Patent Abstract Summarization using Recurrent Neural Networks

Abhishek Jindal
UC - Irvine
ajindal1@uci.edu

Chirag Choudhary
UC - Irvine
cchoudha@uci.edu

Nile Hanov
UC- Irvine
nhanov@uci.edu

Abstract

This work presents an abstractive summarization model for extracting the main ideas from patent abstracts and summarizing them into a title. The model uses an encoder-decoder Recurrent Neural Network (RNN) with attention mechanism. The evaluation is performed on a patent dataset specifically accumulated for this work. The dataset is kept generic and contains patents from multiple domains.

1 Introduction

In abstractive text summarization, the task is to generate a dense summary consisting of a few words or sentences that capture the main ideas of a given document. The generated summary is called abstractive because it is not a mere selection of phrases extracted from the text, but rather a concise paraphrasing of the main thoughts of the document, potentially using vocabulary unseen in the original document.

Text summarization has gained a significant interest due to overwhelming amount of textual information available in electronic format. This information overload fuels the high demand for more capable text summarizers. Many fields can benefit greatly from summarization as it allows us to process and learn more information in a given time. For example, scientific papers and recent news articles can be summarized efficiently such that reading time is significantly reduced.

The main goal of this project is to summarize patent abstracts and use the patent titles to validate the summarization. Patent abstract summarization can be useful for understanding a patent quickly, and also for potentially grouping multiple related patents together. It is important to note that patent industry is oftentimes driven by ”click-bait” objective. In other words, many patents will have a ”catchy” title that doesn't describe the content well. It is essential to generate titles that describe what is actually contained in the abstracts.

Existing patent summarizers are primarily based on extractive summarization techniques and try to capture key-phrases (Brugmann et al., 2014). Extractive summarization techniques usually perform very poorly because they cannot capture higher level abstraction and meaning of the text. There are works also that cluster field specific patents (Trappey et al., 2009). Such works are primarily concerned with unsupervised grouping and do not address summarization.

Our model is a bidirectional encoder-decoder RNN. It was optimized for best performance with tests utilizing Gated Recurrent Units (GRU) and Long short-term memory (LSTM) cells as building blocks. It is trained on corpora that covers various domains. This model allows to effectively capture the main ideas in patent abstracts and summarizes those ideas into titles. Unlike previous works, our model generalizes well to different domains and performs well on out of domain text.

The models was tested on texts that cover broad range of domains, such as: chemistry, physics and biology. For evaluation, we used ROUGE-1, ROUGE-2, ROUGE-L and BLEU metrics. We also performed qualitative evaluation to ensure the generated texts were grammatically sound. We were able to double our best ROUGE score from the last reported results by using new word embeddings and tuning hyperparameters. Complete test results and details of evaluation setup are provided in the experiments section.

2 Related Work

The majority of research in text summarization focused on extractive summarization, where the model identifies key words and phrases from the source documents, and combines them into a

meaningful summary (Narayan et al., 2018; Litvak and Last, 2008; Neto et al., 2002). However, as humans, we tend to generate abstractive summaries by paraphrasing using new words and phrases. With improvements in deep learning, neural networks have been applied as a fully data-driven approach for abstractive summarization. A very relevant model in this domain is an attentional Recurrent Neural Network (RNN) encoder-decoder model (Bahdanau et al., 2014). It uses a bidirectional Gated Recurrent Units (GRU) RNN encoder, and uni-directional GRU-RNN decoder with the same hidden-state size as that of the encoder, and an attention mechanism over the source-hidden states and a soft-max layer over target vocabulary to generate words. This model produced state-of-the-art performance in machine translation (MT).

Unlike in MT, text summarization usually produces a very short output text. Additionally, a key challenge in summarization is to optimally compress the original document in a lossy manner such that the key concepts in the original document are preserved, whereas in MT, the translation is expected to be lossless. Another similar neural attention model (Rush et al., 2015) for abstractive summarization, uses an neural network language model (NNLM) instead of a target-side Long short-term memory (LSTM) and source-side windowed averaging instead of a source-side bidirectional RNN. Nallapati et al. (2016) also provide several novel models over the basic encoder-decoder architecture, such as modeling key-words, capturing the hierarchy of sentence-to-word structure, and emitting words that are rare or unseen at training time.

Previous NLP work on patents is either based on extractive summarization techniques (Brugmann et al., 2014) or tries to cluster patents based on domain similarity. We could not find any work on patent summarization using neural networks. Trappey et al. (2009) summarized patents using a combined ontology based and TF-IDF concept clustering approach. Trappey et al. (2006) introduced a patent document summarization system using an integrated approach of key-phrase recognition and significant information density. Our use of neural networks obviates the domain knowledge required for such models.

3 Approach

3.1 Dataset

We started by scraping the Google patents website for different categories of patents. The scraping was very slow and there were some errors as well while creating the dataset which led to a lot of time consumption. To overcome this limitation we used BigQuery on Google Patents Public Datasets. This allowed us to query thousands of patents at once and have a consistent structure over the used corpora. We used patents whose publication dates were from year 2000 and onwards. This also allowed us to avoid old information in the text. We also made sure that the abstract information was only in English while creating the dataset. The final dataset consisted of 25,743 patents. We split this dataset into training, validation and testing set, containing 16,000, 4,000 and 5,743 patents respectively. To ensure that our model generalized well to different categories of patents, we explicitly collected patents from multiple diverse fields (see Table 1).

Table 1: Dataset Description

Subject	Train	Val	Test
Algebra	189	51	80
Biology	559	118	205
Chemistry	1,620	413	576
Electrical	3,366	867	1,174
General	9,236	2,277	3,347
Operating Systems	476	130	176
Physics	554	144	185

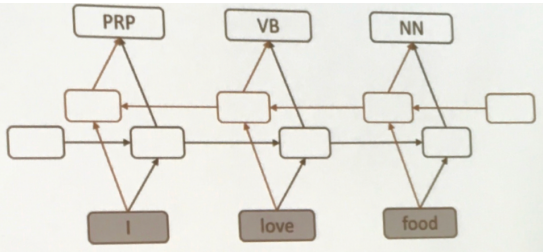
3.2 Data Preprocessing

During the preprocessing step, we cleaned the abstract information. In this step, we expanded the contractions which are usually present in the English language. We also removed the stop words from the abstracts, but kept them for the patent titles. This was done to preserve the syntactic structure of the sentences that the decoder was generating. We then computed the word counts for our vocabulary words. To reduce the size of our vocabulary, we removed low-frequency words (count$<$2). The remaining words were replaced by $<UNK>$ tokens. We also removed the patents for which the abstract or title sentences had more than 5 and 3 $<UNK>$ tokens respectively.

4 Model Overview

Our model is a bidirectional encoder-decoder recurrent neural network (RNN). In bidirectional RNN, the input sequence is fed in normal time order for one network, and in reverse time order for another. This structure allows the networks to have both backward and forward information about the sequence at every time step (see Figure 1). This can provide additional context to the network and result in faster and even fuller learning on the text.

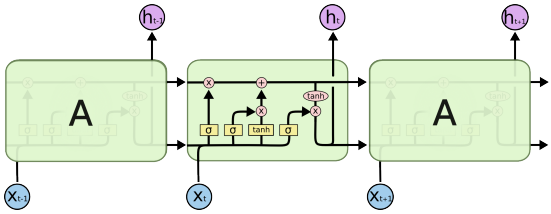
Figure 1: Bidirectional RNN



4.1 Encoder

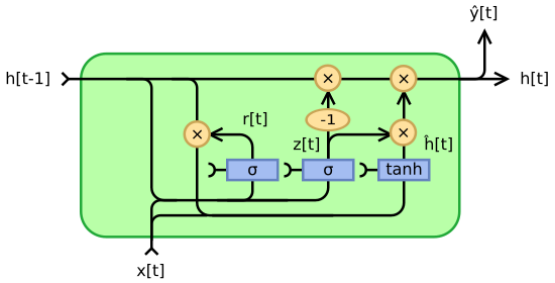
Our encoder-decoder is based on model proposed by (Bahdanau et al., 2014). We use an LSTM cell for the encoder (see Figure 2). This is a special kind of RNN, capable of learning long-term dependencies. Regular RNNs suffer from vanishing gradient problem which prevents them from learning long-term dependencies. LSTMs were designed to combat vanishing gradients through a gating mechanism.

Figure 2: LSTM chain



We also used Gated Recurrent Units (GRU) (see Figure 3) in our model and compare the performance against LSTM cells. GRUs are similar to LSTMs but have fewer parameters and thus can train a bit faster and need less data to generalize. On the other hand, the greater expressive power of LSTMs may lead to better results. This is something we compare empirically in our project.

Figure 3: GRU cell



4.2 Decoder

The decoder implements the mechanism of attention. It decides parts of the source sentence to pay attention to. Each time the model generates an output word, it searches for a set of positions in a source sentence where the most relevant information is concentrated. The model then predicts a target word based on the context vectors associated with these source positions and all the previous generated target words.

5 Experiments and Results

To train the model, we initially used word embeddings called ConceptNet Numberbatch (CN) which is an ensemble of many of other popular word embeddings such as *Word2Vec* and *GloVe*. However, with these embeddings, the model performed suboptimally on various metrics. This was due to large number of $<unk>$ tokens in the summaries. Thus, we switched to GloVe word embeddings (Pennington et al., 2014). The ConceptNet Numberbatch word embeddings consists of 1.9 million words, while GloVe word embeddings has 2.2 million words.

We used the training dataset containing 16,000 samples for our training and validated our model on the validation dataset containing 4,000 samples. After selecting our final model we tested our results on the test dataset containing 5,743 samples.

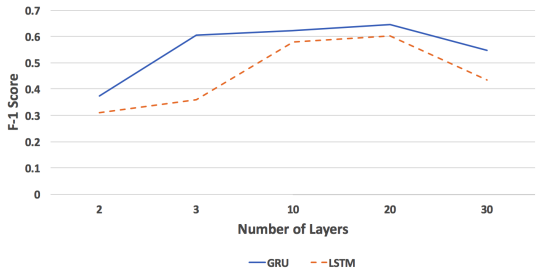
We trained all the models for 25 epochs. We were also using early stopping when the error did not improve for continuous iterations. Previously, the training time for each epoch was approximately 10 minutes for the new dataset. Overall our model used to take 250 minutes to train. But, we were able to parallelize the code correctly on GPU and it ran significantly faster after that. The training time for each epoch reduced to less than 1 minute and overall the 25 epochs completed

in approximately 20 minutes.

5.1 LSTM and GRU comparison

LSTM and GRU cells were used with the model. We varied number of layers, number of units in each layer, batch size and learning rate for each model. We observed diminishing returns when going beyond 256 units in each layer. Varying number of layers was the most impactful for our model. We observed that with more layers the training accuracy tended to consistently increase. However, beyond a certain point, the validation accuracy started to suffer and drop. This was the point when our model started to overfit (see Figure 4).

Figure 4: GRU vs LSTM Comparison



The ROUGE-1, ROUGE-2 and ROUGE-L scores for training and validation dataset are shown in Table 2 and 3. For each model, 20 layer network with 256 units is used. It can be observed that GRU-based model performed better on all metrics. This highlights well the fact that GRU needs less data to generalize.

Table 2: Training dataset performance

GRU			
Accuracy Metric	Precision	Recall	F-1
ROUGE-1	0.8543	0.7387	0.7775
ROUGE-2	0.4827	0.4864	0.5182
ROUGE-L	**0.6519**	**0.7173**	**0.6342**
LSTM			
Accuracy Metric	Precision	Recall	F-1
ROUGE-1	0.7834	0.6873	0.7197
ROUGE-2	0.4505	0.5006	0.4613
ROUGE-L	0.4566	0.6743	0.4554

Table 3: Validation dataset performance

GRU			
Accuracy Metric	Precision	Recall	F-1
ROUGE-1	0.7862	0.6663	0.7064
ROUGE-2	0.4147	0.4313	0.4092
ROUGE-L	**0.6114**	**0.6441**	**0.5784**
LSTM			
Accuracy Metric	Precision	Recall	F-1
ROUGE-1	0.7239	0.6204	0.6549
ROUGE-2	0.3731	0.4035	0.3751
ROUGE-L	0.4194	0.6011	0.4078

The BLEU score for training and validation dataset are shown in Table 4. GRU-based model was superior on this metric as well.

Table 4: BLEU Scores

Evaluation	GRU	LSTM
Train	0.2279	0.1838
Validation	0.1916	0.1328

Through quantitative experiments we have determined that 20-layer GRU-based model with 256 units in each layer performs the best. We then evaluated this model on the test dataset. We compare the final results on all three datasets in Table 5.

Table 5: Best model performance results

Train			
Accuracy Metric	Precision	Recall	F-1
ROUGE-1	0.8543	0.7387	0.7775
ROUGE-2	0.4827	0.4864	0.5182
ROUGE-L	0.6519	0.7173	0.6342
Validation			
Accuracy Metric	Precision	Recall	F-1
ROUGE-1	0.7862	0.6663	0.7064
ROUGE-2	0.4147	0.4313	0.4092
ROUGE-L	0.6114	0.6441	0.5784
Test			
Accuracy Metric	Precision	Recall	F-1
ROUGE-1	0.6727	0.5315	0.5698
ROUGE-2	0.3230	0.3242	0.31033
ROUGE-L	0.5056	0.5150	0.4615

We observe that there is a slight decrease in the

performance on the test dataset as compared to validation dataset which suggests that there might be some overfitting on the validation dataset.

For this model, the also measured the BLEU score:

- Best model BLEU Score: **0.2279**

5.2 Qualitative Evaluation

We also performed qualitative analysis on the summaries generated by the model. Overall, the results seemed to be quite accurate and the main ideas were almost always captured. The sentence structure was well-maintained. In most cases the model generated results and actual titles differed by a few words on the validation dataset like in the example shown below.

- Actual: *automatic chemistry analyzer with sample cup stopper piercing assembly*
- Model: *automatic chemistry analyzer sample cup stopper piercing assembly*

The model sometimes produced titles that were different at first glance; however, after comparing the generated sentence to the original title it was apparent that the main idea was still preserved as shown in the example below.

- Actual: *systems and methods of rf power transmission, modulation and amplification*
- Model: *apparatus system and method for electrical energy conversion*

6 Conclusions and Future Work

Our model performed reasonably well and the chosen approach has proven to be effective. We were able to significantly improve the previously reported score by additional text preprocessing, tuning hyperparameters and using GRU cells instead of LSTM. The model performed well regardless of patent domain.

There are still some problems that need to be addressed in the future work. First is the word embeddings. Although we were able to get some improvement after we switched to using GloVe embeddings, there are still quite a few $<UNK>$ tokens. This can be solved by pre-training training GloVe vectors on patent corpora. Additional tuning of hyperparameters may also lead to further improvement.

We also think that analysing the output on different categories could help us understand the limitations of our model. Moreover, using some categories in the test dataset which are not present in the training dataset should also help us understand how our model is able to generalize.

One final thought to keep in mind for any potential future improvement is that this model should not and cannot be perfect. This comes from our original statement that some patents intentionally have misrepresentative titles designed to be flashy. Therefore, there will be some errors no matter how good the model is.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Soren Brugmann, Nadjet Bouayad-Agha, and Alicia Burga. 2014. Towards content-oriented patent document processing: Intelligent patent analysis and summarization world patent information.

Marina Litvak and Mark Last. 2008. Graph-based keyword extraction for single-document summarization. In *Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization*, MMIES '08, pages 17–24, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning.

Joel Larocca Neto, Alex A. Freitas, and Celso A. A. Kaestner. 2002. Automatic text summarization using a machine learning approach. In *Advances in Artificial Intelligence*, pages 205–215, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685.

A. J. C. Trappey, C. V. Trappey, and B. H. S. Kao. 2006. Automated patent document summarization for r d

intellectual property management. In *2006 10th International Conference on Computer Supported Cooperative Work in Design*, pages 1–6.

Amy J.C. Trappey, Charles V. Trappey, and Chun-Yi Wu. 2009. Automatic patent document summarization for collaborative knowledge systems and services. *Journal of Systems Science and Systems Engineering*, 18(1):71–94.