Computational models for text summarization

Leonid Keselman
leonidk@cs.stanford.edu

Ludwig Schubert
ludwig@cs.stanford.edu

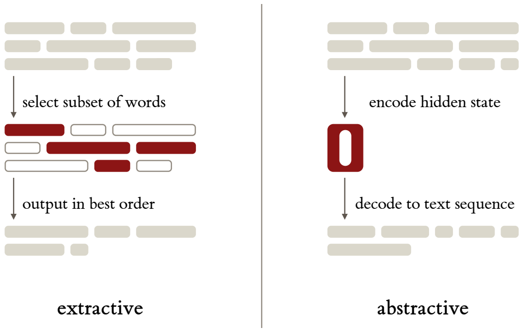
Abstract

Abstractive text summarization is a blossoming area of natural language processing research in which short textual summaries are generated from longer input documents. Existing state-of-the-art methods take long time to train, and are limited to functioning on relatively short input sequences. We evaluate neural network architectures with simplified encoder stages, which naturally support arbitrarily long input sequences in a computationally efficient manner.

1 Introduction

In natural language processing, text summarization refers to the task of taking a document and creating a shorter version of it, that nevertheless retains its primary meaning. While this task has real-world applications, such as saving time reading long news articles, or creating blurbs for news for previews, it is primarily a well-suited task for the reasoning about the scale of different textual models: building neural network models that map from longer sequences to shorter ones inherently has the questions of ”how long” and ”how short”. Classical text summarization techniques were interested in producing a handful of sentences when given a larger article [1]. Recent neural network techniques [2] use seq2seq [3] models, which are naturally limited to sequences of a few hundred words at most. This comes from one of two natural limitations, either encoder-based models lack the proper memory to remember thousands of words ahead, or attention-based models are unable to evaluate attention over giant sequences. In general, these recurrent techniques require non-trivial methods to work over large text sequences, an ongoing area of research [4].

Additionally, many document sources for which we are interested in summaries naturally come with human generated summaries, whether that are headlines for news reports or abstracts for scientific publications. While titles and headlines are not equivalent to summaries, they approximate the same information-compressing ideal. Both the availability of these datasets and the existence of reasonable approximations of summaries makes this problem well suited for supervised deep-learning approaches.



In the absence of perfect text summarization, existing techniques can be classified into two categories; extractive and abstractive.

Extractive methods restrict themselves to choosing words or sentences from their input. While this makes these tasks easier then generating a summary from scratch, any algorithm built with them in mind is ultimately limited to its input data. A human might choose to express a complex idea quite differently at different lengths. The more general problem of abstractive summarization leads to less general architectures, making for a more interesting model that arguably gets closer to general text understanding. Thus this report focuses on abstractive methods and compares different architectures for this task.

1.1 Abstractive text summarization

Abstractive text summarization can be thought of as a two-step process: a sequence of text is first encoded into some kind of internal representation. (Figure **??**) This internal representation is then used to guide the decoding process back into the summary sequence. Models that use this architecture are often called *sequence-to-sequence* models.

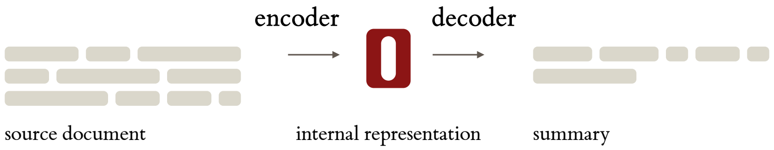


Figure 1: Abstractive text summarization as a two-step process

State of the art architectures use recurrent neural networks for both the encoding and the decoding step; often with attention over the input during decoding as additional help. [5]

While general in nature, architectures like this have been shown to not scale well to arbitrary lengths documents [4]. Thus we were interested in tackling the problem of abstractive text summarization using simple encoding methods (without attention or necessarily even seq2seq encoders). If we are able to focus our energy on building a powerful decoder that works with a simple encoder, we would build an abstractive text summarization system that can handle generating headlines or titles from entire articles. Our report focuses on implementing and testing different architectures for each step in the abstractive summarization process.

2 Related Work

Initial work in natural language processing on text summarization is almost as old as the field itself. In 1958, [6] published statistical method for scoring and selecting sentences from larger blocks of text in order to generate coherent abstracts. This work continued, with a popular model [1] demonstrating compelling results. At the same time, both metrics such as ROUGE [7] and datasets such as DUC [8] were developed in order to compare and contrast various methods for summarization. However, most of this work has been in the form of extractive text summarization.

Abstractive text generation has a recent development in the field, with a variety of ontological [9], rule-based [10] and graph [11] techniques being pursued , combining different ways of tracking important concepts in articles and using a variety of natural language generation models to create valid output.

Recent work, from various research groups at Facebook [2] and IBM [5] have built models that combined the task of extracting with that of generating by using recurrent neural networks. RNNs, usually with generations of the seq2seq model [3] are able to both extract valid information and generate valid language using the same computational framework. They demonstrated state-of-the-art performance on the sentence summarization task using a feed-forward window-based neural network with an attention mechanism. These types of models are in some ways, fairly straightforward and powerful both at extracting high quality concepts and in generating high-quality language, when they are given short snippets of text and large datasets. Often, they are trained on the Gigaword [12] which includes a very large dataset, and then tested on DUC [8], a tiny dataset in comparison, which we will describe below. In our case, we attempt to build simpler models, which are able to perform well on DUC without needing to be trained on Gigaword sized datasets—both in terms of the number of documents and the complexity of the dataset.

3 Datasets

We used four different datasets for training and evaluation of our models:

3.1 DUC 2004

While many datasets are available that approximate the idea of summaries by using abstracts or headlines, this classic dataset from the Document Understanding Conference [8] contains multiple human generated labels per document that were explicitly constructed as summaries. The DUC dataset contains 500 documents of which we used 432 news articles with unique beginnings, with 4 to 8 model summaries each. While these are high-quality gold-standard summaries, the dataset is too small to use as an exclusive training set. Thus it is often used as a test set only; we used it both for training in our smaller models and for testing for all models.

3.2 NewsIR ’16

This “Signal Media One-Million News Articles Dataset”[13] dataset contains one million news articles and their headlines collected from the web. We filtered this dataset to include only news articles (`"media-type": "News"`) in order to be more comparable to the DUC dataset, leaving 572,154 article-headline pairs.

3.3 NIPS articles, abstracts & titles

This kaggle-hosted dataset contains all published papers from the Neural Information Processing Systems (NIPS) conference. [14] We used the title, abstracts, and extracted text to create two subdatasets: `nips-article-abstract` and `nips-abstract-headline`. According to its author Ben Hammer it contains papers “ranging from the first 1987 conference to the current 2016 conference”. We extracted all 3,243 documents for use, and explicitly did not strip LaTeX code.

3.4 SQuAD-flipped

We used this question-answering dataset by flipping some easily identifiable question-answer pairs: for example by replacing the word ”who” with an answer. The results are not always grammatical, but allowed us to use a second dataset with multiple given summarizations—here in the form of factual statements based on the given text—for our evaluation.

For example, we combined the question-answer pair `"Who was the president of Notre Dame in 2012?"`—`"John Jenkins"` into one summary for the question context: `<s> john jenkins was the president of notre dame in 2012 . </s>`

3.5 Preprocessing

We pre-processed datasets to be all lowercase and restricted to the 100000 most common words. We used NLTK[15] to tokenize inputs and add paragraph and sentence start and end markers (`<p>, </p> & <s>,</s>`).

All dataset were split randomly into training, evaluation and test sets, weighted 80% for training, 10% for evaluation and hyperparameter optimization, and 10% for testing.

To simplify our comparison we converted all datasets into a simple `JSON` format:

```
[
  {
    "data": "input document as string",
    "label": ["model summaries as array of strings"],
    "set": "train"|"dev"|"test",
    "prediction": "predicted label as string"
  }
]
```

This specified the output format for our models as well (`prediction` string) and allowed us to run the same evaluation script over the outputs of our various models. We made sure to strip any sentence and paragraph markers before evaluation since they otherwise led to artificially inflated ROUGE-scores.

4 Models

For simplicity of illustration, we omit fully connected layers used for projecting between representations of different dimensions, such as from GLOVE vector embeddings to our hidden state size. Our fully connected layers uniformly consist of Xavier-initialized matrices, bias terms and `tanh` non-linearities.

All models were trained with 50% dropout-rate on the hidden state input to the RNN, optimizing cross-entropy using Adam and numeric gradient clipping.[1]

4.1 State of the Art model: Sequence-to-sequence

State of the art for text summarization as of 2016 is a sequence-to-sequence model with attention over its input. (Figure **??**) These models look up embeddings of their inputs, pass them through a bidirectional recurrent neural net to produce the initial hidden state, which is then decoded into a number of hypotheses by a second recurrent neural network with an attention model over its input, from which a final output is chosen using beam search.

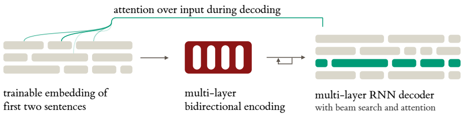

attention over input during decoding
trainable embedding of first two sentences
multi-layer bidirectional encoding
multi-layer RNN decoder
with beam search and attention

Figure 2: A sequence-to-sequence architecture

4.2 Baseline

As a baseline for our models we used a trivial model that repeats the first sentence of the input document. It uses the `</s>` token from the preprocessing step to determine its output. (Figure 3)

4.3 Encoders

In common summarization architectures the input is encoded by a recurrent neural network to allow for flexible input lengths and to capture syntactic information. We experimented with explicitly non-recurrent, pure feedforward encoders. The idea was both to eliminate the dependency on input lengths that models still experience, as well as trying to explicitly model the semantic content of a document in the internal representation–thus we focused on architectures that did not use attention over the input and were instead forced to encode all the content of the input document in their internal representation. These architectures do not beat state of the art approaches using attention, but they are a lot simpler yet not purely statistics based as traditional summarization approaches such as textrank.[1]

4.3.1 Sum-of-GLOVE encoder

The sum-of-Glove encoder (Figure 6) looks up pretrained GLOVE embeddings of all input tokens, sums them, then uses the result as the initial state for a recurrent neural network decoder. Whenever

[1] Clipping by gradient norm had lead to exploding losses for multiple architectures.

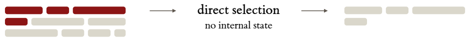


Figure 3: baseline system

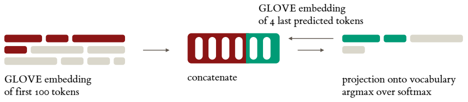


Figure 4: Concat decoder

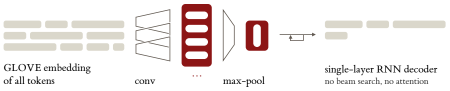


Figure 5: Conv encoder

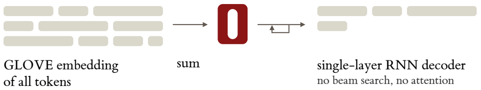


Figure 6: Sum encoder

we refer to summing of GLOVE embeddings in this report we implicitly assume a normalization over the input length, i.e. the "summing" is a reduce-mean operation over the list of input glove vectors.

Additionally, we built models that fine-tune the word embeddings. With our small datasets these tended to exhibit better training set error, but performed similarly to using the pretrained embeddings on dev sets.

4.3.2 Concatenation-of-GLOVE encoder

The concatenation-of-Glove encoder (Figure 4) takes the GLOVE embeddings of the first 100 input tokens, concatenates them, then concatenates the result with the embeddings of a window (here size 4) over the last produced tokens. The resulting vector is projected onto the vocabulary size and turned into a probability distribution via a softmax transformation. It is then sampled word for word, allowing the model to condition its output on its prior output.

4.3.3 Conv/Maxpool encoder

In order to handle more ambiguity in input (such as the relation change that *not* forces), we wanted to build a system that respected word-order relationships, while not being limited to long recurrent relationships that might lose context over time. Thus we implemented a convolutional encoder that pools over the sentence length dimension, similar to other work for CNNs applied to textual data [16]. In our case, the conv/maxpool encoder (Figure 5) looks up GLOVE embeddings of all input tokens, runs a two-layer one dimensional convolution over them, max pools over the activations to create a hidden state.

4.4 Decoders

Our baseline decoder model is a word-predicting recurrent neural network with GRU activations [17]. The "context" vector is projected into the internal state of the RNN, and trained to predict the $t+1$ step word when given the t step word. Our initial token is a unique start of sequence token and we also train an end of sequence token. We limit our vocabulary to about 5,000 words to keep our training speed efficient, replacing unknown words with a special token.

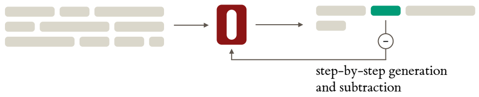


Figure 7: Subtractive decoder

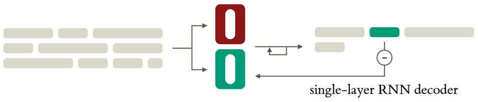


Figure 8: Subconcat encoder

4.4.1 Subtractive decoder

The idea of the subtractive encoder (Figure 7) is to remove (subtract) the semantic content of each predicted word from the sum-of-GLOVE representation after it has been predicted. The idea is to force the model to encode exactly the information it needed to produce a summary, preventing it from just repeating words that are similar to the overall meaning of the input document. The model then predicts the output word-for-word. In this case, we concatenate each input word token with the context, instead of projecting into the initial state of the RNN as with the baseline model.

4.4.2 Subtractive recurrent decoder

To combine the idea of the subtractive decoder with a recurrent neural net for output we duplicated and concatenate the sum-of-GLOVE representation as a global context vector, subtracting only from it while also feeding another copy as the initial internal state of a RNN. (Figure 8)

5 Results & Analysis

5.1 Evaluation metric: ROUGE scores

As a quantitative evaluation measure we use ROUGE[7] [2] scores; particularly we report the F-measures (harmonic mean of precision and recall) for 1-gram and 2-gram overlap as well as longest common subsequence (ROUGE-N and ROUGE-L).

Rouge scores are computed per document (we report the average over our test sets) where M is the set of model[3] summaries, P is the prediction made by the model, n are n-grams in each model summary m.

$$\text{ROUGE-N}(M, P) = \sum_{m \in M} \sum_{n \in m} \frac{|\{n\} \cap P|}{|n|}$$

While this is a commonly used metric we believe that it should not be used exclusively to evaluate a system: extractive methods may have an easier time achieving higher 1-gram overlap while the actual informative content of the summary may be worse than a hypothetical reformulation with no overlap.

5.2 Quantitative Results

		Baseline	Sum-of-Glove + RNN *(F-scores Rouge-1 & -2)*				Textsum *(SOA architecture; stopped after 180k steps)*			
			trained on				*trained on*			
		first sentence	NIPS	SQuAD	NewsIR'16	DUC'04	NIPS	SQuAD	NewsIR'16	DUC'04
tested on	NIPS	3.652 0.060	3.735 0.074	0.052 0	0 0	0 0	3.233 0.165			
	SQuAD	3.687 0	0.087 0.015	0.802 0	0.482 0	0 0		4.018 0.237		
	NewsIR'16	0.034 0	1.017 0	0.196 0	0.016* 0	0 0			1.810 0	
	DUC'04	0.761 0.095	0.631 0	0.020 0	0 0	1.064 0				2.502 0

5.3 Runtime considerations

Compared to our reference models [2], we can train our simple encoder-decoder models on our datasets in just a handful of epochs, or on the order of minutes with a single Azure instance GPU. On the other hand, the implementation notes for our baseline model suggest a training time of weeks on large datasets.

[2] Recall Oriented Understudy for Gisting Evaluation

[3] "model" as in gold-standard, not model-generated—the ROUGE parlance is confusing

5.4 Example summaries

In these two handpicked examples we see how the simple encoders are already able to pick up some semantic context: the "sword of orion" in this summary was a yacht in the mentioned "Sydney To Hobart" race. Those had shown up in the training data together enough for the models to pick up on. The second example from the NIPS dataset shows how even within a topic space the semantic embeddings sometime offer enough resolution to get to a similar meaning even if there is little n-gram overlap.

Ground Truth	Generated Summary
three yachts missing two dead one sailor missing in sydneytohobart race	`sword of orion sailor missing`

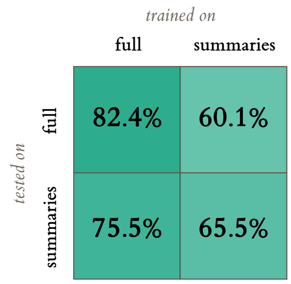
compressive spectral embedding :
sidestepping the svd
	`spectral methods for the`
	`generalized components`

We also wanted to test our system on arbitrary length inputs. We did so by feeding in a whole NIPS paper and asking the model to generate a headline for it.[4]

The paper title is "Learning Distributed Representations for structured Output Prediction" [18] and our model produced the title `the x is of kernel or in the <UNK> to evaluate x for many optimization`—not a good title for this paper. This example shows one of the downsides of our simple encoders: "the input x" shows up a lot in the mathematical sections of the paper and so the model is missing the gist of the paper when it is merely adding all the GLOVE vectors in the document, getting bogged down in the large number of details as it can't accurately which parts of the document are the most important. We detail future work in a separate section, but one approach we want to mention here would be using semantic embeddings of sentences rather than purely on a word level.

6 Transfer task: Sentiment analysis on summarized input

One question we were trying to answer was whether summarization could help with other NLP tasks. We asked ourselves the question: if we run summarization on sentiment analysis datasets, does that allow us to fit better sentiment models? In our case the answer was "no"—the simple models did not improve the quality of an LSTM-based sentiment model whether it was trained or tested on summaries instead of the full data.



		trained on	
		full	summaries
tested on	full	82.4%	60.1%
	summaries	75.5%	65.5%

Results of training an LSTM, binary sentiment classifier on the Stanford Treebank dataset [19]

7 Conclusions and Future Work

We have presented a range of non-attention based models for abstractive summarization. These models do not beat the state of the art in abstractive summarization. Our models produced summaries that were often on topic but with grammatical issues. On the decoder site we saw promising results from training on identity datasets. We could make additional progress both by using larger training data sets and training these systems as autoencoders, where input is expected to match output.

Work in abstractive summarization would also benefit from a different evaluation metric than ROUGE scores. As text summarization moves towards more semantic approaches, an n-gram overlap-based statistic will become less effective at evaluating those systems. A proposed metric

[4] We felt it was only appropriate to use a NIPS paper in our dataset authored by our Professor Chris Manning

would have to strike a balance between two highly opposing desiderata: it would need to measure semantic similarity while requiring no hard to manually create additional labels. Any "platonic" evaluation metric would seem to need a human-level of language understanding, so this seems to be a hard task. For example, our decoders often perform semantically and syntactically valid synonym substitution, which should not be penalized (and makes sense given our compressed context vectors), but is to our detriment under the ROUGE metric.

References

[1] R. Mihalcea and P. Tarau, “TextRank: Bringing order into texts,” in *Proceedings of EMNLP-04and the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004.

[2] A. M. Rush, S. Chopra, and J. Weston, “A neural attention model for abstractive sentence summarization,” *CoRR*, vol. abs/1509.00685, 2015. [Online]. Available: http://arxiv.org/abs/1509.00685

[3] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[4] J. Chorowski and N. Jaitly, “Towards better decoding and language model integration in sequence to sequence models,” *CoRR*, vol. abs/1612.02695, 2016. [Online]. Available: http://arxiv.org/abs/1612.02695

[5] R. Nallapati, B. Xiang, and B. Zhou, “Sequence-to-sequence rnns for text summarization,” *CoRR*, vol. abs/1602.06023, 2016. [Online]. Available: http://arxiv.org/abs/1602.06023

[6] H. P. Luhn, “The automatic creation of literature abstracts,” *IBM Journal of research and development*, vol. 2, no. 2, pp. 159–165, 1958.

[7] C.-Y. Lin, “Rouge: A package for automatic evaluation of summaries.”

[8] NIST, U.S. Department of Commerce, “Document understanding conference dataset,” http://www-nlpir.nist.gov/projects/duc/data/2007_data.html.

[9] C.-S. Lee, Z.-W. Jian, and L.-K. Huang, “A fuzzy ontology and its application to news summarization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 5, pp. 859–880, 2005.

[10] P.-E. Genest and G. Lapalme, “Fully abstractive approach to guided summarization,” in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 2012, pp. 354–358.

[11] K. Ganesan, C. Zhai, and J. Han, “Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions,” in *Proceedings of the 23rd international conference on computational linguistics*. Association for Computational Linguistics, 2010, pp. 340–348.

[12] C. Napoles, M. Gormley, and B. Van Durme, “Annotated gigaword,” in *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*. Association for Computational Linguistics, 2012, pp. 95–100.

[13] D. Corney, D. Albakour, M. Martinez, and S. Moussa, “What do a million news articles look like?” in *Proceedings of the First International Workshop on Recent Trends in News Information Retrieval co-located with 38th European Conference on Information Retrieval (ECIR 2016), Padua, Italy, March 20, 2016.*, 2016, pp. 42–47. [Online]. Available: http://ceur-ws.org/Vol-1568/paper8.pdf

[14] B. Hammer, “Nips papers 1987-2016.” [Online]. Available: https://www.kaggle.com/benhamner/nips-papers

[15] E. Loper and S. Bird, “Nltk: The natural language toolkit,” in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ser. ETMTNLP ’02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 63–70. [Online]. Available: http://dx.doi.org/10.3115/1118108.1118117

[16] Y. Kim, “Convolutional neural networks for sentence classification,” *CoRR*, vol. abs/1408.5882, 2014. [Online]. Available: http://arxiv.org/abs/1408.5882

[17] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *CoRR*, vol. abs/1409.0473, 2014. [Online]. Available: http://arxiv.org/abs/1409.0473

[18] V. Srikumar and C. D. Manning, “Learning distributed representations for structured output prediction,” in *Advances in Neural Information Processing Systems*, 2014, pp. 3266–3274.

[19] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts *et al.*, “Recursive deep models for semantic compositionality over a sentiment treebank.” Citeseer.

[20] P. Ramachandran, P. J. Liu, and Q. V. Le, “Unsupervised pretraining for sequence to sequence learning,” *arXiv*, 2016. [Online]. Available: https://arxiv.org/abs/1611.02683

[21] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[22] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100, 000+ questions for machine comprehension of text,” *CoRR*, vol. abs/1606.05250, 2016. [Online]. Available: http://arxiv.org/abs/1606.05250

[23] Google, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *CoRR*, vol. abs/1603.04467, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467