Abstractive Automatic Text Summarization

Isaac Koak
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
koakx001@morris.umn.edu

ABSTRACT

Information overload — the difficulty one can have understanding and making a decision on an issue due to the availability of too much information[7] — has become a hot topic of research over the past couple of decades because of the increased availability of information the internet. Automatic summarization, the process of condensing input information to produce a concise summary that retains the most important points of the original input, is one possible way of dealing with the issue of information overload. Automatic summarization can be applied to textual documents, video, or images. Methods for automatic summarization can be put into two categories, extractive or abstractive. In this paper we focus on the abstractive approach in its application to document summarization. We investigate the latest approaches in this category and where the field in headed in the future.

Keywords

Automatic Summarization, Abstractive Summarization, Information Overload

1. INTRODUCTION

Automatic summarization has been around since the late 1950s and has mostly focused on single document summarization with emphasis on extracting the most important information at the sentence level using information such as word frequency, position of a word in a text, or keyphrases that capture the main topic of the text. The development of the internet, and its side effect of information overload, has powered a surge of research in the field of automatic summarization.

Summaries can either be *generic* or *query-driven* [4]. Generic summaries return important information from the input document with little regards to the outside world, whereas query-driven summaries generate the output based on keywords and topics of the query. Summaries can also be categorized based on the type of content they produce: *indicative* or *informative* [4]. Indicative summaries tell us what the document is about, while informative summaries can be read in place of the source document. There have been numerous techniques developed over the years for automatic summarization, but they can all be generalized into two categories: *extractive* and *abstractive* [5].

Extractive summarization uses a subset of words, phrases, or sentences from the original document to form a summary. It usually leaves the words in the initial order and wording and uses a cut and paste manner to string together a final summary. This can work fine for single document summarization, but it presents issues when applied to multi-document summarization, especially in the coherence and comprehension of the resulting output; producing a single output summary for multiple input documents introduces the issue of having to consider similarity, differences, and contradictions among the input documents.

Abstractive summarization tries to understand the semantic meaning of what it's summarizing in order to produce the summary. The process is akin to what humans do when they summarize a text. The important parts of the initial document are preserved with the summary possibly being paraphrased.

A majority of summarization methods implemented in the real world today are based on the extractive model, mainly due to the simple fact that it is easier to implement than the alternative, therefore it has attracted most of the research done in the field. The seemingly exponential growth of the internet has brought about different types of documents that may be better suited to being summarized using the abstractive approach. This, coupled with the ever dropping price of computational power, has inspired more research that leverages the abstractive approach in the past two decades. For the rest of this paper, we focus on techniques for abstractive summarization. After introducing some background information, we examine two different examples of how it's been implemented in the last five years: a graph-based implementation that uses Abstract Meaning Representation and a hybrid approach that utilizes several different methods.

2. BACKGROUND

In order to fully make sense of material in the subsequent sections, it's important that some basic background information is provided. We will begin by taking a look at n-grams, which are continuous sequences of n items from a sequence of text. Then, we describe cosine similarity, which is a way of measuring similarity between documents. Finally, we explain ROUGE, a summary evaluation metric, that will help

UMM CSci Senior Seminar Conference, November 2016 Morris, MN.

us better understand the results of the methods presented in section 3.

2.1 N-Gram

An n-gram is a consecutive sequence of n items from a sequence of some tokens $w_1 ... w_n$. A word is the token in our case as we're talking about text summarization. N-grams have many uses, including being used as the main sentence breakdown method in ROUGE (subsection 2.3.2). The name given to an n-gram is based on the value of 'n'. The first three use latin numerical prefixes 'unigram', 'bigram', and 'trigram' respectively, whereas the rest just prepend the number in front, i.e., 'four-gram' or 4-gram for an n-gram of size 4. There's also the concept of skip-gram which allows for gaps in the token. The sentence 'the cat is black' can be split into bigrams as such: 'the cat', 'cat is', and 'is black'. It can also be split into 1-skip-bigrams 'the is' and 'cat black'. Sometimes, skip-gram can be more useful than straightforward n-gram. 1-skip-bigram is able to capture the two most important words in this sentence, 'cat black', while simple bigram failed to do so.

2.2 Cosine Similarity

In making summaries, especially document summaries, there comes a point where you have to compare similarities and differences between the content of documents. One way to do this is by looking at vector representations of the documents and computing the cosine similarity to determine how similar they are. There are several ways of creating vector space models for documents. The simplest of which is tf-idf (term frequency-inverse document frequency).

2.2.1 TF-IDF

Term frequency (TF) is a measure of how many times a word occurs in a document. Term frequency has the bias of scaling up terms with a high count and scaling down terms with a low count. This is an issue as it gives importance to words with high frequency that don't really carry any important information, e.g., the word 'the' or 'a'.

$$TF(t) = \frac{number\ of\ times\ a\ term\ t\ appears\ in\ a\ document}{total\ number\ of\ terms\ in\ the\ document}$$

Inverse Document Frequency (IDF) measures how important a term is in a document. This solves the issue presented by TF by normalizing the weight. Looking at a single document in a collection of documents, IDF is modeled after the observation that the fewer times a word occurs in other documents, the more important that word is in this document

$$IDF(t, D) = log\left(\frac{total\ number\ of\ documents\ (D)}{number\ of\ documents\ with\ term\ t}\right)$$

Finally, TF-IDF of a term is calculated as a product of these two metrics:

$$TFIDF = TF * IDF$$

2.2.2 Computing Similarity

Cosine similarity is the angle difference between two vectors calculated from their inner product. The cosine similarity of any two documents can be used to determine how similar they are based on the resulting angle computed from their vector representations. The closer to 1 the value is, the more similar the documents, since $\cos(0^\circ) = 1$. The computation is done using the following formula:

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

2.3 Summary Evaluation Metric: ROUGE

There are many systems for evaluating automatically generated summaries, but research examined in this paper uses the ROUGE metric, hence only the ROUGE metric will be explained. ROUGE, or Recall-Oriented Understudy for Gisting Evaluation, is a software package for automated evaluation of summaries. It's a recall-based system that measures the overlap, using n-gram, word-sequence, and word-pair information, between the summary being tested and human-generated summaries. It has several different settings that can be used for different types of summaries: ROUGE-N, ROUGE-S, ROUGE-L, and ROUGE-W [2].

ROUGE-N is an n-gram recall measure between a candidate summary and human summaries. ROUGE-S uses a skip-bigram co-occurrence statistic that measures any pair of words in their sentence order; this allows for an arbitrary gap between words. ROUGE-L uses the Longest Common Subsequence (LCS) between the candidate and the reference summary. The issue with ROUGE-L is it doesn't differentiate between LCS's that have a gap and those that don't; it gives them the same score. ROUGE-W solves this issue by giving greater weight to consecutive in-sequence LCS's [2]. Consider the following example:

Summary: police killed the gunman

Ref_1: police kill the gunman

Ref_2: the gunman kill police

We are given the generated summary and two reference summaries. Using ROUGE-N as the measuring method where N = 2, both Ref_1 and Ref_2 would result in the same ROUGE-2 score since *"the gunman"* would be a match on both reference summaries. If ROUGE-L is used instead, Ref_1 would give the summary a better score since the LCS between the summary and Ref_1 is three (*"police the gunman"*) whereas it is only two for Ref_2 (*"the gunman"*) [2].

3. METHODS

3.1 Graph-based Approach

A treebank is a parsed text corpus that annotates syntactic or semantic sentence structure [9]. In this research, Fei Lui et al. present a novel abstractive summarization system that is based on the recent (2014) development of a treebank for Abstract Meaning Representation (AMR). It looks at the viability of an abstractive summarization framework based on transformations of semantic representation such as AMR.

Figure 1 gives an example of this AMR technique being used to summarize two sentences into a single sentence. The summarization process is accomplished in three steps [3]:

1. Parse the input sentences to individual AMR graphs.
2. Combine and transform those graphs into a single summary AMR graph.

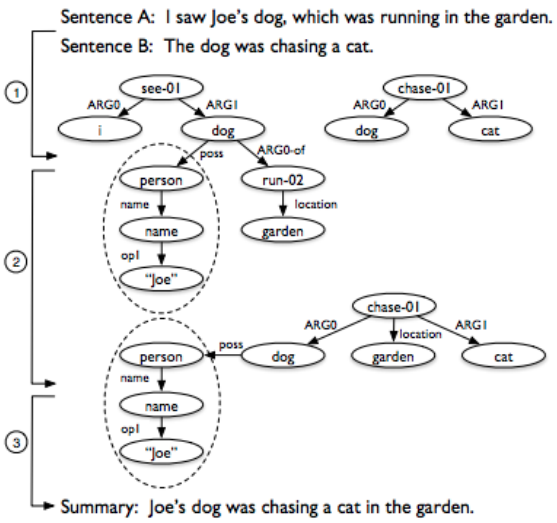

Figure 1: AMR sentence summarization process [3]

3. Generate final summary text from the single graph.

This work focuses on step 2, while leaving full text generation (step 3) for future work. Step 1 assumes the input to be a text document and uses JAMR, an AMR parser and generator, to transform the sentences into AMR graphs. A simple method that reads a bag-of-words (a list of unigrams) off the summary graph is used to get ROUGE-1 evaluations of the generated summaries.

3.1.1 *Individual AMR graphs*

The whole sentence is turned into an AMR graph. Nodes of the graph are labeled with *concepts* and the edges labeled with *relations*. Nodes can be English words ("dog" in Figure 1), PropBank event predicates ("chase-01" in Figure 1), or special keywords("person" in Figure 1). PropBank is a corpus that is annotated with verbal propositions and their arguments. The PropBank event predicate "chase-01" represents a roleset that corresponds to the first sense of the word "chase"; AMR uses approximately 100 of these relations. Other relations in the graph are obtained from the AMR bank, a 20,341-sentence corpus manually constructed by human annotators [3].

3.1.2 *Combine AMR graphs and forming a summary*

In order to combine multiple AMR graphs, two steps are taken: source graph construction and subgraph prediction. In the source graph construction step, identical concepts from the individual AMR graphs are merged into one as illustrated in Figure 2. Then, a subset of the source graph is used to predict the summary graph, which is what is referred to as subgraph prediction.

Source graph construction: This is the step in which multiple sentences' AMRs are merged into one graph called the source graph. The first step of the merge is to create a dummy "ROOT" that connects to the root nodes of the each sentence's AMR graph. Sometimes this step can lead to the resulting graph having coreferenced nodes, i.e, in figure 2 the word "dog" is referenced twice. The two references to the dog are merged to represent the same dog. If needed, an optional graph expansion step is performed where additional edges are added between each pair of concepts to create a fully dense graph. As illustrated in figure 2, the expansion step adds the edge "*dog*" $\rightarrow$ "*garden*", which was never present in either original AMR graph, but is an important part of the resulting summary graph.

Subgraph prediction: Summary graph generation is formulated as a structured prediction problem where a subgraph of the source graph is selected with a goal of preserving meaning, brevity, and fluent language. An integer linear programming (ILP) model is used for the subgraph selection coupled with a learning process using a collection of source graphs and paired with summary graphs. For a predicted subgraph, the ILP seeks to maximize the score computed from the following formula:

$$\sum_{i=1}^{N} v_i \underbrace{\theta^T f(i)}_{\text{node score}} + \sum_{(i,j)\in E} e_{(i,j)} \underbrace{\psi^T g(i,j)}_{\text{edge score}}$$

Starting from the "ROOT", the source graph is indexed by i and j. Let N be the number of nodes in the graph. Let v_i and $e_{i,j}$ represent binary variables where v_i is 1 iff source node i is included in the summary and $e_{i,j}$ is 1 iff the directed edge from node i to j is included. Function f captures the feature representation of node v, while g does the equivalent for edge e. θ and ψ are parameters that estimate the weights of the features and are initially guessed, but adjusted throughout the training process: if a predicted subgraph and has a high score, but ends up not looking like the gold standard structure its parameters are adjusted to penalize the features that led to that prediction by lowering their weights. Likewise, features that lead to a structure that is close to the gold standard are rewarded by increasing their weights.

Given a source graph, the ILP model is constrained to ensure that it picks a connected subcomponent of the source graph. An edge is selected if both of its end points are selected. The subgraph is forced to have a tree structure, where there is at most one incoming edge for each node. A flow constraint is issued to ensure the subgraph is connected. Finally, an optional size constraint can be applied that is based on the number of edges.

3.2 A Real World Application

The English Wikipedia has been growing at a rate of over 1,000 new articles every day since 2005 [8]. Even with these many articles generated each day, there are still some sections of Wikipedia that lack good coverage. Furthermore, adding new articles and editing old ones is very time consuming. There exist methods to automate this process, but the methods do have drawbacks. First, current methods assume that the Wikipedia categories are known. Second, copyright violations can be a problem if the method results in adding long continuous sections of content retrieved from the web into the generated article without paraphrasing. Third, lack of coherence is an issue that plagues these systems. In an effort to address these three drawbacks, Banerjee and Mitra propose WikiWrite, a system for generating Wikipedia articles automatically that doesn't require any information on Wikipedia categories, is abstractive in the way it summarizes (meaning fewer copyright issues), and checks for coherence.

Wikipedia requires articles have a notable and verifiable subject matter in order be retained. Red-linked entities are

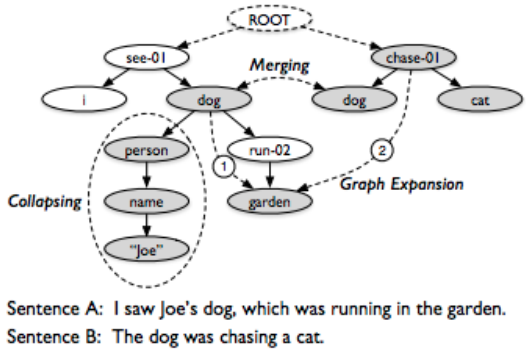

Figure 2: A source graph formed from two individual sentence AMR graphs. A "ROOT" node is added to ensure connectivity. (1) and (2) are among edges added through the optional expansion step, corresponding to sentence- and document-level expansion, respectively. Concept nodes included in the summary graph are shaded [3]

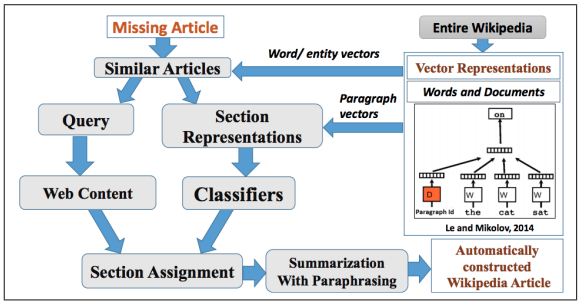

Figure 3: Proposed WikiWrite framework [1]

links to pages on Wikipedia that have not yet been created, but are deemed worthy enough to warrant an article. Red-linked entities are the test subjects for which new articles are generated in this research. As can be seen in Figure 3, the entire Wikipedia corpus is used to obtain vector representations of entities and documents using the paragraph vector distribution memory (PV-DM) model. Similar articles are identified, via cosine similarity, to generate section titles for the missing articles. Classifiers are used to fill in retrieved content from the web into the various sections of the missing articles. Paraphrased summaries for each section are then generated and checked for coherence.

PV-DM: The PV-DM model is based on the idea that several contexts taken from a paragraph can be used to predict the next word. Figure 3, under "Vector Representations", shows the model used where the algorithm is given three word contexts and a paragraph to predict the fourth word. Every word is mapped to a unique vector that is a column in a matrix W, and every paragraph is mapped to a unique vector that is a column in a matrix D. This model can be used on variable-length texts, i.e, sentences, paragraphs, or documents. In WikiWrite the PV-DM model is used for: 1) Identification of similar articles and 2) Inference of vector

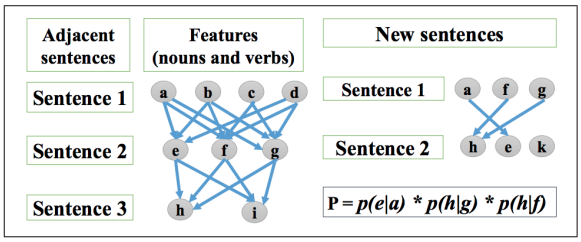

Figure 4: Local coherence estimation between sentences [1]

representations of paragraphs retrieved from the web.

3.2.1 *Content Generation*

Content generation begins with retrieval of information from the web. The text of the entity is not enough by itself to perform a sufficient query. The two most frequent nouns from the introductory sentences of the top 20 similar articles are appended to the entity to create a *reformulated query*. For example, *"Machine Learning"* would get augmented to *"Machine Learning" algorithm intelligence*. The newly constructed query is performed on Google and the informative content from the top 20 search results are retained. Text classifiers, trained from the content of Wikipedia articles similar to the red-link entity, are used to assign the web content into relevant sections of the article being generated.

3.2.2 *Content Summarization*

In this step content is summarized with maximum coherence, informativeness, and linguistic quality. Furthermore, a paraphrasing step is applied in the end to determine the most optimal set of lexical and phrasal transformations to rewrite the summary.

Sentence Generation: A word-graph approach is used in order to generate new sentences from the information retrieved from the web. Bigrams are created from the sentences to represent nodes in the word-graph. Edges between the nodes are constructed if the first word in the bigrams are adjacent in any of the sentences. New sentences are generated by traversing paths along the resulting graph. New sentences that are very similar (cosine similarity ≥ 0.8) to the original sentences are discarded in order to make the results more abstractive and avoid copyright issues.

Further steps are taken in order to decide which sentences to retain and which to discard. The importance of each sentence is determined using the cosine similarity between itself and the reformulated query. Linguistic quality is determined using a trigram language model that finds the best sequences of words. Sentences that are generated from similar initial sentences are weighed heavily, as they are more likely to be coherent together.

Coherence: In order to maximize the global coherence of paragraphs, it is assumed that global coherence is a combined effect of local coherences. Hence, if coherence between each pair of adjacent sentences is achieved, their resulting paragraph will also be coherent. Local coherence is computed by multiplying the individual transition probabilities of features (nouns and verbs) in adjacent sentences. Given three sentences as shown in figure 4, for a transition from sentence 1 to 2, we'd consider all the combinations from 1 to 2: $a \rightarrow e, f, g$; $b \rightarrow e, f, g$, etc. For the entire set of the simi-

lar articles, the total frequency of transitions from $a \rightarrow e$ and $a \rightarrow$ *all other features* is computed. The transition probability of $a \rightarrow e$ is simply the the transition frequency of $a \rightarrow e$ over the transition frequency of $a \rightarrow$ *all other features*. The coherence score between two sentences is the product of the individual transition probabilities of the feature combinations.

Constraints are introduced to optimize the summary. Cosine similarity is used order to mitigate redundancies in the summary. If two sentences have a cosine similarity ≥ 0.5, one of them is dropped. The concept of arcs is used to denote order between sentences. When a summary is being generated dummy sentences marked with arcs are used to indicate the first and last sentence in the summary. The final sentence order of the summary is based on the individual sentence's similarity to the reformulated query: the sentence with the highest similarity is used as the introductory sentence in the article.

Paraphrasing: Each sentence is rewritten by modifying the words and phrases. First, a candidate set of possible modifications is identified using the Paraphrase Database (PPDB). A set of possible modifications for a sentence are examined and the best one is picked based on a readability score. The readability score is computed using a trigram language model of the textual content within a 2-word window either direction of the modification. Consider the two possible modifications to the sentence: *The NSSP initiative will lead to significant economic benefits for both countries.*

1. *significant economic => considerable economic*
2. *economic benefits => financial advantages*

The readability score for 1, can be computed using the sequence – "lead to **considerable economic** benefit for". We compute the cosine similarity using semantic representation from the PVDM of the original sentence and the new paraphrased sentence. This ensures we don't deviate too much from the original meaning. Overlapping modifications are also constrained. For example, the word *economic* is modified twice in the above example. Only one of the modifications is kept for the final paraphrase: *The NSSP initiative will result in major financial advantages for the two countries.*

4. RESULTS

While the two research projects that we have described do not have results that are directly comparable, we will present and describe results from each. Both projects do use F_1 score, also known as F measure, which is a statistical metric used to measure a test's accuracy that is computed from the test's recall and precision [6]. It is not used as widely used as ROUGE in the field of automatic text summarization, hence we won't go into detail explaining it.

4.1 AMR Framework Feasibility

The AMR framework was tested on automatically generated source graphs using JAMR and gold-standard source graphs. The size of the predicted subgraph was based on the number of edges on the gold-standard subgraph to allow for comparable graphs. F_1 scores for both nodes and edges were reported in the subgraph prediction task. JAMR was used to generate a bag-of-words for the summary graphs in order to do ROUGE evaluations. Given a summary graph, JAMR can turn the graph into text by returning the most frequently aligned word for each concept node. Since there's no previous work that uses AMR in the same way as this research, the system's results were compared to "Oracle", a modified version of the AMR system that gives an upper ceiling of the system's performance. For subgraph prediction, the ILP decoder minimizes the cost of the output graph in the way it assigns scores to correct nodes and edges. On the summarization task, Oracle produces a summary by taking the gold-standard AMR parse of the reference summary and using JAMR to produce a bag of words summary [3].

Table 1: AMR Subgraph Prediction Results

	F_1 Score (%)	
	Nodes	Edges
Normal (JAMR)	51.1	20.0
Normal (Gold-Standard)	58.7	39.0
Oracle (JAMR)	68.9	31.1
Oracle (Gold-Standard)	80.7	52.2

Table 2: AMR Summarization Results (ROUGE-1)

	Normal	Oracle
JAMR	44.4	57.8
Gold-Standard	44.3	65.8

Table 1 and 2, shows the results of the subgraph prediction and summarization results respectively. For the subgraph prediction task, JAMR and Gold-Standard denote the origin of the source for the testing data. As expected, the Oracle system did better on both tasks. Node prediction had higher accuracy scores when compared to edge prediction. The low edge prediction values can be accounted for by the fact that the source graph doesn't always predict all the edges that are in the final gold-standard summary.

4.2 WikiWrite

Three measures were used to evaluate WikiWrite's efficiency: 1) Its ability to reconstruct wikipedia articles and accurately assign content to sections. 2) Quality of the articles generated. 3) Retention rate of newly generated articles. For the first two tasks, WikiWrite was compared to two already existing article generation systems on Wikipedia. The first system, Perceptron-ILP, uses a perceptron based ranking algorithm to select informative excerpts in the article. The second, WikiKreator, a system that is designed to improve Wikipedia stub articles.

Reconstructing articles using the existing baseline systems is slow, as they require learning from all articles within a category. To get around this, the reconstruction experiment was restricted to 1000 randomly selected popular articles (articles that are mentioned at least 20 times in other articles). These 1000 articles were not used in training any of the systems. In the reconstruction step, **classification** (assigning content into appropriate sections) and **content selection** (retaining important content from the web in the final article) were evaluated.

4.2.1 Classification

The task here was to predict the section title given the content in the section. As seen in table 3, WikiWrite outperformed WikiKreator in both its accuracy as indicated by the

F_1 scores and time taken to complete the task. WikiWrite was able to assign content to sections about five times faster than WikiKreator mostly due to the classification model in WikiKreator taking significantly longer to run. Perceptron-ILP did not participate in this experiment since the system doesn't involve a classification task.

Table 3: Section Classification Results

Technique	F_1 Score	Average Time
WikiWrite	0.622	∼2 mins
WikiKreator	0.481	∼10 mins

4.2.2 Content Selection

The goal here was to reconstruct Wikipedia articles using knowledge from the web. Articles were constructed from the same 1000 random articles described earlier. In order to evaluate the effectiveness of the query reconstruction model in WikiWrite, a modified system of WikiWrite (WikiWrite (Ref)) was implemented that only uses the references listed in the Wikipedia articles to reconstruct the article. ROUGE was used to evaluate the results. Due to differences in article length, the ROUGE comparisons were restricted to the first 200 words in each section. As seen from table 4, WikiWrite achieved better ROUGE scores than both WikiKreator and Perceptron-ILP. WikiWrite (Ref) out performed regular WikiWrite because it used more reliable and verifiable resources from Wikipedia articles rather than relying upon a Google Search.

Table 4: Content Selection Results

Technique	ROUGE-1	ROUGE-2
WikiWrite	0.441	0.223
WikiWrite (Ref)	0.520	0.257
WikiKreator	0.371	0.183
Perceptron-ILP	0.342	0.169

4.2.3 Generating new articles

The Wikipedia corpus used at the time of the research contained 15,500 red-linked entities that were referenced at least 20 times in other articles. Articles for 50 randomly selected red-linked entities were generated using WikiWrite. As shown in table 5, 47 of them were moved into the Wikipedia mainspace with 12 being retained with zero edits. 35 articles received changes ranging from reference edits to multiple content edits. Seventy-two percent of the references were kept. The three articles that were not retained was due to them containing self-promotional content. At the time of the writing of this paper, all the articles have been removed from Wikipedia due to restrictions on authorship of the articles. Wikipedia is especially concerned with the ethics of such research done without the consent of Wikipedia's contributors or readers.

Table 5: 50 Generated Wikipedia articles

Statistics	
Number of articles in mainspace	47
Entire edit retained	12
Modification of content	35
Average number of edits	11
Percentage of references retained	72%

5. CONCLUSIONS

Abstractive summarization is a summarization technique that tries to emulate how humans summarize. It went through a phase of neglect in terms of research attention, but it's steadily catching on as of late. We've examined an example implementation using semantic representation graphs and a real world application of the process that's used to automatically generate Wikipedia articles.

In the past couple of years the biggest advancement in this field has been achieved through deep learning. The state-of-the-art method for the task of sentence compression, summarizing a sentence to produce a shorter sentence, has been changed and improved three times in the past year alone using neural networks. In a future where information overload will only keep getting worse, it's very exciting to see this much progress in the field.

Acknowledgments

Thanks to Kristin Lamberty, Matthew Linder, and fellow senior seminar students from the class of fall 2016 for their guidance and feedback.

6. REFERENCES

[1] S. Banerjee and P. Mitra. Wikiwrite: Generating wikipedia articles automatically. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2740–2746, 2016.

[2] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In S. S. Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[3] F. Liu, J. Flanigan, S. Thomson, N. M. Sadeh, and N. A. Smith. Toward abstractive summarization using semantic representations. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1077–1086, 2015.

[4] A. Nenkova and K. McKeown. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2âĂŞ3):103–233, 2011.

[5] Wikipedia. Automatic summarization — Wikipedia, The Free Encyclopedia, 2016. [Online; accessed 26-September-2016].

[6] Wikipedia. F1 score — Wikipedia, The Free Encyclopedia, 2016. [Online; accessed 3-December-2016].

[7] Wikipedia. Information overload — Wikipedia, The Free Encyclopedia, 2016. [Online; accessed 26-September-2016].

[8] Wikipedia. Modelling wikipedia extended growth — Wikipedia, The Free Encyclopedia, 2016. [Online; accessed 12-October-2016].

[9] Wikipedia. Treebank — Wikipedia, The Free Encyclopedia, 2016. [Online; accessed 13-October-2016].