PERSONALIZED SUMMARIZATION WITH SHARED ATTENTION AND CONCEPT SPACES

by

Mihai George Mehedint

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science
in Computer Science

Charlotte

2019

Approved by:

_____
Dr. Wlodek Zadrozny

_____
Dr. Samira Shaikh

_____
Dr. Xi Niu

ABSTRACT

MIHAI GEORGE MEHEDINT. Personalized summarization with shared attention and concept spaces. (Under the direction of Dr. WLODEK ZADROZNY)

The information available to users is overwhelming in today's world. Therefore, it is essential to filter and convey only the essential information in a personalized fashion. We explored the automatic summarization of text as a means to address this problem. In addition, the current work explores two mechanisms: the shared attention and conceptual spaces aiming to extract abstract ideas from text and personalize them according to the users' interests.

The CNN_DM database was used as a source for both text and ground truth summarizations. User profiles were extracted from user generate commentaries in NYT, to provide insight into how individuals use abstraction. We utilized several recurrent neural networks with an attached attention mechanism. The results were comparable to the state of the art pointer generator network (0.145 F1 score). The shared attention RNN had an F1 score of 0.13. Moreover the Recurrent Neural Network equipped with a conceptual space mechanism scored 0.079 F1 on the same dataset.

Summarization is the process of condensing the source text with loss of information and preservation of essential ideas.

The existing methods of summarization, whether done by humans or automatic systems, create impersonal summarizations without the user profile in mind. In the current work we show that personalized summarization can be achieved by utilizing

neural networks of cells equipped with attention mechanisms and by introducing semantic information via concept spaces.

The models proposed here achieve similar performance as the state of the art while having user's content as a guide to their interests.

ACKNOWLEDGMENTS

I thank my committee members, Dr. Wlodek Zadrozny, Dr. Samira Shaikh and Dr. Xi Niu.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

RNN …….. Recurrent Neural Network

LSTM …… …….. Long Short Term Memory

ATS …… Automatic Text Summarization

TS …… Text Summarization

Rouge …… Recall-Oriented Understudy for Gisting Evaluation

BLUE …….. BiLingual Evaluation Understudy

NYT ……. New York Times

CNNDM …… CNN & Daily Mail data

CBOW …… Continuous Bag of Words

TF ……Term Frequency

TF-IDF ……..…… Term Frequency Inverse Document Frequency

OOV …….. ……Out of vocabulary words

PG …… Pointer Generator

CHAPTER 1: INTRODUCTION

1.Introduction

Worldwide, digital corpora amass information beyond human mental capacity to interpret and structure in its entirety. This unstructured data represents a difficult task for automatic processing, due to the elusive nature of semantics in various contexts, to language differences and to the lack of clearly defined rules to accomplish text processing. In addition, user interests vary between individuals and therefore it is imperative to process text in a manner that is efficient and relevant to user's needs for information.

The current work shows that automatic personalized summarization can be accomplished via two mechanisms: attention on contexts and concept spaces.

Summarization reduces the length of the original document while preserving the ideas within. During this process, less relevant information from the source document is discarded. Therefore, automatic text summarization is a process of compression with loss of information (Knight 2002).

Creating summaries can be accomplished by humans or by automatic computerized systems. Both avenues have strong and weak points.

The human process of summarization stems from understanding the source documents. In addition, the text being generated is highly influenced by external factors as: knowledge, background, affect, personal opinions, education and life experience. This massive information that gives them identity to the summarizer determines the quality of the summarization. Research has shown that these factors lead to a different summarization outcome of the same source document if the process is repeated after a

few days (Torres-Moreno 2014). Ergo, the summarization is not consistent and the cognitive processes play an important role in it.

In contrast, the automatic summarization of the same source document is consistent. The process is devoid of emotion and personal experience and it is completely rational, objective, and impartial. From this point of view, it comes close to the public sphere concept proposed by Habermas(Habermas 2015). This concept represents an ideal of objectivity and rationality as well as understanding of the topic at hand. However, by embracing a rational and dispassionate approach to the task we reject subjectivity, passion and emotion. By discarding this information, the summarization becomes desirable for a larger public target but too generic and less appealing individually.

The current work aims to improve the automatic summaries created for a specific user profile.

According to Radev et.al., "Text Summarization is the process of identifying salient concepts in text narrative, conceptualizing the relationships that exist among them and generating concise representations of the input text that preserve the gist of its content."(Radev 2000)

Summarizer thus has the task of selecting the important parts of the source document and generate text that contains the information presented in them. This decision is not well defined since summaries pertain to various categories related to a specific topic or query and stemming from one or multiple documents.

Humans approach this task by analyzing the source text in two steps as illustrated by Cremmins et.al. (Cremmins 1992, Cremmins 1996). Local attention to sentence content, but also a global attention to key ideas emerging from the entire document.

Automatic summarization mimics the human approach to the process, and consequently the extractive and the abstractive summarizations types were created. The extractive approach selects sets of words of the original document and quantifies their importance within the source text. Meanwhile, the abstractive approach uses algorithms for generating text with new words while preserving the ideas promoted by the source text.

An important leap forward in natural language processing was made by the introduction of word embeddings (i.e.sets of numbers organized in a vector object with magnitude and direction). By employing vectors we can treat texts as collections of words which map into collections of vectors (i.e. vector spaces). Ergo, word embeddings facilitate a distributed representation of words in a vectorial space (Mikolov, Sutskever et al. 2013). Specifically, this method positions words in space according to their meaning thereby increasing the quality of text processing.

Distributional conceptual spaces are another notable approach to model semantics. This theory makes use of prototypes, and helps the models make higher level abstractive connections between categories of items. Therefore, words can be organized, grouped and understood by their semantic meaning.

The advent of neural networks further enhanced the quality of text summarizations. The Recurrent Neural Networks facilitate the discovery and memorization of complex relationships and temporal distributions of words in a sentence.

2.Goal

To create models for automatic personalized summarizations.

All summarization techniques whether done by humans or software discard information. In particular, automatic summarization systems look for patterns that are appealing to most users and extract words that have the highest statistical probability to represent the source document. This approach however, creates impersonal and generic summarizations learned during training from data that contains no personal information about the user. Consequently, the text generated is less appealing to individual users but universally acceptable. We aimed to improve this approach by introducing user generated content to the model in addition to the source data. Moreover, we attempted to improve the existing machine learning models by allowing complex patterns to be learned from the corpora, thereby leveraging abstraction.

To accomplish this we explored two avenues: the distributed attention mechanism and the concept spaces. Both approaches required word embeddings and machine learning techniques based on recurrent neural networks.

3.Contributions

We proposed two novel approaches to automatic personalized text summarization grounded in the abstraction mechanisms: the attention mechanism and the distributional conceptual spaces. While these methods were applied by others to text processing and even summarization, to our knowledge they have not been used to personalize the generation on text.

The use of the attention mechanism is innovative, and leads to decreased computation time. The summarization model utilizes the information resources available

from the user generated content while learning to summarize. While the model can be further improved, we show that it achieves performances comparable to the state of the art models.

Semantic representations have progressed in recent years, however they are still in their infancy. The goal of creating performant semantics models remains a new stepping stone for high quality summarizations. In this work, we show that conceptual spaces can be used as a platform to extract additional meaning and information from text, and thus providing unstructured data to complex neural networks. This approach can be further refined by introducing the notion of concept memory into the neural cell.

The pointer-generator hybrid architecture was used as a reference model in this study. The attention mechanism is similar to (Bahdanau, Cho et al. 2014) but has a different coverage mechanism. In our approach, the attention layer captures information from two sources and is not used as force-feeding the Decoder.

CHAPTER 2: BACKGROUND AND EXPERIMENTAL SETUP

The model proposed in this work aims to accomplish personalized summarization of text from news articles. The task required several models based on abstractive, extractive and point-generator summarization. In addition, the corpora used for training and evaluating our models was freely available. All resources used here are open source libraries and their functionality is described below.

1. Corpora:

The CNNDM (CNN_Daily-Mail)(Nallapati, Zhou et al. 2016) database provides article bodies along with a few 'highlights' : short paragraphs consisting of one to several sentences generated by humans in reaction to the content. Each article and its 'highlights' provide insight into how humans summarize and react to the ideas within the source text. This database became over the years a reference source for testing new models of summarization (See, Liu et al. 2017). However, this database provides less information about the user generated content.

The aim of the current work is to summarize articles based on user preferences. As such a second source of information was provided to the model in regards to a specific user profile. For this process we gathered information from the New York Times API (NYT). This second source comes with a wealth of information for any given user: names, geolocation, comments to specific articles, topics of interest, occupation, links to the original article as well as the leading paragraph from the article itself. The comments provided by NYT from real users were employed to personalize the summarizations.

The article bodies along with the highlights were utilized to create the reference summaries. Random and unique NYT comments with a length similar to a summarization

were randomly selected and paired with the CNNDM article bodies and highlights. The new article-summary-comment pairs were then subjected to text preprocessing as described below.

comments	[a win for globalist hillary clinton would have strengthened terrorists around the putin knew so of course he wanted trump to because russia has a serious terrorist problem and at the same time putin knows that a political civil war in america will help russia to become a more powerful so russia sowed seeds of discord to instigate and promote animosity between the democrats and]
highlights	[syrian obama climbed to the top of the know how to get, obama sends a letter to the heads of the house and senate, obama to seek congressional approval on military action against syria, aim is to determine whether cw were not by says spokesman]
story	[president barack obama wants lawmakers to weigh in on whether to use military force in, obama sent a letter to the heads of the house and senate on saturday hours after announcing that he believes military action against syrian targets is the right step to take over the alleged use of chemical, the proposed legislation from obama asks congress to approve the use of military force prevent and ...

Figure 1: Sample CNNDM story and highlights with NYT comments.

2. Embeddings:

Word vectors enable us to map words into a metric space. This represents a leap forward in our ability to analyze mathematically and computationally the relations between words and sentences. Moreover, this creates the opportunity to compute and measure the meaning of words.

There are different types of word vectors (word embeddings). (Levy and Goldberg 2014, Levy, Goldberg et al. 2015). For our purpose:

- frequency based and
- prediction based embeddings.

The frequency based embeddings are:

- count vectors,

- TF-IDF vectors,
- co-occurrence vectors

The prediction based embeddings:

- CBOW
- Skip-gram

One type of word vectors used in the current work in the final phase of decoding is the 1of N. This representation establishes a one to one relation between the vectors and the words. Also known as one-hot this mapping attributes a value of 1 to a certain element in the vector corresponding to a unique word while the other elements harbor a value of 0. This representation is commonly used for classification.

Later, Mikolov et. al. used a more refined representation of words in vectors known as Word2Vec (Mikolov, Chen et al. 2013). A distributed encoding allows all the elements of the word vector to contribute to the definition of the current. Ergo, a vector of 100 elements has all 100 relevant values for the definition of a certain word. Moreover, all 100 elements contribute to the definition of all the words. This distributed representation enables a profound approach of semantics. As such, Word2vec can make approximations of a word meaning given extensive training on large data. A famous example of this is:

KING - MEN + WOMAN = QUEEN

PARIS - FRANCE + ITALY = ROME

Concretely, Word2Vec is a shallow 2 layer neural network developed by Mikolov et. al. at Google. It takes an input text and outputs vectors with numerical values as

elements. In other words it takes an input of discrete states and creates a numerical representation where properties like co-occurence and discreetness are translated.

This is achieved via 2 models the CBOW and Skip-grams. The Skip-grams were 66.1% versus CBOW 57.3% superior as accuracy on semantic tasks and 65.1% versus 68.9% on syntactic tasks, with a total of 65.6% Skip-grams to 63.7% CBOW accuracy. CBOW predicts the current word based on its context, while Skip-grams predict the surrounding words given a base word. CBOW performs slightly better on more frequent words 64% versus 59% for Skip-gram syntactic accuracy.

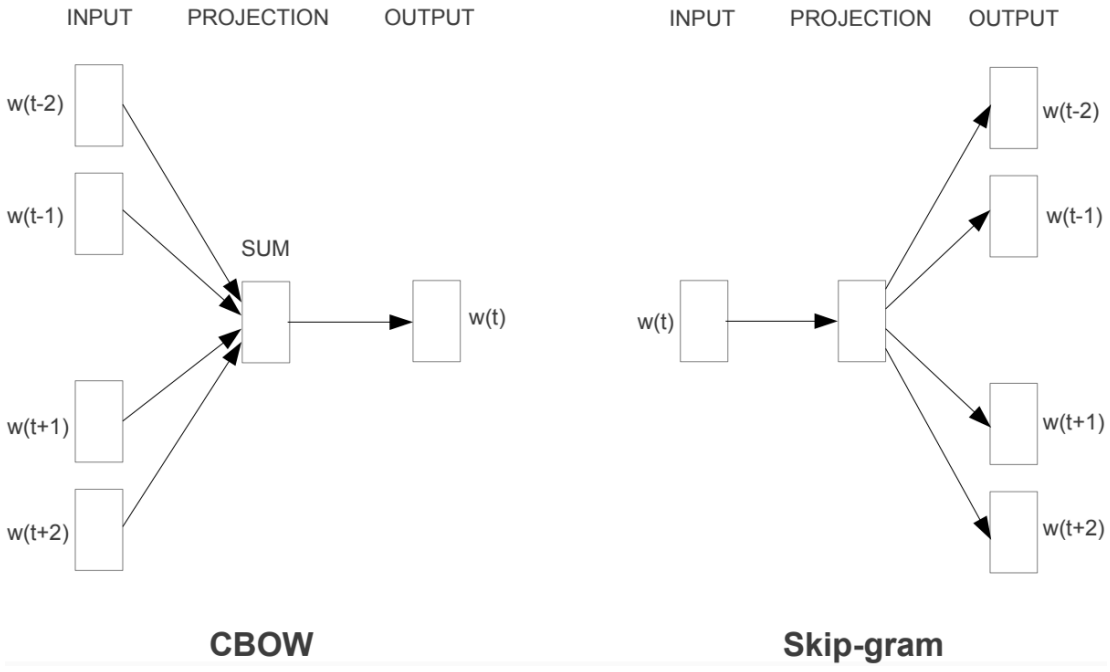


Figure 2: Distributed representations of word vectors. Models for learning Word2Vec distributed representations (Mikolov, Sutskever et al. 2013).

GloVe was created at Stanford by Pennington et.al. and it is an unsupervised learning algorithm for obtaining representation of words based on co-occurrences

(Pennington, Socher et al. 2014). It combines global matrix factorization and local context windows methods. The sole objective is to learn word representations such that the dot product equals the logarithm of word co-occurrence.

Term frequency matrices contain words on rows and documents on the columns (term-document frequencies) or words on columns (term-term frequencies). The global matrix factorization is employed to reduce a large term-frequency matrices via matrix factorization.

Secondly, the local context window makes use of a sliding window over the corpus with the purpose of learning to predict one word (CBOW) or the surrounding words (Skip-gram).

The result of the two combined algorithms is incorporated into a *least squares* regression with f as a weighing function (Pennington, Socher et al. 2014):

$$\hat{J} = \sum_{i,j} f(X_{ij})(w_i^T \tilde{w}_j - log X_{ij})^2,$$

where $\hat{J}$ is the least squares objective function, w_i and $\tilde{w}$ are context vectors.

Pre-trained vectors of GloVe 100 were used in the current work as described (Pennington, Socher et al. 2014)

3. Recurrent neural networks:

Artificial neural networks belong to a class of systems where connections and information processing emulates neurobiological systems. Their purpose is to learn by applying the Hebbian rules of neuroplasticity: as a neuron triggers another neuron the connection between them gets stronger (Hebb 2005). In the original approach, the relation between input and output is governed by propositional logic as demonstrated by (McCulloch and Pitts 1943). This approach evolved, and today in a simple feed-forward

ANN, the input data and the activation function changes the internal state of the artificial neurons and finally produces output data.

This concept is taken one step further by the RNN. The feed forward data flow from the input through the activation in the hidden layers towards the output is followed by a back propagation of the inputs from the output back to the recurrent neurons. Consequently, the neurons receive input from the current step **x**(t) as well as the output from the previous step **h**(t-1) (Géron 2017).

This relatively simple fact gives the network the ability to process data organized in time series: atmospheric events, stock prices, words in a sentence. Upon training, the network is able to make time dependent predictions. In natural language processing, the network is able to process sequences of variable length and predict sentiments, or create speech from text, or translate from a different language.

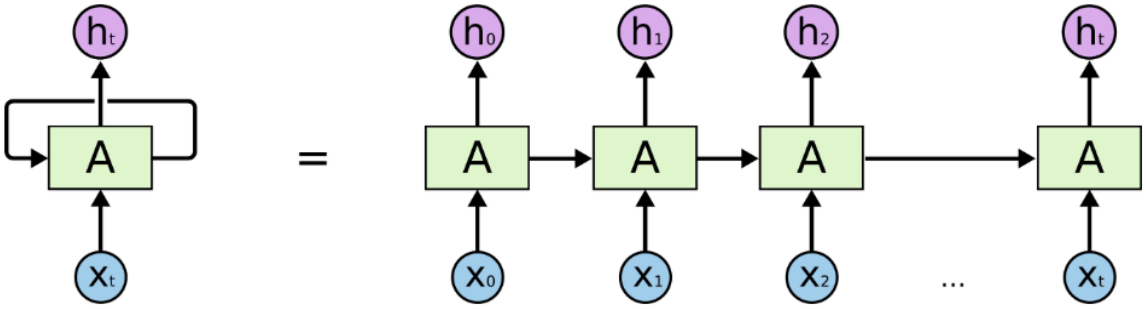


Figure 3: Unrolling RNN. A recurrent neuron (left) unrolled through time, where t is the current time step (Olah 2015).

An RNN unit has two vectorial weights, one for the input **x**(t) at time t and the second for the output **h**(t-1): $\mathbf{w_x}$ and $\mathbf{w_h}$ respectively. For the entire recurrent layer this translates into the matrices $\mathbf{W_x}$ and $\mathbf{W_h}$. Ergo, if Φ is the activation function, and **b** is the bias the output of one instance is:

$$h_{(t)} = \phi(W_x^T \cdot x_{(t)} + W_y^T \cdot h_{(t-1)} + b)$$

4. LSTM

Cells preserve some state over time, therefore RNN is the simplest (memory) cell. In the case of RNN the hidden state is a function of the input at time t and the output at time t-1. However, the hidden state, the state and the output are the same in this case. This means the states are changed with every input.

This model is improved in the Long Short-Term Memory (LSTM) cells created in 1997 (Hochreiter and Schmidhuber 1997). LSTMs harbor two different state vectors: for the cell state **c** and the hidden state **h** (also named output). Each state is dedicated to a different kind of memory: **h**(t) is responsible for the short term memory and **c**(t) is managing the long-term state. Thus, LSTMs are empowered to use long term dependencies to extract complex patterns in time series.

This process starts by recognizing key inputs and storing them in the long-term cell state. The data is retained as needed, managed via forget gates, and consecutively extracted at the opportune moment. To do all this, an LSTM encompasses the following architecture: The gate controllers: forget gate **f**, input gate **i**, output gate **o**, are layers in LSTM that regulate the main layer **g**. Under their control, the cell state **c** and hidden state **h** accomplish the memory function and are calculate as follows:

$$c_{(t)} = f_{(t)} \otimes c_{(t-1)} + i_{(t)} \otimes g_{(t)} \ ,$$

$$y_{(t)} = h_{(t)} = o_{(t)} \otimes tanh(c_{(t)})$$

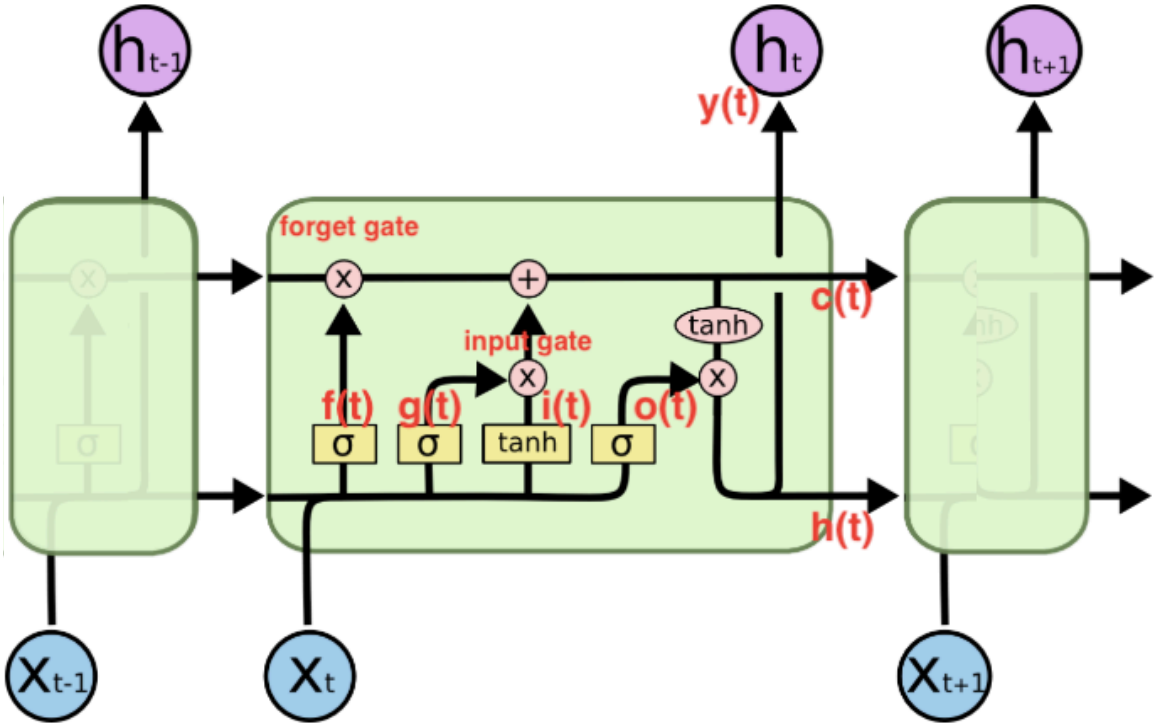


Figure 4: LSTM cell: tanh=tanh activation; σ = logistic activation; $\oplus$ addition element wise; $\otimes$ element wise multiplication (Olah 2015)

5. TensorFlow

It is an end-to-end open-source framework developed by Google Inc. for machine learning, used extensively in this work. It was conceived with versatility in mind and therefore it can be used without changing the code on a plethora of mobile systems, workstations, multicore CPUs and large distributed systems with multiple GPUs. Its purpose is to create deep or shallow nets that can be easily trained and used for inference.

The underlaying architecture is based on directed acyclic graphs (DAGs). Here, layers are the graph nodes and are composed of mathematical operators (Abadi, Barham et al. 2016). In fully connected layers the input is multiplied by the weight matrix, a bias value is added and a non-linear activation function is applied to the result. This is a typical instantiation of an operation in TensorFlow. A scalar loss function is applied to the

output to measure the difference between the predicted values and the ground truth values. The data populates the graph via tensors.

To perform the actions described above, TensorFlow runs computations on tensors. They are a generalization of matrices and vectors to higher dimensions. TensorFlow builds the DAG using the tensor objects as units (Abadi, Agarwal et al. 2016). The researcher builds the graph, performs the transitive closure of nodes and computes the outputs using an interface. After this, the graph can be executed on the data.

6. Keras models and custom layers

Keras functions as an API wrapper for TensorFlow. It is a high-level neural networks API that facilitates prototyping, modularity and extensibility. The framework made the transition seamlessly from CPU to GPU for the experiments ran as part of this work. Keras can be used at any level of an experiment from text preprocessing libraries, to tokenizers, to organizing data batches, custom layers, and optimizer functions (like tanh or the adam optimizer) (Kingma and Ba 2014, Keras 2019).

This API was used in the current experiments to organize the layers of LSTM cells in sequential or non-sequential fashion. Keras also facilitated the implementation of custom layers like the Attention layer constructed for the current experiments. The activation functions (example ReLU) (Ramachandran, Zoph et al. 2017) are attached to a certain layer as function arguments. Similarly, the size of the network can be adjusted in terms of depth and number of cells.

7. Neural models of summarization - state of the art approaches

Summarization using neural networks is far superior to other methods of abstraction in text. The results are concise while preserving the meaning. But the state of the art summarization model used in this work for comparison indicates that summarization methods abstractive and extractive can be used together to improve the results.

The extractive methods have a longer history. Briefly, Hans Peter Luhn, a pioneer in the field of automatic text summarization (Torres-Moreno 2014) created the first extractive summarization. His algorithm used the word frequency distribution to weigh sentences in articles (Luhn 1958). Later, Pollock et. al. (Pollock and Zamora 1975) used sentence compression by deleting words and expressions from the original source. In 1995, Edmundson's method used a Bayes classifier to determine which sentences from the source are likely to be included in the summary. Other, notable approaches to summarization include the Hidden Markov Model (Conroy and O'Leary 2001) and the artificial neural networks (Svore, Vanderwende et al. 2007).

The pointer generator networks model with coverage (See, Liu et al. 2017) can extract words from the source text via pointing and at the same time creates words not encountered in the source text via the generator (Figure 5). The model is based on an Encoder-Decoder with an attention mechanism similar to Bahdanau et. al. (Bahdanau, Cho et al. 2014).

The model created by See et.al. (See, Liu et al. 2017) is aiming to create summarizations from long sequences of text. This is challenging due to the repetitive nature of learning with attention mechanisms. Moreover, longer source texts involve

complex abstractions and the results are not always desirable. However, by employing the coverage mechanism this net avoids most of these shortcomings.We utilized this model as a reference in the current work.

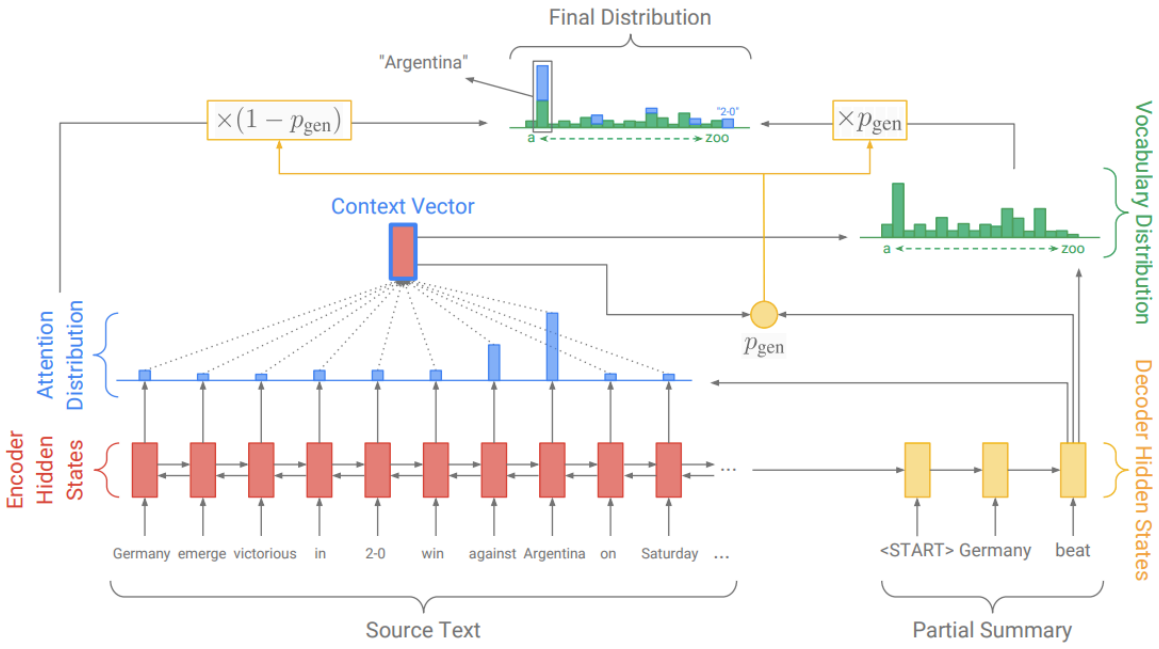


Figure 5: Pointer generator model. PG compares the probabilities for generating versus extracting the next word from the source (See, Liu et al. 2017).

8. Attention

The attention mechanism utilized in our model is derived from the NMT model with attention developed by Badhanau et. al. (Bahdanau, Cho et al. 2014). This mechanism can be defined as a probability distribution over the source words, that focuses the decoder on a specific context from the source that may most likely generate the correct word in the target (i.e. the summary). This process is intentionally non-monotonic and non-sequential. Meaning that the order of the chosen words for the next relevant context and the next generated word does not necessarily depend on the temporal

distribution of the words in the source. Therefore, the abstraction process can be more effective having a more general view over the source text.

This approach is a new way to align sequences by soft-searching for context parts. Before the advent of attention the Encoder-Decoder architectures were based on a fixed length vector which is a vectorial representation of the source sequence. This impedes the process of learning for some tasks. While the LSTM is using long term dependencies to retain information in a sequential fashion, it is not performing as well in tasks where the order of words in a phrase is changed like it is the case in language translations.

This leads to the need to search for relevant information in a focused area containing the relevant information. The task was accomplished by encoding the source sequence in several context vectors harboring just pieces of information instead of one single vector. Consequently, the model can choose from several vectors for each target word, therefore improving the performance on long sentences. In short, the soft alignment can be written as follows:

$$p(y_i \mid y_1, \ldots, y_{i-1}, x) = g(y_{i-1}, h_i, c_i), \text{ where } h_i = f(h_{i-1}, y_{i-1}, c_i).$$

In the above example, p() is the conditional probability for predicting word y_i given the previous words and the current source text x; g() is a non-linear function based on GRU which takes as arguments the previous word the current hidden state h_i and the relevant context for the current word c_i. As we can see, the hidden state h_i depends both on the previous hidden state as well as the previous word and representative context.

The two models of attention proposed in the current work for personalized summarization stem from the approach of Bahdanau et. al. The soft-alignment of sequences permits searching for relevant contexts $c_i^{(c)}$ in the users comments while

learning a summarization for the current word w_i. This approach improves the flow of information from the user content via the hidden state of the Encoder into the linear layer of attention which in turn provides the context vector to the Decoder at step i.

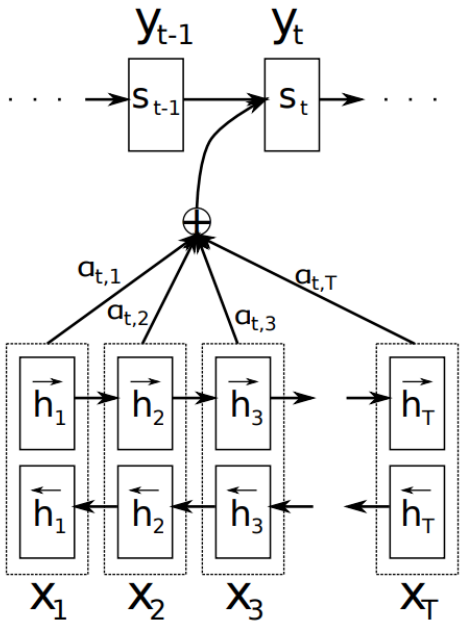


Figure 6: Attention mechanism Bahdanau et. al. (Bahdanau, Cho et al. 2014) At step *t* the model creates the target word y_t given a source sentence$(x_1, x_2, \ldots, x_T)$

However, the use of the context vectors can be multiple. We tested two approaches. The first approach involves creating the context vectors from the comments sequence for each word w_i generated at time *i*. (Figure 32)

$$p(y_i \mid y_1, \ldots, y_{i-1}, x) = g(y_{i-1}, h_i^{(a)}, c_i^{(c)}) \text{ , where } h_i^{(a)} = f(h_{i-1}^{(a)}, y_{i-1}, c_i^{(c)}).$$

The second approach uses a linear attention layer that selects relevant contexts from the source article while the model generates words simultaneously for comments $w_i^{(c)}$ and for summarization $w_i^{(s)}$ at time i (Figure 10).

$$p(y_i \mid y_1, \ldots, y_{i-1}, x) = g(y_{i-1}, h_i^{(a)}, c_i^{(s)}, c_i^{(c)}) \text{ , where } h_i^{(a)} = f(h_{i-1}^{(a)}, y_{i-1}, c_i^{(s)}, c_i^{(c)}).$$

The sequential fixed length static vectors retrieve information relevant to the order of words however, *ipso facto* they limit the non-monotonic extraction of information. With this limitation in mind, both attention methods described in this work extricate the non-monotonic information and relay it to the Decoder.

9. Models (experimental setup)

The models built to complete the summarization tasks are based on the Encoder-Decoder architecture. The source document is fed to the Encoder (E) which converts it into an internal vectorial representation with a fixed length. The Decoder (D) processes the emerging vector into a summarization. This architecture enables the recurrent neural networks to predict sequences with variable numbers of inputs and outputs. As such, given an input sequence of words $(w_1, w_2, \ldots, w_n)$ we can express the expected results as:

$$(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n) = D(E(w_1, w_2, \ldots, w_n))),$$

where D and E are non-linear functions in this case LSTMs.

For all the abstractive models we used 20K pretrained GloVe word embeddings of length 100. The text was fitted with Keras tokenizers and the resulting (word, value) tuples were employed to create a matching embedding matrix as (value, $\overrightarrow{glove}$). The tokenizer objects where saved as pickle objects and later used during the evaluation. Consecutively, the word vectors were fed to the Encoders.

The Keras framework was used to stack the layers of all RNN described in the current work.

The hidden LSTM layer in each Encoder comprises of 128 units which channels the information flow to a Decoder hidden layer of the same size. The information enters changes the weights of the cell states. It further exits into a linear function (i.e linear

dense layer) which conveys it to a single output unit equipped with a Softmax function. Consequently, the Decoder outputs one word at a time t for the Recursive models. As an exception, in the case of the Sequential S-Model, the data is time distributed and thus the output is a sequence of words. The results are in the form of a one hot vector.

The training was relayed by a Batcher module which limits the amount of data stored in the memory at any one time with a size set to 10 sample. Moreover, an early-stopping mechanism supervised the loss function with a patience set to 20 epochs. The learning rate was set to 0.005 for the Sequential model and 0.001 for the Recursive models with or without Attention.

The loss function computed for all the models can be described as the negative logarithmic value of the predicted versus actual word value for each token in the sequence. The optimization was achieved using Adam the adaptive learning rate mechanism.

$$loss = \frac{1}{N}\sum_{n=0}^{N} - log(P(\hat{y}_t)),$$

where N is the number of tokens (time steps) in the sequence and $\hat{y}_t$ is the predicted word at time t.

9.a. Sequential model (S)

The S-model is a simple Seq2seq model built on the Encode-Decoder architecture. Its inputs and outputs are fixed length vectors of size matching the largest number of words in an article and, respectively, in a ground truth summarization. This sequence the sequences encode the words in their original position in the sentence.

The model depicted in Figure 7 is constructed via Keras framework as described in Chapter 2 and was stacked in a sequential fashion.

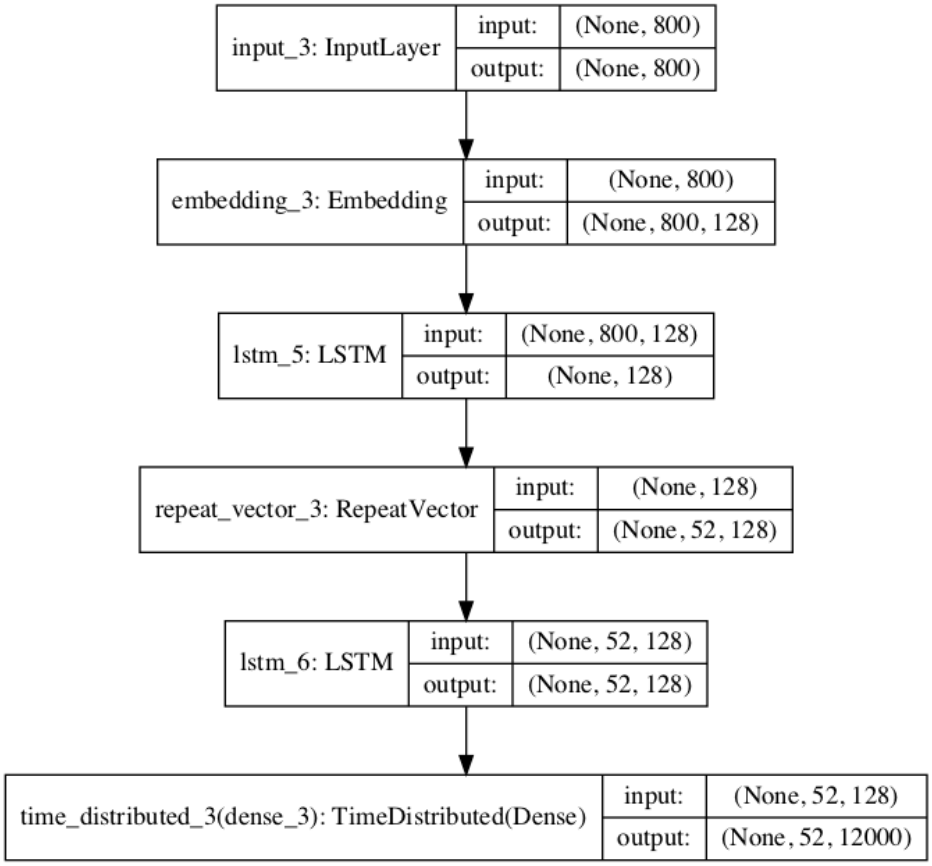


Figure 7: Sequential-Model Schema. Seq2seq model with Encoder-Decoder architecture based on a recurrent LSTM network. The output sequence is generated in one single pass

Here the encoder LSTM generates the entire single context vector of the input sequence and directs it to the Decoder. The output sequence is generated by the Decoder in a single pass with multiple outputs coming from the liner Dense output layer which is time distributed.

The time distributed wrapper applies one layer to each temporal slice of the input as described (Keras 2019). Consequently, the entire output sequence is generated in a single pass.

9.b. Recursive model 1 (R1)

The R1-model, depicted in Appendix B, was created as described (Ludwig 2017). This is a recursive model fundamentally different from the S-model. The model uses 2 Encoders working jointly to embed and relay the input sequences to the Decoder (Figure). The word generated at time t_i is appended to the input of the current summary. In this fashion, the summary built until t_i is used to predict the word w_{i+1} at time-step t_{i+1}.

The prediction of a word can be described as:

$$\hat{y}_i = D(E^a(w_1^{(a)}, w_2^{(a)}, \ldots, w_n^{(a)}), E^s(w_1, w_2, \ldots, w_{i-1})),$$

where E^s and E^a are the encoders for the article and the summary respectively and D is the decoder. They are non linear LSTM functions. This approach puts a heavy burden on the Decoder handling both embedded sequences. However, the R1-model is superior to the sequential model described above since Decoder is aware of the order of words already present in the current summary at time t_i.

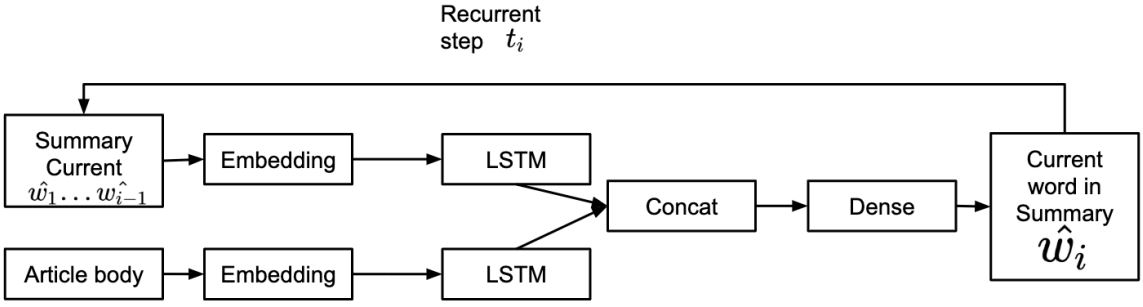


Figure 8: Recursive model R1. The Encoder (Embedding + LSTM) takes as inputs the article bodies and the summary generated until time t. The Decoder (Concat+Dense) generates the words one at the time.

9.c. Recursive model 2 (R2)

The R2-model is a recursive network built as described (Brownlee 2019). This network is similar to the R1 model (Appendix C). It outputs a predicted word-vector at time t_i based on the preceding set of words $\{w_1, w_2, \ldots, w_{i-1}\}$. However, there is a crucial difference in the way it handles the summary input as a set (i.e the order of words is disregarded). This allows for a different kind of output where the information is flowing from the article in a sequential manned and non-sequentially from the summary (Figure). Ergo, a higher level of abstraction can be attained on the summary side while creating the one single context vector, at the expense of losing the temporal order of words in this sequence.

The formula encapsulates this reality:

$\hat{y}_i = D(E^a(w_1^{(a)}, w_2^{(a)}, \ldots, w_n^{(a)}), f^s(\{w_1, w_2, \ldots, w_{i-1}\})$, where $f^{(s)}$ is a linear dense layer function.

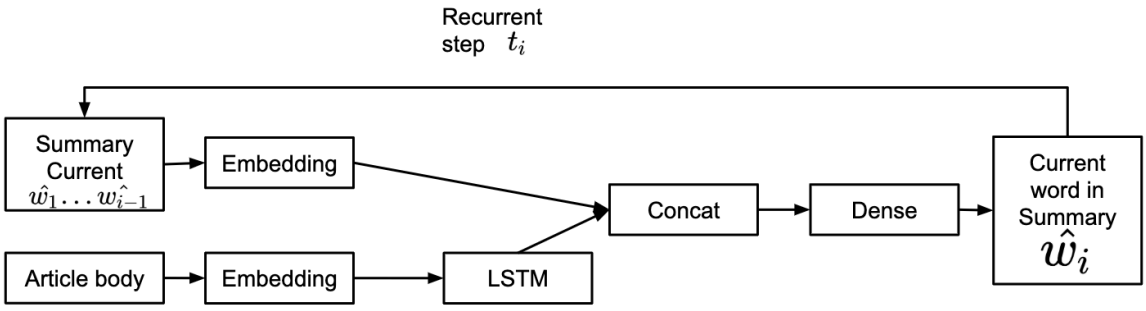


Figure 9: Recursive model R2. Similar to the model R1 with the exception that the Encoder (Embedding + LSTM) takes as inputs the embedded sequence temporal dependent from the article bodies (left) and the embedded summary generated until time t-1. The Decoder (Concat+Dense) generates one word at the time t.

9.d. Extractive model (E)

The sentence level extractive model is fundamentally different from the other models used so far in the current work. Concisely, it extracts the information from the original text without changing the words nor their sequence. To achieve this, all the experimental setup was changed. First, the articles' corpora was preprocessed while preserving the sentence structure. The stop words were eliminated along with the punctuation and numerical features, and the words were lowercased where needed. This step is crucial for selecting the relevant sentences from the text in preparation for the tf-idf sentence ranking.

The frequency of unique words (i.e. document frequency) for a given article was calculated as the ratio between the number of times a given word appears in a document and the number of words in the article body: $df_i = \frac{n_i}{N^{(d)}}$. In terms of term frequency (tf) this value can be expressed as: $tf_i = \frac{N}{df_i}$, where N =1 represents the number of documents analyzed for a given word, and df_i is the document frequency of the word w_i (Wu, Luk et al. 2008).

Based on the above, the inverse document frequency is:

$$idf_i = \begin{cases} 1 + log_e(tf_i) & if\ tf_i > 0, \\ 0 & otherwise \end{cases}$$

The ranking of a sentence is the sum of the idf_i of a given unique word w_i divided by the total number of words in the sentence. This measure prohibits sentences with repeating words to have a higher score.

$$ranking^{(s)} = (\frac{1}{N}) \sum_{i=1}^{N} w_i.$$

The ranking will return a value for each sentence which can be used to create a scoring matrix . The values will allow us to chose the sentences with the highest

cumulative tf-idf scores. In this experiment the threshold was set to top 5 sentences (Figure 21).

The algorithm creates summarizations that contain precise extracts of the original article and thus can be used to see how much of the information from the original article is reflected in the ground truth summarizations.

9.e. Recursive model 2 with attention (R2+A)

This model is derived from the R2-model described above (2.9c). In addition to the R2 Encoder-Decoder, we added a second source of information coming from the user generated content (i.e the comments) via a simple Embedding layer. As in the R2-model, the predicted word vector $\hat{w}_i$ at time t_i is fed back into the input sequence of the current summary $\{w_1, w_2, \ldots, w_{i-1}\}$. The Decoder is therefore aware of the words (in sequence) added to the summary at previous time steps. This information is used to shape the hidden state of the Decoder h_i and further output of the model.

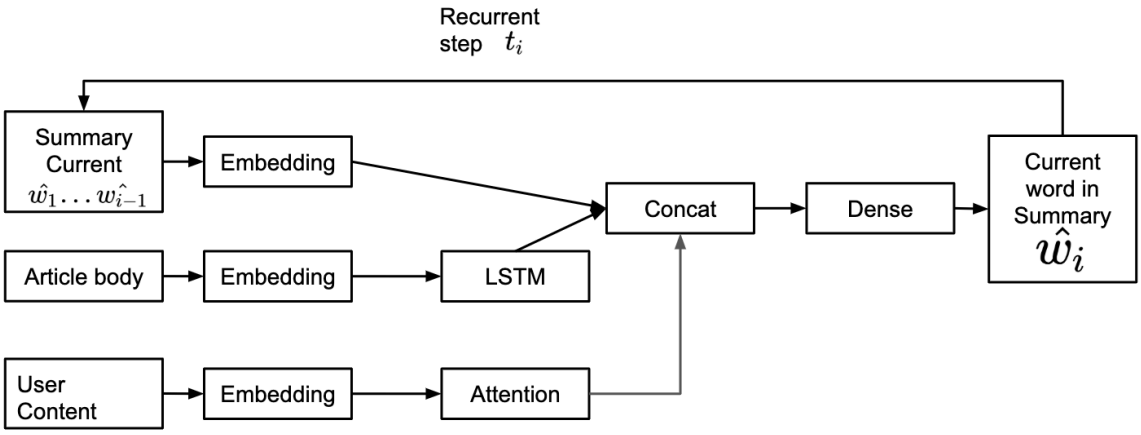


Figure 10: Recursive R2-model with attention. The information from the user is embedded and organized into a set of context vectors and consecutively sent to the Decoder (Concat+Dense) where it alters its hidden state.

The attention model is graphed hierarchically onto the R2 model. It solely handles the task of managing the user information by creating a set of user context vectors. This information is concatenated with the summary hidden state and relayed onto the Decoder LSTMs. Consequently, the trainable weights of the Decoder are shaped both by the user content and the summary being created one word at a time.

9.f. Recursive model with shared attention (R+SA)

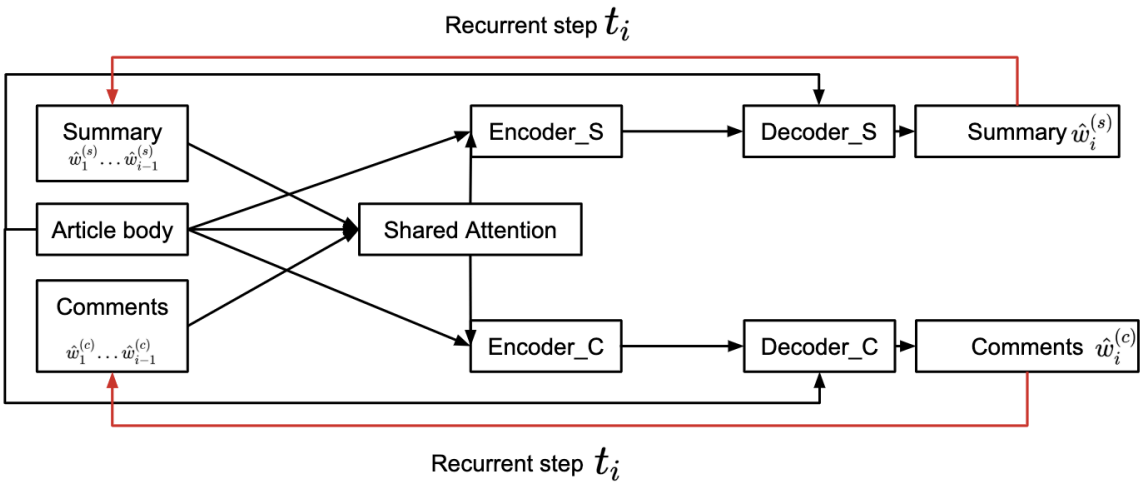


Figure 11: Recursive model with shared attention (R+SA). The Encoder processed the information channeled from 3 input sources: the article, the summary generated at time t and the commentary generated at time t. The two Decoders (S+C) process the information and generate the two words (one for comments and one for the summary) both output at time *t*.

The R+SA model, depicted in Appendix D, uses an Attention mechanism that processes and conveys sets of context vectors from multiple input sources. In short, the recurrent hierarchical network harbors two Encoders and two decoders with trainable weights. The Encoders shared the information conveyed by the Shared Attention (2.8. details on shared Attention) as well as the information coming from the article body at

any given step t_i. The Decoders LSTMs are simple and process the information for the given output: comments and, respectively, the summary.

This architecture learns to produce comments and summaries at the same time and one word at a time.

The combined results of the networks at each one of the two Encoder level can be formulated as:

$$p(y_i | y_1, \ldots, y_{i-1}, x) = g(y_{i-1}, h_i^{(a)}, c_i^{(s)}, c_i^{(c)}),$$

as described in (2.8) where g is a non-linear LSTM function. The context vectors $c_i^{(s)}$ and $c_i^{(c)}$ are added into one vector value by the linear dense function f into a common set of context vectors.

$$c_i^{(shared)} = \sigma(tanh(f(c_i^{(a)}, c_i^{(c)}))).$$

The shared set of context vectors encode relevant segments of information from the article body for any given word in the comments and any word in the summary at the same time. We hypothesize that this shared information is relevant for the creation of personalized summaries as it brings relevant information from the summary and the user content (i.e the comments).

9.g. The pointer-generator reference model with coverage (PG):

This model is considered the state of the art *hoc tempore* and it was utilized in the current work as described (See, Liu et al. 2017).

Briefly, the pointer-generator is a seq2seq recurrent model equipped with an Encoder-Decoder mechanism. The hidden state has 256 dimensions and harbors 128 units for handling the word embeddings. Its pointer-generator (PG) handles a vocabulary set of 50K words which is symmetrically used for the source text and the target summary. This

is roughly half the size used by others (Nallapati, Zhou et al. 2016). This reduction is facilitated by the PG's ability to handle out of vocabulary words (OOVs).

The pre-trained model and weights were downloaded from the source (See 2017 May 1) and ran on the CNN-DM dataset. The authors incorporated an Adagrad mini batch stochastic-gradient mechanism for training the network. The output values are fixed length sequences of vectors.

The attention distribution mechanism follows the same model described by (Bahdanau, Cho et al. 2014) and has the role of producing the relevant context vectors from the encoder hidden states:

$$h_t^* = \sum_i a_i^t h_i ,$$

where a_i^t is the attention vector and h_i is the hidden state.

The pointer generator extractor is built on the following function:

$$p_{gen} = \sigma(w_{h^*}^T * h_t^* + w_s^T * s_t + w_x^T * x_t + b_{ptr}),$$

where the h_t^* is the context vector, s_t is the decoder state and decoder input is x_t and σ is a sigmoid function. All the other weights parameters are subjected to the learning mechanism.

A coverage mechanism is added to the attention vector to eliminate the repetition effect observed during long sequence procession in some nets. The c_t is the coverage vector:

$$c_t = \sum_{t'}^{t-1} a^{t'}.$$

The pointer_generator hybrid network with coverage reduces inaccuracies and repetitions and significantly outperformed previous models by ROUGE estimations. We

used this model to produce test summaries and evaluate the results using ROUGE, BLEU and Jaccard measurements described in Chapter 3.

10. Metrics

The summaries generated by the reference model as well as the extractive and abstractive neural models were evaluated quantitatively using several techniques such as: ROUGE, BLEU and Jaccard.

10.a.ROUGE

Usually, the text generated by the model is compared against a control summary. In our case, the ground truth was defined by human annotated text found in the CNNDM data. ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation (Lin 2004)

$$ROUGE - N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count(gram_n)}$$

and counts the number of overlapping n-grams, sequences or pair of words.

Lin et.al. created four types of ROUGE: ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-S.

- ROUGE-N: In the above formula, *n* is the length of the n-gram, and $Count_{match}(gram_n)$ stands for the maximum number of n-grams co-occurring in the automatic summary and in the reference summary.
- ROUGE-2: (used in Table 1) stands for the ratio between the actual number of co-occurring bigrams from the total number of co-occurring bigrams possible between the automatic summary and the reference summary.

- ROUGE-L: counts only in sequence co-occurrences, and gives results in the interval [0,1] where 0 represents no overlap between a sequence X and a sequence Y and 1 when X=Y. The values in between stem from the longest common subsequence (LCS). ROUGE-L uses LCS based F-measures to compare X and Y.
- ROUGE-W: measures the weighted longest common subsequence of two given sequences X and Y. It gives a better distinction between results with consecutive words and non-consecutive overlapping words by favoring the consecutive sequences.
- ROUGE-S: measures skip-gram co-occurrence. Skip-gram represents any pair of words having the same order as in the original sentence but harboring arbitrary gaps.

10.b.BLEU

BLEU (bilingual evaluation understudy) is an automatic precision-measure typically used in machine translation algorithms like the Open-NMT. It counts the percentage of n-grams in the candidate text that overlap with the source text (Papineni, Roukos et al. 2002), therefore the range is [0,1] similar to ROUGE. The algorithm first computes the n-gram modified precision on blocks of text:

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

BLEU penalizes the candidate summary by introducing the penalty factor for text that is either too long or too short compared to the reference. This measure ensures that

the candidates match the translation length besides the word choice and word order introduced.

$$BP = \begin{cases} 1 & \text{if} c > r, \\ e^{(1-r/c)} & \text{if} c \leq r \end{cases}$$

Therefore:

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

10.c.Jaccard

The Jaccard index computes similarity between two sets (Jaccard 1901). In the current work, the sets are represented by words tokenized from the reference, source and candidate (i.e generated) summaries. In this case the order of words does not matter, which explains the use of sets. Given two tokenized and pre-processed sets of words A and B the J(A,B) (Kosub 2019) is:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{A + B - |A \cap B|}$$

This index allows us to quantify the result of a non-monotonically generated text from the abstractive methods.

CHAPTER 3: RESULTS

Results overview:

Table 3.1: Performance of models. All models were trained on the same data. RNN R1 is the recursive model 1; RNN-R2 is the recursive model 2; R2+Attention is the model R2 with attention; R1+SA is the R1 model with shared attention. ROUGE-2 stands for the ratio between the actual number of co-occurring bigrams from the total number of co-occurring bigrams possible between the automatic summary and the reference summary.

Models	**Rouge-2 F1-values**	**Rouge-2 Precision**	**Rouge-2 Recall**	**BLUE cumm4**	**Jaccard**
Sequential (S)	0.085	0.211	0.24	0.68	0.183
Recursive RNN (R1)	0.126	0.117	0.137	0.61	0.171
Recursive RNN (R2)	0.104	0.104	0.106	0.62	0.214
R2+Attention	0.130	0.137	0.125	0.57	0.237
R1+SA	0.141	0.203	0.109	0.462	0.077
Extractive (E)	0.009	0.3	0.0058	0.68	0.08
Pointer generator (PG)	0.145	0.172	0.123	0.54	0.275

11. Sequential model (S):

The encoder captures the information of the input sequence and relays it to the Decoder. As such, the encoder has the important task of determining how to process the information and what is important in the input sequence. However, in the S-model, by encoding the source vector into a single fixed length vector of a given size and the output of the model is entirely dependent on the order of tokens in it without having the ability to create a higher level abstraction nor to introduce new tokens in the created summarization. While we can observe that the beginning of the sequence resembles closely the summary, it is evident that this pattern is not sustained throughout the sequence (Figure 12).

```
Summary 0:
 syrian obama climbed to the top of of know how to to to to to to to to to the the and and to to to seek seek approva
l on military action action syria aim is is determine whether were not not
Reference 0:
 syrian obama climbed to the top of the know how to get obama sends a letter to the heads of the house and senate oba
ma to seek congressional approval on military action against syria aim is to determine whether cw were not by says sp
okesman
Summary 1:
 usain bolt wins third gold of of championship anchors anchors to xm xm victory eighth eighth at at championships for
bolt bolt double up up relay
Reference 1:
 usain bolt wins third gold of world championship anchors jamaica to xm relay victory eighth gold at the championship
s for bolt jamaica double up in xm relay
Summary 2:
 the employee in kansas city office of among hundreds hundreds workers the the the the and the the the the cost more
more than telecommuting like is is is under review
Reference 2:
 the employee in kansas city office is among hundreds of workers the travel to and from the mainland last year cost m
ore than the telecommuting like all gsa is under review
Summary 3:
 the employee in kansas city office of among hundreds hundreds workers the the the the and and the the the cost more
more than telecommuting like is is is is under
Reference 3:
 a canadian doctor says she was part of a team examining harry burkhart in severe stress disorder and burkhart is als
o suspected in a german arson officials say prosecutors believe the german national set a string of fires in los ange
les
```

Figure 12: Sequential model predicted sequences. Predicted from the sequential model (Summary 0-3) and the corresponding actual summary (Reference 0-3)

This type of networks while simple and easy to use do not scale up very well to large applications with long sequences.

The pattern distribution between the predicted summary and actual summary is more evident in Figure 14, where the similarity between two random words is computed based on their W2V word-vector values and the cosine similarity function. By examining the Rouge values we can see the decreasing trends from ROUGE-1 to ROUGE-4. with increased dimensions. Not surprisingly the trend in the BLEU score cumulative values is increasing.

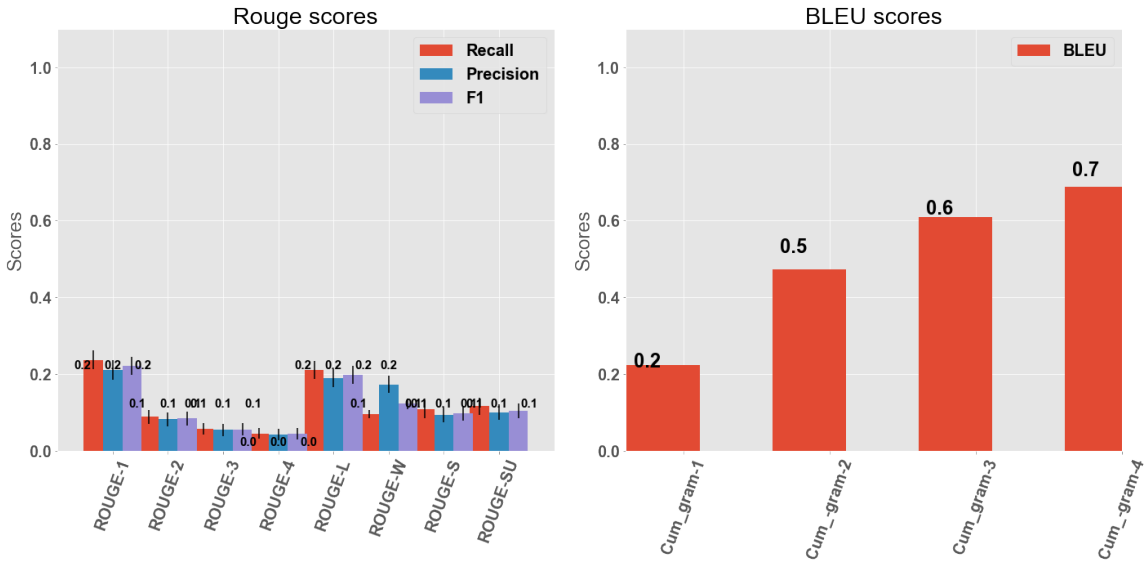


Figure 13:Rouge and BLEU scores for the Sequential model. The F1 score for this model is 0.085. This simple model has a overlap of bigrams between the actual summary and the predicted summary that offers a baseline to compare the other models against.

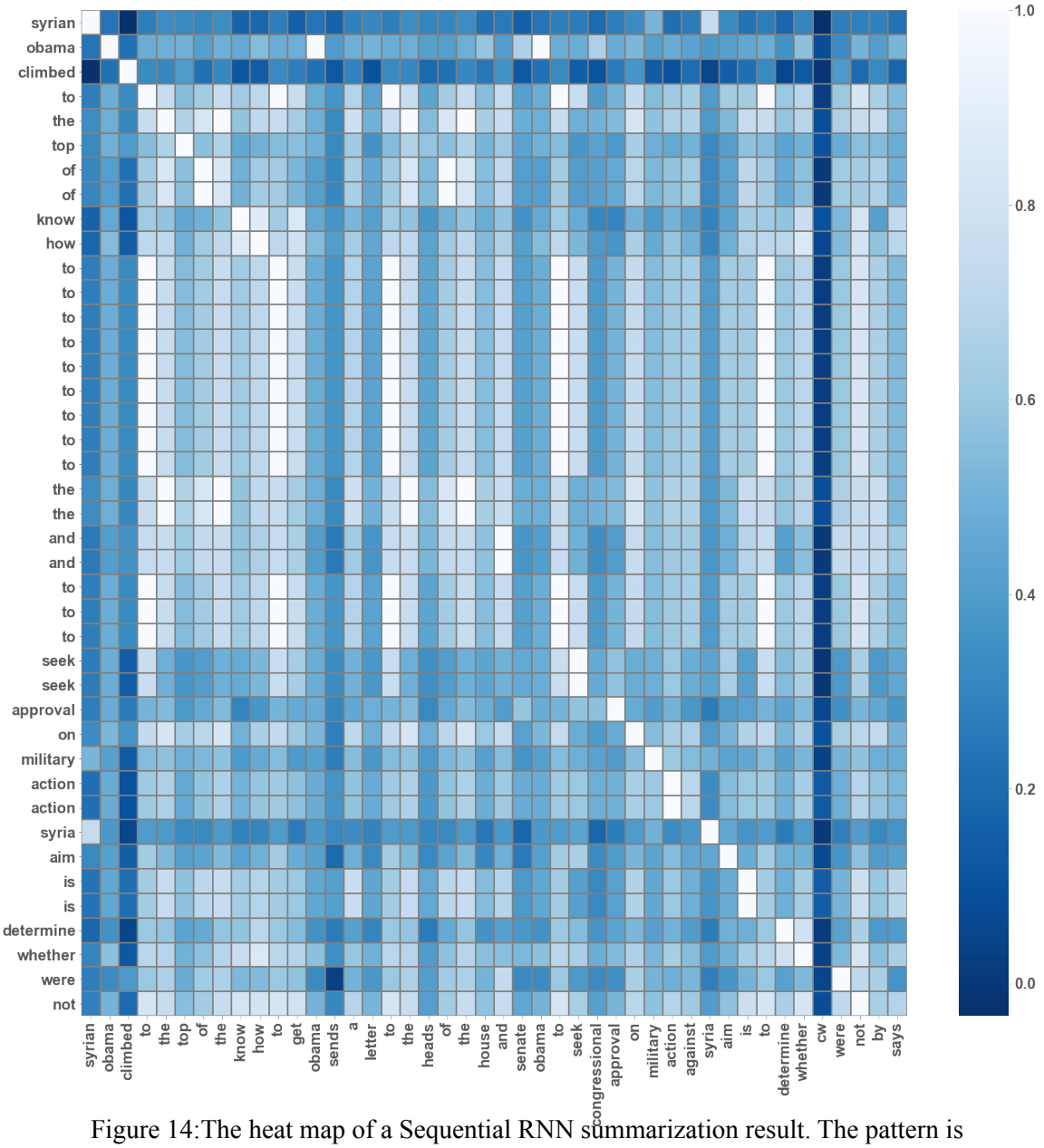


Figure 14:The heat map of a Sequential RNN summarization result. The pattern is calculated by cosine distance between 2 given word vectors. Here, the actual summary is on the X axis and the predicted summary is on the Y axis. Values are between [0,1].

12. Recursive model 1 (R1)

```
Summary 1:
 usain bolt wins third gold of world championship anchors jamaica to xm relay victory eighth gold of world championsh
ip anchors jamaica to xm relay victory eighth gold of world championship anchors jamaica to xm relay victory eighth g
old of world championship anchors jamaica to xm relay victory eighth gold of world championship anchors jamaica to xm
relay victory eighth gold of world championship anchors jamaica to xm
Reference 1:
 usain bolt wins third gold of world championship anchors jamaica to xm relay victory eighth gold at the championship
s for bolt jamaica double up in xm relay

Summary 2:
 the employee in kansas city office is among hundreds of workers the travel to and from from of workers the travel to
and from from of workers the travel to and from from of workers the travel to and from from of workers the travel to
and from from of workers the travel to and from from of workers the travel to and from from of workers
Reference 2:
 the employee in kansas city office is among hundreds of workers the travel to and from the mainland last year cost m
ore than the telecommuting like all gsa is under review
```

Figure 15_: Recursive model R1, summarizations generated.

Model R1 uses a recursive method of training, generating one word at a time (Ludwig 2017). This method allows for a longer sequence to be trained. We observe this aspect in model R2 as well. Moreover, in faster time we achieved better results with 0.126 F1 Score versus 0.145 achieved in the PG control model. Figure 15 indicates a repetitive pattern that does not disappear, typical to RNN trained on longer sequences.

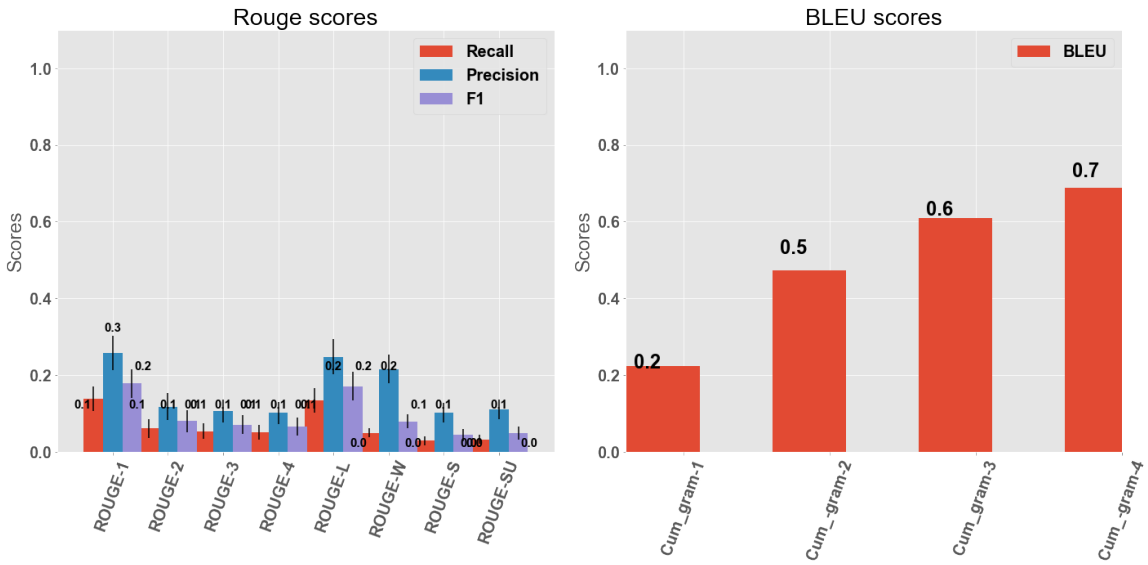


Figure 16: ROUGE and Bleu scores R1 model. The F1score is 0.126 for ROUGE-2 and is comparable to the PG control model at 0.145.

This was the reason from adding coverage to the PG control model which effectively eliminated the patterns. Our scope however was to improve a basic model without added mechanisms.

Similar trends of decrease in trend from for ROUGE-N can be observed with increased N (Figure 16).

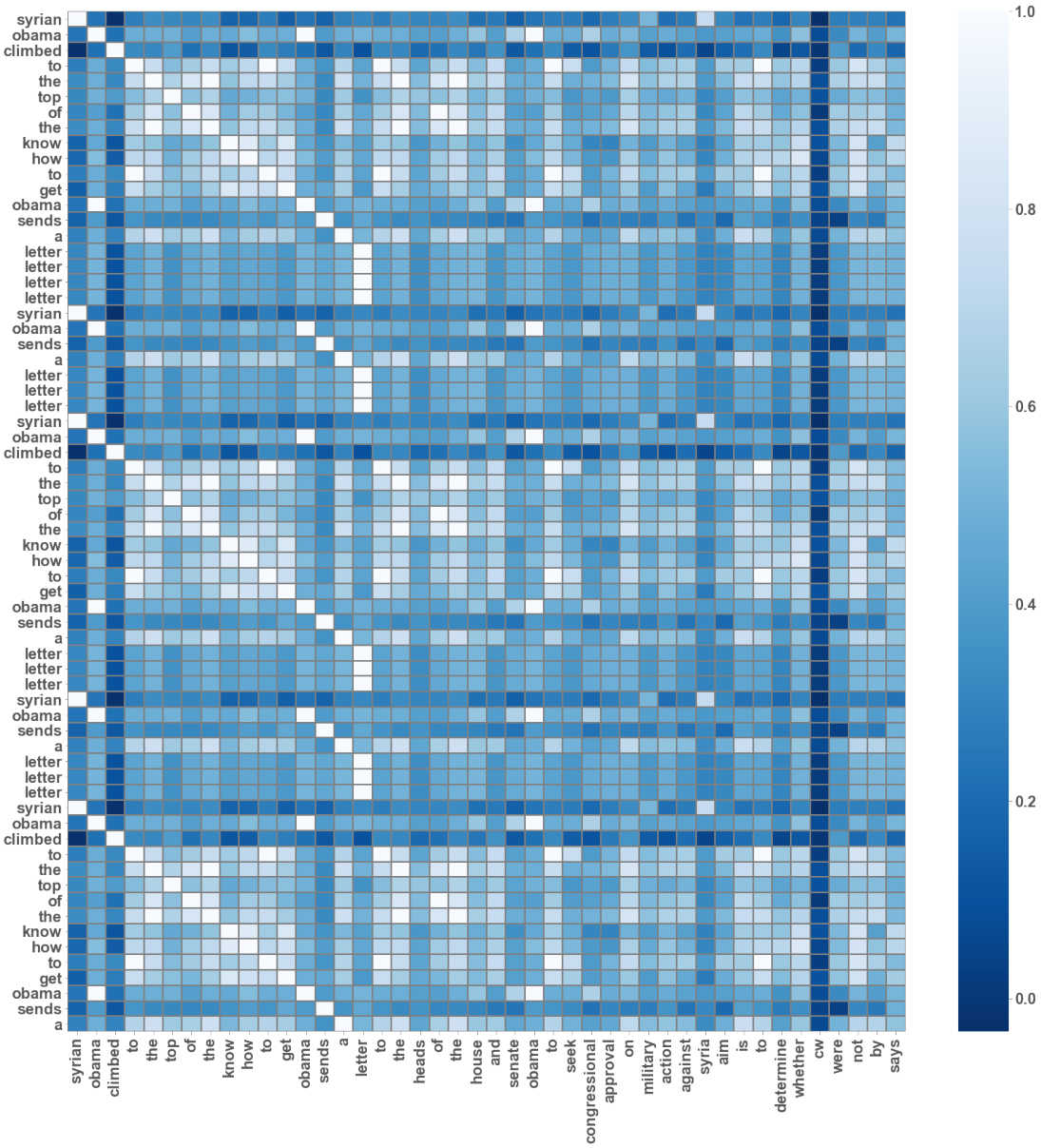


Figure 17: HeatMap comparison in R1 model. Recursive R1 model cosine similarities between the words distributed in the predicted sequence versus the actual summary (X axis). Longer sequences similar to the ground truth summary can be observed. Training is faster.

13. Recursive model 2 (R2)

```
Summary 1:
 usain bolt wins third gold of world championship anchors travel to and from the championships for bolt jamaica doubl
e up in xm relay
Reference 1:
 usain bolt wins third gold of world championship anchors jamaica to xm relay victory eighth gold at the championship
s for bolt jamaica double up in xm relay

Summary 2:
 usain employee in the travel to and from the telecommuting like all gsa is under review
Reference 2:
 the employee in kansas city office is among hundreds of workers the travel to and from the mainland last year cost m
ore than the telecommuting like all gsa is under review
```

Figure 18: Recursive model 2. Summaries generated. The sequence generated is similar but not identical to the actual summary (in Summary 1 word 'Jamaica' is eliminated). The model learns fast and the summarization is meaningful.

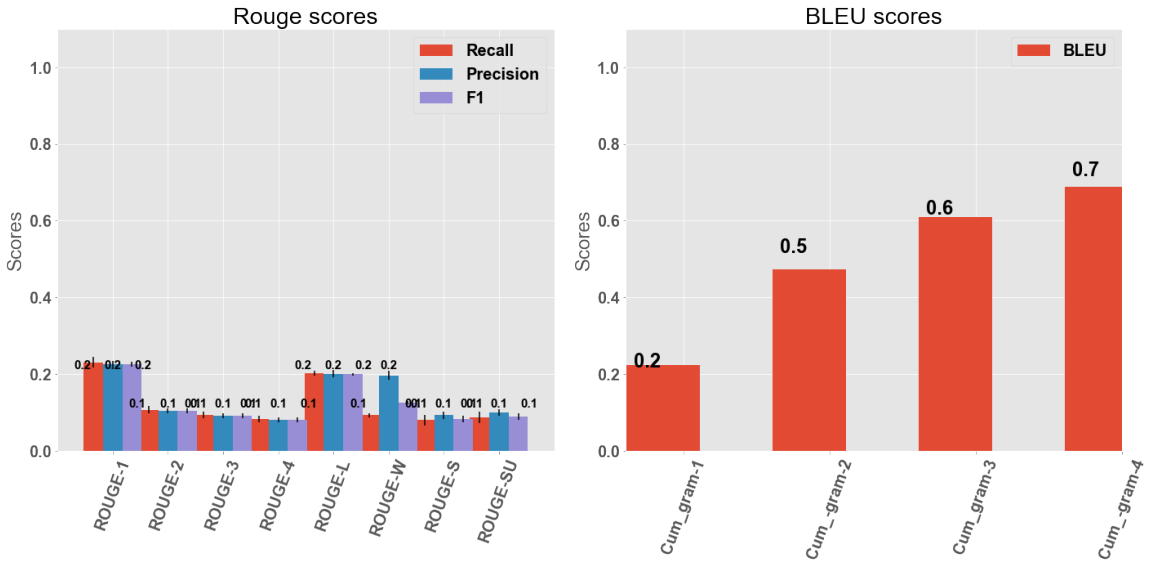


Figure 19: Recursive model 2. BLUE and Rouge scores. Rouge 2 F1 scores values are close to the ground truth values of 0.145 for the F1 score.

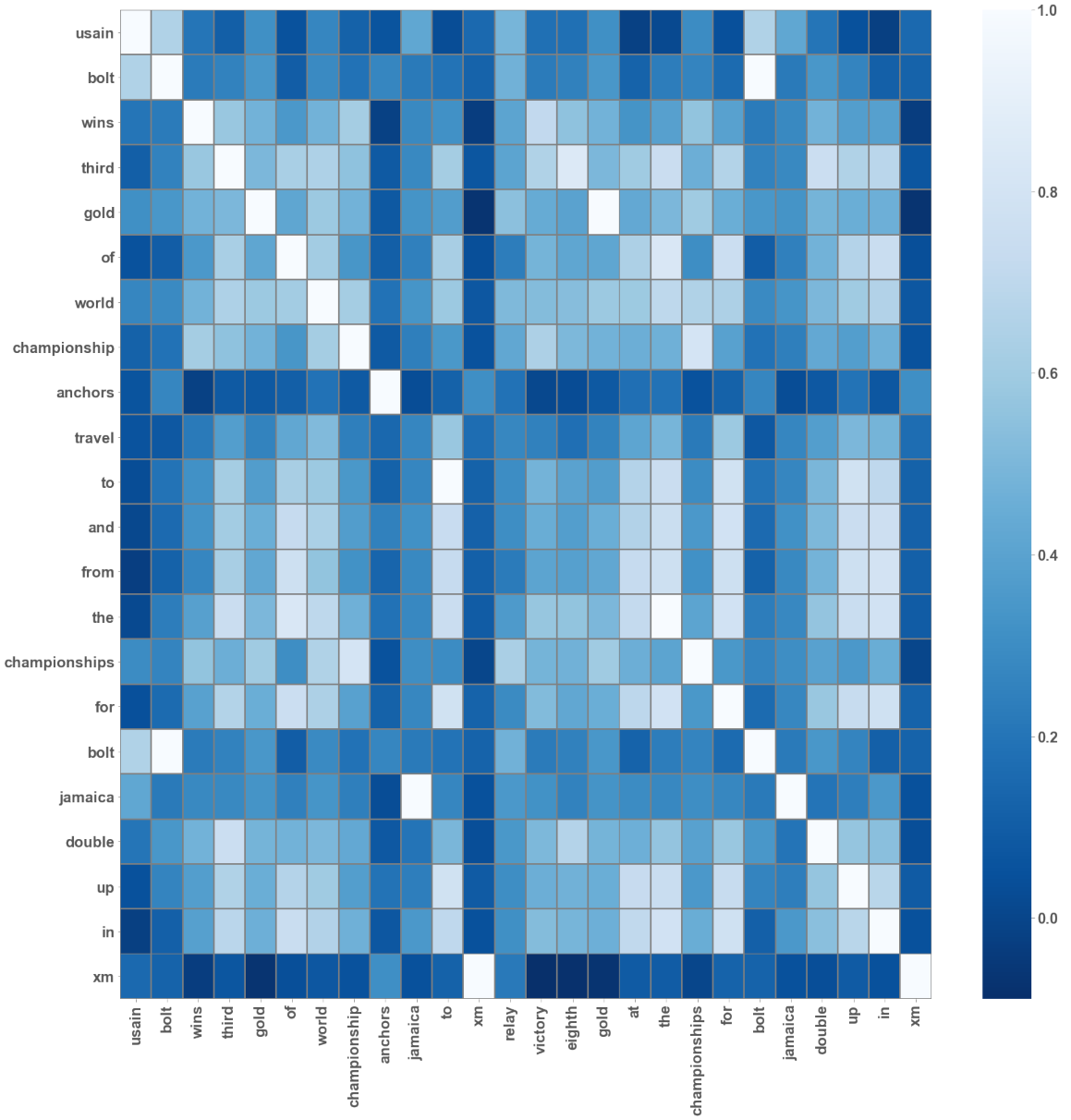


Figure 20: Heat map model R2. Heat map of predicted summarization (y axis) versus actual summarization Complex pattern of words is similar partially to the original but not identical .

14. Extractive (E)

This model is fast and easy to use and, at the same time, it preserves the grammar and accuracy of the original text. However, it is incapable to incorporate novel knowledge from the outside world. Moreover, it cannot use skills like high level abstraction or phrasing to increase the quality of the summary.

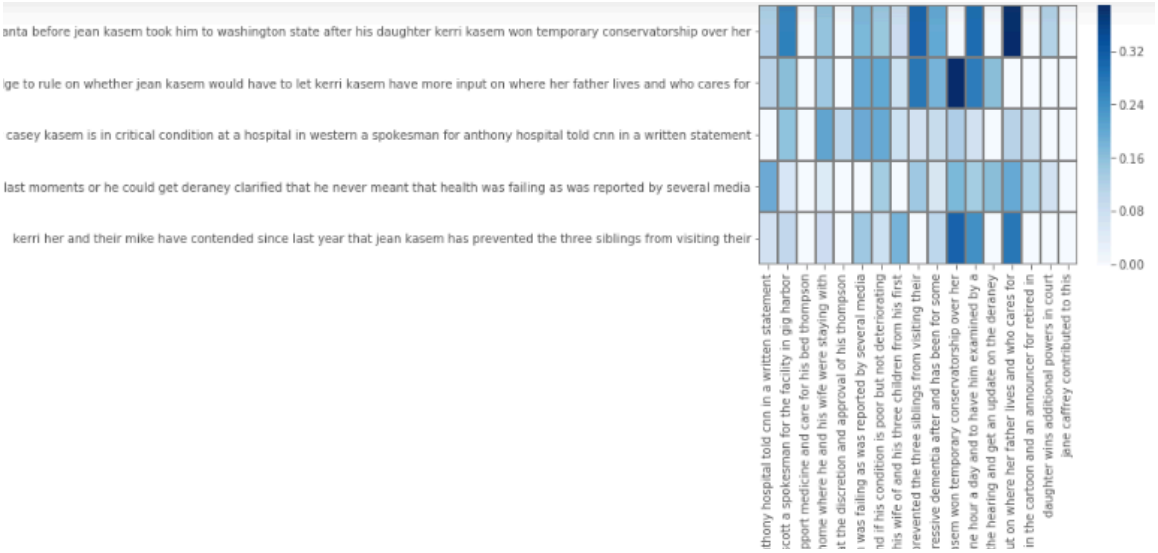


Figure 21: Heat map Extractive Summary. Summarization using the extractive model using sentence scoring with tf_idf at the sentence level followed by extraction. On the y axis, are the top 5 sentences selected from the article body, while on the x axis is the actual article.

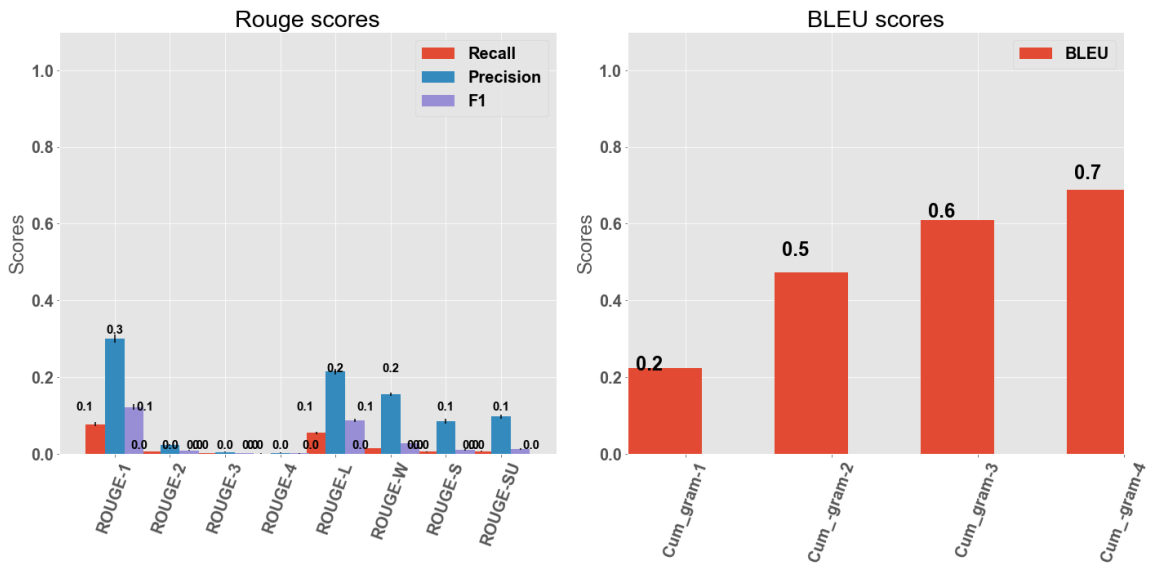


Figure 22: Rouge and BLUE scores for the Extractive model. Surprisingly this simple and fast model scored a ROUGE-2 value that is competitive with the rest of the models.

the consumer electronics show brings a slew of new gadgets tablets running android and windows will debut a new type of thinner are expected to make a splash but some of the largest players in the consumer electronics industry are shunning ces
sara sidner sees another world in a tunnel below tripoli gadhafi may have recorded his taped messages in a studio there rebels are methodically searching through the winding passages

Figure 23: Extractive model: Top 3 sentences are extracted from a given article based on cumulative tf_idf values.

15. Recursive model 2 with attention (R2+A)

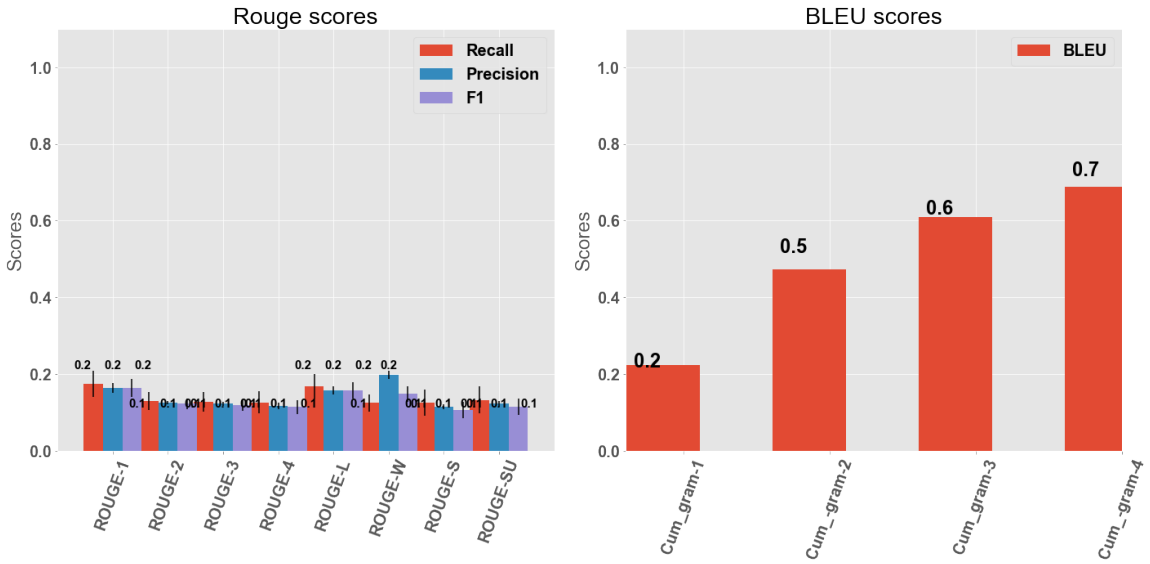


Figure 24: Rouge and BLUE scores for the R2+A model. The F1 score for the ROUGE-2 bigrams is 0.13 which is comparable to the state of the art PG at 0.145.

16. Recursive model 1 with shared attention (R+SA)

The RNN Recursive model 1 was modified by attaching the Attention mechanism. The new architecture gives the Decoder access to non-monotonic context vectors generated from the body of articles. Commentaries and summaries were generated at the same time during prediction. We can observe repetitive patterns due to the long sequences used during training and the long prediction sequences both for the summary and the commentary.

```
Summary 2:
 labor day is the unofficial end of summer and the unofficial start to campaign season as much as billion could could
unofficial start says campaign season as much as billion could could could unofficial start to campaign season as muc
h as billion could could could unofficial start to campaign season as much as billion could could could unofficial st
art to campaign season as much as billion could
Commentary 2:
 trump undermines confidence in the electoral process because your candidate you may not like the electoral college b
ut it has has trump trump undermines confidence in the electoral process because your candidate you may not like the
electoral college but it has has trump trump undermines confidence in the electoral process because your candidate yo
u may not like the electoral college but it has has trump trump
Reference summary 2:
 labor day is the unofficial end of summer and the unofficial start to campaign season as much as billion could be sp
ent on advertising for this midterm election here are five races for these midterms
Reference commentary 2:
 trump undermines confidence in the electoral process because your candidate you may not like the electoral college b
ut it has always been the way we were have chosen our leaders no surprises what undermines the electoral process is p
eople calling winners they like what is supposed to sound like high flying academic thought is dangerously close to s
our very disappointing in a
```

```
Summary 1:
 groups announce legal challenge in phoenix american civil liberties aclu of national immigration law center slam law
mexican american legal legal legal american legal liberties aclu of national immigration law center slam law mexican
american legal challenge in phoenix american civil liberties aclu of national immigration law center slam law mexican
american legal legal legal american legal liberties aclu of national immigration law center slam law mexican
Commentary 1:
 only a few tens of thousands who deal with the pretension of understanding and studying the effect of such will will
will will only a few tens of thousands who deal with the pretension of understanding and studying the effect of such
will will will will only a few tens of thousands who deal with the pretension of understanding and studying the effec
t of such will will
Reference summary 1:
 groups announce legal challenge in phoenix american civil liberties aclu of national immigration law center slam law
mexican american legal defense and educational fund also objects to it they say law encourages racial but supporters
say it involve any illegal acts
Reference commentary 1:
 only a few tens of thousands who deal with the pretension of understanding and studying the effect of such will be t
he rest will do even the defunded or their can come back in a few if and when their models will be defunded academics
will be taken in by the leading french presidential
```

Figure 25: Summaries and commentaries generated by the R1 +Attention shared model. Summaries and commentaries are shown, both for the ground truth and the predicted sequences. We observe repetitive patterns typical to long sequences during training of RNN.

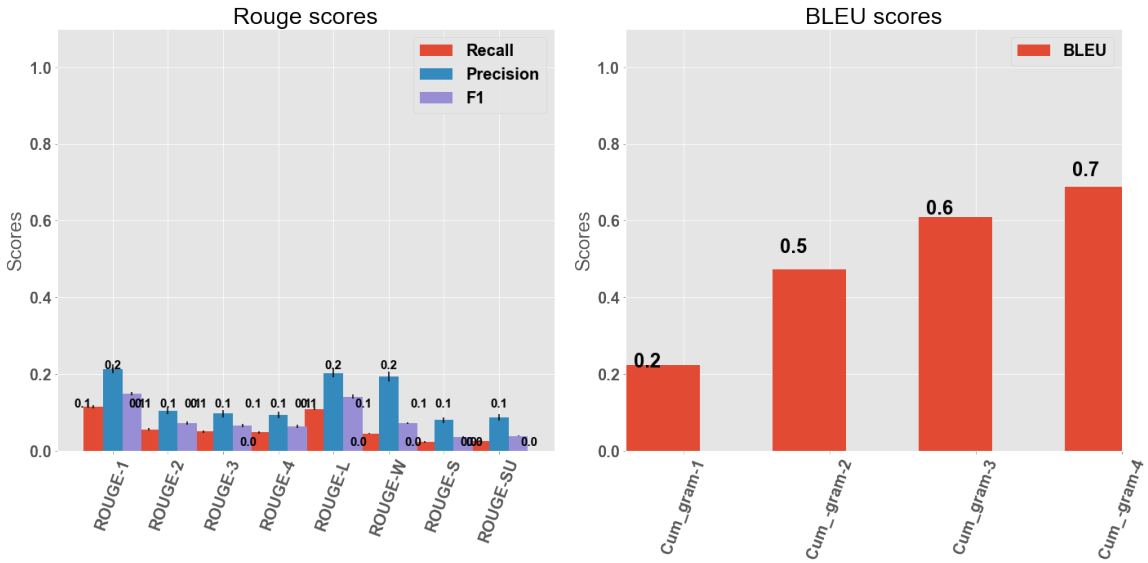


Figure 26: Rouge and BLEU scores for the Recursive model with Shared Attention. The value of Rouge-2 is similar to the reference model pointer generator, not taking into account the differences between the evaluations

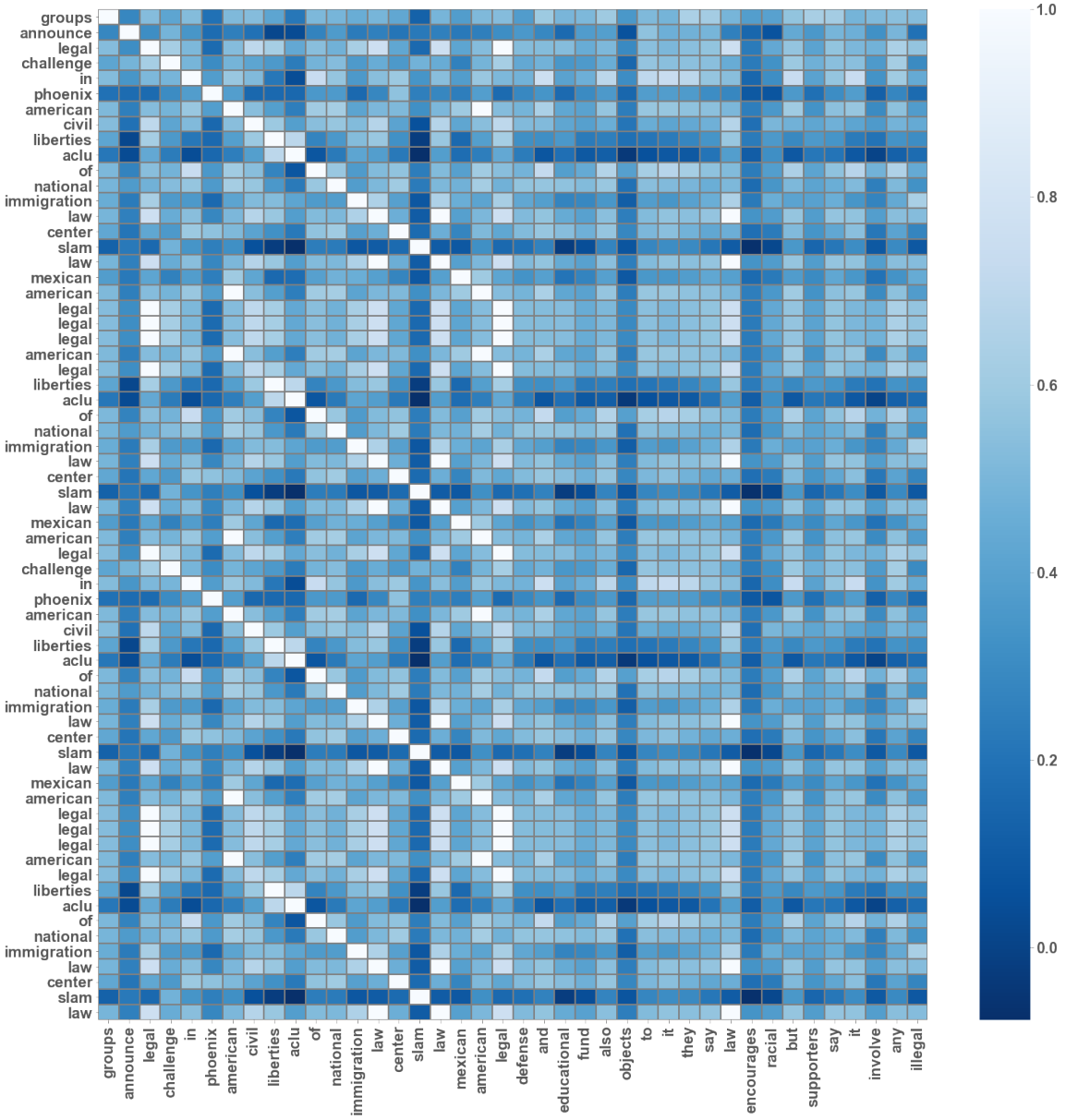


Figure 27: Heat map Recursive Attention Shared model. The attention is focused only on the body of the article. Repetitive patterns are observed.

17. Pointer generator network with coverage (PG)

The pointer generator model and weights were downloaded from (See 2017). The pertained model was fitted on the CNN_DM database and measured against a selected subset of 1000 article summary predictions. The coverage mechanism eliminates the repetitive learning patterns observed in the other models presented above (Figure 28).

```
\Predicted summary 0 french prosecutor says he was not aware of video footage from on board the plane  french prosecu
tor says he was not aware of video footage from on board the plane
\Actual summary 0 marseille prosecutor says `` so far no videos were used in the crash investigation '' despite media
reports  marseille prosecutor says `` so far no videos were used in the crash investigation '' despite media reports

\Predicted summary 1 the palestinian authority became the 123rd member of the international criminal court on wednesd
ay  the palestinian authority became the 123rd member of the international criminal court on wednesday
\Actual summary 1 membership gives the icc jurisdiction over alleged crimes committed in palestinian territories sinc
e last june  membership gives the icc jurisdiction over alleged crimes committed in palestinian territories since las
t june
```

Figure 28: The hybrid Pointer Generator network: predicted summaries. The coverage mechanism eliminates repetitions while the insertion of extracted words and numerical values is evident.

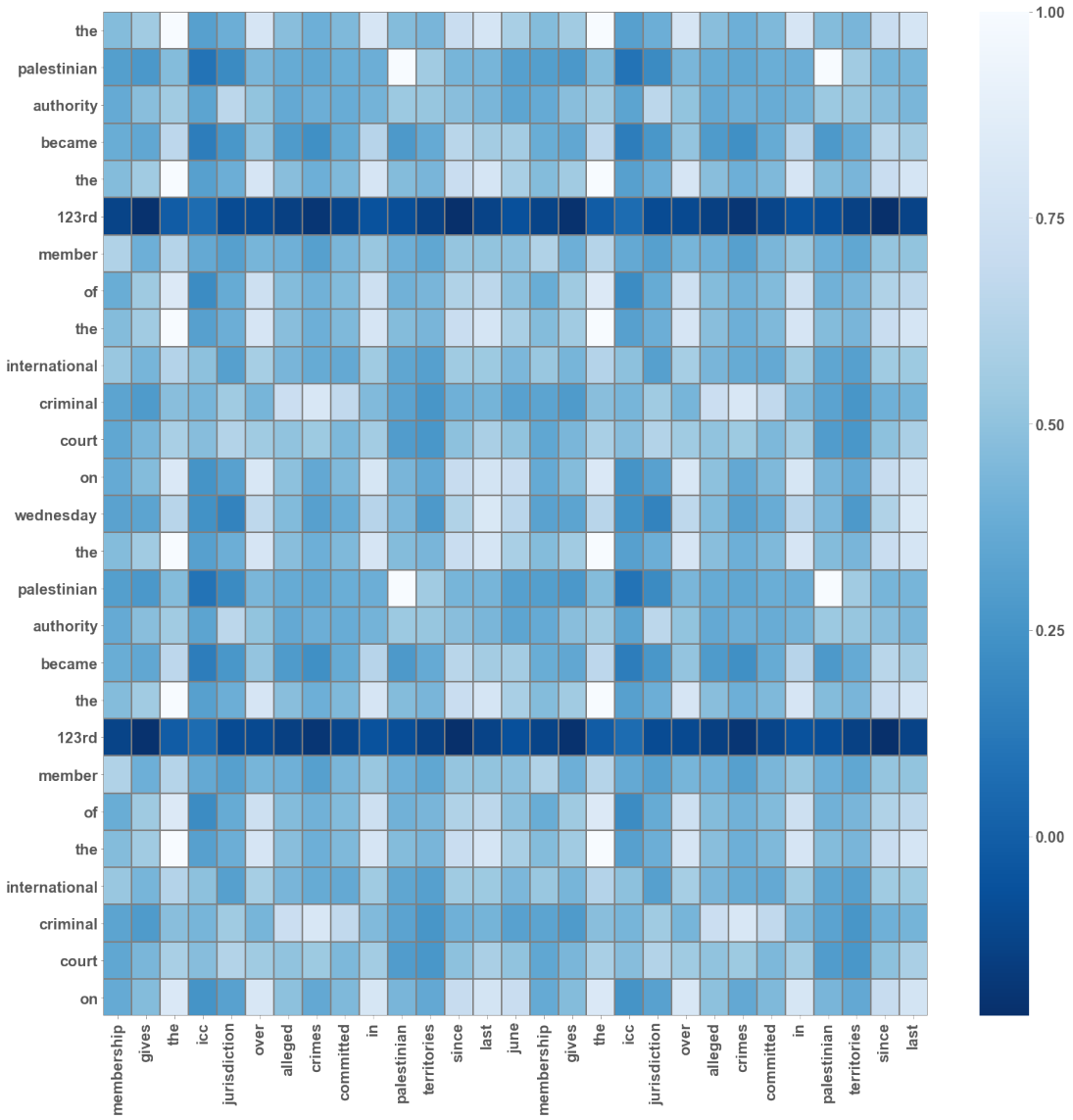


Figure 29: Heat map representation of the PG network with predicted (y axis) versus actual (x axis) summaries and values computed based on cosine similarity. Interestingly the numerical value stand out and are missing from the abstractive reference summary.

We can observe slight differences in the values published before for the PG. These can be explained by the further training and slight differences in preprocessing.

CHAPTER 4: CONCEPT SPACES

18. Background

During learning and communication humans use abstraction as an essential skill. The distinction between “abstract” and “concrete” is important and yet it is hard to define a clear philosophical standard for it (Rosen 2001). In recent years, theories on geometric conceptual spaces put forward a solution to this problem (Gärdenfors and Williams 2001, McGregor, Purver et al. 2016).

Concept spaces were created on a geometrical representation of concepts via Region Connection Calculus. They provide a framework for modeling concepts and consequently for governing semantics. To define the concepts, Gärdenfors (Gärdenfors and Williams 2001) make use of the categorization process.

Creating categories is a fundamental cognitive activity that can be used to clarify the notion of abstraction (i.e. concept). Understanding the process of building categories is instrumental in developing powerful artificial intelligence in general and it was proven very effective on natural language understanding. However, the approach is not without flaw due to inherent complexity of the semantics of words.

McGregor et.al. (McGregor, Agres et al. 2015) note that the ability to map the word meaning to the mental representation of world experiences depends on cognition and language. Since the concept relations are complex, any representation including our natural language is insufficiently effective to describe them completely. Therefore, the language semantics is flexible and vague. Yet the concepts framework make the process of modeling semantics computationally feasible.

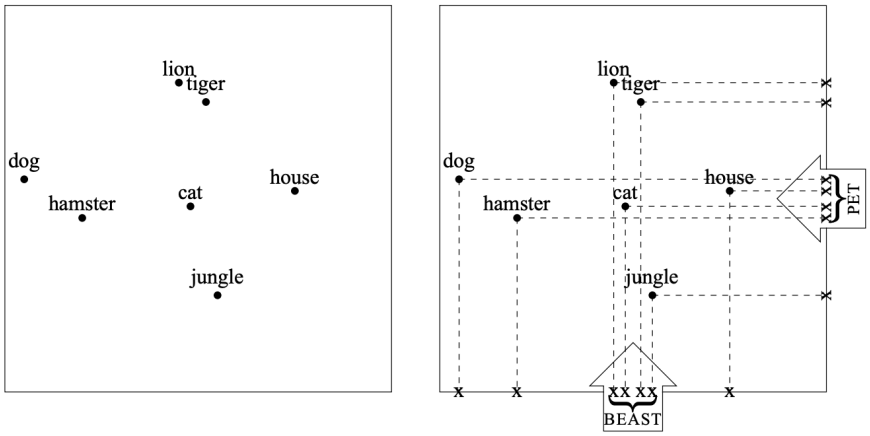


Figure 30: Contextual clustering. (McGregor, Purver et al. 2016), two different perspectives group the words in close proximity to the word-vector $\overrightarrow{cat}$ by their specific contexts.

The neural network approach used to generate the GloVe pretrained word-vectors is leveraging each of the multiple space dimensions. It is capturing the lexical space relations between words and concepts. However, this popular approach is static and hard to use for interpreting words in regards to their contexts to further infer complex semantic mechanisms that link contexts together. A more dynamic mechanism is needed.

The analysis of the word vectors can reveal which dimensions are instrumental for capturing the word meaning in its context (Figure 30). We can select dimensions on the basis of context that will produce contextual spaces where analogical terms always satisfy the relation $A - B \approx C - D$ (e.g. $France - Paris \approx Italy - Rome$). To this end we can create a matrix of word vectors M based on the context co-occurrences of two words w and c and the number of occurrence of words in the vocabulary **W** satisfy the following:

$$M_{w,c} = log_2(\frac{n_{w,c} \times W}{n_w \times (n_c + a)} + 1),$$

where n_w.and n_c are the frequencies of words c and w in the text, while $n_{w,c}$ is the frequency of co-occurrences of the two words.

The dimensions are selected to describe the dynamic distributional concept space by selecting the dimensions with the largest value for:

$$\mu_c = \frac{1}{3}(\sum_{w \in \{A,B,C\}} M_{w,c}).$$

19. Methods

The language model employed to create the distributional concept spaces was utilized as described in (McGregor, Purver et al. 2016, McGregor Dec6 2018). The functional model was downloaded from the repository (McGregor Dec6 2018) and consecutively changed to generate words representative to conceptual spaces automatically. The training was performed on a Wikipedia training corpus (Shaoul 2010). Building the model created a concept space with 35k sparse vectors from 1.15 million word types and a context window size of 5. There were 51551 words accepted by the model and incorporated into the concept spaces. The trained model received 20 dimensions for the subspace selected and the output of word-vectors was set to top 5 for each input word given using the distance from anchor rule.

```
congress FOUND
VECS GROUPED
34022 DIMENSIONS CONSIDERED IN TOTAL
DIMENSIONS CUT
TALLIES NORMED
[(0.0657, 'representatives'), (0.0528, 'speaker'), (0.04728, 'committee'), (0.04725, 'delegates'), (0.046
93, 'passed')]
ANCHOR NORM: 11.180339887498949
```

Figure 31: Concept spaces distance from anchor example. The word “congress” was used as anchor and 5 words were retrieved: representatives, speaker, committee, delegates and passed; with the corresponding distances from anchor. The words are representative for different concept spaces where “congress” is an element of the set.

The user comments were processed by removing the stop words and the most common words were removed by using the inverse term frequency values. The order of words for this particular text was disregarded. Therefore, the neural model processing the concepts emerging from the comment text only used the information stored in the word_vector and not its position in the text. This operation was different to the actual article body and the *ground truth* summary where the order of words is crucial for creating a meaningful text. The resulting words representing the concept space they emerged from created a new set of words that served as input for neural network and helped in the decision made for generating the next word in the predicted summary.

The recurrent-recursive LSTM neural network is using three inputs: article body, the summary and the comment body. Each of them is tokenized, the punctuation signs are removed and the words lowercased where necessary and the stop words where kept in place. The comments body was further processed as described above to select words with increased inverse term frequency.

Input word_vectors (Figure 32) are processed by the Encoder-Decoder. First, the Encoder's embedding layer processes each of the three inputs. LSTM stateful layers are fitting the inputs vectors from the article and the summary and learn the order of words at the same time via long-term dependency loops.

The Encoder-Decoder model is somewhat similar to the recursive RNN model 2 (Appendix B). The tokens from the article body pass through the LSTM layer and capture the information from the current token w_i as well as the information from the previous step w_{i-1}. This type of information retention is also applied for the tokens emerging from the body of the summary. However, the comments' tokens follow a different path.

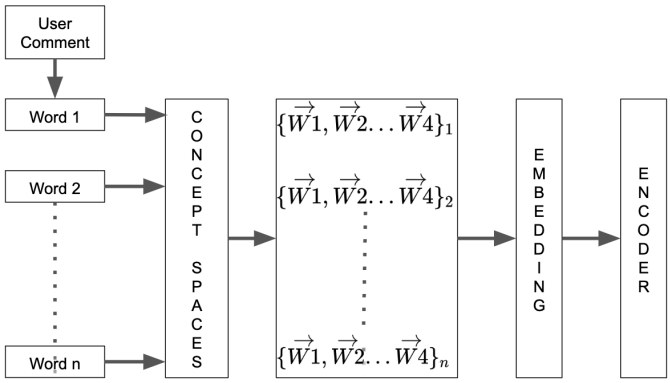


Figure 32: Concept space embedding of user comments. Words provided by the user are mapped into their concept space. The model returns words picked from the concept space using the smallest distance from a central point. These words are fed into the Encoder.

The same network has to channel the information from a given token w_i in the user comments. Since we process these in a manner that disregards the order of tokens in a sentence, we have to use a simpler feed forward Dense layer instead of an LSTM layer for the Encoder part. The information from the 3 inputs is fed to the Decoder after concatenation to allow the prediction of the current word $\hat{w}_i$ (Figure 33). Given this architecture we can calculate the probability of a sequence of N words.

For a given set of words $\{w_1, w_2, \ldots, w_n\}$ and taking into consideration the fact that the order of words matter, the probability distribution is given by the formula:

$$P(\{w_1, w_2, \ldots, w_n\}) = \prod_{i=1}^{N} P(w_i \mid w_1 \ldots w_{i-1})$$

To this end we can calculate the output of the linear Encoder layer $y_t^{(c)}$ for the embedded concept words from comments $x_t^{(c)}$ as $y_t^{(c)} = W_t^{(c)}(x_t^{(c)}) + b^{(c)}$, where $W_t^{(c)}$ are the weights of the layer.

Therefore, the output of the Decoder hidden state h_t is :

$$h_t = W^{(hh)}(h_{t-1}) + W^{(hx)}(h_t^{(a)} + h_t^{(s)} + y_t^{(c)})$$

Here, the $W^{(hh)}$ are the weights of the LSTM for the previous hidden state h_{t-1} and the $W^{(hx)}$ are the input weights for the current state *t.* The $h_t^{(a)}$, $h_t^{(s)}$ are the hidden states of the Encoder layer for article body *(a)*, for the summary *(s)* and $y_t^{(c)}$ is the output from the linear Encoder for the comments.

Finally, the prediction of the current word at time t is calculated by applying the weights and Softmax function to the Decoder's hidden state and can be equated as:

$$\hat{y}_t = W^{(s)} f(h_t)$$

The loss for a given target word $\hat{y}_t$ during training is the negative log likelihood.

Therefore, for the entire predicted summary the loss function is:

$$loss = \frac{1}{N} \sum_{n=0}^{N} - log(P(\hat{y}_t)),$$

where N is the number of tokens (time steps) in the sequence and $\hat{y}_t$ is the predicted word at time t.

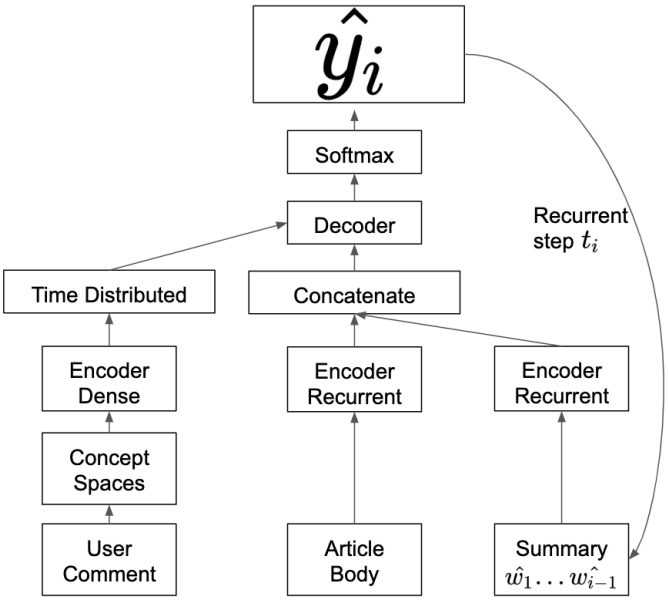


Figure 33: Recursive mechanism for capturing the Concept Spaces information. The words extracted from concept spaces related to user content are fed to the Decoder. Softmax normalizes the output into probability distribution.

20. Results

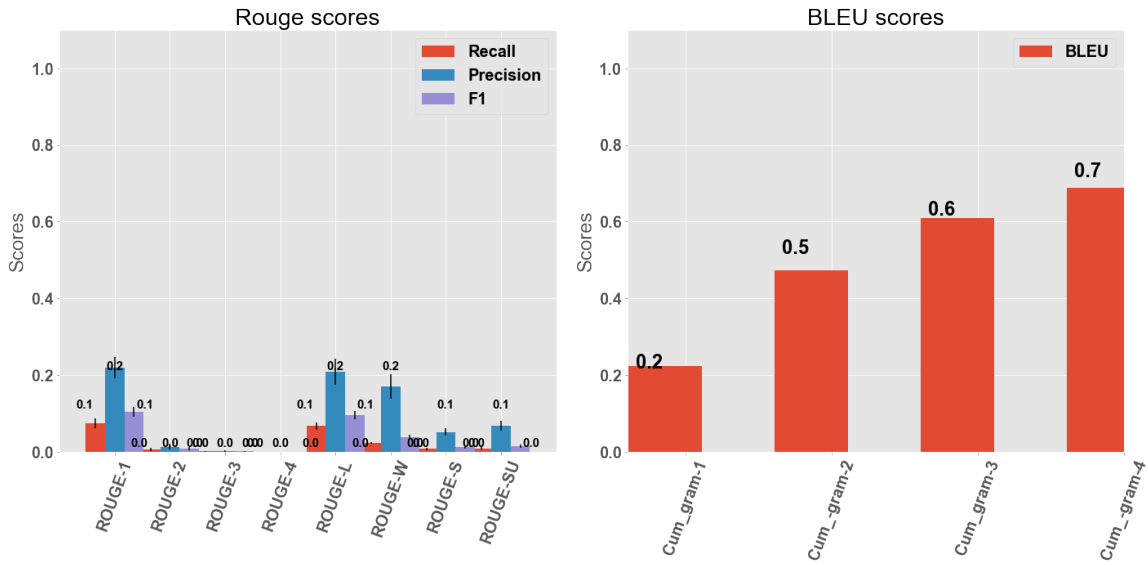


Figure 34: ROUGE-BLUE measurements for RNN with Conceptual Spaces. The F1-score is low compare to the state of the art PG model. However we expect abstraction to alter the number of overlapping bigrams in the predicted summary due to the conceptual spaces method.

The abstraction changes the original text and words, while preserving the main idea. This is why some of the measurements like ROUGE and BLUE which are based on similarity of n-grams to the source text may not reflect accurately the quality of the summarization.

CHAPTER 5: DISCUSSION

There is no precise formula for automatically creating the perfect summary. Even humans are inconsistent in producing the same summarization over time, given the same source document. Moreover, the best professional summarizers admit that the results no matter how good leave something to be desired. This dilemma stems from the simple fact that the summarization process is hard to define exactly.

The main reasons are the loss of information and the abstraction process. The traditional methods of automatic text summarization are based on information extraction and compression (Knight 2002). During this process, some of the original information is discarded. However, this simple process involves cognitive mechanisms that are typically associated with intelligent beings.

Similarly, the quality of the abstraction process is increased by the knowledge of the outside world and experiences (Giunchiglia and Walsh 1992). It leverages the ability to synthesize or incorporate new words and relevant information while preserving the essence of the message from the source text. Ipso facto, this process involves cognitive mechanisms that give humans the ability to categorize and classify various entities material or immaterial. Gardenfors et.al. (Gärdenfors and Williams 2001) propose the prototypes extraction theory as a basis for making abstraction work in biological or artificial intelligent systems.

Language cannot perfectly define the mental representations of the real world experiences (Sperber and Wilson 1986). Some believe this is intentional and not just an undesirable trait (Barsalou 1993). Humans share these linguistic representations and have their own personal unique versions of these experiences. By this very fact, we can infer

that machine learning mechanisms such as the ones presented in the current work, will have a difficult time creating similar abstract representation using the knowledge encoded in the corpora (Gärdenfors and Williams 2001). Secondly, different human individuals have different views of the same representations. As such, there is no perfect summary that will satisfy everyone's impressions of the source text. Therefore, the traditional extractive approaches or more recently the recurrent neural networks for sequence transduction (Vaswani, Shazeer et al. 2017) present only partial solutions to the task at hand.

The personalized summarization presents a solution to the problems described above. Intelligent systems can accomplish their tasks automatically but also in a more individualized and unique fashion. This can be achieved by allowing the models to learn user preferences and create user profiles. The user generated content provides insight into how people use language to express abstract thoughts. Beyond this aspect, written language sheds light on unique user preferences that are part of human personality and make us unique.

Beyond the addition of user related information, information processing is another crucial element in creating quality summarizations. The current work explores the attention based mechanisms and the concept spaces as novel ways to improve transduction of text sequences.

The attention mechanisms described here harvest information encoded in language in a non sequential fashion. Words that are not in the same context and not in their natural position in a sentence convey additional information. This mechanism was

first applied in Neural Machine Translation and lately has been used in most of the areas where machine learning and artificial intelligence can be applied.

Simply put, the attention mechanism searches for a sets of relevant context vectors stemming from one or many words that are part of a source document. These vectors can be utilized to generate words that are part of more refined languages techniques involving paraphrasing, generalizations and possibly even metaphors. However, more recent techniques such as the theory of conceptual spaces are built specifically to govern language abstraction.

In our efforts we show that concept spaces can be used as an efficient platform for machine learning. The seq2seq techniques employing Recurrent Neural Networks can further benefit from utilizing prototyping via concept spaces. The essential information encoded in words is structured dynamically depending on the context. This allows the RNN to learn from the source text not only sequentially (monotonically) but also non-sequentially.

In conclusion, the mechanisms for personalized summarizations via concept spaces or attention are an efficient method to automatically generate text that is appealing to users.

CHAPTER 6: CONCLUSIONS

In a world where the information is constantly surrounding us, it is imperative to present the relevant data succinctly. Inputs have to come at the opportune moment, in an easy to digest form and preferably in tune with our latest preferences. One of the ways we can achieve this is to create models able to deeply understand encoded meaning in text. Here, we presented the abstraction mechanism in two of its latest forms: the attention and the conceptual spaces. Both of these approaches focus on extricating information from text in a non-sequential non-monotonic fashion.

The RNN with shared attention (R+SA) achieved similar performance to the state of the art model (See, Liu et al. 2017). Similarly, the conceptual spaces model creates a deeper understanding of the old concept of meaning via context. The theory puts forward the idea that neighboring words-vectors may share only some dimensions and learning about them can lead us to the discovery of prototypes.

Future work may explore the possibility of integrating concepts into neural networks at the cell level.

REFERENCES

. "Keras Distributed Wrappers." 2019, from https://keras.io/layers/wrappers/.

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean and M. Devin (2016). "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467.

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving and M. Isard (2016). Tensorflow: A system for large-scale machine learning. 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16).

Bahdanau, D., K. Cho and Y. Bengio (2014). "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473.

Barsalou, L. W. (1993). Theories of Memory.

Brownlee, J. (2019). "Machine Learning Mastery." from https://machinelearningmastery.com/encoder-decoder-models-text-summarization-keras/.

Conroy, J. and D. O'Leary (2001). Text summarization via hidden Markov models.

Cremmins, E. T. (1992). "Value-added processing of representational and speculative information using cognitive skills." **18**(1): 27-37.

Cremmins, E. T. (1996). The art of abstracting. Arlington, VA, Information Resources Press.

Gärdenfors, P. and M.-A. Williams (2001). Reasoning about categories in conceptual spaces. IJCAI.

Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems, " O'Reilly Media, Inc.".

Giunchiglia, F. and T. Walsh (1992). "A theory of abstraction." Artificial intelligence **57**(2-3): 323-389.

Habermas, J. (2015). Between facts and norms: Contributions to a discourse theory of law and democracy, John Wiley & Sons.

Hebb, D. O. (2005). The organization of behavior: A neuropsychological theory, Psychology Press.

Hochreiter, S. and J. Schmidhuber (1997). "Long short-term memory." Neural computation **9**(8): 1735-1780.

Jaccard, P. (1901). "Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines." Bull Soc Vaudoise Sci Nat **37**: 241-272.

Keras. (2019). "Keras Distributed Wrappers." 2019, from https://keras.io/layers/wrappers/.

Kingma, D. P. and J. Ba (2014). "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980.

Knight, K., & Marcu, D. (2002). "Summarization beyond sentence extraction: A probabilistic approach to sentence compression." Artificial Intelligence **139**(1): 91-107.

Kosub, S. (2019). "A note on the triangle inequality for the jaccard distance." Pattern Recognition Letters **120**: 36-38.

Levy, O. and Y. Goldberg (2014). Neural word embedding as implicit matrix factorization. Advances in neural information processing systems.

Levy, O., Y. Goldberg and I. Dagan (2015). "Improving distributional similarity with lessons learned from word embeddings." Transactions of the Association for Computational Linguistics **3**: 211-225.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. Text summarization branches out.

Ludwig, O. (2017). "End-to-end Adversarial Learning for Generative Conversational Agents." arXiv cs.CL.

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

McCulloch, W. S. and W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity." The bulletin of mathematical biophysics **5**(4): 115-133.

McGregor, S. (Dec6 2018). "ModelMaker." from https://github.com/masteradamo/ModelMaker.

McGregor, S., K. Agres, M. Purver and G. A. Wiggins (2015). "From distributional semantics to conceptual spaces: A novel computational method for concept creation." Journal of Artificial General Intelligence **6**(1): 55-86.

McGregor, S., M. Purver and G. Wiggins (2016). "Words, concepts, and the geometry of analogy." arXiv preprint arXiv:1608.01403.

Mikolov, T., K. Chen, G. Corrado and J. Dean (2013). "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado and J. Dean (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems.

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

Olah, C. (2015). "Understanding LSTM Networks: Recurrent Neural Networks." https:// colah.github.io/posts/2015-08-Understanding-LSTMs/.

Papineni, K., S. Roukos, T. Ward and W.-J. Zhu (2002). BLEU: a method for automatic evaluation of machine translation. Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics.

Pennington, J., R. Socher and C. Manning (2014). Glove: Global vectors for word representation. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP).

Pollock, J. J. and A. Zamora (1975). "Automatic abstracting research at chemical abstracts service." Journal of Chemical Information and Computer Sciences **15**(4): 226-232.

Radev, D. R. (2000). "Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies." NAACL-ANLP 2000 Workshop on Automatic summarization -.

Ramachandran, P., B. Zoph and Q. V. Le (2017). "Searching for activation functions." arXiv preprint arXiv:1710.05941.

Rosen, G. (2001). "Abstract objects."

See, A. (2017, May 1). "Pointer Generator Github Repository." Retrieved May 12, 2019, from https://github.com/abisee/pointer-generator.

See, A. (2017 May 1). "Pointer Generator Github Repository." Retrieved May 12, 2019, from https://github.com/abisee/pointer-generator.

See, A., P. J. Liu and C. D. Manning (2017). "Get to the point: Summarization with pointer-generator networks." arXiv preprint arXiv:1704.04368.

Shaoul, C. (2010). "The westbury lab wikipedia corpus." Edmonton, AB: University of Alberta: 131.

Sperber, D. and D. Wilson (1986). Relevance: Communication and cognition, Harvard University Press Cambridge, MA.

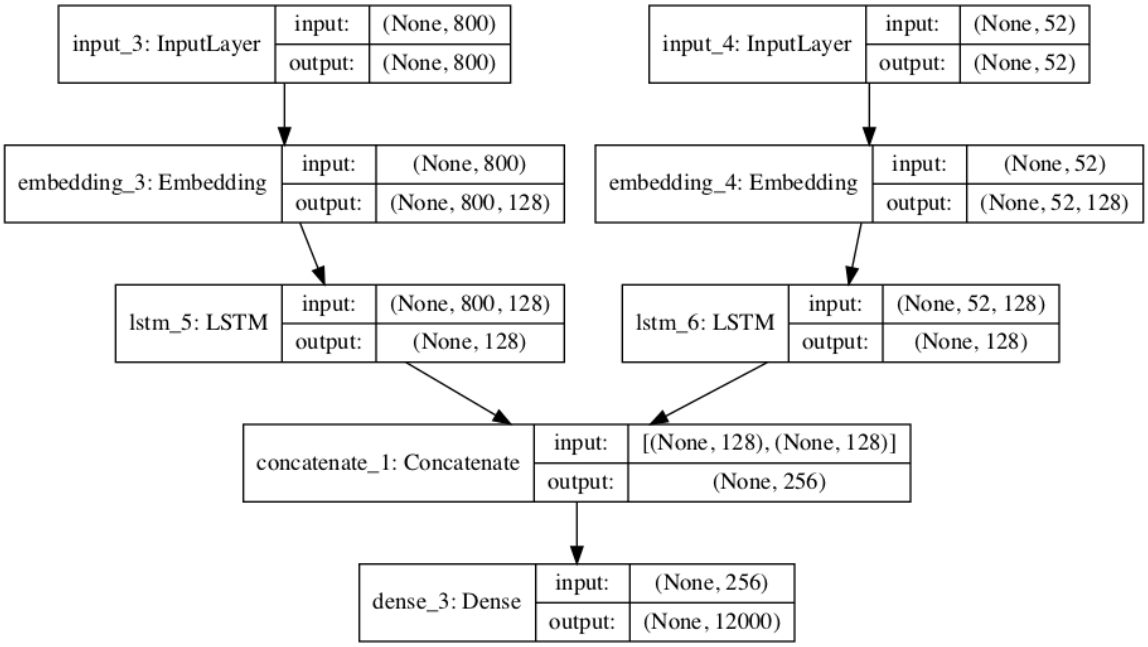
Svore, K., L. Vanderwende and C. Burges (2007). Enhancing single-document summarization by combining RankNet and third-party sources. Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL).

Torres-Moreno, J.-M. (2014). Automatic text summarization, John Wiley & Sons.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin (2017). Attention is all you need. Advances in neural information processing systems.

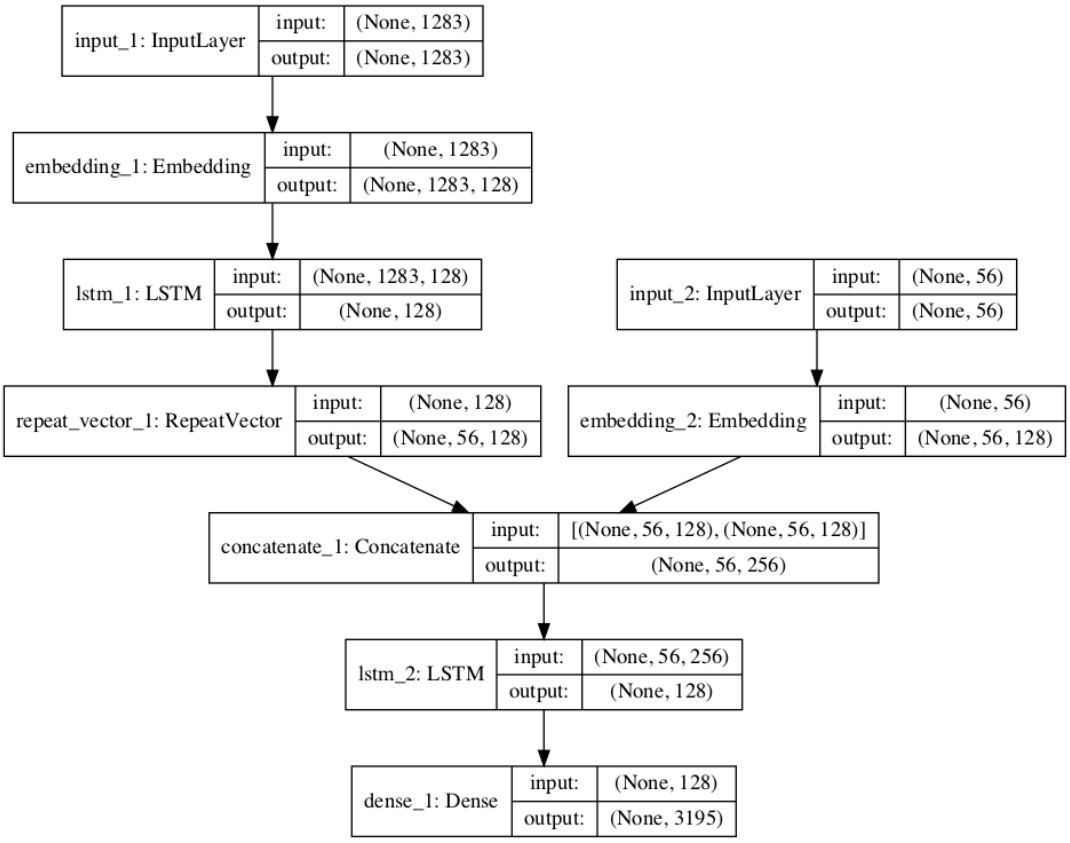
Wu, H. C., R. W. P. Luk, K. F. Wong and K. L. Kwok (2008). "Interpreting tf-idf term weights as making relevance decisions." ACM Transactions on Information Systems (TOIS) **26**(3): 13.

APPENDIX A: RECURRENT MODEL R1.



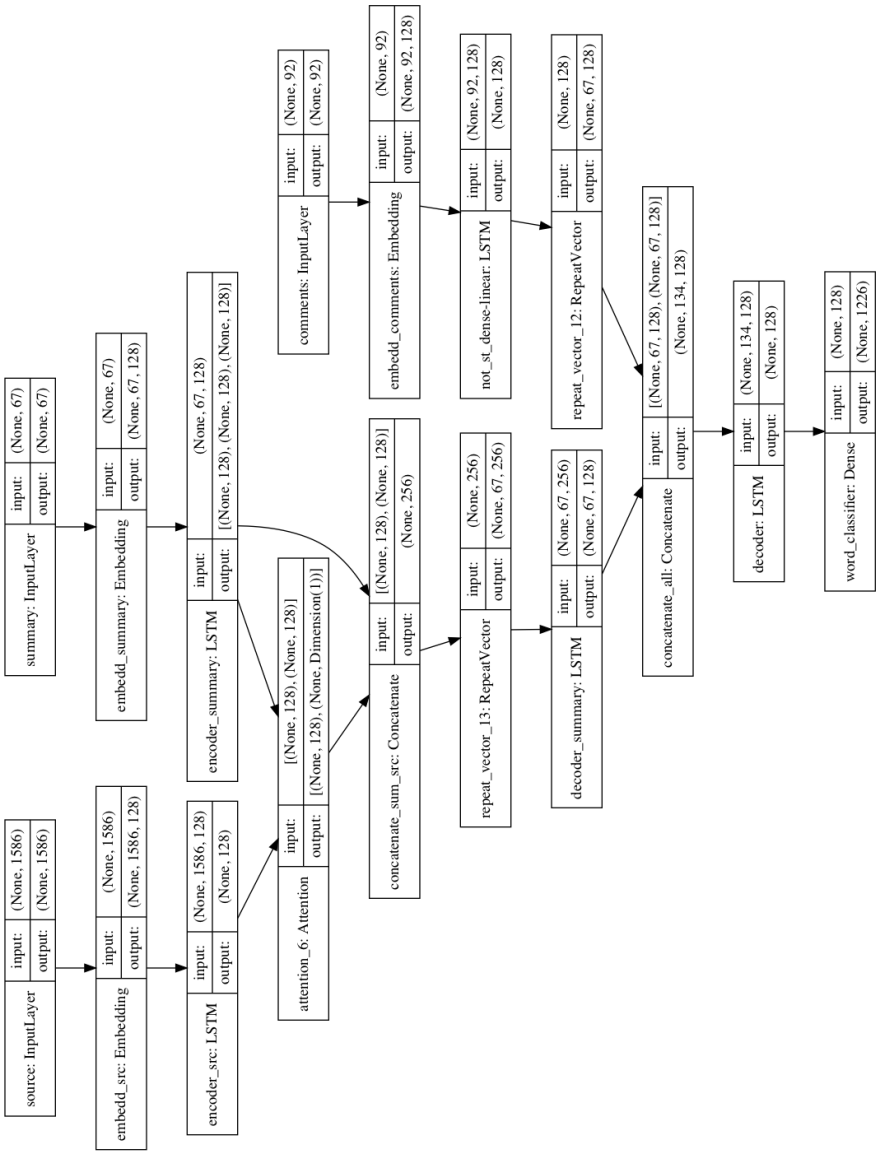
Recursive model R1 - schema. The Encoder takes as inputs the article bodies and the summary generate until time *t*. The Decoder generates the words one at the time.

APPENDIX B: RECURRENT MODEL R2.



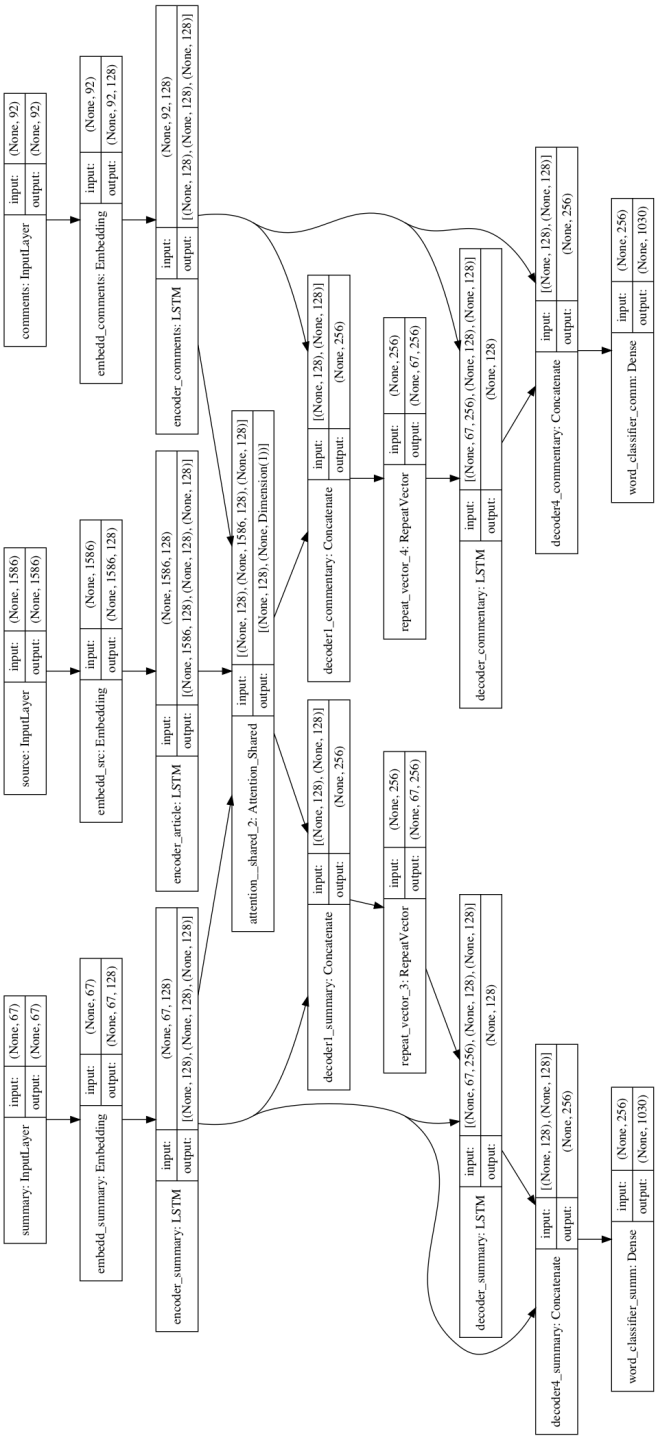
Recursive model R2 - schema. The Encoder takes as inputs the embed sequence temporal dependent from the article bodies (left) and the embedded summary generated until time *t-1*. The Decoder generates one word at the time t.

APPENDIX C: RECURRENT MODEL R2+A.



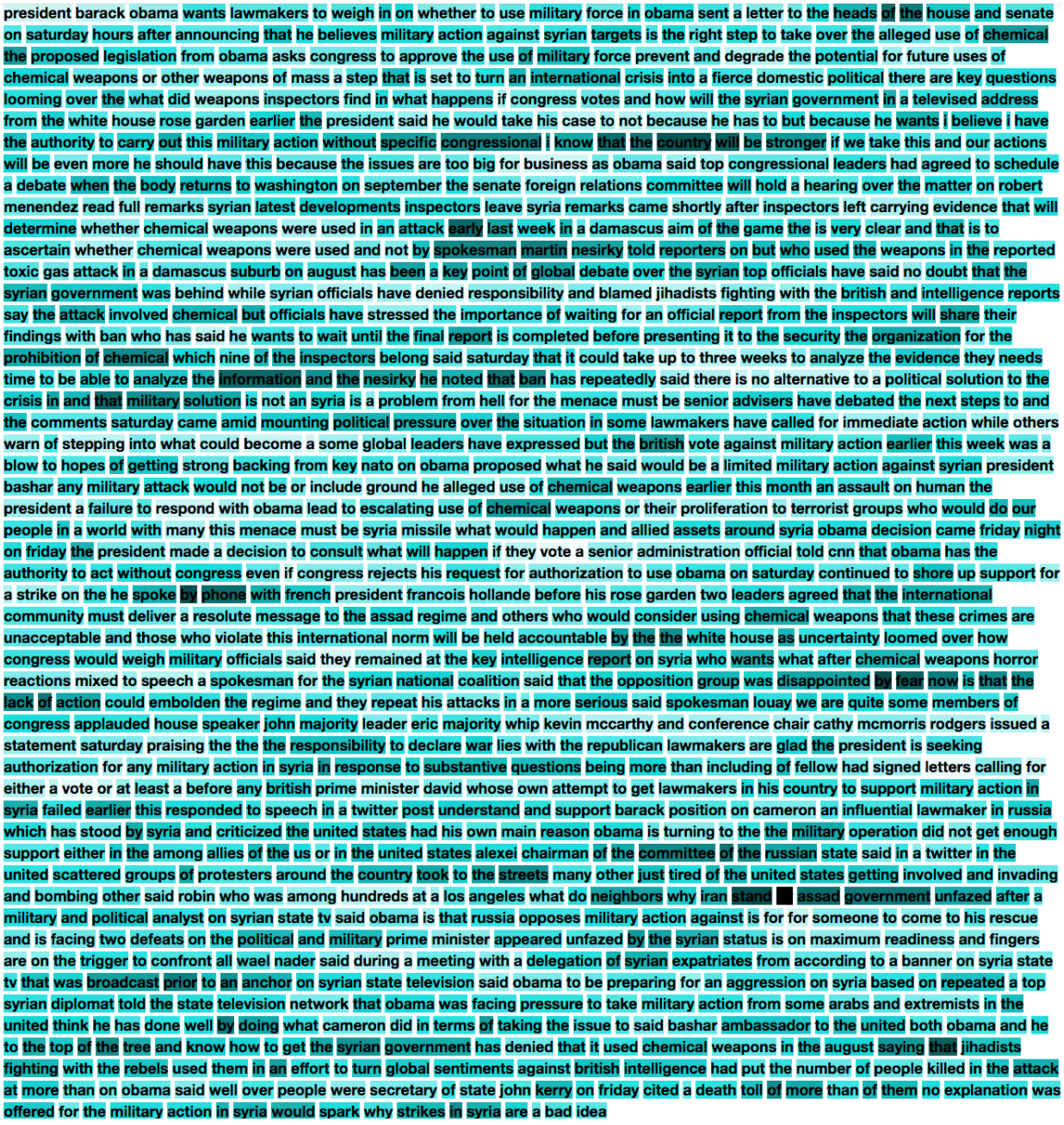
Recursive model R2+A- schema. The comments are passed through a LSTM non-stateful layer as a linear function. The Decoder processes the context vectors and Encoder hidden states for the comments jointly

APPENDIX D: RECURRENT MODEL WITH SHARED ATTENTION.



Recursive model with shared attention (R+SA) - schema. The Encoder processed the information channeled from 3 input sources: article, the summary generated at time t and the commentary generated at time t. The 2 Decoders processes the information and generate the two words (one for comments and one for the summary) both output at time *t.*

APPENDIX E: RNN WITH SHARED ATTENTION - ATTENTION HIGHLIGHTS



The attention context vectors are highlighted based on their weights. The intense colors represents a higher weight. This idea comes from (See, Liu et al. 2017) however the approach and code are original.