Key2Vec: Automatic Ranked Keyphrase Extraction from Scientific Articles using Phrase Embeddings

Debanjan Mahata
Bloomberg
New York, U.S.A
dmahata@bloomberg.net

John Kuriakose
Infosys Limited
Pune, India
John_Kuriakose@infosys.com

Rajiv Ratn Shah
IIIT-Delhi
New Delhi, India
rajivratn@iiitd.ac.in

Roger Zimmermann
NUS-Singapore,
Singapore
rogerz@comp.nus.edu.sg

Abstract

Keyphrase extraction is a fundamental task in natural language processing that facilitates mapping of documents to a set of representative phrases. In this paper, we present an unsupervised technique (*Key2Vec*) that leverages phrase embeddings for ranking keyphrases extracted from scientific articles. Specifically, we propose an effective way of processing text documents for training multi-word phrase embeddings that are used for thematic representation of scientific articles and ranking of keyphrases extracted from them using theme-weighted PageRank. Evaluations are performed on benchmark datasets producing state-of-the-art results.

1 Introduction and Background

Keyphrases are single or multi-word linguistic units that represent the salient aspects of a document. The task of ranked keyphrase extraction from scientific articles is of great interest to scientific publishers as it helps to recommend articles to readers, highlight missing citations to authors, identify potential reviewers for submissions, and analyze research trends over time (Augenstein et al., 2017). Due to its widespread use, keyphrase extraction has received significant attention from researchers (Kim et al., 2010; Augenstein et al., 2017). However, the task is far from solved and the performances of the present systems are worse in comparison to many other NLP tasks (Liu et al., 2010). Some of the major challenges are the varied length of the documents to be processed, their structural inconsistency and developing strategies that can perform well in different domains (Hasan and Ng, 2014).

Methods for automatic keyphrase extraction are mainly divided into two categories: *supervised* and *unsupervised*. Supervised methods approach the problem as a binary classification problem (Hasan and Ng, 2014), whereas the unsupervised methods are mostly based on TF-IDF, clustering, and graph-based ranking (Hasan and Ng, 2010; Mihalcea and Tarau, 2004). On the presence of domain-specific data, supervised methods have shown better performance. The unsupervised methods have the advantage of not requiring any training data and can produce results in any domain.

With recent advancements in deep learning techniques applied to natural language processing (NLP), the trend is to represent words as dense real-valued vectors, popularly known as word embeddings. These representations of words have been shown to equal or outperform other methods (e.g. LSA, SVD) (Baroni et al., 2014). The embedding vectors, are supposed to preserve the semantic and syntactic similarities between words. They have been shown to be useful for several NLP tasks, like part-of-speech tagging, chunking, named entity recognition, semantic role labeling, syntactic parsing, and speech processing, among others (Collobert et al., 2011). Some of the most popular approaches for training word embeddings are Word2Vec (Mikolov et al., 2013), Glove (Pennington et al., 2014) and Fasttext (Bojanowski et al., 2016).

Title: Identification of states of complex systems with estimation of admissible measurement errors on the basis of fuzzy information.
Abstract: The problem of identification of states of complex systems on the basis of fuzzy values of informative attributes is considered. Some estimates of a maximally admissible degree of measurement error are obtained that make it possible, using the apparatus of fuzzy set theory, to correctly identify the current state of a system.
Automatically identified keywords: *complex systems, fuzzy information, admissible measurement errors, fuzzy values, informative attributes measurement error, maximally admissible degree, fuzzy set theory*
Manually assigned keywords: *complex system state identification, admissible measurement errors, informative attributes, measurement errors fuzzy set theory*

Table 1: Keyphrases extracted by using *Key2Vec* from a sample research article abstract.

Word embeddings have already shown promis-

Proceedings of NAACL-HLT 2018, pages 634–639
New Orleans, Louisiana, June 1 - 6, 2018. ©2018 Association for Computational Linguistics

ing results in the process of keyphrase extraction from scientific articles (Wang et al., 2015, 2014). However, Wang *et al.* did not use domain-specific word embeddings and had suggested that training them might lead to improvements. This motivated us to experiment with domain-specific embeddings on scientific articles.

In this work, we represent candidate keyphrases extracted from a scientific article by domain-specific phrase embeddings and rank them using a *theme-weighted* PageRank algorithm (Langville and Meyer, 2004), such that the *thematic weight* of a candidate keyphrase indicate how similar it is to the *thematic representation* or the main theme of the article, which is also constructed using the same embeddings. Due to extensive use of phrase embeddings for representing the candidate keyphrases and ranking them, we name our method as *Key2Vec*. To our knowledge, using multi-word phrase embeddings for constructing *thematic representation* of a given document and to assign *thematic weights* to phrases have not been used for ranked keyphrase extraction, and this work is the first preliminary attempt to do so. Table 1. shows ranked keyphrases extracted using *Key2Vec* from a sample research abstract. Next, we present our methodology.

2 Methodology

Our methodology primarily uses three steps: *candidate selection*, *candidate scoring*, and *candidate ranking*, similar to other popular frameworks of ranked keyphrase extraction (Kim et al., 2013). All the steps depend on the choice of our text processing steps and a phrase embedding model that we train on a large corpus of scientific articles. We explain them next and give a detailed description of their implementations.

2.1 Text Processing

It has been shown (Mikolov et al., 2013), that the presence of multi-word phrases intermixed with unigram words increases the performance and accuracy of the embedding models trained using techniques such as *Word2Vec*. However, in our framework we take a different approach in detecting meaningful and cohesive chunk of phrases while preparing the text samples for training. Instead of relying on measures considering how often two or more words co-occur with each other, we rely on already trained dependency parsing and named entity extraction models. For this work we use *Spacy*[1] as our NLP toolkit along with its default models. The choice of *Spacy* is just for convenience and is not driven by any other factor. We split a text document into sentences, tokenize a sentence into unigram tokens, as well as identify noun phrases and named entities from it. During this process if a named entity is detected at a particular offset in the sentence then a noun phrase appearing at the same offset is not considered.

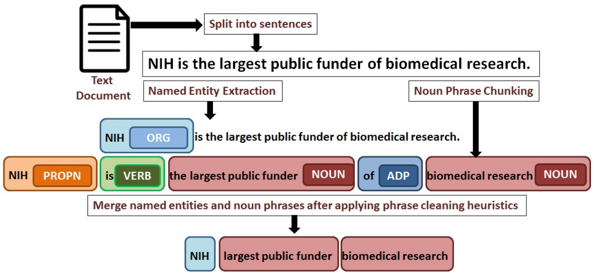
Text
Document
Split into sentences
NIH is the largest public funder of biomedical research.
Named Entity Extraction
Noun Phrase Chunking
NIH
ORG
is the largest public funder of biomedical research.
NIH
PROPN
is
VERB
the largest public funder
NOUN
of
ADP
biomedical research
NOUN
Merge named entities and noun phrases after applying phrase cleaning heuristics
NIH
largest public funder
biomedical research

Figure 1: Text processing pipeline for preparing text samples used for training word embedding models.

We take steps in cleaning the individual single word and multi-word tokens that we obtain. Specifically, we filter out the following tokens.

- Noun phrases and named entities that are fully numeric.
- Named entities that belong to the following categories are filtered out : DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL. Refer, Spacy's named entity documentation[2] for details of the tags.
- Standard stopwords are removed.
- Punctuations are removed except '-'.

We also take steps to clean leading and ending tokens of a multi-word noun phrase and named entity.

- Common adjectives and reporting verbs are removed if they occur as the first or last token of a noun phrase/named entity.
- Determiners are removed from the first token of a noun phrase/named entity.
- First or last tokens of noun phrases/named entities belonging to following parts of

[1] https://spacy.io

[2] https://spacy.io/usage/linguistic-features#section-named-entities

speech: INTJ Interjection, AUX Auxiliary, CCONJ Coordinating Conjunction, ADP Adposition, DET Interjection, NUM Numeral, PART Particle, PRON Pronoun, SCONJ Subordinating Conjunction, PUNCT Punctutation, SYM Symbol, X Other, are removed. For a detailed reference of each of these POS tags please refer Spacy's documentation[3].

- Starting and ending tokens of a noun phrase/named entity is removed if they belong to a standard list of english stopwords and functional words.

Apart from relying on Spacy's parser we use hand crafted regexes for cleaning the final list of tokens obtained after the above data cleaning steps.

- Get rid of leading/trailing junk characters.
- Handle dangling/backwards parentheses. We don't allow '(' or ')' to appear without the other.
- Handle oddly separated hyphenated words.
- Handle oddly separated apostrophe'd words.
- Normalize whitespace.

The resultant unigram tokens and multi-word phrases are merged in the order they appeared in the original sentence. Figure 1, shows an example of how the text processing pipeline works on an example sentence for preparing the training samples that act as an input to the embedding algorithm.

2.2 Training Phrase Embedding Model

The methodology to a great extent relies on the underlying embeddings. We directly train multi-word phrase embeddings using *Fasttext*[4], rather than first training embedding models for unigram words and then combining their dense vectors to obtain vectors for multi-word phrases. Our training vocabulary consists of both unigram as well as multi-word phrases. We are aware of the existing procedures for training phrase embeddings (Yin and Schütze, 2014; Yu and Dredze, 2015), but refrain from using them in this preliminary work. We would like to use them in the future.

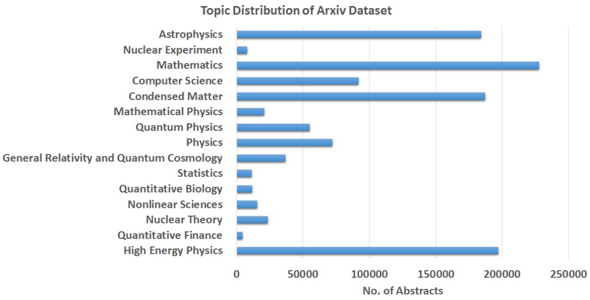

Figure 2: Frequency distribution of topics in the arxiv dataset used for training phrase embeddings.

The main aim of the underlying embedding model is to capture semantic and syntactic similarities between textual units comprising of both single word and multi-word phrases. We chose *Fasttext* over other embedding techniques because it captures both the semantic and morphological similarities[5] between words. For example, if we have *breast cancer* in the content of the theme of a document then intuitively phrases like *breast cancer*, *breast cancer treatment*, should be assigned higher thematic weight than *prostrate cancer* or *lung cancer*, even though the document might mention other forms of cancer as well. Embedding techniques like *Word2Vec* and *Glove*, only takes into account the semantic similarity between words based on their occurrences in a similar context and will not display the desired property that we want to leverage. We would like to study the effects of other types of embeddings in the future.

Dataset: Since this work deals with the domain of scientific articles we train the embedding model on a collection of more than million scientific documents. We collect 1,147,000 scientific abstracts related to different areas (Fig 2) from arxiv.org[6]. For collecting data we use the API provided by arxiv.org that allows bulk access[7] to the articles uploaded in their portal. We also add the scientific documents present in the benchmark datasets (Sections 3), increasing the total number of documents to 1,149,244.

After processing the text of the documents as mentioned above, we train a *Fasttext-skipgram* model using *negative sampling* with a *context win-*

[3] https://spacy.io/api/annotation#pos-tagging

[4] https://fasttext.cc/

[5] https://rare-technologies.com/fasttext-and-gensim-word-embeddings/

[6] http://arxiv.org

[7] https://arxiv.org/help/bulk_data

dow size of 5, *dimension* of 100 and number of *epochs* set to 10. We would like to experiment further in the future on the selection of optimal parameters for the embedding model.

Candidate Selection: This step aids in chosing candidate keyphrases from the set of all phrases that can be extracted from a document, and is commonly used in most of the automated ranked keyphrase extraction systems. Not all the phrases are considered as candidates. Generally, unwanted and noisy phrases are eliminated in this process by using different heuristics (Section 2.1). We split a given document into sentences and to extract noun phrases and named entities as described previously. As an output of this step we get a set of unique phrases ($C_{d_i} = \{c_1, c_2, ..., c_n\}_{d_i}$) for a document d_i to be used later for scoring and ranking in the next two steps.

Candidate Scoring: In this step we assign a *theme vector* ($\hat{\tau_{d_i}}$) to a document (d_i). The *theme vector* can be tuned according to the type of documents that are being processed and the type of keyphrases that we want to get in our final results. In this work, we extract a *theme excerpt* from a given document and further extract a unique set of *thematic phrases* comprising of named entities, noun phrases and unigram words ($T_{d_i} = \{t_1, t_2, ..., t_m\}_{d_i}$) from it. For the *Inspec* dataset we use the first sentence of the document which consists its title, and for the *SemEval* dataset we use the title and the first ten sentences extracted from the beginning of the document, as the *theme excerpts*, respectively (see Section 3). The first ten sentences of a document from the *SemEval* dataset essentially captures the abstract and sometimes first few sentences of the introduction of a scientific article. We get the vector representation ($\hat{t_j}$) of each *thematic phrase* extracted from the *theme excerpt* using the phrase embedding model that we trained and perform vector addition in order to get the final *theme vector* ($\hat{\tau_{d_i}} = \sum_{j=1}^{m} \hat{t_j}$) of the document. The phrase embedding model is then used to get the vector representation ($\hat{c_k}; k \in \{1...n\}$) for each candidate keyphrase in C_{d_i}.

We calculate the *cosine distance* between the *theme vector* ($\hat{\tau_{d_i}}$) and vector for each candidate keyphrase ($\hat{c_k}$) and assign a score ($\kappa(\hat{x}, \hat{y}) \rightarrow [0,1]$) to each candidate, with 1 indicating a complete similarity with the *theme vector* and 0 indicating a complete dissimilarity. To get the final *thematic weight* ($w_{c_j}^{d_i}$) for each candidate w.r.t. a given document (d_i), the candidate scores are scaled again between 0 and 1 with a score of 1 assigned to the candidate semantically closest to the main theme of the document and 0 to the farthest.

Candidate Ranking: In order to perform final ranking of the candidate keyphrases we use weighted personalized PageRank algorithm. A directed graph G_{d_i} is constructed for a given document (d_i) with C_{d_i} as the vertices and E_{d_i} as the edges connecting two candidate keyphrases if they co-occur within a window size of 5. The edges are bidirectional. Weights $sr(c_j^{d_i}, c_k^{d_i})$ are calculated for the edges using the semantic similarity between the candidate keyphrases obtained from the phrase embedding model and their frequency of co-occurrence, as used by Wang *et al.* (Wang et al., 2015), and shown in equation 3. We use *cosine distance* ($\frac{1}{1-cosine(c_j^{d_i}, c_k^{d_i})}$) and *Point-wise Mutual Information* ($PMI(c_j^{d_i}, c_k^{d_i})$) for calculating $semantic(c_j^{d_i}, c_k^{d_i})$ (equation 1) and $cooccur(c_j^{d_i}, c_k^{d_i})$ (equation 2), respectively. The main intuition behind calculating semantic relatedness by using a phrase embedding model is to capture how well two phrases are related to each other in general. Whereas, the co-occurrence score captures the local relationship between the phrases within the context of the given document.

$$semantic(c_j^{d_i}, c_k^{d_i}) = \frac{1}{1 - cosine(c_j^{d_i}, c_k^{d_i})} \quad (1)$$

$$cooccur(c_j^{d_i}, c_k^{d_i}) = PMI(c_j^{d_i}, c_k^{d_i}) \quad (2)$$

$$sr(c_j^{d_i}, c_k^{d_i}) = semantic(c_j^{d_i}, c_k^{d_i}) \times cooccur(c_j^{d_i}, c_k^{d_i}) \quad (3)$$

Given graph G, if $\varepsilon(c_j^{d_i})$ be the set of all edges incident on the vertex $c_j^{d_i}$, and $w_{c_j}^{d_i}$ is the *thematic weight* of $c_j^{d_i}$ as calculated in the *candidate scoring* step, then the final PageRank score $R(c_j^{d_i})$ of a candidate keyphrase $c_j^{d_i}$ is calculated using equation 4, where $d = 0.85$ is the *damping factor* and $out(c_k^{d_i})$ is the out-degree of the vertex $c_k^{d_i}$.

$$R(c_j^{d_i}) = (1-d)w_{c_j}^{d_i} + d \times \sum_{c_k^{d_i} \in \varepsilon(c_j^{d_i})} \left(\frac{sr(c_j^{d_i}, c_k^{d_i})}{|out(c_k^{d_i})|}\right) R(c_k^{d_i}) \quad (4)$$

Next, we evaluate the performance of *Key2Vec*.

	Micro Avg. Precision @5	Micro Avg. Recall @5	Micro Avg. F1 @5	Micro Avg. Precision @10	Micro Avg. Recall @10	Micro Avg. F1 @10	Micro Avg. Precision @15	Micro Avg. Recall @15	Micro Avg. F1 @15
Inspec	61.78 %	25.67 %	36.27 %	57.58 %	42.09 %	48.63 %	55.90 %	50.06 %	52.82 %
SemEval	41 %	14.37 %	21.28 %	35.29 %	24.67 %	29.04 %	34.39 %	32.48 %	33.41 %

Table 2: Performance of *Key2Vec* over combined *controlled* and *uncontrolled* annotated keyphrases for *Inspec* and *SemEval 2010* datasets.

Inspec (Combined)	Key2Vec	Wang et al., 2015	Liu et al., 2010	SGRank (Danesh et al., 2015)	TopicRank (Bougouin et al., 2013)
Micro Avg. F1@10	**48.63 %**	44.7 %	45.7 %	33.95 %	27.9 %

Table 3: Comparison of *Key2Vec* with some state-of-the-art systems (Liu et al., 2009; Danesh et al., 2015; Bougouin et al., 2013; Wang et al., 2015) for Avg. F1@10 on *Inspec* dataset.

SemEval 2010 (Combined)	Key2Vec	SGRank (Danesh et al., 2015)	HUMB (Lopez and Romary, 2010)	TopicRank (Bougouin et al., 2013)
Micro Avg. F1@10	**29.04 %**	26.07 %	22.50 %	12.1 %

Table 4: Comparison of *Key2Vec* with some state-of-the-art systems (Danesh et al., 2015; Bougouin et al., 2013; Lopez and Romary, 2010) for Avg. F1@10 on *SemEval 2010* dataset.

3 Experiments and Results

The final ranked keyphrases obtained using the *Key2Vec* methodology as described in the previous section is evaluated on the popular *Inspec* and *SemEval 2010* datasets. The Inspec dataset (Hulth, 2003) is composed of 2000 abstracts of scientific articles divided into sets of 1000, 500, and 500, as training, validation and test datasets respectively. Each document has two lists of keyphrases assigned by humans - *controlled*, which are assigned by the authors, and *uncontrolled*, which are freely assigned by the readers. The controlled keyphrases are mostly abstractive, whereas the uncontrolled ones are mostly extractive (Wang et al., 2015). The *Semeval 2010* dataset (Kim et al., 2010) consists of 284 full length ACM articles divided into a test set of size 100, training set of size 144 and trial set of size 40. Each article has two sets of human assigned keyphrases: the *author-assigned* and *reader-assigned* ones, equivalent to the *controlled* and *uncontrolled* categories, respectively of the *Inspec* dataset. We only use the test datasets for our evaluations and combine the annotated *controlled* and *uncontrolled* keyphrases.

The ranked keyphrases are evaluated using exact match evaluation metric as used in SemEval 2010 Task 5. We match the keyphrases in the annotated documents in the benchmark datasets with those generated by *Key2Vec*, and calculate micro-averaged precision, recall and F-score ($\beta = 1$), respectively. In the evaluation, we check the performance over the top 5, 10 and 15 candidates returned by *Key2Vec*. The performance of *Key2Vec* on the metrics is shown in Table 2. Tables 3 and 4 shows a comparison of *Key2Vec* with some of the state-of-the-art systems giving best performances on the *Inspec* and *SemEval 2010* datasets, respectively.

4 Conclusion and Future Work

In this paper, we proposed a framework for automatic extraction and ranking of keyphrases from scientific articles. We showed an efficient way of training phrase embeddings, and showed its effectiveness in constructing thematic representation of scientific articles and assigning thematic weights to candidate keyphrases. We also introduced theme-weighted PageRank to rank the candidate keyphrases. Experimental evaluations confirm that our proposed technique of *Key2Vec* produces state-of-the-art results on benchmark datasets. In the future, we plan to use other existing procedures for training phrase embeddings and study their effects. We also plan to use *Key2Vec* in other domains such as news articles and extend the methodology for other related tasks like summarization.

References

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. *arXiv preprint arXiv:1704.02853* .

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL (1)*. pages 238–247.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* .

Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International Joint Conference on Natural Language Processing (IJCNLP)*. pages 543–551.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Soheil Danesh, Tamara Sumner, and James H Martin. 2015. Sgrank: Combining statistical and graphical methods to improve the state of the art in unsupervised keyphrase extraction. *Lexical and Computational Semantics (* SEM 2015)* page 117.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, pages 365–373.

Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *ACL (1)*. pages 1262–1273.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*. Association for Computational Linguistics, pages 216–223.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 21–26.

Su Nam Kim, Olena Medelyan, Min-Yen Kan, and Timothy Baldwin. 2013. Automatic keyphrase extraction from scientific articles. *Language resources and evaluation* 47(3):723–742.

Amy N Langville and Carl D Meyer. 2004. Deeper inside pagerank. *Internet Mathematics* 1(3):335–380.

Zhiyuan Liu, Wenyi Huang, Yabin Zheng, and Maosong Sun. 2010. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, pages 366–376.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*. Association for Computational Linguistics, pages 257–266.

Patrice Lopez and Laurent Romary. 2010. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th international workshop on semantic evaluation*. Association for Computational Linguistics, pages 248–251.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. Association for Computational Linguistics.

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.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. volume 14, pages 1532–1543.

Rui Wang, Wei Liu, and Chris McDonald. 2014. Corpus-independent generic keyphrase extraction using word embedding vectors. In *Software Engineering Research Conference*. volume 39.

Rui Wang, Wei Liu, and Chris McDonald. 2015. Using word embeddings to enhance keyword identification for scientific publications. In *Australasian Database Conference*. Springer, pages 257–268.

Wenpeng Yin and Hinrich Schütze. 2014. An exploration of embeddings for generalized phrases. In *ACL (Student Research Workshop)*. pages 41–47.

Mo Yu and Mark Dredze. 2015. Learning composition models for phrase embeddings. *Transactions of the Association for Computational Linguistics* 3:227–242.