Information Retrieval Engine Part One:

Keyword Extraction and Semantic Similarity

Trevor McInroe

Northwestern University, MSDS-453

August 2, 2020

Introduction and Problem Statement

This work encompasses phase one of the creation of our "Information Retrieval Engine" for the final project of Northwestern's MSDS-453 course. By phase one, we refer to the *Similarity Computation Engine* (SCE) shown in the figure in the Appendix. The SCE consists of a keyword extraction algorithm as well as a model for estimating the semantic similarity between two bodies of text. In this work, we explored the various options for keyword extraction algorithms and tested them on a set of open-source datasets. In addition, we examined two different methods for estimating semantic similarity, one with a static similarity computation and one with a learned neural representation. The purpose of this research was to determine the relative merits of the various approaches and to determine which approaches would be best suited for the Information Retrieval Engine.

Background and Related Works

Information retrieval is an application of natural language processing in which a document or set of documents are retrieved from a given corpus based on their similarity to a given query. These retrieval schemes can range from simple information filtering based on syntactically-strict queries, such as SQL, to complex queries based on open-ended sentences. Information retrieval is a well-studied problem for which there are many solutions (Manning, Raghvan, and Schütze 2018; Birjali, Beni-Hssane, and Erritali 2016; Lin, Jiang, and Lee 2014). Our Similarity Computation Engine (SCE) contains two main natural language processing components. The first is keyword extraction, and the second is semantic similarity.

The goal of keyword extraction algorithms is to detect and return words or phrases that best describe the contents of a given document. As Onan, Korukoğlu, and Bulut (2016) state, keyword extraction methods can be naturally divided into two main types: (a) domain-dependent and (b) domain-independent. (a) requires computation across all documents in the corpus and, therefore, re-computation when new documents are added to the corpus. On the other hand, (b) does not require the analysis of the entire corpus, which allows for corpus scaling without additional computational requirements. An example of domain-dependent methods is TFIDF introduced by Jones (1972), which computes a score for each term in each document through comparisons of the term's prevalence across the entire corpus. The keywords can then be identified as the words in each document that yield the highest TFIDF score. The two examples of domain-independent algorithms explored in this research are Rapid Automatic Keyword Extraction (RAKE), introduced by Rose et al. (2010), and TextRank, introduced by Mihalcea and Tarau (2004).

RAKE works by splitting a document into phrases via stopwords, given domain-specific terms, and punctuation. For example, the sentence "Granny Smith apples, the green ones, can be sour" would be split into three phrases: "Granny Smith apples", "green ones", and "sour". Once the keyword phrases have been identified, the individual terms within them are scored. This score is based on two criteria: (a) the frequency of the term throughout the entire document and (b) the *degree* of the term, or the total length of the keyphrases within which the word appears. The final score is calculated as the ratio of the degree to the frequency. A higher RAKE score indicates that the term plays an important role in defining many keyword-dense keyphrases. The final score for each keyphrase is simply the sum of the term scores within the phrase.

TextRank is a graph-based algorithm that can be used for both keyword and sentence extraction. The authors base their algorithm on the revolutionary PageRank algorithm from Brin and Page (1998). A voting mechanism scans across the graph, where vertices represent individual terms that are nouns or adjectives and edges represent connections between the terms. Vertices are connected by an edge if the two terms they represent occur in the text within a window of N terms, where N is generally set between two and ten (Mihalcea and Tarau 2004). Drawing from the PageRank algorithm, each vertex's score is initialized to one, and the following iterative algorithm is run:

$$S(V_i) = (1-d) + d \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

where $S(V_i)$ is the score of vertex V_i, In is the set of vertices that point to V_i, Out is the set of vertices that V_i points to, and d is a dampening factor which the TextRank authors set to 0.85, the same as the authors of PageRank (Mihalcea and Tarau 2004). For undirected graphs, Out is set equal to In. This algorithm is run until convergence, which is defined as when the change in scores between iterations for any vertex falls below a given threshold, usually set to 0.0001. As a post-processing phase, the top T terms with the best scores are kept, and terms that occur directly next to each other in the original text are collapsed into a single keyword phrase.

Beyond being able to identify keywords in documents, an information retrieval engine must be able to measure the semantic similarity between a query and a set of document candidates. The goal of measuring semantic similarity is to determine the likeness in meaning, not just term contents, of two or more documents. In general, there is a dichotomy of

approaches: (a) computing a distance measure between vector representations of documents or (b) learning a neural representation of the distance between the vector representations of documents. One of the most common distance metrics to use for (a) is *cosine similarity*.

Cosine similarity measures the cosine of the angle between two given vectors, determining if they point in similar directions. The domain of natural language processing uses a modified cosine similarity metric between two vectors, a and b, that outputs values between zero and one:

$$cos(\theta) = 1 - \frac{ab}{||a||_2 \, ||b||_2}$$

where the numerator is the dot product between the two vectors and the denominator is the product between the ℓ_2 norms of the vectors. A similarity measure equal to one indicates complete dissimilarity, and zero indicates identical vectors.

Developing a neural representation of semantic similarity requires a non-standard network architecture. In the normal case, a neural network is meant to learn to map a single input to a single output. In the case of semantic similarity, the network must learn to map two paired-inputs to a single output that compares the inputs. To tackle this problem, several researchers have turned to the idea of a *Siamese network* (Zhu et al. 2018b; Li, Bilodeau, and Bouachir 2018; Zhu et al. 2018a; Kamineni et al. 2018). A Siamese network is made of two neural networks with mirrored architectures that share weights. The networks learn to output an encoded version of their input. These outputs are then passed through a distance measure, which concatenates them into a single value. The most common distance measure for Siamese networks is Manhattan distance, or the ℓ_1 norm of the difference between the

two vectors (Mueller and Thyagarajan 2016). To constrain the output of the network, the result of the ℓ_1 norm is made negative and then passed through an exponential function:

$$\exp(-||x_1 - x_2||_1)$$

where x_1 and x_2 are the outputs from the left and right network, respectively. The intuition here is that the mirrored networks will learn to output vectors whose difference produces a very small ℓ_1 norm for semantically similar sentences. This small ℓ_1 norm, when made negative and exponentiated, will produce a number very close to one.

In the domain of natural language, Siamese networks are usually either made of Long Short Term Memory (LSTM) cells (Hochreiter and Schmidhuber 1997) or Gated Recurrent Unit (GRU) cells (Cho et al. 2014). These types of cells are used in recurrent neural networks and have the ability to learn long-term and short-term information in data that have a serially-dependent structure, such as time-series or text data. GRU cells are a simplified version of the LSTM cell with fewer learnable parameters:

$$z_t = \sigma(W_{xz}^\intercal x_t + W_{hz}^\intercal h_{t-1} + b_z)$$

$$r_t = \sigma(W_{xr}^\intercal x_t + W_{hr}^\intercal h_{t-1} + b_r)$$

$$g_t = \phi(W_{xg}^\intercal x_t + W_{hg}^\intercal (r_t \odot h_{t-1}) + b_g)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot g_z$$

where x_t is the input vector for step t, h_t is the output of the GRU cell at step t, σ is the sigmoid activation function, ϕ is the tanh activation function, $\odot$ is the element-wise product,

W_{ij} is the weight matrix connecting inputs i and j, and b_i is the bias of unit i in the GRU cell. For a graphical depiction, see Figure 1, below.

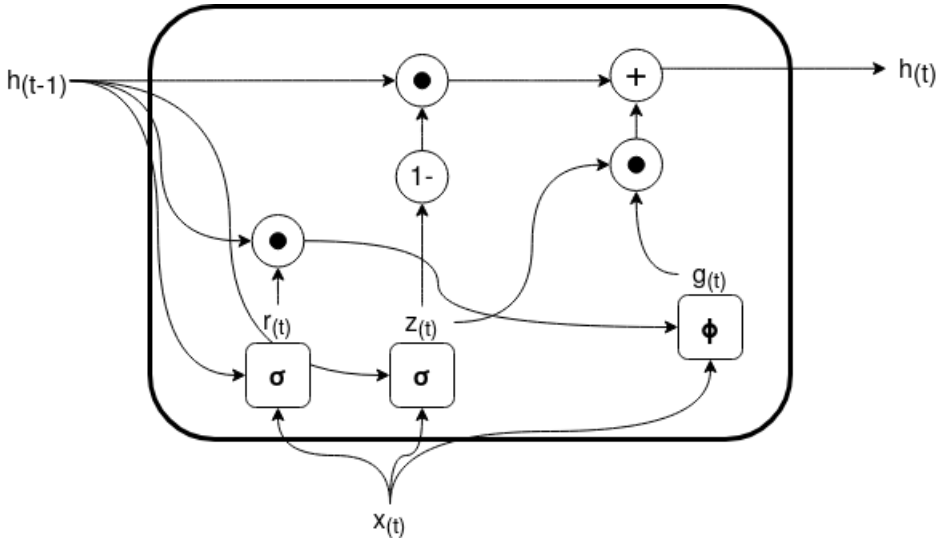


Figure 1. Depiction of GRU cell. Circle with dot depicts element-wise product.

Using text data as input to any sort of mathematical model requires a numerical representation of said text data. Recurrent network-based models require their inputs to be in terms of "time steps". To represent text data this way, each word is transformed into an independent vector and placed sequentially in the same order as the text itself. One of the most popular ways to do this is with the word2vec model introduced by Mikolov et al. (2013). Metric-based measurements, such as cosine similarity, require a single vector that represents each text as a whole. One of the most recent breakthroughs for this transformation, called the Universal Sentence Encoder (USE), was introduced by researchers from Google (Cer et al. 2018). In this paper, the authors present two model architectures for sentence embedding, one based on a transformer architecture introduced by Vaswani et al. (2017), and another based on Deep Average Networks (DAN), introduced by Iyyer et al. (2015). The authors show that the transformer-based architecture produces higher accuracy across various tasks,

while the DAN architecture has computational advantages. The transformer architecture was trained with multi-task learning across a wide variety of natural language tasks. The authors use this method to encourage as much generalization for the embeddings as possible.

Data

For the task of semantic similarity, we used the "Sentences Involving Compositional Knowledge" (SICK) dataset from the SemEval-2014 natural language competition[1]. The SICK dataset was introduced by Bentivogli et al. (2014) and contains about 10,000 sentence pairs, using a designated 50/50 split for the training and testing datasets. These sentence pairs are from a variety of sources such as image and video captions. The sentence pairs were then tagged with a relatedness score $\in [1, 5]$ that captures the degree of similarity between the sentences in the pair. The relatedness score describes the closeness of the subjects and meanings of the sentence pairs, not just the amount of similar words. For examples of sentence pairs and scores, see Table 3, below. To make this relatedness score workable for our experiments, we standardized it to be $\in [0, 1]$.

Relatedness score	Sentence Pair
1.6	A: "A man is jumping into an empty pool" B: "There is no biker jumping in the air"
2.9	A: "Two children are lying in the snow and are making snow angels" B: "Two angels are making snow on the lying children"
3.6	A: "The young boys are playing outdoors and the man is smiling nearby" B: "There is no boy playing outdoors and there is no man smiling"
4.9	A: "A person in a black jacket is doing tricks on a motorbike" B: "A man in a black jacket is doing tricks on a motorbike"

Table 1. Examples of sentence pairs and their relatedness score from Bentivogli et al. (2014)

1. Retrieved from alt.qcri.org/semeval2014/task1/

For the task of keyword extraction, we tested RAKE and TextRank on four open-source datasets. The first dataset is Inspec, originally used by Hulth (2003), which contains abstracts and manually-selected keywords for 500 academic papers. The second dataset is comprised of 1,250 abstracts and keywords from the World Wide Web academic conference. The third includes 450 news articles and keywords from the Marujo dataset procured by Marujo et al. (2013). The last is made of 500 abstracts and keywords from PubMed scientific articles. We chose a wide range of topics and two different document styles to attempt to find a keyword extraction method that generalizes well.

Research Design and Modeling Methods

Keyword Extraction

For keyword extraction, we tested both the RAKE and TextRank algorithms across the four datasets described in the previous section. To determine the success of each algorithm, we used recall, which is the percentage of ground truth keywords the algorithms found. For the comparison mechanism, we allowed a "sub-string" to qualify as a match. For example, if the ground truth keyword is "complex robotics", our comparison mechanism would count the guess "complex robot" as a successful match.

Semantic Similarity

For our task of semantic similarity, we compared a cosine similarity computation, as well as a learned neural representation with Siamese networks. For our sentence-level vectors to be fed into a cosine similarity computation, we used Google's USE. We used the transformer architecture version of USE that outputs an embedding, $x \in \mathbb{R}^{512}$, for each

sentence. For our Siamese networks, we employed GRU cells and Manhattan distance as the concatenation mechanism. Each of the mirrored networks were made of a word2vec embedding layer followed by a single set of GRU cells. The embedding layer made use of a word2vec model that was pre-trained on the Wikipedia corpus[2] that embeds each word in the input text into a 300-dimensional vector. The GRU cells then map this into a vector of length 100. During training, the weights within the embedding layer were frozen. For a graphical depiction of our Siamese network, see Figure 2.

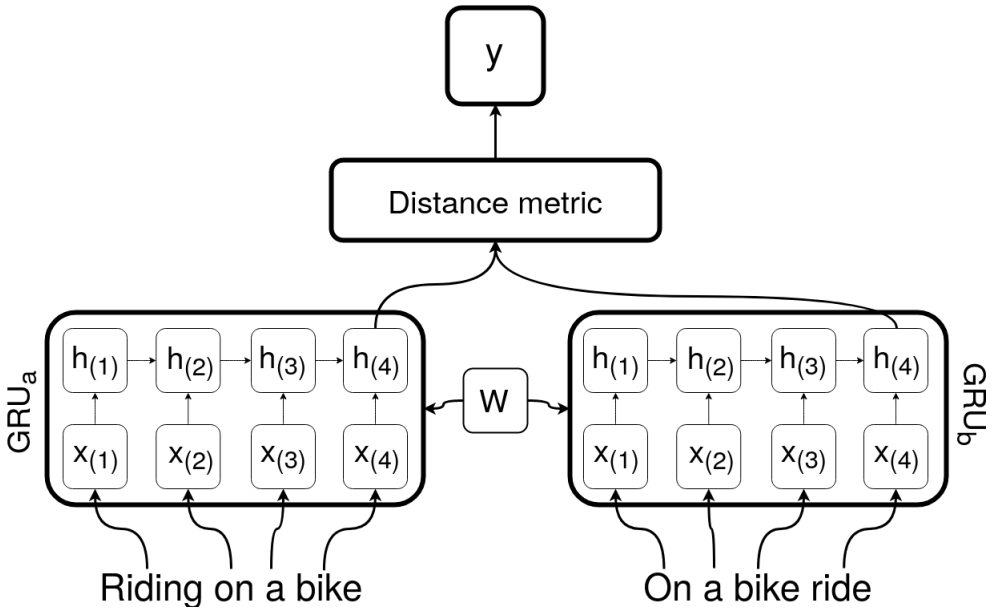


Figure 2. Depiction of Siamese network

To compare the results for our semantic similarity task, we used the recommended evaluation metric from the competition: Pearson's correlation. We took the vector of ground truth relatedness scores and computed their correlation to the relatedness score predictions from the Siamese network and the cosine similarity output. As cosine similarity represents its strength in the reverse order of the relatedness score (zero is the strongest relation), we

2. Retrieved from https://wikipedia2vec.github.io/wikipedia2vec/pretrained/

took the absolute value of the correlation. Using this metric, a score of one indicates perfect performance, and zero indicates the poorest level of performance.

After tuning the Siamese network, we ended with the following hyperparameters: Glorot uniform weight initialization, biases initialized to 2.5, mini-batch size of 64, 500 epochs of training, Adadelta optimizer with an initial learning rate of 1.0, learning rate decay of 0.985, and to help with exploding gradients, gradient clipping for values beyond 2.0.

Analysis of Results

For the task of keyword extraction, we found that the TextRank algorithm outperforms RAKE across all tested datasets. Observing Table 2, we note that, for both keyword extraction methods, the Marujo dataset was "difficult", producing results far below average. This suggests that TextRank is the superior choice for our Information Retrieval Engine.

For the task of semantic similarity, we find that the Siamese network significantly outperformed the combination of USE and cosine similarity. Observing Table 3, we note that the Siamese network was able to nearly-perfectly learn the semantic relationships in the training dataset, ending with a correlation measure of 0.941. This performance generalized well onto the test dataset with a correlation measure of 0.834 versus the 0.776 correlation measure for USE and cosine similarity. We can observe this difference in performance graphically in Figure 3. To display performance, we sorted the sentence pairs by the ground truth relatedness score (orange) and overlaid the estimations (blue). The predictions from the Siamese network (left) follow the pattern of the ground truth values much more closely than the computed cosine similarities (right). In addition, we note that the Siamese network was

able to outperform the best performing system that was submitted to the 2014 competition[3], which produced a correlation measure of 0.828. We note that this comparison is not entirely fair, as we tuned our network to performance on the test set while the original competitors did not have access to the test set.

	Inspec	WWW	Marjuo	PubMed
RAKE	31.8%	12.5%	15.3%	32.7%
TextRank	**50.7%**	**34.0%**	**26.4%**	**43.6%**

Table 2. Recall of keyword extraction algorithms.

	Train data	Test data
USE + cosine	0.771	0.776
Siamnese network	**0.941**	**0.834**
#1 on SICK leaderboard	-	0.828

Table 3. Pearson's correlation on SICK dataset

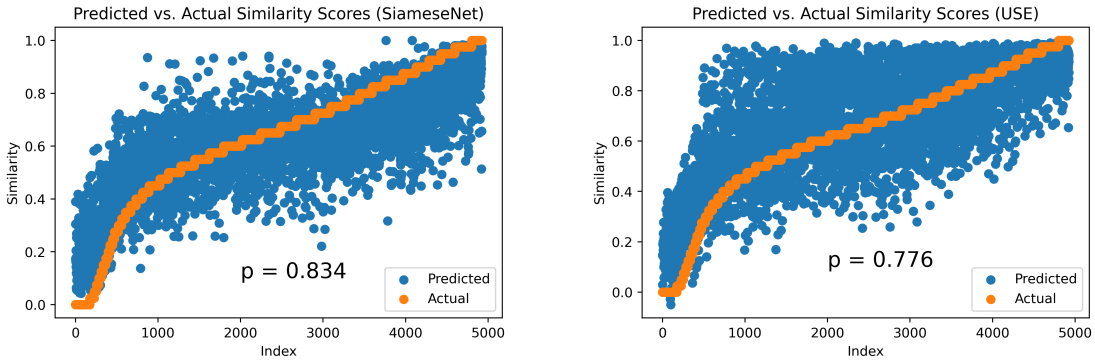


Figure 3. Predicted versus actual relatedness score for Siamese network (left) and USE + cosine similarity (right)

3. Leaderboard: http://alt.qcri.org/semeval2014/task1/index.php?id=results

Conclusion

From this work, we showed the relative performance of various approaches for the two natural language tasks of keyword extraction and semantic similarity. Across four real-world datasets, the TextRank algorithm greatly outperformed RAKE in the task of keyword extraction, while both showed poor performance on the news-article dataset. This suggests that the TextRank algorithm is a strong contender for our Information Retrieval Engine.

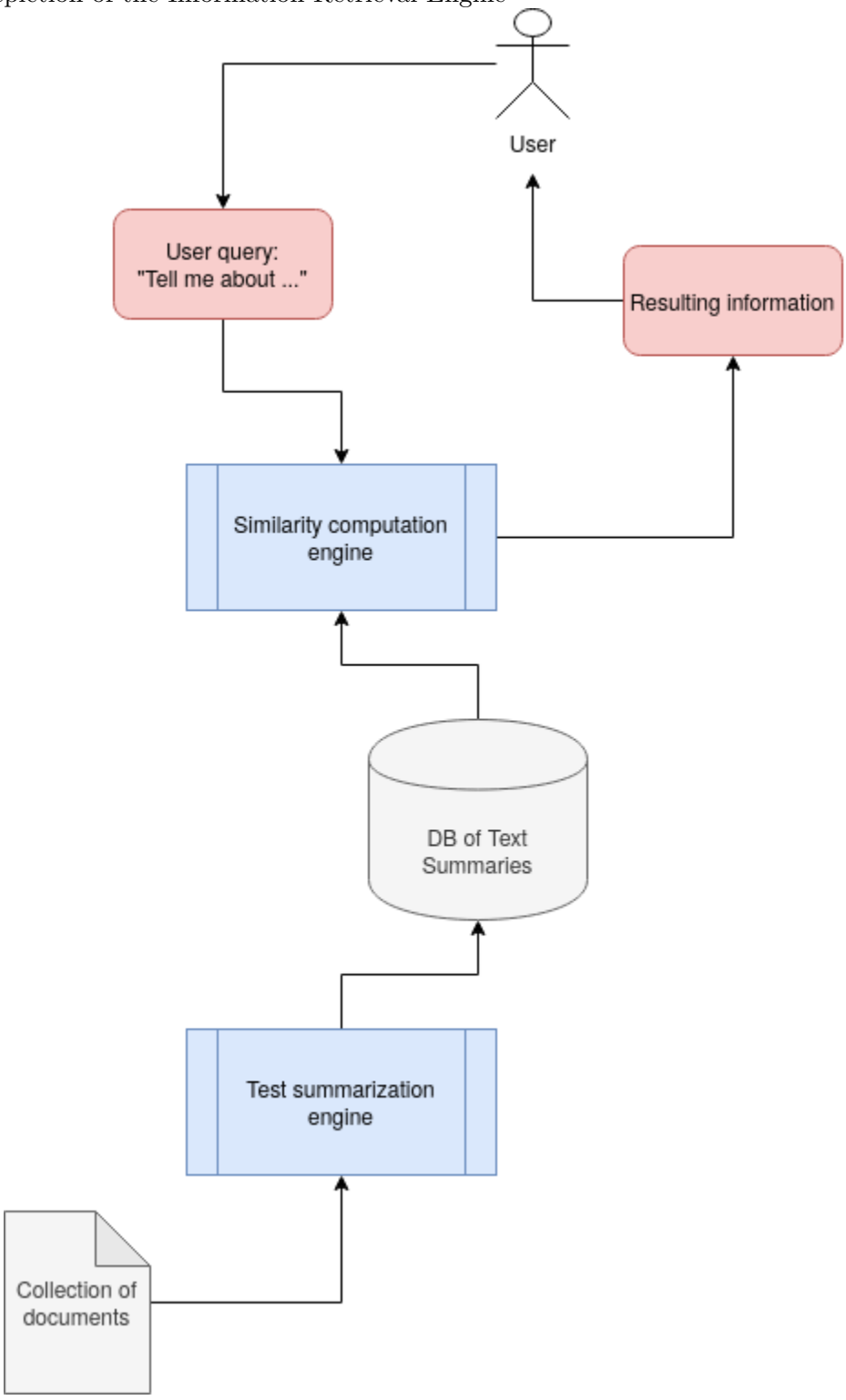
For the task of semantic similarity, the learned neural representation from the Siamese network significantly outperformed the use of the Universal Sentence Encoder embeddings with the cosine similarity computation. We hypothesize that this is due to the domain-specific knowledge gained by the training process of the neural networks as opposed to the sole use of pre-trained embeddings. This suggests that we should choose the Siamese architecture for our Information Retrieval Engine, as well as determine how best to tune the network's weights to the documents we will be storing.

Future Work

While this research shows promising results, several open questions remain. Given that our Similarity Computation Engine will take a short phrase and a long paragraph as input, does the performance of the Siamese network transfer well to two documents of vastly different length? In addition, can we successfully pre-train our network on tasks like SICK and tune it on our final dataset? Finally, what about the Marujo dataset caused both keyword extraction algorithms to perform poorly relative to the other datasets?

Appendix

Figure 4. Depiction of the Information Retrieval Engine



Bibliography

Bentivogli, Luisa, Raffaella Bernardi, Marco Marelli, Stefano Menini, Marco Baroni, and Roberto Zamparelli. 2014. "SICK Through the SemEval Glasses. Lesson Learned from the Evaluation of Compositional Distributional Semantic Models on Full Sentences Through Semantic Relatedness and Textual Entailment". In *9th International Conference on Language Resources and Evaluation*, 216–223. Reykjavik, Iceland: ELRA.

Birjali, Marouane, Abderrahim Beni-Hssane, and Mohammed Erritali. 2016. "Measuring Documents Similarity in Large Corpus Using MapReduce Algorithm". In *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*, 0:24–28. Marrakesh, Morocco: IEEE.

Brin, Sergey, and Lawrence Page. 1998. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". *Computer Networks and ISDN Systems* 30 (1): 107–117.

Cer, Daniel, Yinfei Yand, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yan, Chris Tar, Yun-Hsuan Sun, Brian Strope, and Ray Kurzweil. 2018. "Universal Sentence Encoder". arXiv: `1803.11175v2[cs.CL]`.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation". arXiv: `1406.1078[cs.CL]`.

Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. "Long Short-Term Memory". *Neural Computing* 9:1735–1780.

Hulth, Anette. 2003. "Improved Automatic Keyword Extraction Given More Linguistic Knowledge". In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, 216–223. Sapporo, Japan: ACL.

Iyyer, Mohit, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. "Deep Unordered Composition Rivals Syntactic Methods for Text Classification". In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 1681–1691. Beijing, China: ACL.

Jones, Karen. 1972. "A Statistical Interpretation of Term Specificity and its Application in Retrieval". *Journal of Documentation*. `http://search.proquest.com/docview/57585760/`.

Kamineni, Avinash, Manish Shrivastava, Harish Yenala, and Manoj Chinnakotla. 2018. "Siamese LSTM with Convolutional Similarity for Similar Question Retrieval". In *2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, 1–7. Pattaya, Thailand: IEEE.

Li, Zhenxi, Guillaume-Alexandre Bilodeau, and Wassim Bouachir. 2018. "Multibranch Siamese Networks with Online Selection for Object Tracking". arXiv: `1808.07349[cs.CV]`.

Lin, Yung-Shen, Jung-Yi Jiang, and Shie-Jue Lee. 2014. "A Similarity Measure for Text Classification and Clustering". *IEEE Transactions on Knowledge and Data Engineering* 26 (7): 1575–1590.

Manning, Christopher, Prabhakar Raghvan, and Hinrich Schütze. 2018. *An Introduction to Information Retrieval*. Cambridge, England: Cambridge University Press.

Marujo, Luís, Anatole Gershman, Jaime Carbonell, Robert Frederking, and João Neto. 2013. "Supervised Topical Key Phrase Extraction of News Stories Using Crowdsourcing, Light Filtering and Co-reference Normalization". In *8th International Conference on Language Resources and Evaluation*. Istanbul: ELRA.

Mihalcea, Rada, and Paul Tarau. 2004. "TextRank: Bringing Order into Text". In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, 404–411. Barcelona, Spain: ACL.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Efficient Estimation of Word Representations in Vector Space". arXiv: `1301.3781[cs.CL]`.

Mueller, Jonas, and Aditya Thyagarajan. 2016. "Siamese Recurrent Architectures for Learning Sentence Similarity". In *Thirtieth AAAI Conference on Artificial Intelligence*. Phoenix, Arizona: AAAI Press.

Onan, Aytuğ, Serdar Korukoğlu, and Hasan Bulut. 2016. "Ensemble of Keyword Extraction Methods and Classifiers in Text Classification". *Expert Systems With Applications* 57:232–247.

Rose, Stuart, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. *Automatic Keyword Extraction from Individual Documents*. Sussex, UK: Wiley.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need". In *Neural Information Processing Systems 2017*, 5998–6008. Long Beach, California: Curran Associates, Inc.

Zhu, Wenhao, Tengjun Yao, Jianyue Ni, Baogang Wei, Zhiguo Lu, and Xuchu Weng. 2018a. "Dependency-Based Siamese Long Short-Term Memory Network for Learning Sentence Representations". *PLoS ONE* 13 (3).

Zhu, Zheng, Qiang Wang, Bo Li, Wei Wu, Junji Yan, and Weiming Hu. 2018b. "Distractor-Aware Siamese Networks for Visual Object Tracking". In *ECCV 2018*, 11213:103–119. Munich, Germany: Springer Verlag.