Study of Set Expansion Problem

Arif Hasnat
Department of Computing Science
University of Alberta
Email: hasnat@ualberta.ca

Abstract

In this project, we studied the problem of expanding a small set of entities into a complete set of similar entities. We proposed a modified version of an existing algorithm to overcome some of its limitations. We also presented some experimental analysis using the Web-List dataset to evaluate our proposed algorithm and another existing method for set expansion in terms of Average R-Precision and Mean Average Precision.

1 Introduction

Set expansion refers to the problem of expanding a small set of "seed" entities into a more complete set by discovering other entities that also belongs to the same "concept set". For example, if a given seed set consists of some cities of Alberta like {*Edmonton, Calgary, St. Albert*}, set expansion should return a more complete set of other entities in the same semantic class, such as {*Beaumont, Medicine Hat, Red Deer*} that are also cities of Alberta. Set expansion systems are of practical importance and can benefit various applications, such as knowledge extraction [2], named entity recognition[4], taxonomy induction [6] and web search [1].

Various techniques have been proposed for the set expansion problem over the years using a variety of "contexts" including Web table, Wikipedia list, HTML lists extracted from web pages, Web search query logs, free text corpus etc. Among those, SEISA[3] is an important framework which addressed the set expansion problem using HTML lists extracted from web pages and Web search query logs. Another important framework is SetExpan[5] which used free text corpus for set expansion problem.

In this project, we studied the set expansion problem using Web-List dataset which is a collection of lists of entities extracted from various Web tables. At first, we analyzed some limitations of the SEISA and the SetExpan algorithms. Then, we proposed a modified version of SEISA algorithm (M-SEISA) which improves the run time to a large extent and also overcomes some other limitations. Finally, we evaluated the M-SEISA and the SetExpan algorithms in terms of Average R-Precision and Mean Average Precision using our Web-List dataset to better understand the challenges involved in set expansion problem.

2 Related Work

The problem of expanding an entity set given several seed entities has attracted extensive research efforts due to its practical importance. We will only summarize two important works that we analyzed in this study.

He et al.[3] presented a new general framework based on iterative similarity aggregation and developed a class of algorithms called SEISA (Set Expansion by Iterative Similarity Aggregation) in order to address the set expansion problem. Their Static Thresholding Algorithm starts with computing the relevance score of each entity with the seeds. Then it ranks the entities according to that score and picks top K ranked entities by a thresholding analysis of the score distribution, as the initial estimate. After that, the algorithm iteratively computes new estimate based on the quality score for each entity determined as the weighted combination of its relevance score with the seeds and the coherence score with the previous estimate, and replaces the lowest ranked entity in the previous estimate with the top ranked new entity. Finally, if there is no new entity discovered then the iteration stops, and the latest estimate of expanded entity set is returned. The authors used Otsu's technique for thresholding. Their Dynamic Thresholding Algorithm is similar to the static one except in each iteration, it invokes the thresholding procedure again to find a good value of K, and instead of replacing an entity from previous estimate, it selects top K entities as the new estimate. The convergence is not guaranteed as in the static thresholding algorithm. Therefore, the iteration is stopped after reaching a maximum number. For experimental evaluaton, the authors used general-purpose web data such as the HTML lists extracted from web pages and the web search query logs and evaluated their algorithms in terms of precision and recall.

Shen et al.[5] presented a novel framework for set expansion called SetExpan, based on a denoised context feature selection and a ranking-based unsupervised ensemble method. The SetExpan framework uses two types of context features obtained from the text corpus: skip-grams and coarse-grained types. It assigns weight on the edge between each pair of entity and context feature in the bipartite model using the TF-IDF transformation considering each entity as a "document" and each of its context features as a "term". In context feature selection step, it calculates score for each context features based on its accumulated strength with currently expanded entities and selects top Q features with maximum scores. In entity selection step, the algorithm first generates T subsets of the context features by sampling without replacement. Then, it calculates the rank of each entity based on its similarity score with the currently expanded entities conditioned on each feature subset using weighted Jaccard similarity measure. After that, it obtains the mean

reciprocal rank for each entity by summing up all the reciprocal ranks for that entity based on each feature subset. Finally, all entities with mean reciprocal rank above some threshold are added into the expanded entity set. The algorithm iterates the context feature selection and entity selection steps until the expanded entity set reached the expected output size. The authors presented their experimental results based on three datasets: APR - constructed by crawling all 2015 news articles from AP and Reuters, Wiki - a subset of English Wikipedia and PubMed-CVD – a collection of research paper abstracts about cardiovascular disease retrieved from PubMed and evaluated their algorithm in terms of mean average precision.

3 Methodology

In this section, we will discuss some limitations of the SEISA algorithms in general, compared to the SetExpan algorithm and then propose a better version of the SEISA algorithms to overcome those limitations.

3.1 Discussion of the two algorithms

The SetExpan algorithm resets the context features pool at the beginning of each iteration. Then it selects quality context features again based on the currently expanded set and adds new entities into the expanded set without any replacement in each iteration. Also, the ranking of the entities in the expanded set depends on the insertion order only. Thus, even if the value of expected output size K is larger than the ground truth size of the concept class, then there can be some wrong entities in the output, but they will be ranked lower than the correct entities. Thus, the algorithm can overcome severe semantic drift and entity intrusion problem.

On the contrary, both the SEISA algorithms try to get optimal quality entities by replacing entities with higher quality scores in each iteration and finally, output the entities in order, considering both the similarity with the seeds and other expanded entities. They choose the best output size using Otsu's thresholding technique. If the theshold is larger than the ground truth size, then it can result severe semantic drift problem. The thresholding works well only if the entity scores form bimodal distribution and possess a deep and sharp valley between two peaks. But most of the times, the similarity scores are not ideal for the thresholding technique and it estimates the size of the entity set incorrectly.

In each iteration, the SEISA algorithms calculate similarity score for every entity with respect to expanded entities considering all the features in the dataset or the bipartite graph. Therefore, it suffers huge run time problem for large datasets. On the other hand, the SetExpan chooses a small number of high quality features and calculate similarity scores for entities considering only those selected features. Thus, it takes significantly less time compared to both SEISA algortihms for large datasets.

The following Lemma 1 states a nice property of set expansion:

Lemma 1. *Entities that don't have any features in common with the entities from the seed and the expanded set, can be excluded from consideration without any compromise in accuracy.*

According to the SEISA[3] paper, given the universe of candidate terms U, some seeds $S \subseteq U$, a similarity function $Sim : U \times U \rightarrow [0, 1]$ that measures the similarity of any two terms, the set expansion problem can be stated as identifying the expanded seed set R, $R \subseteq U$ and is of size K, such that the objective function $Q(R, S)$ is maximized, where

$$Q(R, S) = \alpha * S_{rel}(R, S) + (1 - \alpha) * S_{coh}(R)$$

$$S_{rel}(R, S) = \frac{1}{|R| * |S|} * \sum_{r \in R} \sum_{s \in S} Sim(s, r)$$

and

$$S_{coh}(R) = \frac{1}{|R| * |R|} * \sum_{i=1}^{|R|} \sum_{j>i}^{|R|} Sim(r_i, r_j)$$

To prove the Lemma 1, let $Q(R, S)$ and $Q(R', S)$ are the values of the objective function after two subsequent iterations i and i' respectively, and e is an entity such that $e \notin S$, $e \notin R$ and it does not have any common features with the entities from S and R. Hence, the similarity score of entity e with any of the entities from set S and R is zero.

Assume, $e \in R'$ such that $R' = R \cup \{e\}$. Thus, $|R'| = |R| + 1$. That means e is added to the expanded set in i'-th iteration. Hence, $Q(R', S) > Q(R, S)$ as per the problem statement. But,

$$\begin{aligned} Q(R', S) - Q(R, S) = \alpha * (S_{rel}(R', S) - S_{rel}(R, S)) \\ +(1 - \alpha) * (S_{coh}(R') - S_{coh}(R)) \end{aligned}$$

Where,

$$\begin{aligned} S_{rel}(R', S) - S_{rel}(R, S) = (\frac{1}{|R'| * |S|} - \frac{1}{|R| * |S|}) \\ *(\sum_{r' \in R'} \sum_{s \in S} Sim(s, r') - \sum_{r \in R} \sum_{s \in S} Sim(s, r)) \end{aligned}$$

$$\begin{aligned} = (\frac{1}{(|R| + 1) * |S|} - \frac{1}{|R| * |S|}) \\ *(\sum_{r' \in R'} \sum_{s \in S} Sim(s, r') - \sum_{r \in R} \sum_{s \in S} Sim(s, r)) \end{aligned}$$

$\sum_{r' \in R'} \sum_{s \in S} Sim(s, r')$ and $\sum_{r \in R} \sum_{s \in S} Sim(s, r)$ will be equal, as the new entity e in R' has similarity score zero with all the entities in S. Thus,

$$\begin{aligned} S_{rel}(R', S) - S_{rel}(R, S) = (\frac{1}{(|R| + 1) * |S|} - \frac{1}{|R| * |S|}) \\ * \sum_{r \in R} \sum_{s \in S} Sim(s, r) \end{aligned}$$

Since, $\frac{1}{(|R|+1)*|S|} - \frac{1}{|R|*|S|} < 0$,

$$S_{rel}(R', S) - S_{rel}(R, S) < 0$$

Similarly, it can be shown that $S_{coh}(R') - S_{coh}(R) < 0$

Therefore, $Q(R', S) < Q(R, S)$. That contradicts our condition of maximizing objective function. So, e can not be in R' or added to the expanded set.

So, we can say that excluding an entity from consideration that does not have any common features with the entities from the seed and the expanded set, doesn't compromise the accuracy.

Algorithm 1 Modified SEISA Algorithm

M-SEISA$(seeds, graph, K)$:
$f\_terms_0 \leftarrow$ **Filter_Terms**$(seeds, graph)$
for each $term_i$ in f_terms_0 **do**
  $Rel\_Score[i] \leftarrow S_{rel}(term_i, seeds)$
end for
sort $term_i$ by $Rel\_Score[i]$ desc
$threshold \leftarrow$ **Pick_Threshold**$(Rel\_Score[i])$
if $threshold \leq K$ **then**
  $K \leftarrow threshold$
end if
$R_0 \leftarrow$ top K ranked terms by $Rel\_Score[i]$
$iter \leftarrow 1$
while true **do**
  $f\_terms_{iter} \leftarrow$ **Filter_Terms**$(R_{iter-1}, graph)$
  $new\_terms \leftarrow f\_terms_{iter} - f\_terms_{iter-1}$
  for each $term_i$ in new_terms **do**
    $Rel\_Score[i] \leftarrow S_{rel}(term_i, seeds)$
  end for
  for each $term_i$ in f_terms_{iter} **do**
    $Sim\_Score[i] \leftarrow S_{rel}(term_i, R_{iter-1})$
    $g(term_i) \leftarrow \alpha * Rel\_Score[i] + (1 - \alpha) * Sim\_Score[i]$
  end for
  sort $term_i$ by $g(term_i)$ desc
  $R^{'}_{iter} \leftarrow$ top K terms by $g(term_i)$
  sort R_{iter-1} by $g(term_i)$ desc
  if $R^{'}_{iter} \neq R_{iter-1}$ **then**
    let $r \in R^{'}_{iter}$ be the top ranked term not in R_{iter-1}
    let $q \in R_{iter-1}$ be the last ranked term in R_{iter-1}
    $R_{iter} \leftarrow (R_{iter-1} \cup \{r\}) - \{q\}$
  else
    $R_{iter} \leftarrow R_{iter-1}$
    break
  end if
  $iter++$
end while
return R_{iter}

3.2 M-SEISA Algorithm

Based on the Lemma 1, we propose a modified version of SEISA algorithm (M-SEISA) to overcome the run time and thresholding problem of original SEISA algorithms. The M-SEISA algorithm has an extra parameter, expected output size K. Given the currently expanded entities and the bipartite graph, the $Filter\_Terms$ function returns entities that have common features with the expanded entities. Then, instead of considering all the entities in the graph, the algorithm considers only those filtered entities. The $Pick\_Threshold$ function returns the estimated threshold using Otsu's Thresholding technique. The algorithm considers the smallest value between the threshold and the users expected output size K. In each iteration, the algorithm again filters the entities based on currently expanded entities to discover new entities as candidates. Then, it updates the relevance scores for newly discovered entities. After that, it sorts all the entities using the entity scoring function and replace the lowest ranked entity from the previous iteration with newly discovered top ranked entity. Finally, if no new top ranked entity is discovered, then the algorithm returns the sorted entities.

By choosing the smallest value between the estimated threshold and the users expected output size, the algorithm reduces the problem introduced by an incorrectly large threshold value. At the same time, it minimizes the error because of an incorrectly chosen large value of output size by the user.

The algorithm reduces the run time problem by filtering the entities to consider in each iteration. The run time can be reduced further by following some strategies during the implementation. Firstly, in each iteration, only one entity is replaced. As a result, there will be a lot of duplicate entities in the expanded entity set and the filtered entity set in subsequent iterations. So, the raw similarity scores such as Jaccard or Cosine between entities from earlier iterations can be temporarily stored and reused. Secondly, the raw similarity scores can be stored globally in the run time memory. Thus, the run time will be less for similar type of queries in later executions. Thirdly, all the raw similarity scores can be precalculated and stored in a file. Hence, the program will load the scores in its run time memory at the beginning of the execution. The second and third options require a lot of run time memory depending on the dataset size. Therefore, they are in-efficient in terms of memory, but reduce the run time in a great extent. The first strategy can be adopted without much memory requirement and it also reduces the run time a lot, but less than the other two options.

Concept Class	Ground Truth Size
States of India	28
Cities of Alberta	19
Public Universities in Ontario	19
Prime Ministers of Canada	23
Countries of North America	22

Table 1: Concept Classes and Ground Truth

4 Experiments

4.1 Experimental Setup

4.1.1 Dataset

We experimented with the Web-List dataset. It consists of a collection of lists of entities extracted from various web tables. The size of the dataset is 286 MB, with 1,707,913 lists containing 6,312,424 entities, and the resulting bipartite graph has 19,139,143 edges.

4.1.2 Concept Class and Query Construction

A query is a set of seed entities of the same semantic class in a dataset, serving as the input for each system to expand the set. We first selected five concept classes that cover a wide variety of topics, including countries, states, universities, cities as well as prime ministers, and have different degrees of difficulty for set expansion. The "ground truths" or the set of entities that is considered to belong to these classes are well-known and easily verifiable. We determined the ground truth for each concept class using the information available in Wikipedia and based on the availability of each entity in our dataset. Table 1 provides information about the concept classes and ground truths.

Query No.	No. of Seed Entities	Frequent Entities	Less Frequent
1	2	1	1
2	2	2	0
3	2	0	2
4	3	2	1
5	3	1	2
6	3	3	0
7	4	2	2
8	5	5	0

Table 2: Query Description

Concept Class	Truth Size	Threshold	
		Best	Worst
States of India	28	32	37
Cities of Alberta	19	280	14717
Public Universities in Ontario	19	52	1772
Prime Ministers of Canada	23	769	4371
Countries of North America	22	202	228

Table 3: Summary of Threshold Values by Otsu's Algorithm

To construct queries for each concept class, we sorted the entities in the ground truth by their frequency in the dataset in descending order. Then, we constructed eight queries for each concept class, varying the frequency of seed entities and the size of the query. Table 2 provides the information about the queries for each class.

4.1.3 Algorithms

We considered the SEISA[3] and the SetExpan[5] algorithms for our experiments.

We implemented both the Static and the Dynamic Thresholding algorithms with Otsu's Thresholding technique according to the SEISA[3] paper. We set the time limit for running each query to one hour and did not get any results for the queries within that time limit. Furthermore, Otsu's Thresholding returns very large threshold values compared to the ground truth for most of the queries. Table 3 shows the best and the worst threshold values returned by Otsu's algorithm for each concept class. It returns reasonable value for only "States of India" class as the similarity scores of entities of this class are well separated from all other entities. Thus, using the threshold values can produce severe semantic drift and entity intrusion problem for our concept classes. For these reasons, we conducted experiments with our proposed modified SEISA algorithm (M-SEISA) instead of the original one. To get the best results, we used parameter values- $\alpha = 0.5$ as suggested in the SEISA paper, Jaccard Similarity as the similarity metric and K = ground truth size.

The code for the SetExpan[5] algortihm is publicly available from the authors. The algorithm is proposed for corpus-based set expansion. Thus, it uses skip-grams and coarse grained types extracted from the free-text corpus as context features. We modified the implementation to use list identifiers in our dataset as features instead of those text patterns. To get the best results, we used parameter values- $\alpha = 0.6$, $Q = 150$, $T = 60$, $r = 10$ as suggested in the paper and K = ground truth size.

Algorithm	Mean Average R-Precision	Mean MAP
M-SEISA	0.7001	0.9203
SetExpan	**0.7870**	**0.9484**

Table 4: Overall performance of the algorithms

4.1.4 Evaluation Metrics

Since, the ground truths are known for all the concept classes, we calculated R-Precision to evaluate each query for a given concept class. For each query, we also calculated the Average R-Precision over all the concept classes as $\frac{1}{C}\sum_{c=1}^{C} RP_c$, where C is the number of concept classes, RP_c is the R-Precision value of the query for c-th concept class. For each concept class, we calculated the Average R-Precision over all queries as $\frac{1}{S}\sum_{s=1}^{S}\frac{1}{N_s}\sum_{i=1}^{N_s} RP_{si}$, where S is the number of different query sizes, N_s is the number of queries of size s and RP_{si} is the R-Precision value for i-th query of size s. Finally, we calculated the overall Mean Average R-Precision for each algorithm by calculating the average of Average R-Precision values of the concept classes.

Moreover, since, the output of each query is a ranked list of entities, we calculated Average Precision (AP) at K to evaluate each query for a given concept class, where K is the ground truth size for that class. For each query, we also calculated the Mean Average Precision (MAP) over all the concept classes as $\frac{1}{C}\sum_{c=1}^{C} AP_c$, where C is the number of concept classes, AP_c is the average precision value of the query for c-th concept class. For each concept class, we calculated the MAP over all queries as $\frac{1}{S}\sum_{s=1}^{S}\frac{1}{N_s}\sum_{i=1}^{N_s} AP_{si}$, where S is the number of different query sizes, N_s is the number of queries of size s and AP_{si} is the Average Precision value for i-th query of size s. Finally, we calculated the overall mean MAP for each algorithm by calculating the average of MAP values of the concept classes.

4.2 Experimental Results

Table 4 shows the overall performance for M-SEISA and SetExpan. SetExpan performed better than M-SEISA in terms of both Mean Average R-Precision and Mean MAP. But, the results are comparable to each other. In the following sections, we will discuss the performances in detail.

4.2.1 Analysis on Concept Classes

Figure 1 and 2 show the performances of the algorithms for each concept class. The Mean Average R-Precision and the Mean Average Precision of both algorithms are almost perfect for "States of India" class. Because, the entities of that concept class are very similar to each other and also well separated from all other entities in the dataset. The performance is worst for "Cities of Alberta" class. Because, the cities and the towns of Alberta are not well separated and both algorithms incorrectly output some towns instead of cities. That is also the case for "Countries of North America" class, as the algorithms fail to distinguish between North American countries and all the other countries in the world. Overall, although both algorithms perform poorly when the situation is not ideal, but, the SetExpan algorithm performs much better than M-SEISA in terms of Average R-Precision.

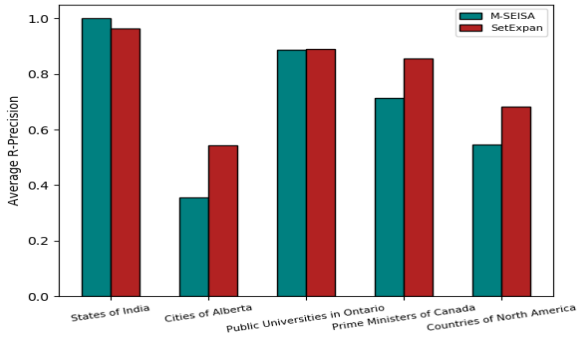

Figure 1: Average R-Precision for concept classes over all queries

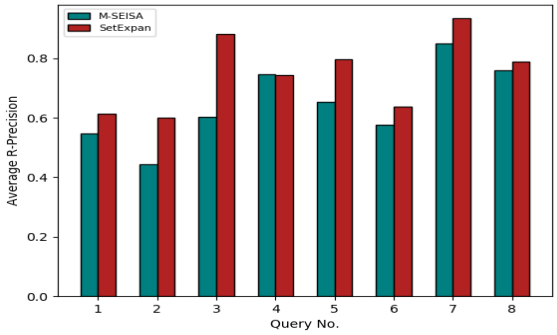

Figure 3: Average R-Precision of queries over concept classes

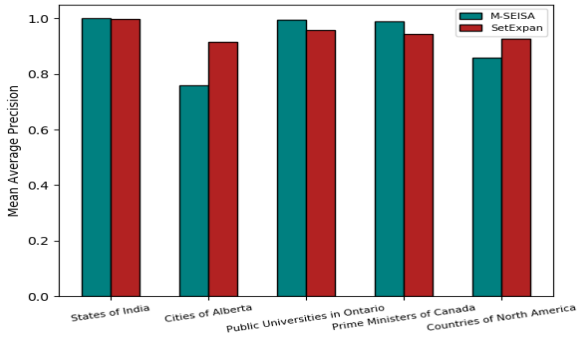

Figure 2: Mean Average Precision for concept classes over all queries

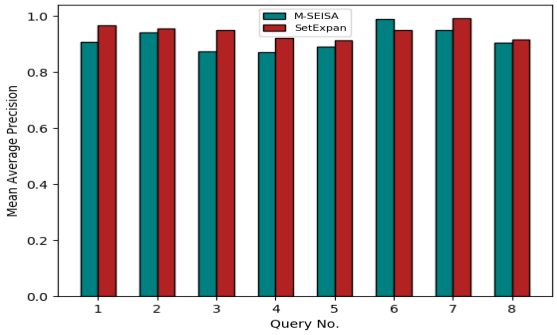

Figure 4: Mean Average Precision of queries over concept classes

4.2.2 Performance per Queries

Figure 3 and 4 summarize the performances of each query over all the concept classes. The Mean Average Precision scores are almost same for all the queries. But, the differences in Average R-Precision are quite significant. For queries 1, 2 and 3 with seed size two, both algorithms performed better with more "less frequent" entities. The performance is best for query 3 which consists of both "less frequent" entities. That is also true for queries 4, 5 and 6 with seed size three. The performance is worst for query 6 which consists of all three "frequent" entities. Even the performance for query 8 is worse than query 7 though it contains one more seed than query 7. Overall, both algorithms perform far better with more "less frequent" entities in the seed set. The reason is that the "less frequent" entities appear in a small number of lists or features and thus, less ambiguously define the concept than the "frequent" entities.

4.2.3 Performance by Seed Size

Figure 5 and 6 show the performances of the algorithms depending on the various seed sizes. The Mean Average Precision scores are almost same for all the seed sizes. But, the Average R-Precision scores are quite different. M-SEISA performs relatively poor for seed size of two compared to SetExpan. The difference between seed size four and five is a bit misleading. Since, there are only one test query for both the seed sizes and seed size five consists of all the "frequent" entities. In general, we can say that both algorithms perform better for seed sizes of at least four and SEISA performs relatively poor for smaller seed sizes than SetExpan.

4.2.4 Run Time Analysis

Table 5 shows the average run time over all queries of each concept class. For M-SEISA, we adopted the temporarily store and reuse of similarity scores strategy discussed in Section 3.2. Both algorithms require less than 3 iterations for most of the queries. The run time of M-SEISA is directly related to the number of unique lists or features containing the ground truth entities of a class. Thus, the run times of M-SEISA for "Cities of Alberta" and "Countries of North America" are higher than the other classes as the number of unique lists are higher for those classes. On the contrary, the differences in run time for SetExpan is small, as it considers a fixed number of high quality features only in each iteration. Moreover, all the TF-IDF scores between each entity and feature pairs are pre-calculated and stored for SetExpan. If we pre-calculate and store all the similarity scores between entities as well for M-SEISA, then the run time can be futher reduced to less than 5 seconds for all the concept classes, which is much better than SetExpan.

5 Conclusion and Future Work

In this project, we proposed a modified version of the SEISA algorithms (M-SEISA) which improves the run time to a large extent and overcomes some other limitations. Then, we evaluated our proposed M-SEISA algorithm and the SetExpan algorithm in terms of Average R-Precision and Mean Average Precision using our Web-List dataset. Both algorithms show relatively poor performances when the seeds are frequent, the seed size is less than four and the ground truth

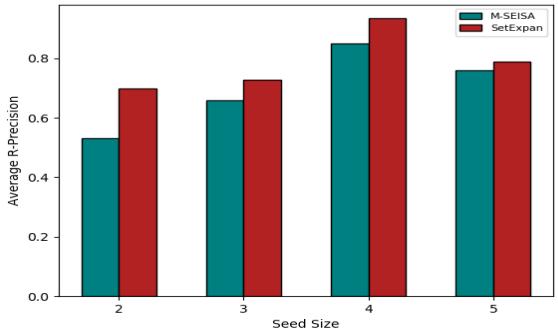

Figure 5: Average R-Precision of seed sizes over concept classes

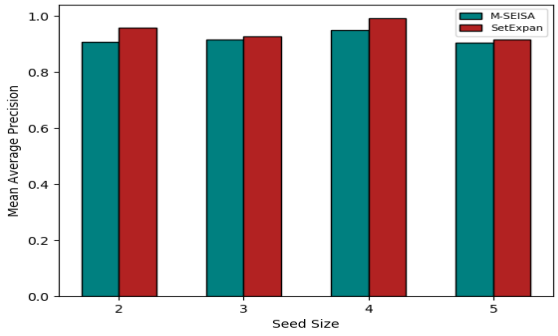

Figure 6: Mean Average Precision of seed sizes over concept classes

entities are not well separated from the other entities in the dataset in terms of entity to entity similarity. Overall, SetExpan performs much better in terms of Average R-Precision and slightly better in terms of Mean Average Precision than M-SEISA but needs more run time.

As future work, to overcome the existing challenges in set expansion, firstly, the basic $Filter\_Terms$ function of M-SEISA can be modified to further filter entities to get a small number of quality entities only. Secondly, it would be interesting to further study the relationships among entities within a concept and outside the concept to propose a better entity ranking function. Finally, instead of depending on user's input of expected size K, a good technique to automatically choose the best size for expanded entity set can be proposed.

Concept Class	**# of Lists**	**Average Run Time(s)**	
		M-SEISA	**SetExpan**
States of India	949	**6.778**	57.180
Cities of Alberta	2119	**40.031**	63.788
Public Universities in Ontario	316	**0.984**	97.643
Prime Ministers of Canada	730	**3.442**	148.329
Countries of North America	43415	1491.201	**49.476**

Table 5: Average Run Time for concept classes

References

[1] Zhe Chen, Michael Cafarella, and H. V. Jagadish. Long-tail vocabulary dictionary extraction from the web. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, WSDM '16, page 625–634, New York, NY, USA, 2016. Association for Computing Machinery.

[2] Sonal Gupta and Christopher Manning. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 98–108, Ann Arbor, Michigan, June 2014. Association for Computational Linguistics.

[3] Yeye He and Dong Xin. Seisa: Set expansion by iterative similarity aggregation. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, page 427–436, New York, NY, USA, 2011. Association for Computing Machinery.

[4] Burr Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, JNLPBA '04, page 104–107, USA, 2004. Association for Computational Linguistics.

[5] Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18-22, 2017, Proceedings, Part I*, volume 10534 of *Lecture Notes in Computer Science*, pages 288–304. Springer, 2017.

[6] Paola Velardi, Stefano Faralli, and Roberto Navigli. OntoLearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707, 2013.