Automatic text summarization

Sütő Evelyne

December 22, 2019

Contents

Abstract

The aim of this paper is to provide an overview of the different methods of summarization developed since it first appeared. Our main goal is to present these methods as an introduction to this vast topic, as a result we will try to cover the systems that have been used as a base for other more complicated models, approaches.

We will walk through the most important statistical approaches, then we will introduce the problem of text connectivity when generating coherent summaries and some possible solutions to avoid this problem, such as, lexical chains, Rhetorical structure theory, G-Flow. Afterwards we will see how summarization problems problem can be represented as a Graph to use graph based rankings for sentence extraction.As a possible abstract generation system we will explore the Encoder-Decoder model and how this should be altered for the summary generation problems.

In our final chapter we will see what evaluation metrics are mainly used to compare different summarization systems.

1 Introduction

The task of automatic document summarization has been an active research topic since the 1950's. The first research has been conducted in 1958 [Luhn, 1958], which had as its main purpose the generation of technical literature abstracts with the purpose of saving the time of the reader by providing them a short summary of the paper to decide if its useful for them.

However, since the appearance of the world wide web our society has been generating data in an alarmingly fast pace and it becomes challenging to search or process them. As a result, summarization systems which are designed to take a single document or a cluster of documents as input, and produce a concise and fluent summary conveying the most important information are used to provide timely access and processing for the exponentially growing data every day. [Yao et al., 2017]

The problem of document summarization is a very complex one and its an active research field today as well in machine learning and natural language processing. The approaches are changing as well, in the beginning the main goal was extract generation, but nowadays we are starting to look for solutions for abstract generation.

Extractive summarization produces summaries by concatenating several sentences taken exactly as they appear in the original documents being summarized. By contrast, abstractive summarization uses different words to describe the contents of the original documents rather than directly copying original sentences. [Yao et al., 2017]

The purpose of this paper is to present the theoretical background of text summarization, its recent advances and experimental results. Furthermore, since the evaluation of this problem is a challenging task in itself we would like to introduce the most used evaluation methods as well.

2 Extractive summarization

Let us first consider the extractive summary generation techniques and its advances. In extractive summarization we are mainly focused on calculating the importance of each sentence in our document, while still considering the grammatical correctness of the overall summary while choosing the most important sentences to represent the overall meaning of the text. As opposed to abstractive summary, where we need to comprehend the text that we are summarizing.

2.1 Word frequency and statistical methods

The first measurement in selecting which sentences should be present in the document extract was the word frequency and the relative position of words in a sentence. Of course, the researchers excluded the so called stop-words, which have little significance in the semantics of a sentence but appear very frequently such as "the, as" etc. Another assumption used in this approach was that the closer some words are associated the most probable that they will appear near to each other in a sentence, conveying important information of the subject. From these considerations a significance factor can be derived which reflects the number of occurrences of significant words within a sentence and the linear distance between them due to the intervention of non-significant words. [Luhn, 1958]

This intuitive idea was expanded, improved in later years. For example in [Edmundson, 1969] they use four methods for sentence selection Cue, Key, Title and Location. The Cue method uses a cue word dictionary, where a cue word is considered to introduce important information in texts such as "significant", "important", "result" etc. This dictionary is defined by calculating word frequency, dispersion and selection ratio (ratio of number of occurrences in extractor-selected sentences to number of occurrences in all sentences of the corpus). Key words are used to consist of topic based words for the specific documents. The Title method is based on the assumption that significant words that are present in the title must be descriptive of the document's subject. The Location method relies on certain characteristics of document structures such as, topic sentences appear very early on in sections. This method can be described with a linear equations using the formula:

$$w_1 * C + w_2 * K + w_3 * T + w_4 * L$$

Where the weights can be learned to maximise the performance of the method.

As a more sophisticated measurement we may use tf*idf weights as well, which uses a corpus from containing documents from the same genre as the document to be summarized in order to see the expected occurence probability of a word in the document. The tf*idf weights of words are good indicators of importance, and they are easy and fast to compute. [Nenkova and McKeown, 2012]

$$tf * idf = c(t) * log\frac{D}{d(t)}$$

where c(t) is the term frequency in the corpus, D is the number of documents and d(t) is number of document t occurred in.

Naturally when thinking of frequent words as a way of classifying important sentences we think of topic specific words. Topic signature is defined as a family of related terms,

$TS = topic, < (t_1, w_1), ..., (t_n, w_n) >$ where topic is the target concept and signature is a vector of related terms. Each t_i is an term highly correlated to topic with association weight w_i . The number of related terms n can be set empirically according to a cutoff associated weight. [Lin and Hovy, 2000] These topic specific terms then can be used in the process of summarization to group related topic specific sentences together. The method for topic signature extraction in [Lin and Hovy, 2000] assumes that semantically related terms co-occur. They use pre-classified texts with a set of relevant and a set of non-relevant texts, and assume that if the probability of a term occurring in the relevant or non relevant set is equal then the term is not descriptive

$$H_1 : P(t|R) = P(t|B) = p$$

, but if the probability of a term occurring in the relevant set is higher than the probability of occurrence in the non-relevant set it indicates strong relevance of the term.

$$H_2 : P(t|R) = p_1 P(t|B) = p_2, p_1 > p_2$$

Assuming binomial distribution,

$$b(k, n, p) = \binom{n}{k} p^k (1 - p)^(n - k)$$

we can calculate the $-2log\lambda$ (where λ is the likelihood ratio, $\lambda = \frac{b(k,n,p)}{b(k_R,n_R,p_r)b(k_B,n_B,p_B)}$) value for each term, determine a cutoff association weight and the number of terms to be extracted. As a result we can rank sentences by relevance to the signature topic of a document, rather than simple frequencies.

2.2 Text connectivity

While the methods mentioned in 2.1 are good for extracting relevant sentences they can fail to convey text that is coherent. By simply choosing sentences with high "relevance" score we might generate parts of the document with anaphoric expressions (expressions, words which refer back to previously mentioned elements of the text), which out of context don't make sense semantically.

A lexical chain is a sequence of related words in writing, spanning short (adjacent words or sentences) or long distances (entire text). A chain is independent of the grammatical structure of the text and in effect it is a list of words that captures a portion of the cohesive structure of the text. Lexical chains as means of summary generation was introduced in [Barzilay and Elhadad, 1999]. Lexical chain calculation algorithms rely on databases, such as Wordnet, in order to determine the relatedness of words. The vocabulary in Wordnet is represented in a hierarchical manner which models the connections between the meaning of the synsets (a synset is a collection of words such as synonyms, antonyms). Lexical chains are computed by selecting a group of candidate words, for each candidate find an appropriate chain relying on relatedness, if found update the chain by adding the candidate to it and updating each member's senses to make sure each word in the chain relates to the same sense. If no chain found we can create a new one. The relatedness can be calculated using the closeness of the words in the Wordnet hierarchical database. As shown in [Barzilay and Elhadad, 1999], the intuition for using lexical chains rather than frequency based methods is that writers tend to use multiple words for the same concept throughout a discourse in order to avoid repetition. This means that even if a concept does appear various times, since the author is using different words to describe it, the frequency of the specific words relevant to that concept might be low, which can re resolved using lexical chains instead. Another challenge in this area is to find a scoring system to select the most significant chain. The used scoring formula in their paper was:

$$Score(Chain) = Length(Chain) * Homogeneity$$

After selecting the strongest chains, for each chain choose the sentence that contains the first appearance of a representative member of the chain in the text, where a representative member is defined as having a frequency greater than the average the average word frequency in the chain. While this method does have a better overall knowledge of the topic it does not resolve the problem of anaphoric expressions.

Another approach for extracting strongly connected summaries is to use Rhetorical structure Theory. Rhetorical structure theory is a descriptive theory of a major aspect of the organisation of a text. It identifies hierarchic structure in text, by providing the relations between text parts in functional terms, transitions of the relations and the extent of the items related to the relation. The theory depends on four objects: Relations, Schemas, Schema Applications, Structures. A relation identifies a particular relationship that can hold between two texts. Based on relations, schemas define patterns in which a particular span of text can be analysed in terms of other spans. The schema applications define a somewhat more flexible ways to initiate a schema. The structure of a text then can be described as a composition of schema applications. A relation is defined by a nuclei, which are the most important part of a text, and satellite which can be considered secondary and can even be viewed as incomprehensible without the nucleus in many instances. As a result, there is a possibility to use the structures given by rhetorical structure theory as summary generation method if we keep the nuclei sentences and remove all satellite texts. The relations are applied recursively, until all text units are classified as part of a relation. As a result we derive a tree representation of the discourse. For an extensive description of this method refer to the cited source. [Mann and Thompson, 1988]

Another approach to improve summary coherence is to use the G-Flow system. [Christensen et al., 2013] G-FLOWs core representation is a graph that approximates the discourse relations across sentences based on indicators including discourse cues, deverbal nouns, co-reference, and more. This graph enables G-FLOW to estimate the coherence of a candidate summary. The main idea of the method is to use a directed graph where the nodes are sentences, and each edge represents a pairwise ordering constraint necessary for a coherent summary. By definition, any coherent summary must obey the constraints in this graph. By using this graph we can estimate how coherent our extracted text is. This method has been applied to multi-document summarization problems, but can be considered as an option in single document summarization as a way of improving coherence. It's also important to note that since the system incorporates other constraints as well other than coherence, like ordering, length, redundancy. In order to build the graph the researchers used deverbal noun references. A deverbal noun reference appears when an event is mentioned in a verbal phrase and subsequent references use deverbal nouns like "attacked" and "the attack". These pairs are identified with the help of Wordnet. For each word they determine potential noun references using a path length of up to two in WordNet. (A path length is defined by the "derivationally related" link.) After generating these pair they filter out the ones that are too generic by using a corpus on which the importance of these pairs can be calculated. As an other indicator they use lexical chains to classify if sentences contain the same events as their topic, which can be used as a new relation. The system also identifies discourse markers like "however", "moreover" as a way of recognising relations. Co-referent mentions are also modelled as relations to avoid the extraction of sentences that need more context in order to be understood. The objective of the system is to maximise the next function:

$$Sal(X) + aCoh(X) - b|X|$$

whith respect to

$$\sum_{n=1}^{N} len(x_i) < B$$

$$redundant(x_i, x_j) = 0, \forall x_i, x_j \in X$$

The coherence function is derived from:

$$Coh(X) = \sum n = 1^{N-1} w_{G+}(x_i, x_{i+1} + \lambda w_{G-}(x_i, x_{i+1})$$

where:

$$w_{G+} = \text{positive edges}$$

$$w_{G-} = \text{negative edges}$$

$$\lambda = \text{tradeoff coefficient}$$

The salience can be calculated as the sum of all the salience values of each sentences that appear in the summary.

As shown in the paper as well, this system is capable of generating promising summaries and the reason behind it is that it combines two important measures, salience and coherence, which are vital in readable, important text generation.

2.3 Graph based methods

While the above mentioned (Section 2.2) system, G-Flow, can be considered a graph based method, we chose to include it in the text connectivity methods since it's main purpose is to improve the coherence and readability of summaries.

One of the used algorithms is called TextRank, which is a graph based ranking algorithm for sentence ranking which has been applied successfully for summary extraction. [Mihalcea and Tarau, 2004] A graph-based ranking algorithm is a way of deciding on the importance of a vertex within a graph, by taking into account global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. The basic idea implemented by a graph-based ranking model is that of voting or recommendation. Each link in the graph can be considered as a vote, the more link a vertex has the greater its importance is. We can also weigh the votes according to the voters, the higher the importance of a voter is the more that vote means. Each vertex has an initial score defined by:

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

Where:

- In(V) = set of vertices pointing to V
- Out(V) = set of vertices to which V points to
- d = damping factor, probability of jumping from a given vertex to another random vertex, usually set to 0.85

The values of the vertices are assigned randomly at first and the algorithm runs until convergence is achieved. Since natural language texts might include multiple or partial links between the edges that are extracted from the text the authors of the paper introduced a new formula which includes the weights of the edges as well.

$$WS(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{w_{j,i}}{\sum_{V_k \in Out(V_j)} w_{j,k}} WS(V_j)$$

In order to convert the text into a graph we can use different text units, like words, sentences, phrases, and the relations can be modelled by lexical relations. For summary generation the chosen units are full sentences. As far as relationships between sentences go, the authors use a similarity measure to model groups of sentences that are referring to the same concepts, have overlaps between theirs subjects. So the recommendations are made based on how similar the concept that they describe is. The similarity of two sentences can be described with the equation:

$$Similarity(S_i, S_j) = \frac{|w_k | w_k \in S_i \& w_k \in S_j|}{log(|S_i|) + log(|S_j|)}$$

After the vertex values are calculated the resulting graph is a highly connected one, we apply the ranking algorithm on it until convergence is achieved, then we sort the edges according to their graph scores and extract the first n sentences based on the scores to achieve the extract. The TextRank algorithm successfully achieves a good summary of the text without having to pre-process any training data, which makes it a desirable for extract generation in languages without corpus for training. It's also important to mention that since the algorithm itself has a recursive step we are able to discover more complicated relationships in the text.

2.4 Sentence reformulation, post-processing

Since the methods mentioned earlier mainly focus on extractive summaries we might need to add an additional step of post-processing to get more informative, shorter, cohesive sentences.

In the last few years there have been increased interest in paraphrase generation due to the recent success of the encoder-decoder model in natural language processing tasks such as machine translation, speech recognition, abstract generation, language modelling etc., which we will cover as well. One method of paraphrase generation is the multi-pivoting method, which uses machine translation to translate a sentence into a chosen language and then translates it back from that language to the original, thus creating a paraphrase for the original sentence. [Mallinson et al., 2017] As a possible improvement, paraphrase generation can be applied as a post-processing task to get closer to human abstracts.

The other big challenge is sentence reordering, which is mainly a problem in multi-document summarization, but it might appear as a room for improvement in single document summarization as well. Classic reordering approaches include inferring order from weighted sentence graph, or perform a chronological ordering algorithm that sorts sentences based on timestamp and position. [Yao et al., 2017]

3 Abstractive summarization

3.1 Towards end-to-end abstract generation systems

Abstractive summaries try to present a shorter version of the text input, based on deeper understanding of the conecpts being represented and it may contain newly added sentences, phrases as well to improve the generated abstract.

One of the promising approaches that has been researched as a possibility of fully abstractive summarization is the use of encoder-decoder models. One of its main advantages is that it only needs an input, which can be raw texts, without needing too much pre-processing, and the desired output in order to train it. Usually it is implemented with some kind of recurrent neural network as the encoder and decoder unit. As a result this model is very sensitive to exploding and vanishing gradients, making it challenging to train. As far as its success as an abstract generation model goes it is still very far from giving us state of the art results, but its successes in sentence and shorter text summarization makes it a worthy candidate for this task as well.

3.2 Encoder-decoder model

3.2.1 Basic elements of the model

This model is also referred to as the sequence to sequence model. One simple representation of this model can be seen on Figure 1

Since the input can be of varying size which could not be modelled by a recurrent network a simple strategy is to map the input sequence to a fixed sized vector using one RNN and then map the vector to the output sentence with another RNN. Since this model might need to learn really long sentences we Long Short-Term Memory Networks are often used in these problems. The idea is to use an LSTM to read the input data one time step at a time to obtain a fixed dimensional vector representation, and then to use another LSTM to extract the output sequence from that vector. So the LSTM's goal will be to estimate the probability of $P(y_1, y_2, .., y_l | x_1, x_2, .., x_t)$ where x is the input sequence and y is the output

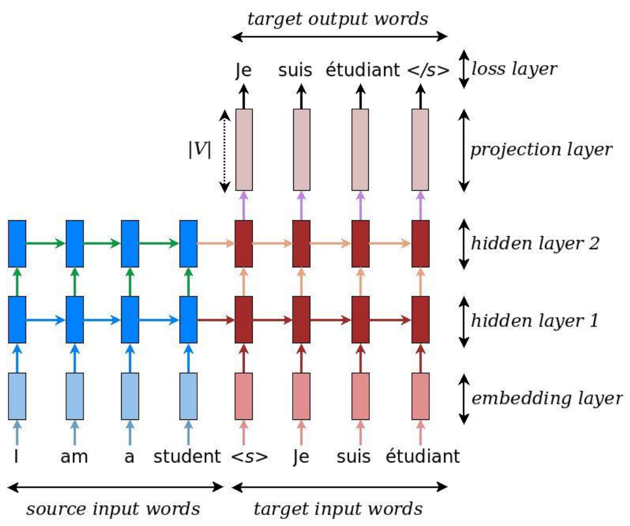

Figure 1: Neural machine translation example of a deep recurrent architecture proposed by for translating a source sentence "I am a student" into a target sentence "Je suis tudiant". Here, "s" marks the start of the decoding process while "/s" tells the decoder to stop. Image source: https://github.com/lmthang/thesis/blob/master/thesis.pdf

sequence whose length might be different. It is also required for the input sequence to end with a special *Stop* character. The model is based on two components: the first one for the input sequence the a second for the output sequence. [Sutskever et al., 2014]

It is also important to mention that the researchers in [Sutskever et al., 2014] have found that reversing the source sentence can lead to better results even though they can't really explain why that might be.

The result of the second layer will be a matrix of probabilities from which we can decode our prediction using two approaches: maximum likelihood method and beam search algorithm which will be discussed in 3.2.2.

As we can see on Figure 1 a conventional Encoder-Decoder model usually has an Embedding layer (in NLP problems) which translates the input sentence to a dense vector representation using the Vocabulary that we provide in the beginning. The output of the embedding is then fed to the encoder model which will return the input for the decoder. The output from the decoder layer then can be processed to retrieve the prediction.

3.2.2 Decoding Algorithms

In the literature there is two main algorithms which are used to extract the output of the decoder.

3.2.3 Greedy Decoding

Greedy decoding is one of the simplest approaches for decoding sentences and it is based on conditional possibility calculations.

In greedy decoding, we follow the conditional dependency path and pick the symbol with the highest conditional probability so far at each node. This is equivalent to picking the best

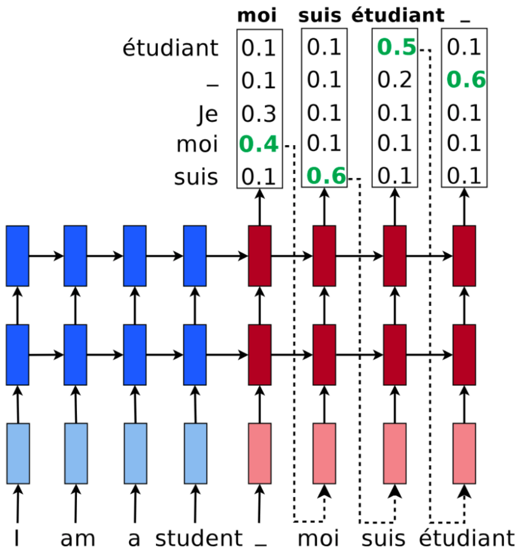

Figure 2: Greedy decoding algorithm example of a deep recurrent architecture proposed by for translating a source sentence "I am a student" into a target sentence "moi suis etudiant" Image source: https://github.com/lmthang/thesis/blob/master/thesis.pdf

symbol, one at a time, from left to right in conditional language modelling. [Gu et al., 2017]

However this does not give us the best results because it only takes into consideration the element at previous time step which means that if it makes a wrong prediction every prediction made after that will be based on false information.

3.2.4 Beam-Search algorithm

Beam search keeps $K > 1$ hypotheses, unlike greedy decoding which keeps only one during decoding. At each time step t, beam search picks K hypotheses with the highest scores

$$\prod_{i=1}^{t} p(y_t|y < t, X)$$

When all the hypotheses terminate (outputting the end-of-the sentence symbol), it returns the hypothesis with the highest log-probability. Despite its superior performance compared to greedy decoding, the computational complexity grows linearly with respect to the size of beam K, which makes it less preferable especially in the production environment. [Gu et al., 2017]

Now that we understand how the basic principle works in this model we can approach their usage in summary generation as well.

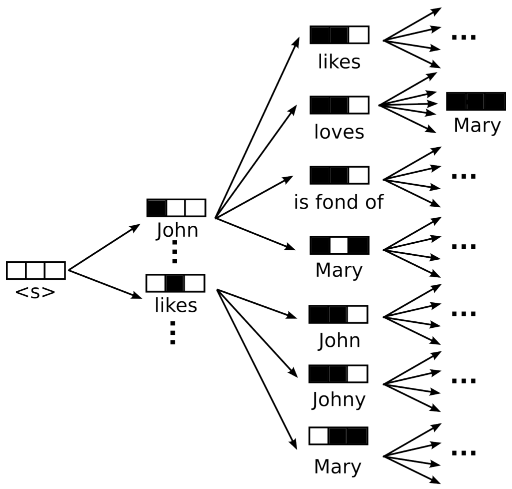

Figure 3: Beam-Search algorithm At each time step each node is expanded with the K most promising new nodes.
Image source: http://mttalks.ufal.ms.mff.cuni.cz/index.php?title=File:Beam-search.png

3.3 Abstract generation with encoder-decoder model

Now that we have seen the basic elements of the encoder-decoder model let us present some researches that use this approach to solve the problem of abstract generation.

The application of sequence to sequence models for abstract generation was done based on the success of neural machine translation. As it is stated in the research paper [Nallapati et al., 2016] besides the similarities, such as sequential data mapping, the problem of abstract generation is very different from machine translation. This is because the model doesn't have the advantages of similar sequence lengths for the input and output, as we usually have for translation problems, summarization has to concentrate the main ideas of the text without loss of information, so we need to gasp the important parts rather than map each part directly. Another advantage that we have in translation is that this mapping is usually a one to one mapping with certain special cases where we need to reformulate the output, as a result it is easier for the model to find the patterns in the training data. In [Nallapati et al., 2016] the researchers have addressed several problems. Their baseline model is identical with the one used for machine translation, containing a bi-directional GRU-RNN encoder and a uni-directional GRU-RNN decoder, and an attention mechanism over the source-hidden states and a soft-max layer over target vocabulary to generate words. They also use the large vocabulary trick (LVT), by restricting the decoder's vocabulary of each mini-batch to words in the source documents of that batch. This trick speeds up the training and the decoding as well, and it is well suited for summarization. Since the most important keys of the specific topic needs to be identified as well, the reasearchers try to capture additional linguistic features such as parts-of-speech tags, named-entity tags, and TF and IDF statistics of the words. These are achieved by creating additional look-up based embedding matrices for the vocabulary of each tag-type, similar to the embeddings for words. For continuous features such as TF and IDF, they convert them into categorical values by discretizing them into a fixed number of bins, and use one-hot representations to indicate the bin number they fall into. This allows them to map them into an embeddings matrix like any other tag-type. They also solve the problem of rare or unseen words by using a switching generator-pointer. The main idea of this approach is to simply point to the location of the out of vocabulary words in the source document instead and use a copy mechanism for them when decoding the generated summary. While this paper has been published a few years ago, in 2016, when they did manage to generate state of the art results as far as the general evaluation, metrics such as Rough, is concerned, they do recognise that this model can fail to capture the essence of the input text, generating comic outputs. This model still can be considered as a very good baseline for future work in abstract generation.

4 Evaluation metrics

In order to easily evaluate and compare the newly developed methods we need to use a widely accepted evaluation metric. This metric not only helps with the comparison, but it implements a way to fully automate the evaluation of the generated summaries. Currently the ROUGE (Recall-Oriented Understudy for Gisty Evaluation) [Lin, 2004] metrics are the standard for automatic evaluation of summarization.

4.1 N-gram Co-Occurrence Statistics

The first recall oriented evaluation metrics that has been introduced for summary evaluation was in 2003 in [Lin and Hovy, 2003]. The paper uses N-gram co-occurrence statistics to evaluate the summaries. The researchers had as a starting point the BLEU score, which is successfully used for machine translation evaluation even today.

The main idea of BLEU is to measure the translation closeness between a candidate translation and a set of reference translations with a numerical metric. To achieve this goal, they used a weighted average of variable length n-gram matches between system translations and a set of human reference translations and showed that a weighted average metric, i.e. BLEU, correlating highly with human assessments. [Lin and Hovy, 2003] The BLEU score calculates an N-gram precision using:

$$BLEU = BP \exp(\sum_{n=1}^{N} w_n \log p_n)$$

Where:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C \in \{Candidates\}} \sum_{n-gram \in C} Count(n-gram)}$$

$$BP = \left\{ \begin{array}{ll} 1, & \text{if } |c| > |r| \\ e^{1-\frac{|r|}{|c|}}, & \text{if } |c| \leq |r| \end{array} \right\}$$

Where $Count_{clip}$(n-gram) is the maximum number of n-grams co-occurring in a candidate translation and a reference translation, and Count(n-gram) is the number of n-grams in the candidate translation. BP is a brevity penalty which prevents very short translations.

This BLEU score can easily be altered for summaries by using:

$$c_n = \frac{\sum_{C \in \{\text{Model Units}\}} \sum_{n-gram \in C} Count_{match}(n - gram)}{\sum_{C \in \{\text{Model Units}\}} \sum_{n-gram \in C} Count(n - gram)}$$

Where $Count_{match}$(n-gram) is the maximum number of n-grams co-occurring in a peer summary and a model unit and Count(n-gram) is the number of n-grams in the model unit. c_n can be seen as a recall metric, rather than a precision metric. They also introduce a brevity bonus, instead of brevity penalty, because in summarization if we express something in a shorter manner that should be awarded rather than penalised.

The resulting equation then becomes:

$$Ngram(i, j) = BB \exp(\sum_{n=i}^{j} w_n \log c_n)$$

According to their experiments, the unigram co-occurrence statistics is a good automatic scoring metric. It consistently correlated highly with human assessments and had high recall and precision in significance test with manual evaluation results.

4.2 ROUGE

A year later the N-gram Co-occurence statistics evaluation metric has been used as the foundation for the Recaull Oriented Understudy for Gisting evaluation (ROUGE) package. [Lin, 2004]

Formally, ROUGE-N is an n-gram recall between a candidate summary and a set of reference summaries.

$$ROUGE - N = \frac{\sum_{S \in \{\text{Reference summary}\}} \sum_{n-gram \in S} Count_{match}(n - gram)}{\sum_{S \in \{\text{Reference summary}\}} \sum_{n-gram \in S} Count(n - gram)}$$

Where n stands for the length of the n-gram, and $Count_{match}$(n-gram) is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries. Note that the number of n-grams in the denominator of the ROUGE-N formula increases as we add more references. Which is understandable because there might exist multiple good summaries. Note that the numerator sums over all reference summaries. This effectively gives more weight to matching n-grams occurring in multiple references. Therefore a candidate summary that contains words shared by more references is favored by the ROUGE-N measure.

If we have multiple reference summaries we may use:

$$ROUGE - N_{multi} = argmax_i ROUGE - N(r_i, s)$$

The researchers also introduce the $ROUGE - L$ measure, which is based on the longest common subsequence. A sequence Z = $[z_1, z_2, ..., z_n]$ is a subsequence of another sequence X = $[x_1, x_2, ..., x_m]$, if there exists a strict increasing sequence $[i_1, i_2, ..., i_k]$ of indices of X such that for all j = 1, 2, ..., k, we have $x_{i,j} = z_j$. Given two sequences X and Y, the longest common subsequence (LCS) of X and Y is a common subsequence with maximum length.

$$R_{LCS} = \frac{LCS(X,Y)}{m}$$

$$P_{LCS} = \frac{LCS(X,Y)}{n}$$

$$ROUGE - L = \frac{(1+b^2)R_{LCS}P_{LCS}}{R_{LCS} + b^2 P_{LCS}}$$

Notice that ROUGE-L is 1 when X = Y; while ROUGE-L is zero when LCS(X,Y) = 0. One advantage of using LCS is that it does not require consecutive matches but in-sequence matches that reflect sentence level word order as n-grams. The other advantage is that it automatically includes longest in-sequence common n-grams, therefore no predefined n-gram length is necessary. By only awarding credit to in-sequence unigram matches, ROUGE-L also captures sentence level structure in a natural way.

It has been proven that these measure correlate very well with the evaluation metrics given by human judgement of the summaries. It is also worth to mention that they work better if we have multiple references to calculate the ROUGE scores.

The package contains some other ROUGE measures as well, but since usually researches use the ROUGE-N and ROUGE-L measure we did not include those here.

5 Conclusions

In this paper we have tried to include some different approaches that one could follow when trying to generate a summary.

As we have seen this research topic has improved a lot since its first appearance when the researchers have only used naive statistical computations to extract relevant knowledge from texts. We can see that in recent years researchers are trying to get away from the ”simple” extractive summaries and try to include methods for abstractive summary generation as well. As a result, different machine learning approaches has been applied to this task, such as the use of neural networks. Another trending approach in the recent years have been to use reinforcement learning as well for coherent extractive summarisation. [Wu and Hu, 2018] Even multi-agent systems have been introduced to solve this task. [Celikyilmaz et al., 2018] These methods were not included in our survey because our goal was to introduce simpler methods in this paper, but it is very interesting to see that there are inifinite probabilities when approaching this specific problem, which might be the reason for its popularity as well.

We also introduced the most used evaluation metric in this paper for a better understanding of the reasearch results in this domain, but it is worth to mention that they do have their limitations. For example, ROUGE scores will remain unchanged after arbitrarily disordering the sentences in a summary, since ROUGE metrics are designed mostly for detecting information coverage rather than coherence or other important quality factors. Studies have shown that for lower-order ROUGE scores they tend to detect significant differences among systems, even though human judges find that they are actually comparable. [Yao et al., 2017]

As a conclusion we can state that document summarization remains an open problem in natural language processig, but we are getting closer to generating good quality results.

References

[Barzilay and Elhadad, 1999] Barzilay, R. and Elhadad, M. (1999). Using lexical chains for text summarization. *Advances in automatic text summarization*, pages 111–121.

[Celikyilmaz et al., 2018] Celikyilmaz, A., Bosselut, A., He, X., and Choi, Y. (2018). Deep communicating agents for abstractive summarization. *arXiv preprint arXiv:1803.10357*.

[Christensen et al., 2013] Christensen, J., Soderland, S., Etzioni, O., et al. (2013). Towards coherent multi-document summarization. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1173.

[Edmundson, 1969] Edmundson, H. P. (1969). New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285.

[Gu et al., 2017] Gu, J., Cho, K., and Li, V. O. (2017). Trainable greedy decoding for neural machine translation. *arXiv preprint arXiv:1702.02429*.

[Lin, 2004] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

[Lin and Hovy, 2000] Lin, C.-Y. and Hovy, E. (2000). The automated acquisition of topic signatures for text summarization. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 495–501. Association for Computational Linguistics.

[Lin and Hovy, 2003] Lin, C.-Y. and Hovy, E. (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

[Luhn, 1958] Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165.

[Mallinson et al., 2017] Mallinson, J., Sennrich, R., and Lapata, M. (2017). Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 881–893.

[Mann and Thompson, 1988] Mann, W. C. and Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text-Interdisciplinary Journal for the Study of Discourse*, 8(3):243–281.

[Mihalcea and Tarau, 2004] Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

[Nallapati et al., 2016] Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

[Nenkova and McKeown, 2012] Nenkova, A. and McKeown, K. (2012). A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

[Wu and Hu, 2018] Wu, Y. and Hu, B. (2018). Learning to extract coherent summary via deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Yao et al., 2017] Yao, J.-g., Wan, X., and Xiao, J. (2017). Recent advances in document summarization. *Knowledge and Information Systems*, 53(2):297–336.