] https://code.google.com/p/word2vec/.

[6] Not to be confused with our proposed skip-gram feature set. The word2vec toolkit does not have positional contraints on individual tokens within a skip-gram.

Representation	Clustering	Purity	NMI	RI
word2vec	k-means	0.168	0.260	0.236
word2vec-net	Louvain	0.166	0.394	0.220
EgoSet-SG	k-means	**0.213**	0.404	0.432
	Louvain	0.210	**0.460**	**0.484**

Table 3: Evaluation of nearest neighbor clusters

4.3 End-to-end Evaluation

Our second set of experiments evaluate the performance of the end-to-end pipeline for multifaceted queries. We explain the input/output, metrics, and baselines, before presenting the results.

4.3.1 Input and Output

The input for each test case is a query, which can be either a single seed or multiple seeds (see Table 2). The output is a set of clusters. Each cluster is a ranked list of entities. The entities in the output are drawn from a vocabulary of 50,000 popular entities in Wikipedia. See Section 4.6 for a discussion about long-tail entities.

4.3.2 Metric

Because the output of multifaceted set expansion is a list of ranked lists, we customize a common ranking metric, mean average precision (MAP), for our evaluations. For a set of queries Q, we define mean-MAP (MMAP) as

$$\text{MMAP}_k = \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{M_q}\sum_{m=1}^{M_q}\text{AP}_k(C_{qi^*}; R_{qm}) \qquad (5)$$

where M_q is the number of MRSCs for q; R_{qm} is the m-th MRSC for q; $\text{AP}_k(c, r)$ is the conventional average precision at k given a ranked list c and an unordered ground-truth set r; and C_{qi^*} is the output cluster that best matches R_{qm} among all clusters generated for q (using Jaccard).

This metric not only captures the internal-coherence of the clusters, but also possesses two important characteristics: (1) it captures how comprehensively the result set covers MRSCs. If an output C fails to capture one of the MRSCs, it will be penalized with a low precision score, since for each MRSC a best matching C_{qi^*} will be selected regardless; (2) it allows the output set to include additional topics, which is important because we cannot guarantee that our MRSC sets cover all possible senses of the input seeds.

The precision values for each baseline is calculated as follows: for each query, for each cluster in the ground truth, we find the cluster among output clusters that have the biggest Jaccard similarity. Then we calculate the precision@k values by comparing this cluster against the ground truth cluster. The precision scores are averaged across all ground-truth clusters of the query, then averaged across all queries, resulting in mean average precision@k (MAP@K). Given the event that the output cluster generates fewer clusters than the ground truth (e.g. the baseline SEAL can only generate one cluster), then the same output cluster will be matched towards multiple ground-truth clusters, which inherently penalizes its precision. Since we cannot guarantee that our ground-truth clusters comprehensively cover all senses of a query, we do not penalize any cluster outside the matched clusters.

		1 seed			2 seeds			3 seeds			4 seeds		
		p@5	**p@10**	**p@20**	**p@5**	**p@10**	**p@20**	**p@5**	**p@10**	**p@20**	**p@5**	**p@10**	**p@20**
baseline	**SEAL**	-	-	-	0.208	0.169	0.138	0.368	0.312	0.269	0.393	0.342	0.298
	NeedleSeek	0.432	0.372	0.325	-	-	-	-	-	-	-	-	-
single	**WikiList**	0.369	0.331	0.292	0.313	0.295	0.250	0.401	0.340	0.284	0.379	0.366	0.325
	word2vec	0.360	0.296	0.249	0.317	0.271	0.219	0.389	0.313	0.247	0.431	0.373	0.320
fusion	**EgoSet-SG & WikiList**	0.465	0.413	0.358	0.357	0.316	0.272	0.366	0.325	0.280	0.447	0.374	0.329
	word2vec & WikiList	0.390	0.331	0.289	0.334	0.313	0.222	0.373	0.303	0.240	0.352	0.333	0.308
	EgoSet-ALL	**0.490**	**0.427**	**0.372**	**0.369**	**0.323**	**0.274**	**0.432**	**0.370**	**0.313**	**0.453**	**0.399**	**0.356**

Table 4: End-to-end performance evaluation.

4.3.3 *Baselines*

We perform comparison across the following baselines.

- **SEAL:** Set Expander for Any Language [35, 36]. The published implementation was used with default settings.[7] We only managed to execute multi-seed queries using SEAL, and acquired a single ranked list per query.
- **NeedleSeek:** Web semantic mining prototype by Microsoft Research [38, 30].[8] Semantic maps were retrieved for each single-seed query using an API provided by the authors.
- **WikiList:** Wikipedia lists only. For each input query q, we find its 200 nearest neighbors using Hamming distance of list memberships. Then an ego-network is built using pairwise distance of its neighbors, and community detection (Louvain) is performed to acquire clusters. Cluster members are ranked towards each cluster centroid.
- **word2vec:** Same as **word2vec-net** in Section 4.2.
- **fusion models:** set expansion pipelines with two or three sources. **EgoSet-SG** represents our skip-gram feature set. **EgoSet-ALL** is the method that fuses all three sources (see Section 3.4).

4.3.4 *Results*

Table 4 contains three separate groups of results, including queries of single seeds, double seeds, and three or more seeds. Three major observations can be made: (1) Between single-source methods, **WikiList** consistently outperforms **word2vec**; (2) Fusion-based methods consistently outperform single-source methods; (3) The complete **EgoSet-ALL** method outperforms two-source fusion methods (**EgoSet-SG & WikiList** and **word2vec & WikiList**).

These observations support that our ensemble solution can indeed effectively purify the clusters generated by skip-grams. Note that although the precision numbers reported here may seem lower than those in other state-of-the-art work (e.g., [34]), they may not reflect the actual perceived quality of the results. This is because our evaluation metric, MMAP$_k$, takes into account the coverage of semantic classes, and automatically penalizes situations where there lacks sense coverage for a given query.

4.4 Parameter Analysis

The parameters in our proposed pipeline fall into two categories. The first category is associated with corpus processing and ego-network construction. Here we list a set of parameters that directly influence the quality of ego-networks:

- range of skip-gram lengths (e.g., 3–5 tokens): longer skip-grams may capture more semantics of the entity, but will lead to a sparser feature space;
- maximum degree of a skip-gram in a word-feature bipartite network (e.g., 2500): a larger value means more skip-grams will be considered with the risk of creating more spurious links between words and poorer efficiency;
- number of min-hash dimensions used for sampling skip-grams (e.g, 300): a larger value means better preservation of the "true" word similarity value, at the cost of computation efficiency. This parameter is analogous to the vector dimension in word2vec;
- number of nearest neighbors to index in an ego-network (e.g., 250): a larger value improves the long-tail performance, at the cost of computational efficiency.

The second category of parameters are associated with the process of fusing ego-communities with ontologies (Section 3.4). As can be seen in Algorithm 1, there are two important parameters (θ_1 and θ_2):

- θ_1, minimum cut-off Jaccard similarity for matching ontology lists (e.g., 0.75): a larger value means more strict selection of ontology lists;
- θ_2, a factor for determining the cut-off threshold in purification (see Algorithm 1): a larger value means more aggressive cut-offs in the purification process.

Figure 5 shows the influence of the relevant parameters on the system performance. According to Figure 5(a), the strictness of matching ontology lists plays an important role, and requires proper trade-off between two extremes. A proper value of θ_1 should be set

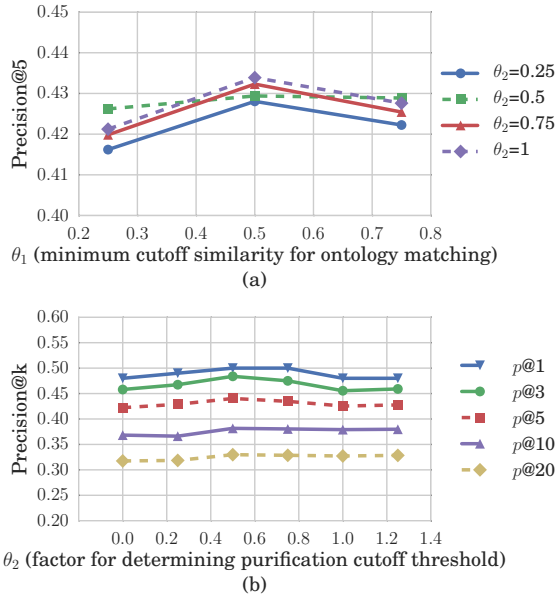

Figure 5: Parameter analysis for fusing ego-network communities with ontologies.

[7] https://github.com/TeamCohen/SEAL.
[8] http://needleseek.msra.cn/

Multi-seed Query	Identified Ego-clusters	Top Skip-gram
brown white	Cluster 1: brown white green williams roberts johnson jackson smith evans jones ... (popular English last names)	kenneth ___ ,
	Cluster 2: white red blue black yellow brown green purple light_blue ... (colors)	colors are ___ and
beijing shanghai	Cluster 1: tianjin guangzhou shanghai shenzhen chongqing beijing wuhan chengdu dalian shenyang ... (major Chinese cities)	in ___ , china .
	Cluster 2: liaoning heilongjiang hebei tianjin shanghai jilin inner_mongolia shanxi beijing hunan ... (Chinese province-level administrative regions)	of ___ , china .
beaver elk	Cluster 1: coyote moose elk cougar beaver bison opossum marten wolverine fisher ... (animals)	deer , ___ ,
	Cluster 2: westmoreland schuylkill fayette crawford beaver elk greene fulton chester ... (counties in Pennsylvania)	in ___ county , pennsylvania

Table 5: Case study of multi-seed multifacted queries

between 0.5 and 0.75. According to Figure 5(b), the quality of the result is not very sensitive to the selection of θ_2. This is because in the actual implementation each type of similarity metrics (i.e., f_{WikiList}, f_{EgoSet}, and f_{word2vec}) also has a flat minimum cutoff threshold, which is independent of θ_2.

Note that word2vec also has a number of tunable parameters, which will not be discussed here.

4.5 Case Study

Table 5 presents three case studies for multi-seed expansion. In all cases, the algorithm succeeds in finding at least two facets of the query. On the right column is a top weighted skip-gram identified for each cluster. This gives us an idea how skip-grams capture the commonality of words and coercing them into a cluster. Note that all phrases (e.g., "light blue" or "inner mongolia") are converted unigrams by concatenating their tokens during pre-processing.

4.6 Discussions

There are several details that are worth discussing:

Number of semantic classes: The number of semantic classes returned by EgoSet is automatically determined. There are two factors: (1) the community detection algorithm (Louvain) returns a partition of the graph with the number of communities that yields the best modularity; (2) in post-processing, if two clusters are too similar (Jaccard index greater than a threshold), they will be merged, which will reduce the number of classes (see Section 3.5).

Scalability: Our proposed pipeline is highly scalable. Although the candidate entities are drawn from a large vocabulary, we only need to keep in memory the ego-networks, i.e., the nodes and links related to the nearest neighbors of the seed words (hence the name "EgoSet"). We experimented with ego-network construction of large n-gram vocabularies: by using Map-Reduce on a moderate cluster (70 cores), the process of extracting features and indexing nearest neighbors can be finished within 5 hours for a vocabulary of 1 million n-grams. After this pre-processing step, during serving time, we only need to look up input seeds in a nearest-neighbor index, without having to iterate through every word in the vocabulary. In addition, similar to [11], most of the computation carried out in the pipeline can be done using sparse matrices, which may further improve the overall efficiency.

Interplay of three similarity metrics: We exploit three different similarity metrics in our system, including skip-gram-based similarity, ontology-based similarity, and word-embedding-based metric. While there is a great amount of commonality in terms of what is captured by the three metrics, they all have their own distinct characteristics: our proposed skip-gram-based similarity metric is great for producing networks with good community structure and performs especially well on geolocation terms (e.g., Figure 1), but it tends to under-perform in some domains, such as movie titles and musical records. In contrast, Wikipedia lists, although quite noisy, provide good coverage on almost every domain, and thus can compensate the skip-gram-based metric to improve recall. Moreover, embedding vectors produced by word2vec has similar characteristics to our skip-gram-based metric, but its performance on multifaceted queries is not as satisfactory. Due to reasons explained in Section 3.1, word2vec-generated embeddings tend to mix non-sibling entities together (e.g., *apple* and *ipad*), which makes it difficult to generate high-quality clusters in word ego-networks. This difference is also shown in Table 3. However, through proper fusing techniques, such as our proposed ensemble model, these three metrics can facilitate each other and lead to improvement of set expansion performance.

Long-tail terms: We intentionally distinguish our work from previous work that focuses on optimizing the performance of long-tail expansion, and thus we do not particularly attempt to address the coverage of long-tail terms in the expansion results (we do care about supporting long-tail queries, which is why we develop the ensemble model). Since there is already plenty of existing work that optimizes the precision and recall given an unambiguous set of seeds, some of which can achieve near-perfect performance (e.g., [24]), we choose to instead focus our study on addressing the ambiguity that exist among many entity set expansion queries. We aim to make the top 5 to 20 ranked entities in each result cluster good enough, so that one can potentially supply the result of our system to downstream expansion algorithms that specialize in long-tail expansions to achieve overall good performance.

Handling n-grams: We treat n-grams as unigrams by concatenating the tokens. This is done by using Mikolov's word2phrase tool published alongside word2vec [20].

Choice and availability of ontologies: The Wikipedia lists used in our pipeline may be replaced with other similar ontologies. When an existing ontology is not readily available, an automatically created ontology (e.g., using techniques of [33]) may be used instead. We do not yet know how well our proposed method works in such scenarios. We leave these topics for future work.

Future work: It would be interesting to conduct a theoretical analysis in order to explain the characteristics of skip-gram-based distributional similarity metric, and how it improves local ego-community structure for multifaceted seeds. Another potential improvement is to integrate more sources of information into the system, including co-occurrence between entities in the Web corpus, Web tables and lists, as well as Wikipedia categories. In addition, our reviewers raised several important points which we could not address within the scope of this paper and would leave

as future work, including: (1) building a system pipeline that handles a mixture of both "single-faceted" and multifaceted queries; (2) evaluating the system against a collection of queries in which multiple terms possess one single common sense but each of the term itself is multifaceted; and (3) developing a simpler or more elegant solution with fewer components or without having to rely on a number of different data sources.

5. RELATED WORK

The problem of completing an entity set given a few seeds has attracted a great amount of research effort. Google Sets was among the earliest work that supported open-domain set expansion [32]. It used proprietary algorithms and is no longer publicly accessible. Ghahramani and Heller [11] formulate the same task as a Bayesian inference problem, where they develop a probabilistic model that tests whether a candidate entity belongs to some unknown cluster that contains the input seeds. Their model exploits the co-occurrence relations between entities in documents, and is shown to have achieved comparable performance to Google Sets. Wang and Cohen [35, 36] propose the *SEAL* system, which submits input seeds as queries to a search engine, and mines the top-ranked webpages using a graph-based ranking model to obtain a ranked list of entities as result. The graphs constructed in SEAL contain heterogeneous sets of nodes, including documents, entities, and wrappers (prefixes and suffixes close to named entities in HTML documents). SEAL effectively leverages the analogous patterns observed around sibling entities in HTML pages, and is shown to outperform Google Sets. In more recent work, Pantel et al. [24] build a highly scalable set expansion pipeline with MapReduce; He and Xin [12] design an iterative similarity aggregation process; and Wang et al. [34] achieves near-perfect precision for completing a set of named entities with not only a few input seeds, but also an explicitly specified label (in natural language) for the intended entity set. Chen et al. [7] improve performance on long-tail term expansions by leveraging "page-specific" extractors built in a supervised fashion. While these approaches perform well for expanding a well-defined semantic class, they are not designed for handling multifaceted seeds. In real usage, the user or upstream applications of an entity set expander may not always be able to provide a clear and unambiguous set of seeds, and the capability of handling multifacetedness in input seeds is important. In comparison to these studies, our work has community detection algorithms embedded in the pipeline with proper post-processing, and can thus correctly handle multifaceted seeds by returning multiple semantic classes as results.

There are also existing techniques that can handle multifaceted seeds for set expansion. These techniques usually involve a batch process that mines the Web to automatically create semantic classes and acquire their corresponding member instances [1, 9, 25, 26, 30, 31, 33, 37]. The core to these techniques include two aspects: (1) mining hyponym (*is-a*) relations between an entity and a class label, for which contextual patterns (e.g., Hearst patterns [13]) are often used; (2) mining sibling relations among entities, which can be achieved by using contextual patterns and distributional similarity on a text corpus [25, 30, 33], or leveraging structural clues in Web tables and lists [9], or a combination of both unstructured and structured data [26, 31]. Compared to these techniques, our pipeline does not stop at the batch process: we introduce effective post-processing techniques that are specially designed to accommodate individual sets of input seeds, especially multifaceted ones.

Perhaps the most relevant work to ours is by Zhang et al. [38]. The authors address multifaceted queries by using topic modeling to refine raw semantic classes for a specific multifaceted query. We show that our method outperforms theirs on multifaceted queries, and can handle multi-seed queries whereas theirs cannot. In addition, Kong and Allan [15] also target multifacetedness of queries but are aimed at finding relevant entities instead of strict siblings of the query. To the best of knowledge, our work is the first to provide a rigorous evaluation against multi-seed multifaceted seeds for entity set expansion.

Recent advances in neural embedding models provide alternative solutions to finding semantic classes, such as [2, 17]. The neural embedding model proposed in these studies are trained with syntactic contexts rather than windows of words. They use a dependency parser to parse the text and contextual words are taken from the proximity of the target word on a parse tree. Syntactic relations between the words are also used to construct the context. As a result, such models can have the effect of grouping words that share similar semantic roles or even semantic classes. Our proposed method may be benefited by the strict sibling entity relations recovered in such dependency-based embedding models.

More generally, our work is also related to word sense disambiguation [5, 4], sense discovery [23], and clustering [18, 19].

6. CONCLUSIONS

We discovered that multifaceted terms possess highly modular ego-community structure in their word ego-networks. To exploit such a nice property for multifaceted set expansion, we designed novel skip-gram features to enhance such modular structure, and proposed the EgoSet pipeline to fuse ego-communities with user-created ontologies, as well as word embeddings, through an ensemble model. As result, we are able to provide accurate entity set expansion for multifaceted seeds through a simple procedure that outperforms state-of-the-art baselines. By only indexing and serving with ego-networks, our proposed pipeline is also efficient enough to support Web-scale set expansion tasks.

Acknowledgments

This work is partially supported by the National Science Foundation under grant numbers IIS-1054199 and CCF-1048168. We thank our reviewers for very helpful comments and suggestions.

References

[1] Ramnath Balasubramanyan, Bhavana Dalvi, and William W Cohen. From topic models to semi-supervised learning: Biasing mixed-membership models to exploit topic-indicative features in entity clustering. In *MLKDD*, 2013.

[2] Mohit Bansal, Kevin Gimpel, and Karen Livescu. Tailoring continuous word representations for dependency parsing. In *ACL*, 2014.

[3] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[4] Jordan L Boyd-Graber, David M Blei, and Xiaojin Zhu. A topic model for word sense disambiguation. In *EMNLP-CoNLL*, 2007.

[5] Junfu Cai, Wee Sun Lee, and Yee Whye Teh. Improving word sense disambiguation using topic features. In *EMNLP-CoNLL*, 2007.

[6] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In *KDD*, 2008.

[7] Zhe Chen, Michael Cafarella, and H. V. Jagadish. Long-tail vocabulary dictionary extraction from the web. In *WSDM*, 2016.

[8] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29, 1990.

[9] Bhavana Bharat Dalvi, William W Cohen, and Jamie Callan. Websets: Extracting sets of entities from the web using unsupervised information extraction. In *WSDM*, 2012.

[10] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, and Ni Lao. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[11] Zoubin Ghahramani and Katherine Heller. Bayesian sets. In *NIPS*, 2005.

[12] Yeye He and Dong Xin. Seisa: set expansion by iterative similarity aggregation. In *WWW*, 2011.

[13] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Computational linguistics*, 1992.

[14] Sergey Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. In *ICDM*, 2010.

[15] Weize Kong and James Allan. Extracting query facets from search results. In *SIGIR*, 2013.

[16] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80, 2009.

[17] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *ACL*, 2014.

[18] Dekang Lin. Automatic retrieval and clustering of similar words. In *ACL*, 1998.

[19] Dekang Lin and Xiaoyun Wu. Phrase clustering for discriminative learning. In *ACL*, 2009.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[22] Marie-Francine Moens, Juanzi Li, and Tat-Seng Chua. Knowledge extraction from wikis/bbs/blogs/news web sites. In *Mining User Generated Content*, pages 129–166. CRC Press, 2015. ISBN 9781466557413.

[23] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *KDD*, 2002.

[24] Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. Web-scale distributional similarity and entity set expansion. In *EMNLP*, 2009.

[25] Marius Pasca. Acquisition of categorized named entities for web search. In *CIKM*, 2004.

[26] Marius Pasca. Open-domain fine-grained class extraction from web search queries. In *EMNLP*, 2013.

[27] John Prager. Question answering using constraint satisfaction. In *ACL*, 2004.

[28] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[29] Abeed Sarker and Graciela Gonzalez. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. *Journal of Biomedical Informatics*, 2014.

[30] Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *Computational Linguistics*, 2010.

[31] Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *EMNLP*, 2008.

[32] Simon Tong and Jeff Dean. System and methods for automatically creating lists, March 25 2008. US Patent 7,350,187.

[33] Benjamin Van Durme and Marius Pasca. Finding cars, goddesses and enzymes: Parametrizable acquisition of labeled instances for open-domain information extraction. In *AAAI*, 2008.

[34] Chi Wang, Kaushik Chakrabarti, Yeye He, Kris Ganjam, Zhimin Chen, and Philip A Bernstein. Concept expansion using web tables. In *WWW*, 2015.

[35] Richard C Wang and William W Cohen. Language-independent set expansion of named entities using the web. In *ICDM*, 2007.

[36] Richard C Wang and William W Cohen. Iterative set expansion of named entities using the web. In *ICDM*, 2008.

[37] Seon Yang and Youngjoong Ko. Extracting comparative entities and predicates from texts using comparative type classification. In *ACL*, 2011.

[38] Huibin Zhang, Mingjie Zhu, Shuming Shi, and Ji-Rong Wen. Employing topic models for pattern-based semantic class discovery. In *ACL*, 2009.