Term Set Expansion based NLP Architect by Intel AI Lab

Jonathan Mamou, Oren Pereg, Moshe Wasserblat,
Alon Eirew, Yael Green, Shira Guskin, Peter Izsak, Daniel Korat
Intel AI Lab, Israel
`firstname.lastname@intel.com`

Abstract

We present SetExpander, the term set expansion system based NLP Architect by Intel AI Lab. SetExpander is a corpus-based system for expanding a seed set of terms into a more complete set of terms that belong to the same semantic class. It implements an iterative end-to-end workflow and enables users to easily select a seed set of terms, expand it, view the expanded set, validate it, re-expand the validated set and store it, thus simplifying the extraction of domain-specific fine-grained semantic classes. SetExpander has been used successfully in real-life use cases including integration into an automated recruitment system and an issues and defects resolution system.[1]

1 Introduction

Term set expansion is the task of expanding a given partial set of terms into a more complete set of terms that belong to the same semantic class. For example, given a seed of personal assistant application terms like 'Siri' and 'Cortana', the expanded set is expected to include additional terms such as 'Amazon Echo' and 'Google Now'. Many NLP-based information extraction applications, such as relation extraction or document matching, require the extraction of terms belonging to fine-grained semantic classes as a basic building block. A practical approach to extracting such terms is to apply a term set expansion system. The input seed set for such systems may contain as few as 2 to 10 terms, which is practical to obtain. SetExpander uses a corpus-based approach based on the *distributional similarity hypothesis* (Harris, 1954), stating that semantically similar words appear in similar contexts. Linear bag-of-words con-

text is widely used to compute semantic similarity. However, it typically captures more *topical* and less *functional* similarity, while for the purpose of set expansion, we need to capture more functional and less topical similarity.[2] For example, given a seed term like the programming language 'Python', we would like the expanded set to include other programming languages with similar characteristics, but we would not like it to include terms like 'bytecode' or 'high-level programming language' despite these terms being semantically related to 'Python' in linear bag-of-words contexts. Moreover, for the purpose of set expansion, a seed set contains more than one term and the terms of the expanded set are expected to be as functionally similar to *all* the terms of the seed set as possible. For example, 'orange' is functionally similar to 'red' (color) and to 'apple' (fruit), but if the seed set contains both 'orange' and 'yellow' then only 'red' should be part of the expanded set. However, we do not want to capture only the term sense; we also wish to capture the granularity within a category. For example, 'orange' is functionally similar to both 'apple' and 'lemon'; however, if the seed set contains 'orange' and 'banana' (fruits), the expanded set is expected to contain both 'apple' and 'lemon'; but if the seed set is 'orange' and 'grapefruit' (citrus fruits), then the expanded set is expected to contain 'lemon' but not 'apple'.

While term set expansion has received attention from both industry and academia, there are only a handful of available implementations. Relative to prior work, the contribution of this paper is twofold. First, it presents an iterative end-to-end

---

[1]A video demo of SetExpander is available at https://drive.google.com/open?id=1e545bB87Autsch36DjnJHmq3HWfSd1Rv (some images were blurred for privacy reasons).

[2]We use the terminology introduced by (Turney, 2012): the *topic* of a term is characterized by the nouns that occur in its neighborhood while the *function* of a term is characterized by the syntactic context that relates it to the verbs that occur in its neighborhood.

workflow that enables users to select an input corpus, train multiple embedding models and combine them; after which the user can easily select a seed set of terms, expand it, view the expanded set, validate it, iteratively re-expand the validated set and store it. Second, it describes the SetExpander application that provides these abilities. SetExpander is based on a novel corpus-based set expansion algorithm. This algorithm combines multi-context term embeddings using a neural classifier in order to capture different aspects of semantic similarity and to make the system more robust across different semantic classes and different domains. The algorithm is briefly described in Section 3. Our system has been used successfully in several real-life use cases. One of them is an automated recruitment system that matches job descriptions with job-applicant resumes. Another use case involves enhancing a software development process by detecting and reducing the amount of duplicate defects in a validation system. Section 5 includes a detailed description of both use cases. The system is distributed as open source software under the Apache license as part of NLP Architect by Intel AI Lab.[3]

2 Related Work

State-of-the-art set expansion techniques return the k nearest neighbors around the seed terms as the expanded set, where terms are represented by their co-occurrence or embedding vectors in a training corpus. Vectors are constructed according to different context types, such as linear bag-of-words context (Pantel et al., 2009; Shi et al., 2010; Rong et al., 2016; Zaheer et al., 2017; Gyllensten and Sahlgren, 2018), explicit lists (Roark and Charniak, 1998; Sarmento et al., 2007; He and Xin, 2011), coordinational patterns (Sarmento et al., 2007) and unary patterns (Rong et al., 2016; Shen et al., 2017). SetExpander looks at additional context types that can capture functional semantic similarities and combines context type embeddings using a neural classifier.

Google Sets, now discontinued, was one of the earliest web applications for term set expansion. It used methods like latent semantic indexing to pre-compute lists of similar words from the web. Word Grab Bag[4] is another web application

based on a method that builds lists dynamically using word2vec embeddings based on linear bag-of-word contexts, but their algorithm is not publicly described. Later, Wang and Cohen (2007) proposed the SEAL (Set Expander for Any Language) system which automatically finds semi-structured web pages that contain 'lists of' items, and then aggregates these lists so that the most promising items are ranked higher. In our paper, we describe an iterative end-to-end system, including model training and using additional context types.

Pantel et al. (2009) propose a highly scalable algorithm, implemented in the MapReduce framework, for computing semantic similarity, where terms are represented by large and sparse co-occurrence vectors. SetExpander ensures scalability by representing terms with small and dense embeddings vectors.

The current paper extends (Mamou et al., 2018) paper.

3 Algorithm Overview

3.1 Term Extraction and Representation

Our approach is based on representing any term of an (unlabeled) training corpus by its word embeddings in order to estimate the similarity between seed terms and candidate expansion terms.

Noun phrases provide good approximation for candidate terms and are extracted in our system using a noun phrase chunker.[5] Term variations, such as aliases, acronyms and synonyms, which refer to the same entity, are grouped together.[6] Next, we use term groups as input units for embedding training; this enables obtaining more contextual information compared to using individual terms, thus enhancing embedding model robustness. In the remainder of this paper, by language abuse, *term* will be used instead of term group.

While word2vec originally uses a linear bag-of-words context around the *focus term* to learn the term embeddings, the literature describes other possible context types. For each focus term, we extract *context units* of different types, as follows (see examples in Table 1).

Context type	Example sentence	Focus term	Context units
Linear $win = 5$	*Siri uses voice queries and a natural language user interface.*	*Siri*	*uses, voice queries, natural language user interface*
List	*Experience in Image processing, Signal processing, Computer Vision.*	*Image processing*	*Signal processing, Computer Vision*
Dep	*Turing studied as an undergraduate ... at King's College, Cambridge.*	*studied*	*(Turing/nsubj), (undergraduate/- prep_as), (King's College/prep_at)*
SP	*Apple and Orange juice drink ...*	*Apple*	*Orange*
UP	*In the U.S. state of Alaska ...*	*Alaska*	*U.S. state of __*

Table 1: Examples of extracted contexts per context type.

Linear Bag-of-Words Context. This context type is defined by neighboring context units within a fixed length window of context units, denoted by win, around the focus term. Both terms and other words can be context units. One of its implementations is word2vec (Mikolov et al., 2013), widely used for NLP tasks including set expansion.

Explicit Lists. Context units consist of terms co-occurring with the focus term in textual lists such as comma separated lists and bullet lists (Roark and Charniak, 1998).

Syntactic Dependency Context (Dep). This context type is defined by the syntactic dependency relations in which the focus term participates (Levy and Goldberg, 2014). Context units consist of terms or other words, along with the type and the direction of the dependency relation. This context type has not been used for set expansion in prior work. However, Levy and Goldberg (2014) showed that this context yields more functional similarities of a co-hyponym nature than is yielded by linear bag-of-words context, which suggests its relevance for set expansion.

Symmetric Patterns (SP). Context units consist of terms co-occurring with the focus term in symmetric patterns (Davidov and Rappoport, 2006). For example, the symmetric pattern 'X rather than Y' captures a certain semantic relatedness between the terms X and Y. This context type generalizes coordinational patterns ('X and Y', 'X or Y'), which have been used for set expansion.

Unary Patterns (UP). This context type is defined by the unary patterns in which the focus term occurs (Rong et al., 2016). Context units consist of n-grams of terms and other words, in which the

focus term occurs; '__' denotes the placeholder of the focus term in Table 1.[7]

We found that indeed in different domains and for different semantic classes, better similarities are found using different context types. The different contexts thus complement each other by capturing different types of semantic relations. For example, explicit list contexts worked well for the automated recruitment system use case, while unary patterns contexts worked well for the issues and defects resolution use case (discussed in Section 5). Moreover, explicit lists, syntactic dependency, symmetric patterns and unary patterns context types tend to capture functional rather than topical semantic similarities. We train a separate term embedding model for each of the five context types and thus, for each term, we obtain five different representations.

Terms are represented by their linear bag-of-words window context embeddings using the word2vec toolkit [8] and by arbitrary context embeddings using the generic word2vecf toolkit.[9]. For each focus term in the corpus, <focus term, context unit> pairs are extracted from the corpus and are then fed to the embeddings training algorithm. Concerning linear bag-of-words context type, some hyperparameters of the term embeddings training can be tuned to optimize the set expansion task; in particular, a smaller window size

---

[7] Following Rong et al. (2016), we extract six n-grams per focus term. Given a sentence fragment c_{-3} c_{-2} c_{-1} t c_1 c_2 c_3 where t is the focus term and c_i are the context units, the following n-grams are extracted: $(c_{-3}\ c_{-2}\ c_{-1}\ t\ c_1)$, $(c_{-2}\ c_{-1}\ t\ c_1\ c_2)$, $(c_{-2}\ c_{-1}\ t\ c_1)$, $(c_{-1}\ t\ c_1\ c_2\ c_3)$, $(c_{-1}\ t\ c_1\ c_2)$, $(c_{-1}\ t\ c_1)$.

[8] http://code.google.com/archive/p/word2vec

[9] http://bitbucket.org/yoavgo/word2vecf

seems to capture functional rather than topical semantic similarities (Levy and Goldberg, 2014).

3.2 Multi-Context Term Similarity

To make set expansion more robust, we aim to combine multi-context embeddings. Following (Berant et al., 2012), who train a Support Vector Machine (SVM) to combine different similarity score features, we train a Multilayer Perceptron (MLP) classifier that predicts whether a candidate term should be part of the expanded set based on ten similarity scores (considered as input features) obtained by the five different context types and two different similarity-scoring methods. The two similarity scores are estimated by the cosine similarity between the centroid of the seed terms and each candidate term, and by the average pairwise cosine similarity between each seed term and each candidate term; both methods ensure that the candidate term is similar to all the seed terms. MLP is trained on a labeled training set of seed terms and candidate terms.

3.3 Implementation and Evaluation

NLP Architect by Intel AI Lab [10] has been used for noun phrase chunking, dependency parsing and term embeddings model training. The performance of the algorithm was first evaluated by the Mean Average Precision at different top n values (MAP@n). MAP@10, MAP@20 and MAP@50 on an English Wikipedia based dataset [11] are respectively 0.83, 0.74 and 0.63. These figures indicate the quite useful performance of the algorithm, which was further assessed by the use cases described in Section 5.

4 System Workflow and Application

This section describes the iterative end-to-end workflow of SetExpander, as depicted in Figure 1.

Steps 1 & 2: Selecting an Input Corpus and Training Models. The first step of the flow is to **select an input corpus**, performed by selecting *Open* (not shown) from the *File* menu (see the red rectangle in Figure 2). The second step of the flow is to **train the models** based on the selected corpus, performed by selecting *Train Models* (not

shown) from the *Tools* menu (see the yellow rectangle in Figure 2). The "train models" step extracts term groups from the corpus, trains the combined term groups embedding models (Section 3.1) and the MLP classifier that predicts whether a candidate term should be part of the expanded set (Section 3.2).

Steps 3 & 4: Selecting and Expanding a Seed Set. Figure 2 also shows the seed set selection and expansion user interface. Each row in the displayed table corresponds to a different term group. The top 5000 term group names are displayed under the *Expression* column, sorted by their TF-IDF based importance score. Term groups that include more than one term are highlighted in bold, and are represented in the display, by the term with the highest importance score among the terms of the group. Hovering over such a group opens a drop-down list that displays all the terms within the group. The user can choose to exclude specific terms from the group if their semantic meaning does not align with that of the group. The *Filter* text box is used for searching for specific term groups. Upon selecting (clicking) a term group, the context view on the right hand side of Figure 2 (blurred) displays text snippets from the input corpus that include terms that are part of the selected term group (highlighted in green). The context view enables the user to verify the semantic meaning of terms in various contexts in the topical domain.

The user can **create a seed set** assembled from specific term groups by checking their *Expand* checkbox (see the red circle in Figure 2). The user can set a name for the semantic category of the seed set. This name will be used for displaying and storing the seed set and the resulting expanded set of terms. The category name can be selected from a predefined list of category names or added as a new custom category name (see the drop down list in Figure 2). Once the seed set is assembled, the user can **expand the seed set** by selecting the *Expand* option (not shown) in the *Tools* menu.

Steps 5 & 6: Edit, Validate and Re-expand. Figure 3 shows the output of the expansion process. The *Certainty* score represents the relatedness of each expanded term group to the seed set, as determined by the MLP classifier (Section 3.2). The Certainty scores of term groups that were manually selected as part of the seed set are
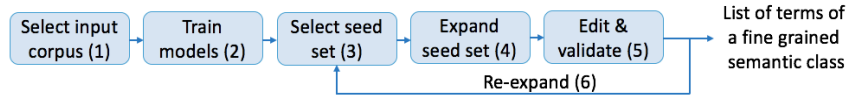

Figure 1: SetExpander end-to-end workflow.
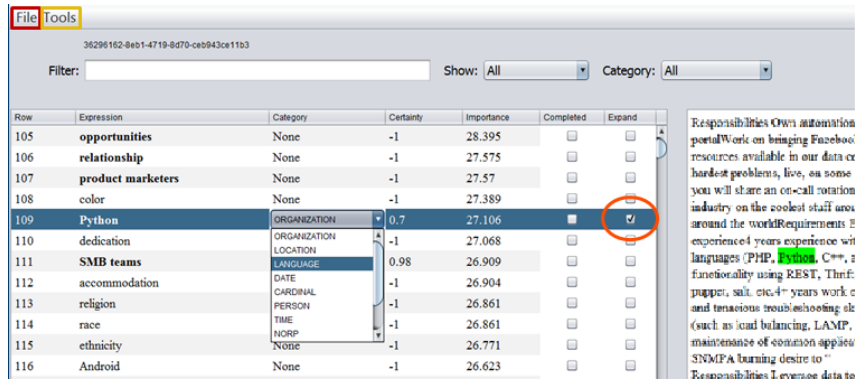
Figure 2: SetExpander user interface for seed selection and expansion.

Row	Expression	Category	Certainty
0	java	LANGUAGE	1
1	Python	LANGUAGE	1
2	JavaScript	LANGUAGE	0.92
3	SQL	LANGUAGE	0.88
4	perl	LANGUAGE	0.84
5	PHP	LANGUAGE	0.82
6	C++	LANGUAGE	0.82
7	TCL	LANGUAGE	0.81
8	ruby	LANGUAGE	0.79
9	visual basic	LANGUAGE	0.77

Figure 3: SetExpander user interface for expansion results output. Seed terms are 'java' and 'python'.

set to 1. The user can **validate** each expanded item by checking the *Completed* checkbox. The validated list can then be saved and later used as a fine-grained semantic class input to external applications. Following validation, the user can perform **re-expansion** by creating a new seed set based on the validated expanded terms and the original seed set terms.

5 Field Use Cases

This section describes two use cases in which SetExpander has been successfully used.

Automated Recruitment System. Human matching of applicant resumes to open positions in organizations is time-consuming and costly. Automated recruitment systems enable recruiters to speed up and refine this process. The recruiter provides an open position description and then the system scans the organizations resume repository

searching for the best matches. One of the main features that affect the matching is the skills list, for example, a good match between an applicant and an open position regarding specific programming skills or experience using specific tools is significant for the overall matching. However, manual generation and maintenance of comprehensive and updated skills lists is tedious and difficult to scale. SetExpander was integrated into such a recruitment system. Recruiters used the system's user interface (Figures 2 & 3) to generate fine-grained skills lists based on small seed sets for eighteen engineering job position categories. We evaluated the recruitment system use case for different skill classes. The system achieved a precision of 94.5%, 98.0% and 70.5% at the top 100 applicants, for the job position categories of Software Machine Learning Engineer, Firmware Engineer and ADAS Senior Software Engineer, respectively.

Issues and Defects Resolution. Quick identification of duplicate defects is critical for efficient software development. The aim of automated issues and defects resolution systems is to find duplicates in large repositories of millions of software defects used by dozens of development teams. This task is challenging because the same defect may have different title names and different textual descriptions. The legacy solution relied on manually constructed lists of tens of thousands of terms, which were built over several weeks. Our term set expansion application was integrated into

23

such a system and was used for generating domain specific semantic categories such as product names, process names, technical terms, etc. The integrated system enhanced the duplicate defects detection precision by more than 10% and sped-up the term list generation process from several weeks to hours.

6 Conclusion

We presented SetExpander, a corpus-based system for set expansion which enables users to select a seed set of terms, expand it, validate it, re-expand the validated set and store it. The expanded sets can then be used as a domain specific semantic classes for downstream applications. Our system was used in several real-world use cases, among them, an automated recruitment system and an issues and defects resolution system.

References

J. Berant, I. Dagan, and J. Goldberger. 2012. Learning entailment relations by global graph structure optimization. *Computational Linguistics*, 38:73–111.

Dmitry Davidov and Ari Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 297–304. Association for Computational Linguistics.

Amaru Cuba Gyllensten and Magnus Sahlgren. 2018. Distributional term set expansion. *arXiv preprint arXiv:1802.05014*.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Yeye He and Dong Xin. 2011. Seisa: set expansion by iterative similarity aggregation. In *Proceedings of the 20th international conference on World wide web*, pages 427–436. ACM.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 302–308.

Jonathan Mamou, Oren Pereg, Moshe Wasserblat, Ido Dagan, Yoav Goldberg, Alon Eirew, Yael Green, Shira Guskin, Peter Izsak, and Daniel Korat. 2018. Setexpander: End-to-end term set expansion based on multi-context term embeddings. In *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 58–62.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 938–947. Association for Computational Linguistics.

Brian Roark and Eugene Charniak. 1998. Noun-phrase co-occurrence statistics for semiautomatic semantic lexicon construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2*, pages 1110–1116. Association for Computational Linguistics.

Xin Rong, Zhe Chen, Qiaozhu Mei, and Eytan Adar. 2016. Egoset: Exploiting word ego-networks and user-generated ontology for multifaceted set expansion. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 645–654. ACM.

Luis Sarmento, Valentin Jijkuon, Maarten de Rijke, and Eugenio Oliveira. 2007. More like these: growing entity classes from seeds. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 959–962. ACM.

Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. Setexpan: Corpus-based set expansion via context feature selection and rank ensemble. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 288–304. Springer.

Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen. 2010. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 993–1001. Association for Computational Linguistics.

Peter D Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Richard C Wang and William W Cohen. 2007. Language-independent set expansion of named entities using the web. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 342–350. IEEE.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3394–3404.