Automatic Slide Generation for Scientific Papers

Athar Sefid
Pennsylvania State University
azs5955@psu.edu

Jian Wu
Old Dominion University
jwu@cs.odu.edu

Prasenjit Mitra
Pennsylvania State University
pmitra@ist.psu.edu

C. Lee Giles
Pennsylvania State University
giles@ist.psu.edu

ABSTRACT

We describe our approach for automatically generating presentation slides for scientific papers using deep neural networks. Such slides can help authors have a starting point for their slide generation process. Extractive summarization techniques are applied to rank and select important sentences from the original document. Previous work identified important sentences based only on a limited number of features that were extracted from the position and structure of sentences in the paper. Our method extends previous work by (1) extracting a more comprehensive list of surface features, (2) considering semantic or meaning of the sentence, and (3) using context around the current sentence to rank the sentences. Once, the sentences are ranked, salient sentences are selected using Integer Linear Programming (ILP). Our results show the efficacy of our model for summarization and the slide generation task.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Classification and regression trees*; • **Information systems** → *Document topic models*; *Retrieval models and ranking*.

KEYWORDS

Natural Language Processing, Text Mining, Slide Generation, Summarization, Deep Learning

1 INTRODUCTION

Scientific results are usually disseminated by publications and research presentations. The latter has been a convention for almost all scientific domains since it provides an efficient way for others to grasp the major contribution of a paper. Currently, PowerPoint automates slide formating and thus significantly reduces the effort to make professional and useful slides. However, PowerPoint cannot automatically generate slides based on the content of research papers. Automatically generating slides would not only save the presenters' time in preparing presentations, but it also provides a way to summarize paper and it can also be used to generate summaries for papers that do not have publicly available slides.

The two major approaches for summarization are abstractive and extractive methods. Abstractive approaches summarize text using words not necessarily appearing in the original document. Extractive summarization focuses on identifying important constituents usually at the sentence level and connecting them to generate a text snippet, which is usually significantly shorter than the original document. This method has been applied in slide generation tasks, e.g., [6]. This is because (1) the selected sentences have correct grammatical structure and (2) they are scientifically/technically consistent with the original article.

The two major components of extractive summarization are **sentence scoring** and **sentence selection**. The goal is to keep salient sentences while excluding redundant information. Sentence scoring is usually converted to a regression problem. The scores depend on features extracted from the current sentence and its contextual sentences.

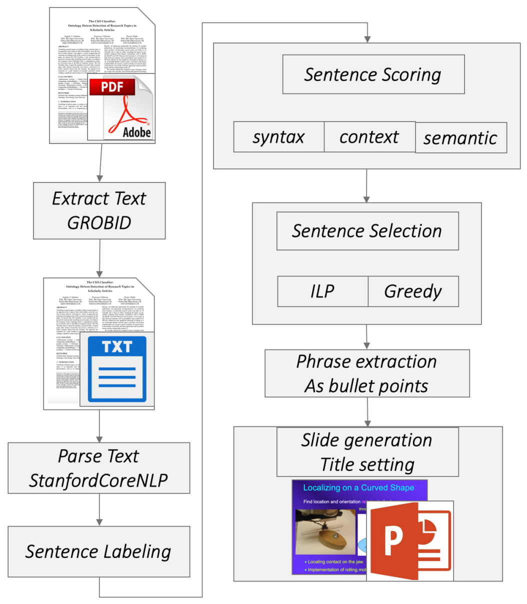

Figure 1: Slide Generation flow chart

The main contributions of this paper are the following.

(1) To the best of our knowledge, we are the first to use Deep Neural models to encode sentences and its context as new features in sentence ranking to build slides.

(2) We combine regression with integer linear programming (ILP) to select salient sentences. Then, noun phrases are extracted from the selected sentences to build the first-level bullet points in slides.

Here, we focus on the textual components. This approach can later be enhanced with visual effects, e.g., figures and tables, in order to make better more complex slides.

2 RELATED WORK

There has been much work on text summarization using both abstractive and extractive methods. However, automatic generation of slides, which can be seen as a specific form of summarization of scholarly papers, has not been well studied.

2.1 Summarization

Unsupervised Models. Identifying important sentences for generating a limited length summary can be formalized as an optimization problem which can be NP-hard. Maximum Marginal Relevance [2] has been used to heuristically select salient sentences while keeping redundancy with the selected summary at the lowest point. Sentence selection could be converted to the Knapsack problem by maximizing the total scores of the selected sentences given the length limit and can be solved by Integer Linear Programming (ILP). ILP is an optimization method with linear constraints and objective functions [13]. Graph-based summarization methods model a document as a graph, in which vertices are sentences and edges are the similarity of vertices. One example is TextRank [14] which extends Google's PageRank. TextRank selects important sentences that have a high degree of overlapping tokens with other sentences. LexRank [5], a stochastic graph-based method, measures the sentence importance by eigenvector centrality of the sentences in the graph.

Feature-Based Models. Many traditional extractive summarization approaches use feature-based models such as term frequency, an effective feature. Other features include the length and position of sentences [18]. The Support Vector Regressor was applied to score and extract salient sentences [9]. Others have modeled a document as a sequence of sentences and label each sentence with 1 or 0 in which 1 indicates that the corresponding sentence belongs to the summary [19] and a 0 not. Hidden Markov models (HMM) have been adopted to solve such sequence labeling problems [4].

Deep Neural Network Models. State-of-the-art models now use deep neural networks which benefit from both the semantics of the sentences and their structure [7, 8]. Zhou et al. used multiple layers of bidirectional Gated Recurrent Units (GRU) to jointly learn the scoring process and the selection of sentences [23]. This work applies bi-GRU to make sentence embeddings and then combines sentence vectors with another bi-GRU layer to make document level representations.

2.2 Slide Generation

Previously, a method was proposed to generate slides from documents by identifying important topics and then adding their related sentences to the summary [12]. A tool called PPSGen applies a Support Vector Regressor for sentence ranking and then ILP to select important sentences with a set of sophisticated constraints [6]. Another method generates slides by phrases extracted from papers [22]. The model learns the saliency of each phrase and the hierarchical relationship between a pair of phrases to make the bullet points and to determine their place in the slide. However, their model was tested on a limited set of only 175 paper-slide pairs.

Compared with previous works, we use a combination of feature based and deep neural network methods to score sentences. Then, ILP is applied to build the summary. To transform the summary into a slide format, first-level bullet points are generated using key phrases extracted from the selected sentences.

3 MODEL

The summarization system consists of 2 steps. The first is to train a model to calculate scores used to identify important sentences. The second is to extract salient sentences under constraints. The model is trained on a corpus of aligned pairs of papers and slides. In the training process, we investigate Convolutional Neural Networks (CNN), Gated Recurrent Unit (GRU), and Long Short Term Memory (LSTM) using pre-trained word embeddings (WE) to represent the semantic feature of sentences. This semantic representation is enhanced by combining other surface features of the current sentence and its contextual features. The high-level architecture is shown in Figure 1.

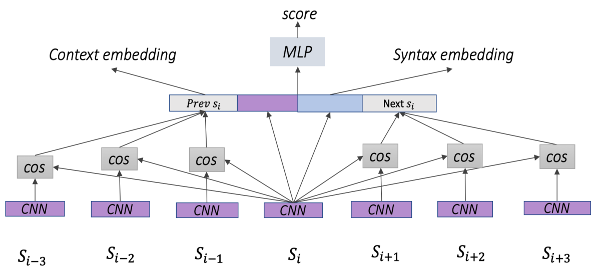

Figure 2: Architecture to predict salience score of a sentence S_i. $S_j (j \neq i)$ are contextual sentences. Three types of embeddings are combined as an input to MLP to output the score of sentence S_i. The CNN could be replaced with LSTM or GRU.

3.1 Sentence Labeling

In this section, we describe how to generate the ground truth by assigning scores to sentences in a paper given a paper-slide pair. An academic paper D can be represented as a sequence of sentences $D = \{s_1, s_2, \ldots, s_n\}$. Each sentence is represented by the set of its unigrams and bigrams, such as $s_i = \{t_1, t_2, \ldots, t_k, t_1t_2, t_2t_3, \ldots, t_{k-1}t_k\}$ where t_i is the i-th token in the sentence. The corresponding presentation of the document D can be represented as a sequence of slides $P = \{p_1, p_2, \ldots, p_m\}$. The salience score of each sentence is determined by the highest Jaccard similarity of unigrams and bigrams between a sentence and all slides in the corresponding presentation.

$$\forall s_i \in D, score_{s_i} = \max_{p_j \in P} Jaccard(s_i, p_j) \tag{1}$$

The Jaccard similarity of s_i and p_j sets is defined as $Jaccard(s_i, p_j) = \frac{s_i \bigcap p_j}{s_i \bigcup p_j}$.

Having the labeled sentences with their scores, we can start training different models to learn the scoring functions.

3.2 Sentence Ranking

The goal of the sentence ranking module is to train a model $f(s_i|\phi)$ that minimizes the Mean Square Error (MSE) defined as below:

$$\forall s_i, 0 \leq f(s_i|\phi) \leq 1 \tag{2}$$

$$MSE = \sum_{s_i} (f(s_i|\phi) - score_{s_i})^2 \tag{3}$$

in which ϕ is sentence embedding combined with syntactic and contextual features.

The sentence ranking model consists of three modules. The first models the meaning of a sentence using a deep neural network that embeds the sentence itself into a vector. The second evaluates the capability of the sentence to summarize its contextual sentences. The last extracts a variety of surface features from the sentence structure and its position in the paper. The results from all of the three modules are combined to build the input for the final Multi-Layer Perceptron (MLP) with two hidden layers which acts as a regressor to predict the final score for a sentence. Figure 2 depicts the embedding layers for the construction of sentence representation vectors.

3.2.1 *Semantic Embedding.*

In this layer, the semantic of a sentence is encoded into a vector. We compare the performance of both convolutional and recurrent networks in order to choose the best system for semantic embedding.

Convolutional Neural Network (CNN): In this layer, the semantic representation of a sentence s_i is generated by concatenating embeddings of bi-grams. The intuition is that bi-grams preserve the sequential information of the text. Convolutional layers and element-wise max-pooling on top of the bigram embeddings capture important patterns in the sentence that match similar patterns in the reference slides.

The bigram representations are calculated by concatenating pre-trained word embeddings of tokens in the bigram.

$$bigram_j = [v_j, v_{j+1}] \tag{4}$$

In which v_j and v_{j+1} stand for the current and the next word vectors in sentence s_i.

$$V_{bigram_j} = tanh(bigram_j * W_c + b) \tag{5}$$

Where filter $W_c \in \mathbb{R}^{2|v_j|*|v_j|}$ is applied on a vector of two words to make new features. The activation function is the Hyperbolic Tangent and b is bias term.

$$V_{s_i} = \max_{0<j<|s_i|-1} V_{bigram_j} \tag{6}$$

where the max function is an element-wise max pooling and V_{s_i} is the semantic embedding of sentence s_i [3].

Recurrent Neural Network (RNN): The convolutional layers above can model the sequential patterns up to 2 consecutive tokens. However, RNNs are able to capture the dependencies of tokens in long sentences. Both LSTM and GRU have been widely used to remember long term dependencies by keeping memory/state from previous activations instead of replacing the entire activation like a vanilla RNN.

LSTM: The LSTM tracks long term dependencies via input (i_t), forget (f_t), and output (o_t) gates; the input gate regulates how much of the new cell state to keep, the forget gate forgets a portion of existing memory and the output gate sends important information in the cell state to the next layers of the network.

GRU: The GRU operates using a reset gate and an update gate. The reset gate decides how much of the past information to forget, and the update gate helps the model to determine how much of the past information needs to be passed to the future.

The input to the recurrent units are pre-trained word vectors and the unit outputs are combined to build the sentence semantic embeddings.

3.2.2 *Context Embedding.*

The context of a sentence is defined as sentences before and after the current sentence. Intuitively, the sentence that includes an abstract of its context is more important and should have a higher chance to be selected for the summary of the paper. For instance, some paragraphs are ended by a sentence that summarizes the whole paragraph. This sentence probably has a high word overlap with other sentences of the paragraph. Contextual sentence relations have been used to model attention weights of sentences and then to identify important sentences [17]. The context information is embedded as the following vectors.

$$prev_{S_i} = \begin{bmatrix} cos(V_{s_i}, V_{s_{i-1}}) \\ cos(V_{s_i}, V_{s_{i-2}}) \\ cos(V_{s_i}, V_{s_{i-3}}) \end{bmatrix}, next_{S_i} = \begin{bmatrix} cos(V_{s_i}, V_{s_{i+1}}) \\ cos(V_{s_i}, V_{s_{i+2}}) \\ cos(V_{s_i}, V_{s_{i+3}}) \end{bmatrix} \tag{7}$$

3.2.3 *Syntactic Embedding.*

The models suggested in the above focus on representing the meaning of the sentences and their context into vectors. Although semantic features are strong metrics for identification of salient sentences, surface features are also proved to be important [6]. The surface features we extracted are tabulated in Table 1.

The "section" feature is a one-hot vector to represent the section that sentence belongs to. Certain sections, such as *conclusion*, can be more important to indicate sentence importance compared with other sections such as *acknowledgment*. Table 2 shows the average scores of the sentences in a set of pre-defined sections calculated from the labeled paper-slide pairs. The table indicates that *Introduction* and *Conclusion* sections are relatively more important than the other sections.

The second feature is the relative position of the sentence in the section in units of sentences.

The other features #NP, #VP, #S, and "height" are obtained from the parse tree of the sentence. #NP shows the number of noun phrases in the sentence, #VP counts verb phrases, #S indicates the number of sub-sentences and "height" is the height of the sub-tree. We used Stanford CoreNLP that outputs tree structures consisting of constituent noun and verb phrases.

The stop_word%, #tokens, and "length" are ratios of the stop words among tokens, number of tokens and number of characters in the sentences, respectively.

The "paper_title_similatiry" is the Jaccard similarity between the tokens in the title of the paper and the sentence. The "section_title_similarity" is the Jaccard similarity of the section title and the current sentence.

The "average_IDF" feature is the average of IDF scores of tokens of the sentence and "average_TF" takes the average of the TF scores.

3.3 Sentence Selection

The model above generates a salience score for each sentence in the paper. Next, we compared a greedy method and the ILP for sentence selection.

3.3.1 Greedy Method. In this intuitive approach, sentences are ranked based on their salience scores and then are added to the summary while the bigram overlap is less a threshold θ and the summary size does not exceed a maximum limit. The threshold parameter is tuned to be $\theta = 0.7$ as higher or lower amounts result in worse performance.

3.3.2 Integer Linear Programming (ILP). ILP is an optimization problem with linear constraints and objective functions. ILP restricts some of the variables to be integers and it is NP-complete. We adopted the IBM CPLEX Optimizer[1], which has been used to solve ILP problems efficiently.

We aim to maximize the following objective function:

$$\max \sum_{i \in N_s} l_i W_i x_i, \quad \text{subject to:} \tag{8}$$

$$\sum_i l_i x_i < maxLen, \quad \forall i,\ x_i \in \{0, 1\} \tag{9}$$

where W_i is the sentence scores generated by the deep model suggested in the paper. The x_i is a binary variable showing whether sentence i is selected for the summary or not. The l_i is the number of characters in sentence i and Eq. (9) sets a constraint to limit the size of the summary to a maximum length $maxLen$. Existence of l_i in the Eq. (8) helps to penalize short sentences.

3.4 Slide Generation

Typical presentation slides include a limited number of bullet points which is often the first-level of the slide structure. These bullet points are usually noun phrases or shortened versions of sentences. Some slides may contain second-level bullet points for further breakdowns. Less than 8% of the content of the presentations in the ground truth corpus is covered in third-level bullets. Therefore, we generate slides with up to 2 bullet levels. Slide titles on average contains 4 words and either Level 1 or Level 2 bullets contains on average 8 words. Each slide is on average made of 36 words in 5 bullets and each level-1 bullet includes 2 second-level bullets.

The deep neural model first ranks the sentences and then ILP helps to select salient sentences by considering the length of the sentences and length limit for the generated presentation. These selected sentences are considered as the second-level bullets. The first-level bullets are the noun phrases extracted from the sentences. Some of the noun phrases are removed from the candidate set if (1) They have more than 10 tokens.(2) They have only 1 token. (3) Their DF is higher than 10. Noun phrases with a high degree of document frequency are not informative enough to be added as a bullet point such as "the model".

Once a set of candidate noun phrases is selected, they are sorted based on the average frequency of their tokens. The noun phrases and the sentences containing them are added to the slides in order of their frequency until all of the selected sentences by the model are covered in the slides. The slide generation process is demonstrated in Algorithm 1.

In order to set the titles of the slides, the section of the first sentence in the slide is determined and then the heading of the section is borrowed from the paper as the title of the slide. The heading is truncated to the first 5 tokens. We constrain a maximum of 4 sentences per slide. If a topic has more than 4 related sentences, the slide is split into two distinct ones.

```
Data: sentences
Result: slides
ranked_sents = rank(sentences);
selected_sents = ILP(ranked_sents);
NPs = {} ;     // dictionary of NPs and a list of its sentences
for $s \in selected\_sents$ do
    for $np \in noun\_phrases(s)$ do
        NPs[np].add(s)
    end
end
freq_rank(NPs);     // rank noun phrases based on frequency of tokens
slides=[];
while $selected\_sents \neq \emptyset$ do
    for $np, sents \in NPs$ do
        splides[np] = [];
        for $s \in sents$ do
            if $s \in selected\_sents$ then
                slides[np].add(s);
                selectet_sents \ {s};
            end
        end
    end
end
distribute_sents(slides) ;     // allow a maximum of 4 sentences per slide
```

Algorithm 1: Bullet Point Generation.

4 EXPERIMENTS

4.1 Data

The dataset is a collection of 1200 scientific papers and their corresponding presentation slides made by their authors. This dataset is adopted from a previous project called PPSGen [6]. The dataset contains conference proceedings in computer science in PDF format. To extract the header metadata and content of the paper from PDF files, GROBID [11] is used and information is transformed into an XML file with the TEI (Text Encoding Initiative) format, a comprehensive encoding standard for machine-readable texts.

[1]https://www.ibm.com/products/ilog-cplex-optimization-studio

Table 1: Sentence surface features

Feature	Type	Description
section	one-hot vector	Section (Intro, Related, Model, Result, Conclusion, Acknowledge) of the sentence
position	index	Position of the sentence in the section
height	integer	Height of the Parse tree of the sentence
#NP	integer	Number of noun phrases in the sentence
#VP	integer	Number of verb phrases in the sentence
#S	integer	Number of sub-sentences in the sentence
length	integer	Number of characters in sentence
stop_word%	float	Ratio of the stop words in the sentence
#tokens	integer	Number of tokens in the sentence
paper_title_similarity	float	Jaccard similarity of the sentence to the title of the paper
section_title_similarity	float	Jaccard similarity of the sentence to the title of the section
average_IDF	float	Average of the IDF scores of tokens
average_TF	float	Average of the frequency of tokens in the paper

The slides are either in PDF or PPT format. The PPT files are first converted to PDFs. Then, the text in presentations is extracted by the LibreOffice pdftotext tool.

4.2 Model Configuration

Stanford CoreNLP 3.9.2 is used to tokenize and lemmatize sentences to the constituent words and also to build the parse trees. The word embeddings are initialized with GloVe 1.2 [16] 50-dimensional vectors trained on Wikipedia. The pre-trained GloVe vectors contain 400,000 words and cover 98.9% of our model vocabulary. The rest of the word embeddings are randomly initialized using a Gaussian distribution. Drop out layers with probability $p = 0.3$ are added to the recurrent model to randomly remove some of the neurons from the network in order to prevent over-fitting [20]. Adam optimizer with L2-norm regularization and mini-batches of size 100 are used to train the model. The learning rate was set to 0.0001 and the model is trained for 50 epochs. We truncate each sentence containing more than 200 words.

4.3 Evaluation Metric

The Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [10] is a standard metric for automatic evaluation of machine-generated summaries. ROUGE-N is an n-gram overlap between a candidate summary and a reference summary. The recall-oriented score (Re) and the precision-oriented score (Pr) are computed as follows:

$$Re = \frac{\sum_{g \in Reference} Count_{\text{match}}(g)}{\sum_{g \in Reference} Count(g)} \tag{10}$$

$$Pr = \frac{\sum_{g \in Reference} Count_{\text{match}}(g)}{\sum_{g \in Candidate} Count(g)} \tag{11}$$

Table 2: Score distribution of sentences in the ground truth corpus.

Section	Intro	Related Work	Model	Result	Conclusion	Acknowledgement
Avg Score	**44.29**	38.92	39.09	42.42	**51.02**	31.18
#sentences	25494	18316	8924	6955	7036	722

where g stands for an n-gram in the reference summary $Reference$, n is its length, $Count_{\text{match}}(g)$ is the number of shared n-grams between the candidate $Candidate$ and the reference summary and $Count(g)$ is the number of n-grams in reference or candidate summary. The F-measure score(F) is harmonic mean of Pr and Re.

ROUGE-L [10] is similarly defined based on the Longest Common Sub-sequence (LCS) and ROUGE-W is Weighted Longest Common Sub-sequence developed to differentiate LCSs of different spatial relations. ROUGE-W gives more weight to the sequences that are spatially closer to each other [10].

Empirically, ROUGE-2 is more suitable for single document summaries compared with ROUGE-1 [10, 15].

4.4 Result

We compared our model with the following baselines listed as follows:

(1) AvgTFIDF: This model ranks sentences based on the Avg(TF)× Ave(IDF) named AvgTFIDF scores of the tokens in sentence. The model greedily adds the sentences to the summary until it hits the size limit.
(2) TextRank: TextRank is a graph-based algorithm where vertices are sentences and the edge weights are the cosine similarity of the sentence embeddings [1].
(3) MEAD: MEAD is a publicly available platform for multilingual summarization. The platform implements a combination of position-based and centroid-based algorithms to score the sentences [21].

The results for the ROUGE scores are shown in Table 3. Our model with the CNN neural structure and ILP sentence selector achieved the best F scores in terms of ROUGE-2, ROUGE-L, and ROUGE-W compared with the baselines.

Intuitively, LSTMs should have shown a better performance in representing sentences with long dependencies. However, they are prone to over-fitting since there are many parameters to be tuned.

ILP always outperforms the simple greedy algorithm. The reason is that it is able to recognize if adding multiple shorter and less important sentences will result in better ROUGE scores compared with having fewer long and more important sentences.

Table 3: ROUGE Scores

model	selector	ROUGE-1			ROUGE-2			ROUGE-L			ROUGE-W		
		Pr%	Re%	F%	Pr%	Re%	F%	Pr%	Re%	F%	Pr%	Re%	F%
AvgTFIDF	Greedy	41.94	30.02	32.06	11.50	7.89	8.46	21.37	15.77	16.93	8.45	1.56	2.46
AvgTFIDF	ILP	41.45	30.53	32.24	11.31	8.00	8.48	23.17	17.36	18.51	9.21	1.72	2.71
TextRank	Greedy	41.04	31.87	32.79	11.09	8.28	8.60	22.61	17.75	18.49	8.77	1.71	2.66
MEAD	Greedy	**44.58**	**35.00**	**35.86**	**12.49**	9.31	9.65	22.84	18.19	18.84	8.90	1.76	2.73
GRU	Greedy	40.53	33.63	35.54	11.18	8.86	8.91	20.09	16.80	17.01	7.77	1.66	2.53
GRU	ILP	40.28	34.17	33.73	11.06	8.96	8.92	22.55	18.99	19.16	8.84	1.89	2.88
LSTM	Greedy	40.04	34.26	33.61	11.60	9.44	9.37	22.20	18.87	18.93	8.71	1.88	2.86
LSTM	ILP	40.26	34.94	34.08	11.58	9.46	9.59	22.58	19.46	19.40	8.83	1.94	2.94
CNN	Greedy	41.92	33.93	34.28	12.33	9.50	9.71	21.65	17.67	18.10	8.49	1.76	2.73
CNN	**ILP**	41.80	34.74	34.74	12.19	**9.70**	**9.79**	**23.45**	**19.46**	**19.79**	**9.24**	**1.94**	**2.98**

5 CONCLUSION AND FUTURE WORK

Although summarization has been well studied, the task of automatically generation of slides is relatively new. This paper uses neural network models to encode syntax, semantics, and context of sentences as features to automatically generate slide content. A multilayer perceptron on top of sentence embeddings acts as a regressor that ranks the sentences. Our model then selects salient sentences with the constraint of a limit on summary length. Our method shows higher ROUGE scores compared with the introduced baselines which we interpret to mean that the slides generated have a high overlap with manually made slides.

The software implementation of the model is available at GitHub: https://github.com/atharsefid/Automatic-Slide-Generation.

6 ACKNOWLEDGEMENTS

The National Science Foundation is gratefully acknowledged for partial support.

REFERENCES

[1] Federico Barrios, Federico López, Luis Argerich, and Rosa Wachenchauzer. 2016. Variations of the Similarity Function of TextRank for Automated Summarization. *CoRR* abs/1602.03606 (2016). arXiv:1602.03606 http://arxiv.org/abs/1602.03606

[2] Jaime G Carbonell and Jade Goldstein. 1998. The Use of MMR and Diversity-Based Reranking for Reodering Documents and Producing Summaries. (1998).

[3] 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 (2011), 2493–2537.

[4] John M Conroy and Dianne P O'leary. 2001. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 406–407.

[5] Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research* 22 (2004), 457–479.

[6] Yue Hu and Xiaojun Wan. 2015. PPSGen: Learning-based presentation slides generation for academic papers. *IEEE transactions on knowledge and data engineering* 27, 4 (2015), 1085–1097.

[7] Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. 2015. Summarization based on embedding distributions. In *Proceedings of the 2015 conference on empirical methods in natural language processing.* 1984–1989.

[8] Chen Li, Xian Qian, and Yang Liu. 2013. Using supervised bigram-based ILP for extractive summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1004–1013.

[9] Sujian Li, You Ouyang, Wei Wang, and Bin Sun. 2007. Multi-document summarization using support vector regression. In *Proceedings of DUC*. Citeseer.

[10] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* (2004).

[11] Patrice Lopez. 2009. GROBID: Combining Automatic Bibliographic Data Recognition and Term Extraction for Scholarship Publications. In *Proceedings of the 13th European Conference on Research and Advanced Technology for Digital Libraries (ECDL'09).* 473–474.

[12] Utiyama Masao and Hasida Kôiti. 1999. Automatic slide presentation from semantically annotated documents. In *Proceedings of the Workshop on Coreference and its Applications.* Association for Computational Linguistics, 25–30.

[13] Ryan McDonald. 2007. A study of global inference algorithms in multi-document summarization. In *European Conference on Information Retrieval.* 557–564.

[14] Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing.*

[15] Karolina Owczarzak, John M Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An assessment of the accuracy of automatic evaluation in summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization.* Association for Computational Linguistics, 1–9.

[16] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).* 1532–1543.

[17] Pengjie Ren, Zhumin Chen, Zhaochun Ren, Furu Wei, Jun Ma, and Maarten de Rijke. 2017. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *Proceedings of the 40th International SIGIR Conference on Research and Development in Information Retrieval.* 95–104.

[18] Pengjie Ren, Furu Wei, CHEN Zhumin, MA Jun, and Ming Zhou. 2016. A redundancy-aware sentence regression framework for extractive summarization. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers.* 33–43.

[19] Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document summarization using conditional random fields.. In *IJCAI*, Vol. 7. 2862–2867.

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[21] DR Timothy, T Allison, S Blair-goldensohn, J Blitzer, A Elebi, S Dimitrov, E Drabek, A Hakim, W Lam, D Liu, et al. 2004. MEAD a platform for multidocument multilingual text summarization. In *International Conference on Language Resources and Evaluation.*

[22] Sida Wang, Xiaojun Wan, and Shikang Du. 2017. Phrase-based presentation slides generation for academic papers. In *31st AAAI Conference.*

[23] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. Neural document summarization by jointly learning to score and select sentences. *arXiv preprint arXiv:1807.02305* (2018).