Keyphrase Extraction via an Attentive Model

Marco Passon(✉), Massimo Comuzzo(✉), Giuseppe Serra(✉), and Carlo Tasso(✉)

Artificial Intelligence Laboratory, University of Udine, Udine, Italy
{passon.marco,comuzzo.massimo}@spes.uniud.it,
{giuseppe.serra,carlo.tasso}@uniud.it

Abstract. Keyphrase extraction is a task of crucial importance for digital libraries. When performing automatically a task of this, the context in which a specific word is located seems to hold a substantial role. To exploit this context, in this paper we propose an architecture based on an Attentive Model: a neural network designed to focus on the most relevant parts of data. A preliminary experimental evaluation on the widely used INSPEC dataset confirms the validity of the approach and shows our approach achieves higher performance than the state of the art.

1 Introduction

The continuous growth of textual digital libraries, in terms of both importance and size, urgently requires advanced and effective tools to extract automatically, for each document, the most relevant content. To achieve this goal, the Natural Language Processing Community exploits the concept of "Keyphrases" (KPs), which are phrases that "capture the main topics discussed on a given document" [33].

Extracting KPs from a document can be done manually, employing human judges, or automatically; in the latter case we talk about Automatic Keyphrase Extraction (AKE), a task whose importance has been growing for the last two decades [13]. In fact, the ability to extract automatically keyphrases from documents will make it possible to build more effective information retrieval systems or to summarize [37] or cluster [12] textual documents. Other fields worth mentioning where AKE can be applied are social network analysis [26] and user modeling [27].

Classic AKE approaches rely on Machine Learning algorithms. More specifically, supervised techniques have been used for this task: Naive Bayes [34], C4.5 decision trees [33], Multilayer Perceptrons [3,20], Support Vector Machines [20], Logistic Regression [3,11], and Bagging [16]. Relevant works that investigated the unsupervised extraction of Keyphrases used a language model approach [32] or a graph-based ranking algorithm [23]. However, these approaches achieved lower performance than the one obtained in the supervised case.

M. Passon and M. Comuzzo—Equally contributed.

P. Manghi et al. (Eds.): IRCDL 2019, CCIS 988, pp. 304–314, 2019.
https://doi.org/10.1007/978-3-030-11226-4_24

Due to this difference in performance, in the last years research focus shifted towards the *features* exploited by supervised algorithms. The kind of knowledge encoded in the model can be used to discriminate between different families of approaches: *statistical knowledge* (number of appearances of KPs in the document, TF-IDF, number of sentences containing KPs, etc.), *positional knowledge* (first position of the KP in the document, position of the last occurrence, appearance in the title or in specific sections, etc.), *linguistic knowledge* (part-of-speech tags of the KP [16], anaphoras pointing to the KP [3], etc.), *external knowledge* (presence of the KP as a page on Wikipedia [7] or in specialized domain ontologies [20], etc.). Despite being a subject on several studies, AKE is still an open problem in the NLP field: in fact, even the best techniques for this task reach at best an average performance F1-Score around 50% [16,18].

Although Deep Learning techniques have been recently established as state-of-the-art approaches in many NLP tasks (i.e. sentiment classification, machine translation, etc.), to the best of our knowledge, only a few Deep Learning models addressed the AKE task. Zhang et al. [36] proposed a deep Recurrent Neural Network (RNN) model that combines keywords and context information to be exploited in the AKE task in Twitter domain. In particular, their model consists of a RNN model with two hidden layers: the first captures the keyword information, the second extracts the keyphrases according to the keyword information. Meng et al. [22] addressed the challenge of generating keyphrases that are not present in the text, investigating Encoder-Decoder Neural architecture [31]: the underlying idea is to compress the text content into an hidden representation using an encoder and generate the corresponding keyphrases with a decoder. Basaldella et al. [2] proposed an architecture for AKE based on Bidirectional Long Short-Term Memory (BLSTM) RNN, which is able to exploit both previous and future context of a specific word, differently from simple RNNs that can exploit only the previous context.

In parallel with these initiatives, the class of Attentive Models [31,35] started gaining more and more interest in NLP community, because they have been successfully applied in various text understanding tasks, like neural machine translation from a language to another [1,21,31], abstracting text [25] and sentence summarization [30]. To our knowledge, however, these Models have not been employed in AKE tasks.

In this paper, we investigate the usage of Attentive models in the Keyphrase Extraction domain. The rationale behind this choice is that Attentive Models provide weights that indicate the relevance of a word with respect to its context [1,35] and thus can help in extracting keyphrases. Preliminary experimental results on the widely used INSPEC dataset [16] confirm our hypothesis and show that our approach outperforms the competitors.

2 Keyphrase Extraction Approach

We aim at developing a system that, given a text document, is able to automatically extract its keyphrases. Our solution consists of a neural network architecture that combines Recurrent Neural models with an Attentive component. The

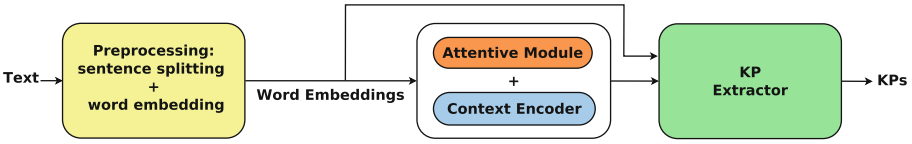

Fig. 1. Overview of the proposed approach.

proposed model takes as input a text and returns as prediction an annotated text (see Fig. 1).

First, text is split into sentences, and then tokenized in words using the library NLTK [4]. Each word is then mapped into a continuous vector representation, called word embedding, that according to recent studies [6,24] represents the semantics of words better than the "one hot" encoding word representation. For our work we used Stanford's GloVe Embeddings [29], since the common datasets adopted for AKE task are rather small, making it difficult to build custom embeddings.

However, when dealing with the keyphrase extraction, words cannot be treated with the same importance (for example, a stop word is less important than a noun, adjectives and adverbs enrich a speech but add almost nothing to the core meaning of it, etc.).

To encode this core feature, we propose to integrate in our pipeline an attentive neural component. In fact, attention models are inspired by human attention mechanisms, that do not use all the available information at a given time, but select the most pertinent piece of information, leaving out the parts considered irrelevant [8,17].

The goal of our *Attentive Model* is to associate to each word of the text an attentive value, i.e. a weight representing the attention level of the word: this information is then exploited in the subsequent processing phase and we claim it can have a significant role for a more effective and precise identification of KPs. To compute such attentive values, the Attentive Module needs, for each word w in the text, (i) the corresponding word embedding and (ii) the so called context, that is a representation of the semantics of the words appearing in the text before and after the word w. The context is computed by a specific module, the *Context Encoder*, which exploits a BLSTM network capable of producing a non-linear combination of the word embeddings belonging to the previous and future surroundings of w.

Finally, to extract keyphrases, the *Extractor module* combines word embeddings and the attentive values (concatenating their results) by means of a BLSTM neural model which is able to analyze word embeddings and their sequential features and to effectively deal with the variable lengths of sentences. For each word, the output consists of three possible classes: `NO_KP` for words that are not keyphrases, `BEGIN_KP` for words corresponding to the first token of a keyphrase, `INSIDE_KP` for a token, other than the first one, belonging to a keyphrase (see Fig. 2 for more details).

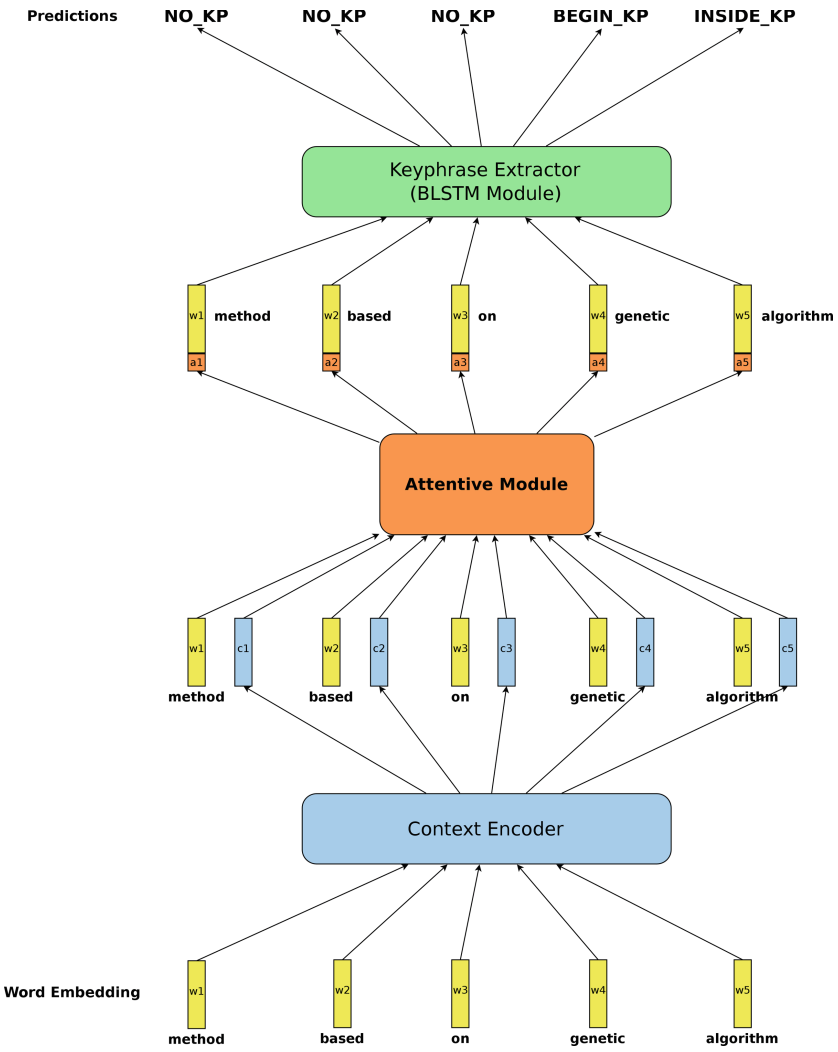

Fig. 2. Detailed schematization of our solution.

2.1 Attentive Module and Context Encoder

To extract the importance of a word in a given sentence we propose to use an attentive model. Our attentive model exploits the semantic representation of each word (the word embeddings) and the context in which the word is located. The main idea is to identify words that are more related to the context. In our architecture the context is defined as the output results of a BLSTM that takes in input the word embeddings of the text.

Let S be the matrix formed by the word embeddings $w_1, w_2, \ldots, w_s$ and C the matrix formed by the context vectors $c_1, c_2, \ldots, c_s$ (see Fig. 2). The size of

each context vector is equal to size of the word embedding vector. Therefore, S and C are both $s \times d$ matrices, where d is the dimension of the word embeddings.

Our attentive model first performs the matrix multiplication between C and the transpose of S, resulting in a new matrix M. More formally, each element of M, $m_{i,j}$ (where i is the row and j is the column) is computed as follows:

$$m_{i,j} = \sum_{k=1}^{d} (C_{i,k} \cdot S_{j,k}) \tag{1}$$

In other words, each element of $m_{i,j}$ is given by the sum, for each word embedding dimension k, of the product between the element in the context matrix $C_{i,k}$ and the element in the word embedding matrix $S_{j,k}$.

Then, we compute the normalization of the matrix M using a softmax layer (that behaves almost like a argmax, but is differentiable) as follows:

$$m_{i,j} = \frac{e^{m_{i,j}}}{\sum_{k=1}^{s} e^{m_{i,k}}} \tag{2}$$

Every item of M is now the range of 0 and 1 and the sum of each row is equal to 1. The element $m_{i,j}$ represents the attention score of the word w_j in the classification context of the word w_i. Through matrix multiplication between the matrices M and S we compute the matrix A, where each row represents the output of the attentive model.

The attentive output is then used as input in a Dense layer in order to manage the contribution of the attentive model in the final word representation. The output vectors coming out from this Dense layer are finally concatenated with the initial embeddings and fed into the classifier BLSTM.

Figure 3 illustrates a single iteration of the attentive model that we just outlined. Specifically, we represented the case where the importance of the single words is computed against the context vector $c3$ and saved into the vector $a3$ (in our case, this represents the third row of A). It is important to point out that there are as many context vectors as there are word embeddings and each context vector is different from the others, thus each subsequent iteration will use a different context vector and will consequently compute a different vector of weights.

2.2 Bidirectional Long Short-Term Memory (BLSTM)

Differently from feedforward neural networks, where the inputs are independent of each other, Recurrent Neural Networks (RNNs) keep an internal state that allows them to process sequences of inputs, with each input related to each other, thus granting the persistence of the information. RNNs are often employed in NLP tasks where the context is an important component needed to compute the predictions.

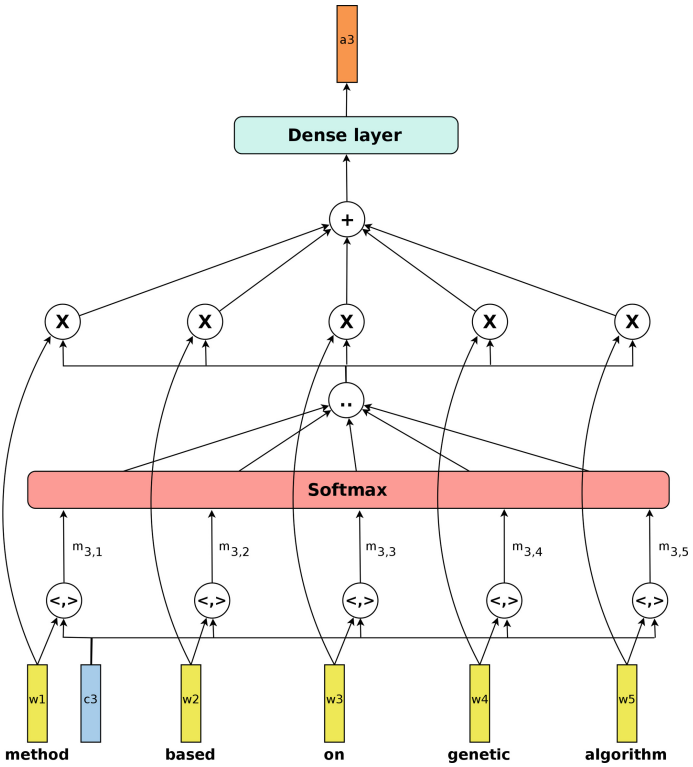

Fig. 3. Outline of a single iteration of the attentive layer.

As Recurrent Neural Network, we adopt the Long Short-Term Memory (LSTM) architecture [15], a common and effective solution employed to reduce the *vanishing gradient* problem [14] that typically affect plain RNNs. In particular an LSTM is defined as follows [9]:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{3}$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{4}$$
$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{5}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{6}$$
$$h_t = o_t \tanh(c_t) \tag{7}$$

where σ is the logistic sigmoid function, i, f, o, and c are the input gate, forget gate, output gate and cell activation vectors, and all b are learned biases.

The first step (Eq. 3) deletes the information coming from the previous element in the input sequence. Next (Eqs. 4 and 5) comes the decision of what information is going to be stored in the cell's state, replacing the one previously forgot: this is done having an input gate i deciding the values to update and then creating a candidate value c. In the last step (Eqs. 6 and 7) the output is

computed: the cell's state passes through a sigmoid layer σ and finally through the tanh function in order to push the values between -1 and 1.

This kind of architecture, however, contemplates only previous information, but in our case, dealing with the AKE task, future information can support the identification of a possible keyphrase. For this reason a variant of the LSTM architecture is used, namely the Bidirectional LSTM architecture [10], that allows us to employ both past and future information. In a BLSTM the output y_t is obtained combining the forward hidden sequence $\overrightarrow{h_t}$ and the backward hidden sequence $\overleftarrow{h_t}$. A BLSTM is then defined as follows:

$$\overrightarrow{h_t} = H(W_{x\overrightarrow{h}} x_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}) \tag{8}$$

$$\overleftarrow{h_t} = H(W_{x\overleftarrow{h}} x_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}) \tag{9}$$

$$y_t = W_{\overrightarrow{h}y} \overrightarrow{h}_t + W_{\overleftarrow{h}y} \overleftarrow{h}_t + b_y \tag{10}$$

3 Experimental Results

In order to validate our solution, we used the well-known INSPEC dataset [16], which consists by 2000 abstract papers written in English extracted from journal papers from the disciplines Computer and Control, Information Technology. The dataset is split in: 1000 documents as training set, 500 documents as validation set, 500 documents as test set.

To write the implementation of our approach we used Pytorch [28]. The GPU employed in our experiments is a GeForce GTX 660 Ti. We train our network aiming at minimizing the Crossentropy Loss and the training is done using the Root Square Mean Propagation optimization algorithm [19]. The data is loaded into the network in batches, where each batch has a size of 32 input items. The experiments have been run with different configurations of the network's parameters, finally obtaining the best results with a size of 30 neurons for the Attentive Model, 150 neurons for the BLSTM used for classification, 150 neurons for its hidden dense layer and a value of 0.5 for the Dropout layer before the final Dense layer. The Pytorch framework does not implement a early-stopping mechanism; for this reason, we empirically set the number of epochs to 14.

Table 1. Performance obtained varying the dimension of the attentive layer.

Embedding	Attentive Dim.	Precision	Recall	F1-score	MAP	F1@5	F1@10
Glove-200 (Baseline)	-	0.326	0.643	0.432	0.356	0.286	0.353
Glove-200	10	0.291	0.624	0.397	0.327	0.271	0.326
Glove-200	20	0.348	0.654	0.455	0.370	0.297	0.371
Glove-200	30	**0.373**	**0.658**	**0.476**	**0.388**	**0.313**	**0.394**
Glove-200	40	0.321	0.648	0.429	0.356	0.287	0.350

The first of our experiments aimed at reproducing Basaldella et al. [2] results (an approach based on word embeddings and BLSTM) in order to create a solid baseline against which we can compare the results obtained by our approach based on an attentive module. Baseline results presented in Table 1 are slightly better than the original ones [2], because here we adopted a different value of the dropout layer.

Table 1 also presents performance varying the dimension of the attentive module: the size of the Dense layer that allows us to weight the attention effect. Note that a small attentive layer does not bring any benefit, on the contrary the performances are lower compared to the baseline where no attentive module is present. The reason behind this behavior may be the attentive layer focusing on a part of information that is too small and leaving other important parts out. Further increasing the dimension of the attentive module causes an improvement to the performances achieving the best score (in all metrics) with an attentive layer of dimension 30. An additional increase of the attentive layer makes it focus on a part of information that is too big, actually not focusing on anything in particular thus not making use of the attention mechanism. In fact, performance are similar to the ones obtained without attentive model.

Finally, we compare our results with state-of-the-art systems, which rely on supervised and unsupervised machine learning techniques (see Table 2). The first system is the one proposed in [2] that uses a BiLSTM architecture (our baseline); the second technique proposed an approach based on Encoder-Decoder Neural architecture [22]; the next three are works presented in [16] that use three different techniques, respectively: n-grams, Noun Phrases (NP) chunking and patterns; the last one [5] relies on a topical representation of a document, making use of graphs to extract keywords. Note, our proposed approach achieves state of the art performance under every measure considered. It is worth noting that we perform better than the results presented in [2] and [22], two recent works that make use of Deep Learning techniques.

Table 2. Comparison results on INSPEC dataset

Method	Precision	Recall	F1-score	F1@5	F1@10
Proposed approach	*0.373*	*0.658*	*0.476*	*0.313*	*0.394*
BiLSTM [2]	0.340	0.578	0.428	-	-
KP Generation [22]	-	-	-	0.278	0.342
n-grams with tag [16]	0.252	0.517	0.339	-	-
NP Chunking with tag [16]	0.297	0.372	0.330	-	-
Pattern with tag [16]	0.217	0.399	0.281	-	-
TopicRank [5]	0.348	0.404	0.352	-	-

4 Conclusion

In this work, we proposed a network that uses an Attentive Model as its core in order to perform automatic keyphrase extraction. The approach has been validated on the well-known INSPEC dataset and the experiments have been performed varying the size of the attentive model. Comparison evaluation shows that our approach outperforms competitive works on all metrics. As future works we intend testing the proposed architecture on other Keyphrase datasets and we will investigate advanced attentive architectures, such as Tree and Graph Attention models that can deal with complex text representation.

Acknowledgements. This project was partially supported by the FVG P.O.R. FESR 2014-2020 fund, project "Design of a Digital Assistant based on machine learning and natural language".

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Basaldella, M., Antolli, E., Serra, G., Tasso, C.: Bidirectional LSTM recurrent neural network for keyphrase extraction. In: Italian Research Conference on Digital Libraries, pp. 180–187 (2018)
3. Basaldella, M., Chiaradia, G., Tasso, C.: Evaluating anaphora and coreference resolution to improve automatic keyphrase extraction. In: Proceedings of International Conference on Computational Linguistics, pp. 804–814 (2016)
4. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python. OReilly Media Inc., Sebastopol (2009)
5. Bougouin, A., Boudin, F., Daille, B.: TopicRank: graph-based topic ranking for keyphrase extraction. In: Proceedings of International Joint Conference on Natural Language Processing, pp. 543–551 (2013)
6. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2498–2537 (2011)
7. Degl'Innocenti, D., De Nart, D., Tasso, C.: A new multi-lingual knowledge-base approach to keyphrase extraction for the Italian language. In: Proceedings of International Conference on Knowledge Discovery and Information Retrieval, pp. 78–85 (2014)
8. Desimone, R., Duncan, J.: Neural mechanisms of selective visual attention. Ann. Rev. Neurosci. **18**(1), 193–222 (1995)
9. Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with LSTM recurrent networks. J. Mach. Learn. Res. **3**, 115–143 (2002)
10. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. **18**(5–6), 602–610 (2005)
11. Haddoud, M., Abdeddaïm, S.: Accurate keyphrase extraction by discriminating overlapping phrases. J. Inf. Sci. **40**(4), 488–500 (2014)
12. Hammouda, K.M., Matute, D.N., Kamel, M.S.: CorePhrase: keyphrase extraction for document clustering. In: Perner, P., Imiya, A. (eds.) MLDM 2005. LNCS (LNAI), vol. 3587, pp. 265–274. Springer, Heidelberg (2005). https://doi.org/10.1007/11510888_26

13. Hasan, K.S., Ng, V.: Automatic keyphrase extraction: a survey of the state of the art. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, (Volume 1: Long Papers), vol. 1, pp. 1262–1273 (2014)
14. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
16. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, pp. 216–223 (2003)
17. Itti, L., Koch, C., Niebur, E.: A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. Patt. Anal. Mach. Intell. **20**(11), 1254–1259 (1998)
18. Kim, S.N., Medelyan, O., Kan, M.Y., Baldwin, T.: SemEval-2010 task 5: automatic keyphrase extraction from scientific articles. In: Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 21–26 (2010)
19. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
20. Lopez, P., Romary, L.: HUMB: automatic key term extraction from scientific articles in GROBID. In: Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 248–251 (2010)
21. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
22. Meng, R., Zhao, S., Han, S., He, D., Brusilovsky, P., Chi, Y.: Deep keyphrase generation. arXiv preprint arXiv:1704.06879 (2017)
23. Mihalcea, R., Tarau, P.: TextRank: bringing order into texts. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (2004)
24. 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, pp. 3111–3119 (2013)
25. Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al.: Abstractive text summarization using sequence-to-sequence RNNs and beyond. arXiv preprint arXiv:1602.06023 (2016)
26. De Nart, D., Degl'Innocenti, D., Basaldella, M., Agosti, M., Tasso, C.: A content-based approach to social network analysis: a case study on research communities. In: Calvanese, D., De De Nart, D., Tasso, C. (eds.) IRCDL 2015. CCIS, vol. 612, pp. 142–154. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41938-1_15
27. De Nart, D., Degl'Innocenti, D., Pavan, A., Basaldella, M., Tasso, C.: Modelling the user modelling community (and other communities as well). In: Ricci, F., Bontcheva, K., Conlan, O., Lawless, S. (eds.) UMAP 2015. LNCS, vol. 9146, pp. 357–363. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20267-9_31
28. Paszke, A., et al.: Automatic differentiation in PyTorch (2017)
29. 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)
30. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685 (2015)
31. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)

32. Tomokiyo, T., Hurst, M.: A language model approach to keyphrase extraction. In: Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment, vol. 18, pp. 33–40 (2003)
33. Turney, P.D.: Learning algorithms for keyphrase extraction. Inf. Retrieval **2**(4), 303–336 (2000)
34. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: KEA: practical automatic keyphrase extraction. In: Proceedings of ACM Conference on Digital Libraries (1999)
35. Xu, K., et al.: Show, attend and tell: neural image caption generation with visual attention. In: International Conference on Machine Learning, pp. 2048–2057 (2015)
36. Zhang, Q., Wang, Y., Gong, Y., Huang, X.: Keyphrase extraction using deep recurrent neural networks on Twitter. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 836–845 (2016)
37. Zhang, Y., Zincir-Heywood, N., Milios, E.: World wide web site summarization. Web Intell. Agent Syst. Int. J. **2**(1), 39–53 (2004)