API recommendation based on Word Embeddings

Ana Martínez Saucedo[1], Leonardo Henrique da Rocha Araujo[2], and Guillermo Rodríguez[2]

[1] Universidad Argentina de la Empresa (UADE), Instituto de Tecnología (INTEC), Buenos Aires, Argentina
anmartinez@uade.edu.ar
[2] ISISTAN Research Institute (CONICET - UNICEN), Tandil, Buenos Aires, Argentina
{leonardo.araujo, guillermo.rodriguez}@isistan.unicen.edu.ar

Abstract. In this new era where web services are trending and businesses constantly develop and expose APIs that can be used by third parties, finding one which fits a functional requirement is a daunting task. For this reason, websites such as ProgrammableWeb and APIs.guru offer a directory of API definitions that can be filtered and searched by developers. However, searching for APIs that conform to a requirement on those platforms is still a manual task, and searches are based on the inclusion or exclusion of query words in an API description that does not provide relevant results. For this reason, we have explored the application of word embeddings in the problem of API recommendation using Word2Vec, FastText and GloVe algorithms, as well as pre-trained domain-general and software engineering embeddings. We have constructed a dataset from APIs.guru and retrieved services descriptions to obtain their embeddings and calculate their similarity with a given query embedding. To this end, we created ten test queries with their relevant APIs using a subset of the original dataset. With a recall at 10 recommendations of 69.8% and a nDCG at 10 of 81.4%, we have obtained promising results which demonstrate embeddings can alleviate developers' searches for relevant APIs.

Keywords: API recommendation · word embedding · APIs · microservices · software development

1 Introduction

Software reusability is very important in software development since it can assist developers to build new and more complex software without the need to re-write code [8]. Application Programming Interfaces (APIs) – applications that focus on abstracting a problem and providing an interface for a user to interact with – are an alternative to provide re-usable code [7]. Nowadays, there are over 2500 publicly available documented API providing services that range from stock and forex data to text translation[3].

[3] https://apis.guru

Given the number of available APIs, it is important for the developers to evaluate whether it is necessary to implement each part of the project they are working on, or if an API can be used for certain tasks to increase productivity. However, it is not feasible for developers to manually search through over 2500 API in order to find an interface that provides them the feature they need. Therefore, the objective of our research is to use the documentation available for the APIs in a dataset to suggest APIs that match a user query, given that a direct query in APIs.guru portal is not possible since it only allows keywords search.

In the literature several works have addressed web service recommendation in different contexts. Xiong et al. [10] describe a deep learning based hybrid approach for web service recommendation by combining collaborative filtering and textual content to support mashup development. Nonetheless, since their approach is based on matrix factorization, it is liable to cold start and gray sheep problems. Similarly, Cao et al. [3] propose an approach based on integrated content and network-based service clustering for mashup development. However, both approaches focus on mashups and its granularity is at API-level. Thus, finer granularity could be achieved by considering API endpoints instead.

A relevant recommendation is related to the input query. The user inputs a description of the feature needed and the recommendation is given considering the meaning of the words, instead of keywords. This approach is used since it is possible for the user to provide words that were not in the APIs dataset. Therefore, to achieve this goal, the recommendation evaluates the input using the Natural Language Processing (NLP) technique called Word Embeddings. This algorithm models words as a vector of numbers that represent their meaning, given other words in the vocabulary [1].

In our experimental analysis, we evaluate different word embeddings algorithms in order to recommend relevant services given a user query. The remainder of this paper is organized as follows: the methodology is presented in the Section 2. The results are presented in Section 3, the conclusion and future work are presented in section 4.

2 Methodology

In order to obtain API recommendations given a developer query string, we constructed an API dataset by extracting APIs.guru directories from its GitHub repository[4]. For each API, APIs.guru provides a swagger.yaml or openapi.yaml document (OpenAPI documents adhering to the specification) that describes the API and each endpoint path that constitutes it. In particular, for each endpoint we extracted its description for the purpose of obtaining their embeddings and compare them with a query embedding afterwards.

The first step of our methodology consists of extracting and crawling APIs.guru directories. To this end we have developed a Python script to automatically

[4] `https://github.com/APIs-guru/openapi-directory`

download, transform and construct an API dataset. It is important to highlight that during this step we create metadata as well, such as the corresponding commit hash for a created dataset, and malformed OpenAPI documents which could not be crawled.

The resulting dataset has the following structure:

```
{
    [api]: {
        "description": "api description",
        "endpoints": {
            [endpoint-path]: "endpoint description"
        }
    }
}
```

The second step involves API and endpoint descriptions preprocessing to ensure quality embeddings in the next step. For this purpose, for each string the following transformations are applied:

- HTML code removal (such as code snippets and custom formatting).
- Sentences mapping to list of words.
- Creation of bigrams (two-word sequence of words) and trigrams(three-word sequence of words) that convey a relevant meaning.
- Stop words removal.
- Word lemmatization.

This process results in API endpoint descriptions represented by lists of processed words. Therefore, the next step consists of obtaining for each processed word its corresponding embedding using Word2Vec, FastText or GloVe algorithms (considering that a API endpoint description word exists in the model vocabulary). To this end, we have employed the Gensim[8] library and pretrained embeddings.

Finally, given a query string the last step is executed to obtain its corresponding embeddings and compute the cosine distance (Equation 1) [2] between the query embedding and each API endpoint description embeddings. In this way, the most relevant APIs are retrieved given a query.

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{te}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \quad (1)$$

In order to evaluate our approach, we created eight test queries (Table 1) to determine which API endpoints from our dataset can fulfill a query requirement. For this purpose, we randomly split 20% of the original dataset (44708 endpoints, corresponding to commit `fb28391`[5]) to obtain a test dataset (8941 endpoints).

[5] `https://github.com/APIs-guru/openapi-directory/commit/fb28391bf36639c0473935cbe1ebaa484156b786`

Splitting the original dataset was necessary to perform an unbiased evaluation of our approach, and to be able to manually determine the relevance of each endpoint so that the query requirements are meet.

Query	Relevant APIs
current weather	4
upload a video on a platform	3
send mail	1
weather forecast	2
get text from audio	1
translate text	6
object recognition	3
get nearby restaurants	1

Table 1. Test queries and their relevant APIs

Test queries are employed to determine how useful our approach is to recommend APIs based on textual queries. To this end, for each test query and word embedding technique several metrics utilized in the literature are calculated, namely recall, precision, normalized discounted cumulative gain (nDCG) and F1 score [9]. Our strategy results are reported in the following section.

3 Results

For each test query presented in Section 2, we evaluate our approach performance in terms of recall, precision, nDGC and F1 score at 10 recommendations (Table 2). An example of API recommendations output for a given query is shown in Figure 1. Pre-trained embeddings used for each algorithm are depicted in Table 3.

Metric \ Embedding	Word2Vec	FastText	GloVe
Recall at 10	69.8%	28.12%	18.75%
Precision at 10	16.3%	6.25%	5%
nDCG at 10	64.1%	22.7%	81.4%
F1 score at 10	24.3%	9.7%	9.18%

Table 2. Recommendation metrics for each embedding technique

[6] `https://nlp.stanford.edu/projects/glove/`

Model	Pre-trained embeddings
Word2Vec	SO_vectors_200[5] (Stack Overflow posts)
FastText	crawl-300d-2M-subword[6] (Common Crawl)
GloVe	Wikipedia 2014 + Gigaword 5[6]

Table 3. Pre-trained embeddings used for each model

```
Query: ['upload', 'a', 'video', 'on', 'a', 'platform']
1: api.video/1/videos/post
2: cpy.re/peertube/3.3.0/videos/upload/post
3: vimeo.com/3.4/me/videos/post
4: api.video/1/videos/{videoId}/thumbnail/post
5: api.video/1/live-streams/{liveStreamId}/thumbnail/post
6: lgtm.com/v1.0/snapshots/{project-id}/{language}/post
7: googleapis.com/identitytoolkit/v2/v2/{project}/identityPlatform:initializeAuth/post
8: googleapis.com/analyticsadmin/v1alpha/v1alpha/{parent}/displayVideo360AdvertiserLinks/post
9: azure.com/containerregistry/2019-08-15-preview/{nextBlobUuidLink}/patch
10: amazonaws.com/nimble/2020-08-01/2020-08-01/studios/{studioId}/streaming-images/get
```

Fig. 1. API recommendations for query *"upload a video on a platform"*

Results reveal that Word2Vec model had the best performance on almost all evaluation metrics, which could be explained as a consequence of using pre-trained embeddings specific to the software engineering domain rather than general domain pre-trained embeddings such as the ones trained with Google News or Wikipedia entries. In fact, for FastText and Glove models we used the publicly available Common Crawl pre-trained embeddings, which may explain why they could not capture relations of analogy between an information technology query and API descriptions.

A similar approach that combines word embeddings and a topic model to extract high quality talent topics from short texts and then recommend APIs for a target mashup is presented in [4]. To this end, Word2Vec along Wikipedia pre-trained embeddings were used. Authors report a recall at 10 of ∼50%, precision at 10 of ∼99%, nDCG at 10 of ∼80%, and a F1-score at 10 of ∼60%. Even though their approach has a different objective (recommending APIs for target mashup), we demonstrate that by using domain-specific (software engineering) word embeddings solely our approach could outperform some metrics.

Moreover, our Word2Vec-based recommender could successfully retrieve 69.8% of relevant APIs from the test dataset (recall metric), which are 64.1% useful in terms of API relevance and position in the recommendation list (nDCG metric). Nonetheless, the fraction of relevant APIs in the recommendation list is low (16.3%).

4 Conclusion and future work

In this work we have presented an approach to recommend relevant API to developers using word embeddings. To this end we created an API dataset by extracting endpoint descriptions from APIs.guru OpenAPI documents. Our initial

results are encouraging in terms of usefulness to developers needing to search for specific APIs that could solve a problem, with a recall at 10 of 69.8% and a nDCG at 10 of 64.1% using a Word2Vec model. We consider these results promising as a consequence of using software engineering pre-trained embeddings.

In order to improve each evaluation metric, we plan to fine-tune Stack Overflow embeddings employing API descriptions from the extracted dataset, with the expectation of obtaining better analogies and interpretation of words commonly used to describe APIs. Furthermore, we aim to develop a web application to enhance user usability of our recommender, allowing periodic updates of the API dataset to ensure that not only provided API recommendations are relevant but also current.

References

1. Almeida, F., Xexéo, G.: Word embeddings: A survey. arXiv preprint arXiv:1901.09069 (2019)
2. Baroni, M., Dinu, G., Kruszewski, G.: Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 238–247 (2014)
3. Cao, B., Liu, X.F., Rahman, M.M., Li, B., Liu, J., Tang, M.: Integrated content and network-based service clustering and web apis recommendation for mashup development. IEEE Transactions on Services Computing **13**(1), 99–113 (2020). https://doi.org/10.1109/TSC.2017.2686390
4. Chen, T., Liu, J., Cao, B., Peng, Z., Wen, Y., Li, R.: Web service recommendation based on word embedding and topic model. In: 2018 IEEE Intl Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom). pp. 903–910 (2018). https://doi.org/10.1109/BDCloud.2018.00133
5. Efstathiou, V., Chatzilenas, C., Spinellis, D.: Word embeddings for the software engineering domain. In: Proceedings of the 15th International Conference on Mining Software Repositories. p. 38–41. MSR '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3196398.3196448, https://doi.org/10.1145/3196398.3196448
6. Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., Joulin, A.: Advances in pre-training distributed word representations. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
7. Reddy, M.: API Design for C++. Elsevier (2011)
8. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010), http://is.muni.cz/publication/884893/en
9. Schütze, H., Manning, C.D., Raghavan, P.: Introduction to information retrieval, vol. 39. Cambridge University Press Cambridge (2008)
10. Xiong, R., Wang, J., Zhang, N., Ma, Y.: Deep hybrid collaborative filtering for web service recommendation. Expert Systems with Applications **110**, 191–205 (2018). https://doi.org/https://doi.org/10.1016/j.eswa.2018.05.039, https://www.sciencedirect.com/science/article/pii/S0957417418303385