Generation of Research Paper Titles

Christopher Gossmann
Colegio Interamericano
Guatemala City, Guatemala

1. ABSTRACT

Can NLP accurately and effectively generate research paper titles? In this research paper, an effective and accurate artificial intelligence NLP model is tried to be determined by evaluating various models and methods for title generation. Titles are a vital part of any work. People tend to struggle to find good titles for their work, including researchers. A title must be representative of the whole paper and its topic, so it must be carefully chosen. Titles are the main reason people read a paper, so it's very important to attract the audience of the researcher. Apart from this, normalizing research paper titles with generated ones can eliminate bias and exaggeration from them. To determine an accurate and effective model, various methods for title generation were tested given the abstract of a research paper. The methods tested vary from very simple, like "bag of words", to complex with GPT-2. The best result was getting a sentence similarity of 0.9032, which was 66% higher than the initial model. Through this project it is confirmed that generating good titles with NLP is a hard task, but the results from the models can facilitate the process of creating a good and compact research paper title.

2. INTRODUCTION

Titles are a compact representation of a paper, which lets people quickly capture what it is about without going into the details. Automatic research paper title generation is a complex process that not only has to extract meaning out of the text and put it into a few words, but also has to construct the title in a human-readable form. These programs could potentially save researchers time and effort used in trying to determine the best possible way to express what the paper is all about in just a few words. The used NLP model for this task needs to be determined as efficient and accurate, so researchers can confidently use it to generate titles for their works.

NLP stands for natural language processing, which means that language data -in this case research papers- is worked with in the models. Title generation given the abstract of a research paper is a supervised problem since research papers titles are the goal so they can be used to check how well the model is doing. As is evident, the output of the models are words in the form of a title.

The approach to effectively create a title generating program is to develop a model that learns the correlation between each word in an abstract and the words composing the title. Then the learned correlations between the corpus of texts and the corresponding titles to generate titles for validation research abstracts, which the model hasn't seen or learned from yet. The models compared in this paper convert each word into numerical data in various forms so they can process and make numerical calculations to learn correlations and potentially generate a good research paper title.

3. BACKGROUND

Background approaches for this task.

3.1 Neural Networks in NLP tasks

Neural networks are widely the most used models for almost every artificial intelligence task, including natural language. Neural networks are inspired by the human brain, trying to replicate the connections made by neurons to identify patterns in data. Here, there are layers of neurons: input layer, hidden layers, and output layer. Each neuron connects with another, and has a specific weight and threshold, which determines the output passed to the next neuron. Each neuron has its own activation formula, which is somewhat like:

$$\sum_{i=1}^{m} w_i x_i + bias = w_1 x_1 + w_2 x_2 + \ldots + bias$$

Before using this activation formula to generate the output, weights are assigned to determine the importance of each variable in the model. These weights are changed during the training phases to best suit the problem. In the case of title generation, the weights indicate how important is each word in the input layer. For this "simple" neural network, the limitation is the word order, which leads to a title not readable or understandable by humans.

3.2 GPT-3 as a natural language generator model

Open AI's GPT-3 is a massive natural language model designed to replicate human language thinking as closely as possible. This is a model aimed to be the most human-like possible, trying to solve any problem in a zero- one- or few shots situation. A shot is an example of the solution of a like-problem the model is trying to solve. This provides an easy method to try to solve NLP tasks, but it also means that it isn't very open for fine tuning options. This model was developed with 175 billion parameters, which gives it a massive power for a variety of tasks in natural language. Unfortunately, the size of this model makes it difficult to train on a personal computer. Therefore, this model isn't a very efficient way of generating titles for research papers.

4. DATASET

The dataset used in this project was provided by the Inspirit AI collection of datasets for their mentorship program. This dataset consists of language and numerical data. It includes a total of 20972 rows of data for research papers of a variety of subjects. There are a total of 8 columns, being the ones useful for this NLP task the first two: title and abstract. The other 6 columns represent the categories which the research paper belongs to, being the value 1 if the paper fits the category or 0 otherwise.

Data preprocessing was done in different ways depending on how the model used worked. The first and second models, most common words and adjacency word matrix, were used with tokenized and cleaned text. For this process, each word in the data of titles and abstracts had to be separated from the rest of the text. Now that each word was taken out individually, the text had to be cleaned. All punctuation was removed, to avoid the model interpreting punctuation as important words. Every word was converted to lowercase to avoid doubling the vocabulary due to capitalization of letters. Then, all stop words - like pronouns or articles - were removed from the list of word tokens to avoid giving importance to these kinds of words. Finally, each word token was lemmatized, converting every word to the room word.

Figure 1: Example of word tokenization
Source: Towards Data Science

The 3rd model, a long-short term memory recurrent neural network, had to work with numbers, so the language data had to be converted into numerical data. For this, each word in the corpus of text used in the model had to be converted into a vector. To do this, Gensim's Word2Vec model was used. Word2Vec learns the relationships between words of unchanged text, producing an output of one vector for each word. These vectors have dimensions with linear relationships, which allow for various mathematical operations with these.

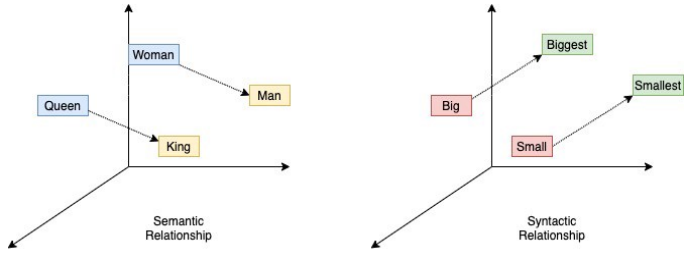


Figure 2: Visualization of relationships established by Word2Vec model
Source: Towards Data Science

5. METHODOLOGY / MODELS

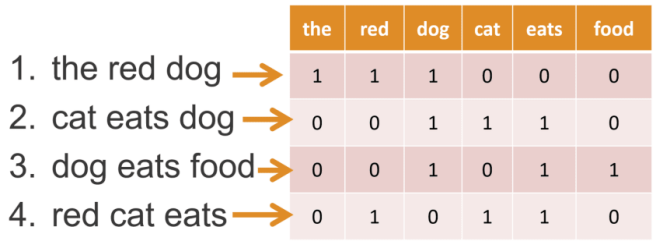
Models tested for title generation

5.1 Most Common Words

The first method evaluated was a most common words model. The method was carried out with the help of sklearn's CountVectorizer. This model, also referred to as "bag of words," creates a one-hot-encoding for each word in the text, generating a vocabulary for known words and a measure of the presence of those words in the text. As the name describes, it's just a "bag of words", which doesn't take into account order or sentence structure, just the two features mentioned before.

The model was fit to the entire dataset, of both titles and abstracts.This was done preprocessing the data with tokenization and cleaning as described earlier in order to

avoid having insignificant words with high presence used by the model.. Then the nature of the model, measuring presence of words in text, was used to determine the most frequent words of an abstract. Since it was the most basic model used, the length of the title was generated with a random number between 6 and 10. Then the most frequent words were put together to form a title of this length. As expected, the generated titles didn't make much sense because it was generated by a simple model, which can't put words in an order that makes sense to humans.



	the	red	dog	cat	eats	food
1. the red dog →	1	1	1	0	0	0
2. cat eats dog →	0	0	1	1	1	0
3. dog eats food →	0	0	1	0	1	1
4. red cat eats →	0	1	0	1	1	0

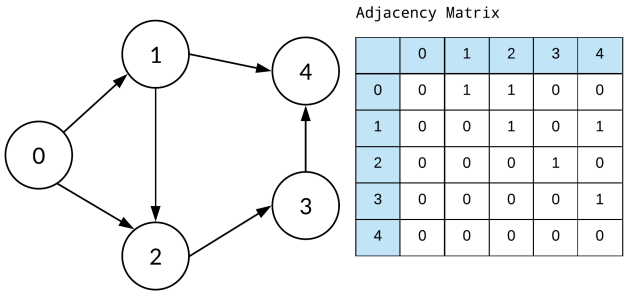
Figure 3: Depiction of how bag of words works
Source: Medium: Spam filtering using Bag-Of-Words

5.2 Adjacency matrix of words

This second model was a little more complex than the first one, and therefore generated slightly better results. The idea of an adjacency matrix of words is to create an adjacency matrix that counts how frequently words occur next to each other. Generating titles with this method rather than the "bag of words", which ignores word order, has the advantage that it generates titles in a more human readable way. Humans have put together words that make sense together in the research papers used for this study, and the model uses these words that make sense together to ensemble a more reasonable set of words for the title.

First, the corpus of text for each sample was cleaned and tokenized. With those tokenized and clean words, a vocabulary of tokens and a list of cleaned tokens were made. The vocabulary was made by removing repeated words from it. Each word had an index specific to it, which will be later used to retrieve the word. The list of cleaned tokens kept the original order of the words, without removing any. The adjacency matrix of words was initialized by creating a list of lists, the amount of words in the tokens vocabulary. Each list of the matrix was filled with zeros corresponding also to the amount of words in the tokens vocabulary, creating a symmetric matrix, which is easier to handle. To write the frequencies of the adjacency of words, a loop with an inner loop was utilized. The first loop went through the vocabulary of tokens, and the second one went through the list of cleaned tokens. Whenever the token of the vocabulary and the token of the list of cleaned tokens were the same, the tokens on the right and left of the token of the cleaned list were looked for in the vocabulary, and then in the intersection of the word and the adjacent word, 1 was added to the frequencies. This process was made for all words except for the first and last words, which don't have one of the adjacency words

In order to generate the actual titles the order of the words must be defined. To define the first word used, sklearn's TfidfVectorizer. This model converts words into numerical data by converting each token to a weighted number, which is a measure of originality of the word. This model was used to search for the most important word in the sample being used in the model. After determining this word, it was put at the beginning of a list of words to form the title. Then a loop was used to go through the adjacency matrix and append the most frequent adjacent word of the previous word. This process was repeated for 5 to 9 times, to get a total title length between 6 and 10 words. To avoid having repeated words, each word index had to be tracked to prevent the model from using that same word more times. This method got slightly better results than the most frequent words model.



	0	1	2	3	4
0	0	1	1	0	0
1	0	0	1	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	0	0	0	0	0

Figure 4: Visualization of an adjacency matrix
Source: Codepath: Graphs

5.3 Long-Short Term Memory neural network

This third, more advanced and complex method, achieved greater results than the previous ones. This was the first model used that was actually very complex. LSTMs neural networks are an area of deep learning, and therefore it's troubling to understand it in great detail. These are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is of great usage for this problem because it learns the correlation between words, creating a more

human readable and understandable text. One of the main advantages of this model is that it can keep information about past data depending on inputs, meaning it can retain important information for later use. The main use for LSTMs is to generate output by learning the context of the input, which is very useful for this task.

Most neural networks require a fixed input and output size, so to make the process simpler, the data had to be altered in order to fit the model. The chosen input size was 150 words, and the chosen output size was 6 words. In order to make this plausible, the data had to be cut only to those research papers that had an abstract length of more than 150 words and a title length of more than 6 words. Then two new lists of abstracts and titles were made, having each abstract cut down to the first 150 words and each title cut down to the first 6 words. After that, each abstract and title in the lists were separated into individual tokens of words. These tokens were then converted to their Word2Vec representation, consisting of a vector of 10 dimensions. The vectors consisted of only 10 dimensions, even though it is not as exact as vectors of more dimensions, because more dimensions wouldn't have run in a reasonable time, and the notebook software used couldn't process all the data, running into a RAM memory problem.

To train and test the model, the data was split into 80% training data and 20% testing data. This consisted of 9148 research paper training samples and 2287 research paper testing samples. The model used was developed with tensorflow's keras, which is used to create neural networks. A sequential model was used for this task. The first layer of the neural network was a LSTM layer of 10 nodes, which had an input size of 150 vectors of 10 dimensions each. This layer had an output of the same dimensions as the input, which went through a normalization layer. After that, the data went through a reshaping layer, to yield an output of 6 by 250 dimensions. The data was reshaped to fit the output shape looked for. Finally, the output layer was a dense layer consisting of 10 nodes, which generated the desired final output of 6 vectors of 10 dimensions. The model was then fit to the training and testing data, being trained for 10 epochs. This model generated pretty accurate results.

5.4 GPT-2 summarizer

The final model used for this research was huggingface's summarization model. This is by far the most complex one, being developed to efficiently solve the task of summarization for texts. This model uses GPT-2 for this task. It has been used for research paper summarization, to allow researchers spend less time choosing what to read, but that's what titles are for, to provide an even more summarized version of a research paper.

To use the model, huggingface's transformer library was downloaded. The model was fine tuned to use the t5-small model with the research paper's abstracts and titles. The samples of research papers were divided into 80% training data and 20% testing data. The model was trained with these samples, taking 873 minutes and 55.7 seconds. Even though this was a lot of time, the complexity of the model required such an elevated training time. To see the output of the model, it had to predict titles for the research paper abstracts.

First, these titles weren't as good as they should have been, they were too long. To correct this, the model was again used to predict titles given the previously generated "long" titles. This method decreased the title length significantly, but was still a little longer than expected. To correct this again, the same process was carried out with the new generated titles. This time the titles were accurate and short enough, so they were left as generated this final time.

6. RESULTS AND DISCUSSION

Research findings and results.

6.1 Approach to measure performance of the models.

Comparing sentences automatically with their meaning is not as easy as it sounds. One of the challenges of generating a score of similarity was that the models used generated strings of different length than the actual title, which made the process a little more complex. Therefore, the model used to score the titles in comparison to the actual title was SpaCy's similarity metric. SpaCy's model measures the models performance by converting each word in the actual title and in the generated title to vectors. The vectors are then averaged for the title. This method allows for evaluation of variable length titles, but is insensitive to order, which may not be as representative of the titles as optimal. Nevertheless, it is a good metric to measure quantitatively the performance of the models tested.

The method used by SpaCy's similarity metric is cosine similarity. Cosine similarity is a measure of the distance of vectors. These vectors are the average of the token vectors, creating a vector representative of the meaning of the sentence. In order to get the cosine similarity, the following formula is used. This method yields a (0, 1] score, being 0 the least similar and 1 the exact equal.

$$sim(A, B) = cos(\theta) = \frac{A \cdot B}{||A||||B||}$$

6.2 Most Common Words Model Performance

The first method, the most common words model, was the simplest one. What was basically done was to put a title together from the most common words used in the abstract. As described in the previous section, this model wasn't the best for many reasons. The model's titles and actual titles were converted to vectors to test the similarity between the texts. The cosine similarity score was 0.5436 on the testing data, which is a good score for such a simple baseline model like this one. The performance of this model is surely low because the model just put frequent words of the abstract into the title without actually them being important or having any sense together.

6.3 Adjacency Matrix of Words Model Performance

The second method, an adjacency matrix of words, was an improvement on the first model. The function of this method was to take the most frequent and important word out of the abstract, and then put the most frequent adjacent words next to each other until forming a list of words to create a title. While this was better than the previous one, it still has some limitations like human readability and the importance of words. Then the generated titles and actual titles were converted to vectors to test their cosine similarity. The cosine similarity score for this model was 0.6022 on the testing data. The score increased by 10.78% from the last model. This points out that the increased complexity of the model improved results. These results, which are still a little low, may be due to the use of words that are adjacent to the initial important word, but the following may not be as important in the title as they are frequent in the abstract.

6.4 Long-Short Term Memory Model Performance

The third method tested, a long-short term memory recurrent neural network, accomplished a great leap from the previous examples. This model was the first actual neural network. What it did was to convert everything to vectors, learn the correlations between the word vectors in the abstract and the word vectors in the title, and be able to make predictions of titles. In this case the testing data had to be cut down because not all research papers met the requisites to be processed with this model. The model had a huge leap from the previous ones, achieving a cosine similarity of 0.9032, which is very good. This model had a score 66.15% better than the first method and 49.98% better than the second one. The complexity of the model can be reflected in the results achieved by it. Neural networks are very powerful and complex, and this made possible such a high score. Due to the limitations of this kind of neural networks, the score most probably doesn't reflect the true capabilities of the model. This model was only trained to produce 6 vectors, which had to be as close as the vectors of the first 6 words of the title. This may overstate the true capability of the model, because it's a simplified version and didn't process the whole meaning of the title.

6.5 GPT-2 Summarizer Model Performance

The fourth and final method tested, a GPT-2 summarizer, didn't do as well as expected. This was the last, most complex, and better developed model of all of the tested ones. This was a fine-tuned version of a model created to summarize texts. Unfortunately, the model generated strings that were too long, but still were useful text generations. The model was the one with the most human-like language, which isn't measured by the cosine distance, but is a good qualitative measure to keep in mind. The cosine similarity score for the testing data was 0.6574, which is good but not as expected. This score is 20.93% better than the first model, 9.12% better than the second one, but it performed 27.21% worse than the LSTM neural network, according to the cosine similarity. This result may be due to the high amount of words generated, which may have altered the average vector by having the same weight as highly important words. The result may also be due to the model not trying to replicate exactly the actual title, but rather generating the title through a human-like thought process, which places words in the correct order and makes more sense.

Actual title	Rate Optimal Binary Linear Locally Repairable Codes with Small Availability
Model 1	code binary linear rate associated class size 2 A
Model 2	r -availability r,3 2 code linear
Model 3	*Future work*
Model 4	code symbol said to have (r,t)-availability if it can be recovered. if rate is maximum, it is said to be 'rate-optimal'

Table 1: Comparison of a sample of generated titles vs actual title.

7. CONCLUSION

This project was developed to determine an accurate and efficient model to generate research titles with NLP models. There were four different models or methods tested and compared. The best result was of a cosine

similarity of 0.9032, which belongs to the long-short term memory model. This model generated fairly good results mostly because it was done with only the initial 6 words of the title, and generating only 6 words, given that the output shape has to be fixed for this process. This made the model have a much higher similarity because it had the same exact amount of words it was looking for, and that it tried to replicate the word vectors of the actual title as closely as possible. The model that actually produced the most well composed titles was the GPT-2 summarizer. Unfortunately, this model generated titles that were too long, which affected the average sentence vector. The model also tried to capture all the important details of the abstract, so it didn't have the same exact topics or details as the actual title. This model could be improved by fine tuning it even more to produce titles of less length. Despite this low similarity score, the GPT-2 summarizer model can be used by researchers to have an even more compact representation of the research paper than the abstract. This reduced content might help researchers choose the title for their work, because they just have to put the content of the few generated words into a compact title form. Title generation has a lot of development opportunities, and that's why it's necessary to keep looking for- and creating better performing NLP models for this task.

8. ACKNOWLEDGEMENTS

Special thanks to Inspirit AI for providing me with this amazing opportunity to do this research paper. Also great special thanks to Sean Konz, my mentor through the development of this project, for helping me greatly with the problems I encountered and for his guidance throughout the whole development of this paper.

9. REFERENCES

Birajdar, Nikhil. "Word2Vec Research Paper Explained | by Nikhil Birajdar." *Towards Data Science*, 15 March 2021, https://towardsdatascience.com/word2vec-research-paper-explained-205cb7eecc30. Accessed 5 November 2022.

Codepath. "Graphs." *CodePath Cliffnotes*, 2018, https://guides.codepath.com/compsci/Graphs. Accessed 5 November 2022.

Gensim. "Word2Vec Model — gensim." *Radim Řehůřek*, 6 May 2022, https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html#sphx-glr-auto-examples-tutorials-run-word2vec-py. Accessed 4 November 2022.

huggingface. "Summarization." *huggingface / transformers*, https://github.com/huggingface/transformers/tree/v4.21.0/examples/pytorch/summarization#with-trainer.

Lang, Niklas. "Stemming vs. Lemmatization in NLP | by Niklas Lang." *Towards Data Science*, 19 February 2022, https://towardsdatascience.com/stemming-vs-lemmatization-in-nlp-dea008600a0. Accessed 5 November 2022.

Mikolov, Thomas, et al. "Efficient Estimation of Word Representations in Vector Space." vol. 3, no. NLP, 2013, p. 12. *arXiv*, https://arxiv.org/pdf/1301.3781.pdf.

Mukerjee, Aditi. "Spam Filtering Using Bag-of-Words | by Aditi Mukerjee | The Startup." *Medium*, 31 December 2021, https://medium.com/swlh/spam-filtering-using-bag-of-words-aac778e1ee0b. Accessed 5 November 2022.

Németh, D. "BLEU-BERT-y: Comparing sentence scores | by Gergely D. Németh." *Towards Data Science*, 6 November 2019, https://towardsdatascience.com/bleu-bert-y-comparing-sentence-scores-307e0975994d. Accessed 4 November 2022.

Open AI. "Language Models are Few-Shot Learners." *arxiv*, no. NLP, 2020, p. 75. *arXiv*, https://arxiv.org/pdf/2005.14165.pdf.

Open AI, et al. "Language Models are Unsupervised Multitask Learners." *Cloudfront*, no. NLP, 2018, p. 24. *Cloudfront*, Language Models are Unsupervised Multitask Learners.

Spacy. "Linguistic Features · spaCy Usage Documentation." *spaCy*, https://spacy.io/usage/linguistic-features#vectors-similarity. Accessed 2 November 2022.

SpaCy. "Linguistic Features · spaCy Usage Documentation." *spaCy*, https://spacy.io/usage/linguistic-features#vectors-similarity. Accessed 4 November 2022.

Yalçın, Orhan G. "3 Pre-Trained Model Series to Use for NLP with Transfer Learning." *Towards Data Science*, 5 December 2020, https://towardsdatascience.com/3-pre-trained-model-series-to-use-for-nlp-with-transfer-learning-b2e45c1c275b. Accessed 4 Novembe