A Question Answering Model Based on Semantic Matcher for Support Ticketing System

Suyog Trivedi, Gopichand Agnihotram(✉), Balaji Jagan, and Pandurang Naik

Wipro Technology Limited, Wipro CTO Office, Wipro, Bangalore 560100, India
strivedi2505@gmail.com, {gopichand.agnihotram, balaji.jagan, Pradeep.naik}@wipro.com

Abstract. In this paper, we propose an approach to search for the best semantic match of a user query for the question answering system. To achieve this, we make use of word embeddings with a help of trained model using the question answering corpus and its variations to detect the word senses of search queries by the user and show the top best matches which belongs to the same class of question answering pairs and retrieves the corresponding answer to the user. This solution is deployed in ticketing system in large IT industry to automate the user query to retrieve the answers. Word level to context level semantics are achieved through trained model of semantic knowledge with word embeddings.

Keywords: Question answering model · Feature vectors · SpaCy model GloVe · Cosine similarity · Ticketing system

1 Introduction

Semantic matching is one of the important tasks in many natural language processing (NLP) applications, such as information retrieval [1], question answering [2], etc. Considering question answering system as an example, given a pair of question and answer, a matching function is required to determine the degree of matching between two sentences such as question and user query to match with the questions. Moreover, matching the user's query or question with the list of question answer pairs is a difficult task where the user queries are not always complete, grammatically, and syntactically correct. In addition, capturing the intent of the query involves finding the semantic information at a deeper level to capture all the variations of the user questions. Nowadays, deep neural network based models have been applied to overcome such issues. A lot of deep learning models follow a criterion to represent the question and answer in a single distributed representation, and then compute similarities between the query vectors and the question answer pair vectors to output the matching score.

To properly represent words in a continuous space, the idea of a neural model [3] is employed to enable jointly learn to embed of words into an n-dimensional vector space and to use these vectors to predict how likely a word is given its context. Skip-gram model [4] is a widely used approach to compute such an embedding. The skip-gram

M. Singh et al. (Eds.): ICACDS 2018, CCIS 906, pp. 167–176, 2018.
https://doi.org/10.1007/978-981-13-1813-9_17

networks are optimized via gradient ascent, the derivatives modify the word embedding matrix $L \in R\ (n \times |V|)$, where $|V|$ is the size of the vocabulary. The word vectors inside the embedding matrix capture distributional syntactic and semantic information via the word co-occurrence statistics [3, 4]. Once this matrix is learned on an unlabeled corpus, it can be used for subsequent tasks by using each word's vector (a column in L) to represent that word.

In this paper, we propose an approach on semantic matcher for a user query where the top K results are obtained based on the neural embedding models. To achieve this, the unlabeled corpus is pre-processed by POS (parts of speech) tagging the words, lemmatization and then building the random word vectors for both questions answers of the corpus. The word embeddings/word feature vector is created with the help of pre-trained models (spaCy Model, [14]) and the words that are not present in the spaCy model are then given to the GloVe (Global vector representation) [5, 15] which creates the word co-occurrence form the corpus. In the same way, with the user query vectors/questions are created and matched with the reference corpus question answering model. The cosine similarity measure/Euclidean Distance is then applied on the matched vectors to compute the similarity and perform the ranking. The matched items with low ranking are filtered out and shows the top K results of matching question answering model and retrieve the user the corresponding answer. This solution is deployed in ticketing system in large Information Technology (IT) industry to automate the user query to retrieve the answers. For example, the system addresses the user queries related to leave policies, visa related queries, salary related queries etc.

The rest of the paper is organized as follows. Section 2 describes the related work. Section 3 discuss the proposed approach on how semantic matching is carried out using the distributed vector representation. Section 4 deals with the Results and discussion using distributed semantic models which discusses about building the word vectors using the pre-trained vectors based neural models and GloVe word vector representation. Section 5 describes the conclusions and future enhancements followed by References.

2 Related Work

In this section, we discuss the recent works carried out on semantic mapping of query with document, advertisements, passages, etc. using distributed vector representation.

Shen et al. [6] integrated the advantages of translation model and word embedding model to capture the word-to-word relation called a Word Embedding Correlation (WEC) model. The words in the query are mapped to vectors to identify the co-occurrence relationships between words. The word level relationship is extended to the sentence level to calculate the relevance between question and answers. Kutuzov and Kuzmenko [7] proposed an approach for identifying the senses of search queries and perform the semantic clustering on each search engine page results. The word sense disambiguation is performed with the use of distributed word vector representations with the help of prediction-based neural embedding models.

Wan et al. [8] tackled the problem of matching two sentences with multiple positional sentence representation using bidirectional LSTM (Long Short-Term

Memory) in which the contextual information is captured. The interaction between two representations is performed through k-max pooling and a multi-layer perceptron by which the matching score is generated. Using this positional independent matching procedure, any part of the queries can be matched. This type of approach is useful in handling the data of fully and/or partially free-word order languages.

Grbovic et al. [9] presented a search2vec model, a semantic embedding based approach, for queries and ads in which the embedding is learned using various components such as search queries, clicked ads, search links, dwell time and implicit negative signals. In case of the absence of information on new ads, the vector learns the context information from the textual content of the new ads including the bid term context vector. Guo et al. [10] introduced a novel semantic matching based retrieval model based on the Bag of Words Embedding (BoWE) representation. The semantic matching between queries and documents can be viewed as a non-linear word transportation (NWT) problem (based on document word capacity and transportation profit).

Molino and Aiello [11] proposed a semantic matching approach using skip gram model, a distributed representation learned with neural networks for matching questions and answers based on their semantic similarity. The question and answers are represented at different linguistic levels to extract various features that overlap each other. The overlapping features are then used to obtain the linguistic similarity. Giordani and Moschitti [12] proposed an approach to translate a natural language question into a SQL queries where the semantic mapping is carried out with the use of syntactic analyzer. The syntactic trees of questions and queries are represented as relational pairs and are encoded using the SVM based kernel functions.

3 Semantic Matching Approach

The proposed approach deals with the semantic matching of the user query/question/utterances and produces top K questions for a search query. The semantic matching of a query is determined using the word embeddings/feature vectors created with the help of pre-trained models such as spaCy and GloVe vectors to detect the closest match of the user search query. The method proposed an idea of building query words based vector representation. The words and its context information are captured through these vectors where the similarity between the words are measured through the trained models using the domain specific corpus. In addition, words that are semantically similar and/or semantically related to the query but are not part of the vectors, are added to the word embedding vectors incrementally using GloVe of co-occurrence words. Therefore, two words are semantically similar and/or semantically related, if their distributional vector representation are similar based on some established collocation measure. The semantic similarity between the query word vectors and the document word vectors is computed using a traditional measure such as Cosine Similarity or Euclidian distance measure. The cosine distance between word vectors in the trained model is used as a feature determining whether the distance is closer to these words in the user query words.

The corpus is constructed using the question answers utterances (1) such as

$$C = \{(Q1, A1), (Q2, A2), (Q3, A3) \ldots, (Qn, An)\} \tag{1}$$

where Qi is the i^{th} question and $i = 1, 2, \ldots, n$; Ai is the i^{th} answer $i = 1, 2, \ldots, n$.

The following steps involves creating word embeddings using question answering utterances with the pre-trained models.

3.1 Preprocessing

The data under consideration provides a list of question answers utterances as given in (1). The preprocessing module consists of tokenizer, lemmatizer, and POS tagger. The tokenizer tokenizes all the words in the question and answers utterances after removing the stop words and special characters such as "is", "the", "are", "_", "%", etc. For all tokenized words, the nltk WordNet Lemmatization and Stemming is used to obtain the lemma of each word with the help of the nltk POS tagger for example: car, cars, car's, cars to the root word car "fishing", "fished", and "fisher" to the root word, "fish etc. If the word is not in the lemmatizer, then the actual word itself is returned as a lemma. To obtain the correct lemma of a word in a sentence, the POS tag of a word is also considered while finding out the lemma through WordNet lemmatizer. The question answer vector represented after preprocessing is as given in (2) below.

$$(Qi, Ai) = (Sq, Sa) = \{(w1q, w2q, \ldots wnq),\ (w1a, w2a, \ldots, wna)\} \tag{2}$$

where Sq is the sentence corresponding to question and Sa is the sentence corresponding to answer, wiq is the i^{th} word corresponding to the question after preprocessing and similarly wia is the i^{th} word corresponding to the answer, $i = 1, 2, \ldots, n$.

After preprocessing these words are given to the spaCy pre-trained model to create the word embeddings/feature vectors where the semantic matching criterion is accomplished along with the help of the GloVe vector representation and semantic similarity measure. The next step/subsection discusses the word embedding/feature vector creation using the corpus after preprocessing.

3.2 Word Embedding/Feature Vector Creation

In this method, we have adopted the pre-trained neural embedding model for building the feature vectors. Word vectors encode semantic meaning and capture many different degrees of similarity as explained in the paper [13]. There are a variety of computational models that implement the distributional hypothesis, including word2vec, GloVe, dependency-based word embeddings, spaCy and Random Indexing.

In this paper, we use the dependency-based word embeddings implemented in spaCy for troubleshooting data consists of a set of questions answers. This set is used as a training data to build the spaCy word embedding model with the help of the already built pre-trained model [14]. The pre-trained model is available in spaCy where the model uses huge corpus comprises of news articles, Wikipedia documents, weblogs, newsgroups, blogs etc. for building the model. The spaCy model has different features such POS tagger, NER (Named Entity Recognition), sentiment analysis, along

with creating semantic word embeddings. The spaCy model builds semantic word embeddings/feature vectors and internally uses the GloVe vectors and computes the top list words matching with distance measures such as Cosine Similarity and Euclidian distance approach. The spaCy has trained for one million-word vectors. Here vectors ranging from −1 to 1 of each word represent feature vectors.

Along with the use of spaCy model, the question answer set is used to build the domain specific word vectors representation using the one million-word vectors. To build the word vector representation, the sentences in the corpus are preprocessed to derive the words and trained with the pre-trained spaCy model to build the feature vectors. The word embeddings are initialized to the 300-dimensional feature vectors. The word embeddings are given in (3)–(7) below for example sake,

From (2)

$$w1q = [0.145357, -0.227473, \ldots, -0.297674,]_{1 \times 300} \quad \ldots\ldots \tag{3}$$

$$wnq = [-0.545021, 0.064370, \ldots, 0.246844]_{1 \times 300} \tag{4}$$

$$\text{similarly,} \quad w1a = [0.167175,\ 0.287581, \ldots, -0.165552]_{1 \times 300} \quad \ldots\ldots \tag{5}$$

$$wna = [0.299463, 0.317821, \ldots, 0.026345]_{1 \times 300} \tag{6}$$

Finally,

$$(Sq, Sa) = \left[((1 \times 300), (1 \times 300), \ldots, (1 \times 300))_{n \times 300}, ((1 \times 300), \ldots, (1 \times 300))_{n \times 300}\right] \tag{7}$$

In case of the absence of the word in the pre-trained model, the natural language word searches for the vectors in the GloVe word vector representation as explained in the next section.

Once the word vectors are created from the data, the similar procedure is applied to the user query/user question to build the query word vectors representation using preprocessing steps and spaCy model. The GloVe word representation for the missing words are explained in the next step/section.

3.3 GloVe Vector Representation

GloVe is an unsupervised machine learning algorithm for obtaining vector representations for words. The GloVe builds the training model using the aggregated global word-word co-occurrence statistics from a corpus. The method uses semantic similarity words using Euclidian distance or cosine similarity measure to build the resulting word representation with the word vector space.

In the GloVe vector representation for question answering model uses question answering corpus and builds the co-occurrence of unigram, bigram and trigram words using the GloVe to compute the words of 300-dimentinal features vectors. Here we use

the GloVe to the words of our corpus which are not find in spaCy model. From (2) the GloVe representation of the words are given from (8) – (9) below.

$$(Qi, Ai) = \left\{\left(w1q^1, w2q^1, \ldots wnq^1\right), \left(w1a^1, w2a^1, \ldots, wna^1\right)\right\} \tag{8}$$

$$\begin{aligned}\text{Finally,}\quad (Sq, Sa) = \{(w1q, w2q, \ldots wnq), (w1a, w2a, \ldots, wna)\}U \\ \left\{\left(w1q^1, w2q^1, \ldots wnq^1\right), \left(w1a^1, w2a^1, \ldots, wna^1\right)\right\}\end{aligned} \tag{9}$$

where $w1q^1, w2q^1, \ldots wnq^1$ are the words form question, which are not part of spaCy model. $w1a^1, w1a^1, \ldots, w1a^1$ are the words from answers which are not part of spaCy model. The feature vector representation of the words in the question answering corpus are given from (10) – (12) below as example.

$$\begin{aligned} w1q^1 = &[0.064101, -0.544901, \ldots, 0.364538]_{1\times 300} \\ &\ldots. \end{aligned} \tag{10}$$

$$\begin{aligned}\text{Similarly,}\quad w1a^1 = &[0.167175, 0.287581, \ldots, -0.165552]_{1\times 300} \\ &\ldots\ldots\end{aligned} \tag{11}$$

$$wna^1 = [0.299463, 0.317821, \ldots, 0.026345]_{1\times 300} \tag{12}$$

The feature vectors of the question answering corpus will be created using spaCy and Glove vector as explained in the above steps (Subsects. 3.1, 3.2 and 3.3). The next subsection discusses the user query processing in above steps or Subsect. 3.1 to 3.3.

3.4 User Query Processing

The user query will be processed in the above steps and arrive 300-dimensional feature vectors using spaCy and GloVe vectors. Let the user query/utterance is *Ut* which comprises of sequence of sentences or unique sentence. The *Ut* is processes through preprocessing step and removes all stop words and special characters as explained in Subsect. 3.1. The user utterance is shown in (13) below

$$Ut = (S_1, S_2, \ldots, S_n) = (w_1, w_2, \ldots, w_n) \tag{13}$$

here $S_1, S_2, \ldots, S_n$ are the sentences from the utterance and it can be unique sentence and sequence of sentence. $w_1, w_2, \ldots, w_n$ are the words extracted after preprocessing step. The spaCy and GloVe feature vectors of the utterance is shown from (14) – (15) below as example.

The spaCy feature vector of words in the utterance (16) is given as

$$\begin{aligned} w1S = &[0.356557, -0.348621, \ldots, -0.569231]_{1\times 300} \\ &\ldots\ldots \end{aligned} \tag{14}$$

$$wnS = [-0.002453, 0.026352, \ldots, -0.0369124]_{1\times 300} \tag{15}$$

Here $w1S$, $w2S$, …, wnS are the feature vectors of the words representation using spaCy model. There may be few words in Ut where spaCy model is not able to address those words will be addressed using GloVe feature vector as explained in Subsect. 3.3. The GloVe feature vectors are shown in (16) – (17) below.

$$w1G = [-0.378294,\ 0.629482, \ldots, -0.72359]_{1\times 300} \quad \ldots\ldots \tag{16}$$

$$wnG = [0.452625,\ 0.252418, \ldots, -0.0013528]_{1\times 300} \tag{17}$$

Here $w1G, w2G, \ldots, wnG$ are the words represent GloVe feature vectors. The utterance Ut is rewritten as given below in (18)

$$Ut = (w_1,\ w_2, \ldots, w_n) = (w1S,\ w2S, \ldots, wnS)U(w1G,\ w2G, \ldots, wnG) \tag{18}$$

Once the user query/utterance is encoded into a feature vector representation, it is matched with the question answering corpus feature vectors created in Subsects. 3.1–3.3 using the pre-trained models of spaCy and GloVe vector representation. The next subsection discusses the matching of the feature vectors.

3.5 Semantic Similarity Matching

The distance between the feature vectors of user query/question/utterance vs question answering corpus is measured by computing the cosine similarity or Euclidian distance methods. Here we adapt cosine similarity measure to compute the distance and cosine similarity is as given below (19) and (20)

$$(\underline{a}, \underline{b}) = ||\underline{a}||||\underline{b}|| \cos(\theta) \tag{19}$$

$$\cos(\theta) = \frac{\underline{a} \cdot \underline{b}}{||\underline{a}||||\underline{b}||} \tag{20}$$

where $\underline{a}$ is the feature vectors derived from (Qi, Ai) for each $i = 1, 2, \ldots, n$ and $\underline{b}$ is the feature vectors derived from Ut. Based on the confidence of the similarity, the best describes the context information, the top K matched questions and corresponding answers are obtained.

The architecture diagram given Fig. 1 depicts entire flow of Semantic model and semantic matcher explained in Sect. 3.

4 Results and Discussions

The solution is deployed in support ticketing system for large IT services to automate the user queries/questions based on leave, attendance, salary, travel, finance related queries. We have captured all the question answers related to leave, salary, travel etc. in to the systems and performed steps given in Sect. 3. The system can predict the answer

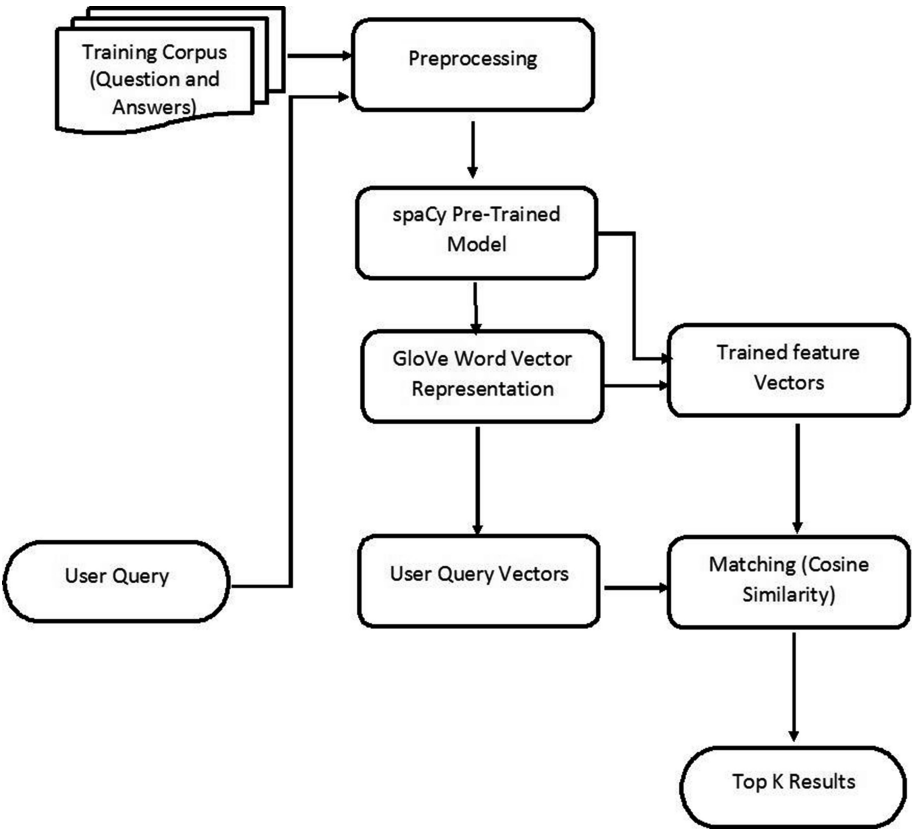

Fig. 1. Detailed architecture diagram for semantic model and matching

to the user query/question using question answering model. We can automate more than 80% of user queries using this solution and reduced to a large extent manual assistance. The current solution build on more than 10k question answering utterance and tested more than one lakh user utterances. We have achieved around 85% accuracy to address the right answer to the user queries using this solution.

The sample utterance for question answers and words which are used for feature vector computation are given in below tables as examples. Let the user utterance related to paternity leave is "**Is paternity leave subject to manager's approval**," and users also asked the same user utterances in different way such as {"**paternity leaves require manager approval?", "any pointers on paternity approval related stuff", "after applying paternity leave whether It will go for any approval?", "what is the process to apply paternity leave"**}, these variations are processed through steps given in Sect. 3 and arrive the unique matching question 1 given in the Table 1 and corresponding answer is retrieved to the user.

For example, the question answers utterances are captured in the database and a few utterances are given in the Table 1. The utterances are preproceed and derive words

Table 1. Support ticketing system: question answers capture for training

Question	Answer
1. Is paternity leave subject to manager's approval	Paternity leave is an auto-approved leave however your manager gets an intimation of the same
2. What is the vertical level travel or FTR (Foreign Travel Request)	Dear user, you cannot raise the FTR/foreign travel request for vertical level. FTR is always project based
3. I am trying to raise the amendment but getting the error as "reservation is mandatory for raising amendment"	Employees must be reserved for an Onsite US (United States) indent before raising the Amendment, it's a valid message

Table 2. Words used for creating the feature vectors

Keywords separated by comma	
Question	Answer
Paternity leave, subject, manager, approval	Paternity, leave, auto, approve, leave, manager, intimation, same
Vertical, level, travel, FTR	User, cannot, raise, FTR, foreign, travel, request, vertical, level, FTR, project
Raise, amendment, get, error, reservation, mandatory, raise	Employee, reserve, Onsite, US, indent, before, raise, Amendment, valid, message

using preprocessed step explained in Sect. 3. The words which are shown in Table 2 are used to create the feature vectors with the help of spaCy and GloVe vectors.

5 Conclusions and Future Work

The solution proposed in this paper discusses the semantic model for question answering utterances. This model built based on pre-trained spaCy and GloVe vectors. The cosine similarity distance measure helps in matching the user query with the question answering utterances. The proposed solution is deployed in support ticketing system for large IT industry to automate user queries/utterances by in large. The solution able to capture more than 85% of accuracy on testing data sets of user queries.

In future, we want to extend this solution to multi-lingual support system such as Spanish, French, Dutch etc. to resolve user queries on support ticketing system or any other domains such as health care, finance, automobile etc.

References

1. Li, H., Xu, J.: Semantic matching in search. Found. Trends Inf. Retr. **7**, 343–469 (2013)
2. Berger, A., Caruana, R., Cohn, D., Freitag, D., Mittal, V.: Bridging the lexical chasm: statistical approaches to answer-finding. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 192–199 (2000)
3. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**(Feb), 1137–1155 (2003)
4. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems (NIPS), pp. 3111–3119 (2013)
5. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543 (2014)
6. Shen, Y., Rong, W., Jiang, N., Peng, B., Tang, J., Xiong, Z.: Word embedding based correlation model for question/answer matching. In: Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI-2017), pp. 3511–3517 (2017)
7. Kutuzov, A., Kuzmenko, E.: Neural embedding language models in semantic clustering of web search results. In: Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016), pp. 3044–3048 (2016)
8. Wan, S., Lan, Y., Guo, J., Xu, J, Pang, L., Cheng, X.: A deep architecture for semantic matching with multiple positional sentence representations. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI 2016, pp. 2835–2841 (2016)
9. Grbovic, M., Djuric, N., Feng, A., Ordentlich, E.: Scalable semantic matching of queries to ads in sponsored search advertising. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2016, pp. 375–384 (2016)
10. Guo, J., Fan, Y., Ai, Q., Croft, W.B.: Semantic matching by non-linear word transportation for information retrieval. In: Proceeding CIKM 2016 Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pp. 701–710 (2016)
11. Molino, P., Aiello, L.M.: Distributed representations for semantic matching in non-factoid question answering. In: Proceedings of Workshop on Semantic Matching in Information Retrieval Co-located with the 37th International {ACM} {SIGIR} Conference on Research and Development in Information Retrieval, SMIR@SIGIR 2014, pp. 38–45 (2014)
12. Giordani, A., Moschitti, A.: Semantic mapping between natural language questions and SQL queries via syntactic pairing. In: Horacek, H., Métais, E., Muñoz, R., Wolska, M. (eds.) NLDB 2009. LNCS, vol. 5723, pp. 207–221. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12550-8_17
13. Levy, O., Goldberg, Y.: Dependency-based word embeddings. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Baltimore, Maryland, pp. 302–308 (2014)
14. Models and Languages: spaCy Usage Documentation. https://spacy.io/. Accessed 08 Nov 2017
15. Pennington, J., Socher, R., Manning, C.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)