Abstractive text summarization based on deep learning and semantic content generalization

Panagiotis Kouris
School of Electrical & Computer Engineering
National Technical University of Athens, Greece
pkouris@islab.ntua.gr

Georgios Alexandridis
School of Electrical & Computer Engineering
National Technical University of Athens, Greece
gealexandri@islab.ntua.gr

Andreas Stafylopatis
School of Electrical & Computer Engineering
National Technical University of Athens, Greece
andreas@cs.ntua.gr

Abstract

This work proposes a novel framework for enhancing abstractive text summarization based on the combination of deep learning techniques along with semantic data transformations. Initially, a theoretical model for semantic-based text generalization is introduced and used in conjunction with a deep encoder-decoder architecture in order to produce a summary in generalized form. Subsequently, a methodology is proposed which transforms the aforementioned generalized summary into human-readable form, retaining at the same time important informational aspects of the original text and addressing the problem of out-of-vocabulary or rare words. The overall approach is evaluated on two popular datasets with encouraging results.

1 Introduction

Text Summarization (TS) aims at composing a concise version of an original text, retaining its salient information. Since *manual* TS is a demanding, time expensive and generally laborious task, *automatic* TS is gaining increasing popularity and therefore constitutes a strong motivation for further research.

Current efforts in automatic TS mainly focus on summarizing *single documents* (e.g. news, articles, scientific papers, weather forecasts, etc.) and *multi-documents* (e.g. news from different sources, user reviews, e-mails etc.), reducing the size of the initial text while at the same time preserving key informational elements and the meaning of content.

Two main approaches to automatic TS have been reported in the relevant literature; *extractive* and *abstractive* (Gambhir and Gupta, 2017; Allahyari et al., 2017). In the former case, those sentences of original text that convey its content are firstly identified and then extracted in order to construct the summary. In the latter case, new sentences are generated which concatenate the overall meaning of the initial text, rephrasing its content. Abstractive TS is a more challenging task; it resembles human-written summaries, as it may contain rephrased sentences or phrases with new words (i.e. sentences, phrases and words that do not appear in the original text), thereby improving the generated summary in terms of cohesion, readability or redundancy.

The main contribution of this work is a novel abstractive TS technique that combines deep learning models of encoder-decoder architecture and semantic-based data transformations. Since the majority of literature in abstractive TS focuses in either of the aforementioned parts, the proposed approach tries to bridge this gap by introducing a framework that combines the potential of machine learning with the importance of semantics. The said framework is comprised of three components; (i) a theoretical model for text generalization (Section 3) (ii) a deep learning network whose input is the text and its output a summary in generalized form (Section 4) and (iii) a methodology of transforming the "generalized" summary into a human-readable form, containing salient information of the original document (Section 5). Additionally, the proposed framework is capable of coping with the problem of *out-of-vocabulary* (OOV) words (or words of limited occurrences), thereby achieving semantic content generalization. The overall architecture is evaluated on `Gigaword` (Napoles et al., 2012; Rush et al., 2015) and `Duc 2004` (Over et al., 2007), two popular datasets used in TS tasks, with the obtained results being promising, outperforming the current state-of-the-art.

The rest of this paper is organized as follows; Section 2 overviews the related work and Sections 3-5 outline the components of the proposed framework. Section 6 describes the experimental procedure in detail and discusses the obtained results. Finally, the paper concludes in Section 7, where

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5082–5092
Florence, Italy, July 28 - August 2, 2019.

possible future extensions are examined.

2 Related work

Abstractive TS methods can be broadly classified into *structure* and *semantic* based approaches (Moratanch and Chitrakala, 2016). The former make use of pre-defined structures (e.g. ontologies, trees, templates, graphs and rules), whereas the latter utilize the semantic representation of text along with natural language generation systems (based on information items, predicate arguments and semantic graphs). Recently, deep learning architectures have been widely adopted in abstractive TS and they have since become the state-of-the-art (Gupta and Gupta, 2019), especially in short text summarization (Paulus et al., 2017) that is the focus of the current work. The proposed approach further extends the said architectures with semantic-based concept generalization, in an effort to improve the overall system performance.

In particular, semantic-based approaches utilizing (semantic) graphs produce the desired summaries through the extraction of ontological and syntactical relations in text, mainly by reducing the graph or by locating its key concepts (Khan et al., 2018; Joshi et al., 2018; Moawad and Aref, 2012). Item-based solutions, on the other hand, employ the notion of *information item* (the smallest unit of coherent textual information such as subject, verb and object triplets) in order to generate the summary out of the top-rated sentences. For example, the information items, along with temporal and spatial characteristics, are used in (Genest and Lapalme, 2011) in order to produce the abstractive summary.

Predicate argument-based approaches merge the respective structures of text (i.e. verbs, subjects and objects) and the summary is being formed from the top-ranked such structures (Alshaina et al., 2017; Zhang et al., 2016). Nevertheless, semantic-based methods are not able to achieve comparable performance to deep learning approaches (Gupta and Gupta, 2019) and for this reason, a framework utilizing semantic-based data generalization for the enhancement of *sequence-to-sequence* (*seq2seq*) deep learning abstractive summarization is presented in this work. *Seq2seq* architectures require a sequence of words at their input and also emit a different, in the general case, sequence of words at their output.

An early approach to using semantic resources for the generalization of concepts connected with a conjunctive or disjunctive relation is due to (Belkebir and Guessoum, 2016), which replaces two or more consecutive concepts by one more general word, entailing the meaning of the initial ones (e.g. the phrase "apples and oranges" may be replaced by the word "fruits"). Our proposed methodology, however, is not limited to conjunctive and disjunctive relations and can, therefore, generalize every concept of a text.

The state-of-the-art in abstractive TS deep learning systems employ *seq2seq* models of encoder-decoder architectures along with attention mechanisms, primarily based on *recurrent neural networks* (RNNs) and especially on *long short-term memory* networks (LSTMs) and *gated recurrent units* (GRUs) (Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017; Song et al., 2018; Chen et al., 2016; Gupta and Gupta, 2019). In these cases, the encoder input is a sequence of words which are subsequently converted into a vector representation and the decoder, assisted by the attention mechanism which focuses on specific words at each step of the input sequence (Bahdanau et al., 2014), determines the output, emitting the next word of the summary based on the previous ones.

The methodology described above is further extended in (Rush et al., 2015), where a neural attention-based model is trained end-to-end on a large amount of data (article-summary pairs) that learns to produce abstractive summaries. Similarly, Nallapati et al. (2016) and See et al. (2017) train encoder-decoder models with attention mechanisms in order to face the problem of unseen (out-of-vocabulary) words, incorporating a *pointer generator network* in their system. Furthermore, See et al. (2017) avoid repetition of the same words in the summary through the inclusion of a coverage mechanism, while Lin et al. (2018) address the same problem by proposing a model of a convolutional gated unit that performs global encoding for the improvement of the representation of the input data. Finally, Song et al. (2018) propose a deep LSTM-CNN (*convolutional neural network*) framework, which generates summaries via the extraction of phrases from source sentences.

The presented approach in this work is also based on a *seq2seq* deep learning model (See et al., 2017). In contrast to the systems outlined above,

the novelty of our technique lies in the device of a semantic-based methodology for text generalization, which is going to be presented in detail in the forthcoming sections.

3 Text generalization

The basic assumption of text generalization is the existence of a *taxonomy of concepts* that can be extracted from text (Definition 3.1). More specifically, the said taxonomy contains *concepts* and their *hypernyms* (Definition 3.2) in a hierarchical structure. Once the concepts have been extracted, the *taxonomy path* (Definition 3.3), containing the ordered sequence of concepts according to their *taxonomy depth* (Definition 3.4), is used for generalizing text. Figure 1 illustrates an example taxonomy of five concepts, where concept c_4 has a taxonomy depth equal to 3 and a taxonomy path $P_{c_4} = \{c_4, c_2, c_1, c_0\}$.

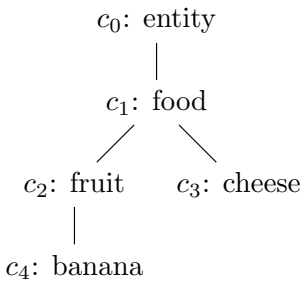

Figure 1: A taxonomy of concepts.

Definition 3.1 (Taxonomy of concepts) *A taxonomy of concepts consists of a hierarchical structure of concepts which are related with an* is-a *type of a relationship.*

Definition 3.2 (Hypernym) *Given a taxonomy of concepts, concept c_j is a hypernym of c_i if and only if c_i semantically entails c_j ($c_i \models c_j$).*

Definition 3.3 (Taxonomy path of concept) *Given a taxonomy of concepts, a taxonomy path P_{c_a} of c_a is an ordered sequence of concepts $P_{c_a} = \{c_a, c_{a+1}, \ldots, c_n\}$ where $c_i \models c_j, \forall i < j$ and c_n is the root concept of the taxonomy.*

Definition 3.4 (Taxonomy depth of concept) *Given a taxonomy path of concepts $P_{c_a} = \{c_a, c_{a+1}, \ldots, c_i, \ldots, c_n\}$, the taxonomy depth of concept c_i is the number of concepts from c_i to the root concept c_n in the path of concepts ($d_{c_i} = n - i$). By definition, the depth of the root concept is equal to zero.*

A piece of text can be generalized only when it contains *generalizable concepts* (Definition 3.5). A concept c_i with a taxonomy path P_{c_i} is said to have been generalized when it has been replaced by a concept $c_j \in P_{c_i}$ such that $d_{c_j} < d_{c_i}$. Accordingly, a text excerpt is said to have been generalized when it contains at least one generalized concept (Definition 3.6). The minimum taxonomy depth of a generalized concept constitutes the *level of generalization* of the given text (Definition 3.7).

Definition 3.5 (Generalizable concept) *A concept c_i of taxonomy depth d_{c_i} is said to be generalizable when at least one concept of its taxonomy path has a taxonomy depth less than d_{c_i}.*

Definition 3.6 (Generalizable text) *A text excerpt is said to be generalizable when it contains at least one generalizable concept.*

Definition 3.7 (Level of generalization) *The level of generalization of a text excerpt is equal to the minimum depth of its generalized concepts.*

3.1 Text generalization strategies

Given the above definitions, two novel strategies for text generalization are presented, which take into account the frequency of a concept in the source text. The intuition behind this transformation is the fact that machine learning systems tend to require a sufficient number of training samples prior to producing accurate predictions. Therefore, low-frequency terms should ideally be replaced by respective high-frequency hypernyms that semantically convey the original meaning.

Text generalization strategies are used to generalize both the training set (i.e. the articles and their respective summaries) as well as the test set (i.e. the unseen text). As it shall be described next, the machine learning model of Section 4 generates a generalized summary that is transformed to a readable text through the post-processing methodology of Section 5.

3.1.1 Named Entities-driven Generalization (NEG)

NEG only generalizes those concepts whose taxonomy path contains particular *named entities* (NEs) such as location, person and organization (Algorithm 1). For example, given the set of named entities $E = \{location, person\}$, the sentence *"John has been in Paris"* can be generalized to *"_person_ has been in _location_"*, where NEs

are enclosed in underscores in order to be distinguished from the corresponding words that may appear in the dataset.

Algorithm 1 requires: (i) the input $text$, (ii) the taxonomy of concepts T, (iii) the set C of tuples of extracted concepts c_i along with their respective taxonomy paths P_i and frequency f_i ($C = \{(c_1, P_1, f_1), (c_2, P_2, f_2), \ldots, (c_n, P_n, f_n)\}$), (iv) the set E of named entities ($E = \{e_1, e_2, \ldots\}$) and (v) the threshold θ_f of the minimum number of occurrences of a concept. In lines $2 - 4$ of Algorithm 1, a term can be generalized when both its frequency in the input text is less than the specified threshold θ_f and its taxonomy path P_i contains a named entity $c \in E$. In this case, c_i is replaced by its hypernym c (line 4). The output of the algorithm is a generalized version of the input text ($genText$). It should be noted that when $\theta_f = \infty$, the operation of the NEG algorithm resembles that of *named entity anonymization* (Hassan et al., 2018).

Algorithm 1 Named entities-driven text generalization (NEG)

Require: $text$, T, C, E, θ_f

1: $genText \leftarrow text$
2: **for all** $(c_i,\ P_i,\ f_i) \in C$ **do**
3: **if** $f_i \leq \theta_f$ **and** $\exists c \in P_i$ s.t $c \in E$ **then**
4: $genText \leftarrow$ replace c_i with c
5: **end if**
6: **end for**
7: **return** $genText$

3.1.2 Level-driven Generalization (LG)

LG generalizes the concepts according to the given level of generalization d (Definition 3.7), as illustrated in Algorithm 2. For instance, given the taxonomy of Figure 1 and $d = 1$, the sentence *"banana is nutritious"* may be generalized to *"food is nutritious"*.

Similarly to Algorithm 1, Algorithm 2 requires (i) the input $text$, (ii) the taxonomy T, (iii) the set of tuples C, (iv) the threshold θ_f and (v) the level of generalization d. In lines $6 - 25$, a term c_i is candidate for generalization when its frequency f_i is below the specified threshold θ_f (line 7). More specifically, c_i is replaced by its hypernym c_h (line 11) only when the depth d_{c_h} of the latter is at least equal to d (line 9).

When a term is generalized, the set of concepts C is either updated by merging c_i with its hypernym c_h (lines $14 - 18$) or a new entry is added in C, if c_h is not already a member of the set (lines $20 - 21$). Both the outer while-loop and the inner for-loop are terminated when no more generalization can be applied to the text because either the frequency of all concepts is greater than θ_f or all concepts have a taxonomy depth less or equal to d. In this case, the algorithm returns the generalized version of the input text (line 27) and terminates.

Algorithm 2 Level-driven text generalization (LG)

Require: $text$, T, C, d, θ_f

1: $genText \leftarrow text$
2: $inLoop \leftarrow$ **true**
3: **while** $inLoop$ **do**
4: $C_{new} \leftarrow C$
5: $inLoop \leftarrow$ **false**
6: **for all** $(c_i, P_i, f_i) \in C_{new}$ **do**
7: **if** $f_i \leq \theta_f$ **then**
8: $c_h \leftarrow$ hypernym of c_i from P_i
9: **if** $d_{c_h} \geq d$ **then**
10: $inLoop \leftarrow$ **true**
11: $genText \leftarrow$ replace c_i with c_h
12: $C \leftarrow C \setminus \{(c_h, P_h, f_h)\}$
13: **if** $\exists c_h \in C$ **then**
14: $P_h \leftarrow$ get P_h from C
15: $f_h \leftarrow$ get f_h from C
16: $f_{h_{new}} \leftarrow f_h + f_i$
17: $C \leftarrow C \setminus \{(c_h, P_h, f_h)\}$
18: $C \leftarrow C \cup \{(c_h, P_h, f_{h_{new}})\}$
19: **else**
20: $P_h \leftarrow$ get P_h from T
21: $C \leftarrow C \cup \{(c_h, P_h, f_i)\}$
22: **end if**
23: **end if**
24: **end if**
25: **end for**
26: **end while**
27: **return** $genText$

The strategies described above are not limited to a single text; they may also be applied to datasets of concatenated documents.

4 Deep learning model

After the text generalization phase outlined in the previous section completes, the summaries are produced by an encoder-decoder deep learning model, inspired from the "Sequence-to-sequence attentional model" (See et al., 2017). The en-

coder consists of a bi-directional LSTM (Graves et al., 2013), the decoder of a unidirectional LSTM and the attention mechanism employed is similar to that of Bahdanau et al. (2014). Words are represented using a *neural language model* like *word2vec* (Mikolov et al., 2013) and the overall model is trained on article-summary pairs. Once the training phase is over, the model is expected to predict an output vector of tokens $Y' = (y'_1, y'_2, ...)$ (summary) given an input vector of tokens $X = (x_1, x_2, ...)$ (text).

During training, the sequence of tokens (word embeddings) of the source text $X = (x_1, x_2, \ldots, x_n)$ is given to the encoder one-by-one in forward and reverse order, producing a hidden state $h_i = bi\_lstm(x_i, h_{i-1})$ for each embedding x_i. Then, the target sequence of tokens $Y = (y_1, y_2, \ldots, y_m)$ is given to the decoder, which learns to predict the next word y_t given the previous one y_{t-1}, the state of the decoder $s_t = lstm(s_{t-1}, y_{t-1}, c_t)$ and the context vector c_t, as computed by the attention mechanism. More specifically, the context vector c_t is computed as a weighted sum of the encoder hidden states h_i, according to the Equations 1-3 below

$$c_t = \sum_{i=1}^{|X|} a_{ti} h_i \quad (1)$$

$$a_{ti} = softmax(e_{ti}) \quad (2)$$

$$e_{ti} = tanh(W_h h_i + W_s s_{t-1} + b) \quad (3)$$

where a_{ti} is the weight, at each time step t, of the hidden state of the encoder h_i (i.e. a_{ti} indicates the importance of h_i), e_{ti} indicates how well the output of step t matches with the input around word x_i, s_{t-1} is the previous state of decoder, W_h, W_s and b are the weights and bias, respectively.

Summary prediction is achieved using *beam search* (Graves, 2012; Boulanger-Lewandowski et al., 2013); for each time step of the beam search-based decoder, the w candidate tokens with the highest log-probability are kept in order to determine the best output summary, where w is the beam width.

5 Post-processing of the predicted summary

Since the output of the deep learning model described in Section 4 is in generalized form, a post-processing technique for determining the specific meaning of each general concept is necessary. More specifically, a method should be devised that would match the generalized concepts of the predicted summary with the appropriate tokens of the original text.

Essentially, this is a problem of *optimal bipartite matching*, between the general concepts of the (generalized) summary and candidate concepts of the original text. To address this issue, Algorithm 3 is proposed, which performs the best matching based on the similarity of the context around the generalized concepts of the summary and the candidate concepts of the text.

Algorithm 3 Matching Algorithm

Require: $genSum$, $text$, T
1: $cr \leftarrow \{\}$ ▷ candidate replacements of generalized concepts
2: $gc \leftarrow \{\}$ ▷ generalized concepts
3: $summary \leftarrow genSum$
4: **for all** $token_s \in genSum$ **do**
5: **if** $token_s$ is generalized **then**
6: $gc \leftarrow gc \cup \{token_s\}$
7: **for all** $token_a \in text$ **do**
8: **if** $\exists c \in P_{token_a}$ s.t. $token_s = c$ **then**
9: $s \leftarrow$ similarity$(token_s, token_a)$
10: $cr \leftarrow cr \cup \{(token_s, token_a, s)\}$
11: **end if**
12: **end for**
13: **end if**
14: **end for**
15: sort cr in descending order of s
16: **for all** $(token_s, token_a, s) \in cr$ **do**
17: **if** $token_s \in gc$ **then**
18: $summary \leftarrow$ replace $token_s$ with $token_a$
19: $gc \leftarrow gc \setminus token_s$
20: **end if**
21: **end for**
22: **return** $summary$

Algorithm's 3 input is the generalized summary $genSum$, the original text $text$ and the taxonomy of concepts T. In the first loop (lines $4 - 14$), the similarity s between the context of each generalized token $token_s$ and each token $token_a$ of the source text that has a hypernym c similar to $token_s$ is computed (line 9) and the tuple $\{(token_s, token_a, s)\}$ is added to the set cr of candidate replacements of the generalized concepts (line 10). When all the generalized concepts of the (generalized) summary have been examined, cr is sorted in descending order according to s (line 15). In the second loop (lines $16 - 21$), $token_s$ is replaced by $token_a$ of maximum s (line 18) and is subsequently removed from gc (line 19). Eventually, Algorithm 3 returns the final summary $summary$ (line 22) in human-readable

form, which also contains specific information according to the source text.

Algorithm 3 works for both strategies of Section 3.1. In the LG strategy (Section 3.1.2), it is trivial to check whether $token_s$ exists in the taxonomy path of $token_a$ and therefore become candidate for replacement. In the case of NEG (Section 3.1.1), $token_s$ (e.g. a general concept of the summary such as *location* or *person*) may be replaced by a concept of the article, when the taxonomy path of the latter contains the former.

Finally, an important aspect affecting the performance of Algorithm 3 is the choice of the similarity function (line 9), which is a hyperparameter of the approach. Candidate similarity functions range from some well established indices like the cosine distance or the Jaccard coefficient to more complex measures like the word mover distance (Kusner et al., 2015) and the Levenshtein edit distance (Yujian and Bo, 2007). Of course, the optimal choice is highly dependant on the available data and we further reason on this subject on the experimental part of this submission.

6 Experiments & Results

The experimental methodology followed in this work is in accordance with some widely-adopted practices in the relevant literature (Rush et al., 2015; Nallapati et al., 2016; Chopra et al., 2016; See et al., 2017; Gao et al., 2019).

6.1 Datasets

Two popular datasets used in automatic TS tasks have been selected; `Gigaword` (Napoles et al., 2012) and `DUC 2004` (Over et al., 2007). The first dataset, `Gigaword`, is obtained as it is described by Rush et al. (2015) and further preprocessed in order to remove duplicate entries, punctuation and summaries whose length is either greater than or equal to the length of the articles they summarize. Moreover, the dataset has been normalized by expanding the contractions in the text (e.g. "I've" to "I have")[1]. After the completion of this step, the training set contains about 3 million article-summary pairs which consist of $99,224$ unique words (out of a total of 110 million words). The average article and summary length is 28.9 and 8.3 words, respectively. Finally, $4,000$ pairs have been selected randomly from the test set to form the validation set and another 4.000 pairs were also randomly selected to form the final test vectors as it is commonly done in the relevant literature (Rush et al., 2015; Nallapati et al., 2016; Chopra et al., 2016; Gao et al., 2019).

The `DUC 2004` dataset, on the other hand, contains 500 news articles and 4 human-generated summaries for each one of them. The same preprocessing methodology is applied to this dataset as well, but since it contains very few instances it is solely used for evaluation purposes (and not during model training). As it is a common practice in relevant experimental procedures, only the first sentence of the articles is used and the summaries are set to have a maximum length of 75 bytes (Rush et al., 2015; Nallapati et al., 2016; Gao et al., 2019).

6.2 Baseline and competitive approaches

The deep learning model outlined in Section 4 serves as the baseline approach. Its optimal hyperparameters are reported in the subsequent section; however, no generalization scheme is used. The baseline approach is tested on both datasets (`Gigaword` and `DUC 2004`).

Additionally, the results of some other approaches (ABS+ (Rush et al., 2015), RAS-Elman (Chopra et al., 2016), words-lvt5k-1sent (Nallapati et al., 2016) and GLEAM (Gao et al., 2019)) are also reported on the `DUC 2004` dataset. A direct comparison is possible, since the same evaluation methodology is adopted.

Such a direct comparison is not possible for the `Gigaword` dataset, due to the extra preprocessing steps of our approach and the random sampling of the testing data.

6.3 Parameter tuning

The methodology outlined in this work is dependant on a number of parameters and hyperparameters. Initially, the neural language model for the vector representation of words must be decided upon; after a brief experimentation with various representations and vector-spaces, pre-trained word2vec embeddings of size 300 were selected (Mikolov et al., 2013).

Following, a suitable similarity function for Algorithm 3 (line 8) must be specified. Several notions of word similarity have been considered, ranging from simple indices in-between single words (e.g. cosine similarity, Jaccard coefficient) to more advanced measurements like the word

[1]Expanding of contractions is performed by pycontractions package: https://pypi.org/project/pycontractions/

mover distance (Kusner et al., 2015) and the Levenshtein Edit distance (Yujian and Bo, 2007). The approach that achieved the best result was that of the combination of cosine similarity of averaged word2vec vectors and cosine similarity based on bag of words. In particular, the best performance was achieved when the windows around the candidate and the generalized concepts were set to 10 and 6, respectively.

The optimal hyper-parameters of the deep learning model (Section 4) have been determined to be as follows; The encoder (bi-directional LSTM) consists of two layers (of size 200 each), while the decoder (unidirectional LSTM) is single-layered, again of size 200. The batch size has been set to 64, the learning rate to 0.001 and the training data were randomly shuffled at each epoch. The employed optimization method has been the Adam algorithm (Kingma and Ba, 2014), with gradient norm clipping (Pascanu et al., 2013) and cross-entropy as the loss function (Golik et al., 2013). Finally, all words of the vocabulary have been considered in the training phase and a beam search of width equal to 4 has been used in the evaluation phase.

In order to assess the effect of the two generalization strategies discussed in Section 3.1, three distinct system configurations have been evaluated. The first one is the baseline approach of Section 6.2. The second system is an extension of the baseline, using NEG as the generalization methodology and the third one is also an extension of the baseline, employing the LG strategy.

Input text	police raided several locations near nobe after receiving word of a threat but no evidence of a planned attack was found
Generalized text	police raided several locations near _location_ after receiving word of a threat but no evidence of a planned attack was found
Generalized summary	police raided several locations near _location_
Output summary	police raided several locations near nobe

Table 1: An example of NEG strategy from the input text to the output summary

6.4 Procedure

As it has been discussed above, the experimental procedure includes three sets of experiments in total, with two of them based on the generalization strategies of Section 3.1. The *WordNet* taxonomy of concepts has been used (Miller, 1995; Fellbaum, 1998), out of which the hypernyms and the taxonomy paths have been extracted. To select the appropriate synset for extracting its taxonomy path, we use the *WordNet first sense*, as it has proved to be a very hard baseline in knowledge-based word sense disambiguation approaches (Raganato et al., 2017). Both generalization strategies are only applied to nouns in text, which are identified by the application of part-of-speech tagging and more specifically, the *Stanford log-linear* part-of-speech tagger (Toutanova et al., 2003).

Input text	for the second day in a row astronauts boarded space shuttle endeavour on friday for liftoff on nasa first space station construction flight
Generalized text	for the second day in a row astronauts boarded space equipment endeavour on friday for rise on nasa first space station construction flight
Generalized summary	astronauts boarded spacecraft for rise
Output summary	astronauts boarded spacecraft for liftoff

Table 2: An example of LG strategy from the input text to the output summary

The set of named entities used in NEG strategy (Section 3.1.1) is $E = \{Location, Person, Organization\}$, as the datasets contain news articles which are dominated by relevant entities. The named entities are extracted from the text using a *named entity recognizer* (NER) (specifically, the *Stanford* NER, Finkel et al., 2005) in conjunction with the *WordNet* taxonomy. Firstly, the pre-trained NER is executed and then the remaining named entities are extracted from *WordNet*; when a term in the text has a hypernym in the predefined set of named entities E, this word is annotated as a named entity. The performance of this generalization strategy is assessed for various thresholds of word frequency θ_f (as stated in the respective Section, a word is generalized only if its frequency in the dataset is less than θ_f).

The level of generalization (i.e. the taxonomy depth of a generalized concept) used in LG (Section 3.1.2) has been determined to be $d = 5$. This level has been chosen as the concepts become very general when $d < 5$, rendering the production of

Model	θ_f	ROUGE-1	ROUGE-2	ROUGE-L
NEG-100	100	45.95	23.52	43.30
NEG-200	200	46.20	23.86	43.45
NEG-500	500	**46.30**	**23.88**	**43.94**
NEG-1k	1000	46.14	23.31	43.35
NEG-infinity	∞	44.45	21.91	41.34
LG-100	100	**46.34**	**24.02**	**43.65**
LG-200	200	46.09	23.91	43.34
LG-500	500	46.04	23.64	43.25
LG-1k	1000	45.57	23.09	42.77
LG-infinity	∞	42.49	19.52	39.53
Baseline	-	44.35	22.43	41.87

Table 3: ROUGE scores on the `Gigaword` dataset.

the final summary a difficult task (Section 5). In a similar fashion to the NEG strategy, the performance of the LG approach is assessed for various thresholds of word frequency θ_f.

The overall architecture and all model configurations were trained on single Titan XP GPU[2]. Each training epoch took approximately 3.5 hours and all models converged around epoch 15.

Finally, the performance of all systems is measured on the official *ROUGE* package (Lin, 2004) of *ROUGE-1* (word overlap), *ROUGE-2* (bigram overlap) and *ROUGE-L* (longest common sequence). More specifically, for Gigaword testing data the F-measure of ROUGE score is reported while for the DUC dataset the evaluation metric is the standard ROUGE recall (Nallapati et al., 2016; Chopra et al., 2016; Gao et al., 2019).

Table 1 illustrates an example NEG approach which includes the input text, the generalized text (after the application of the NEG algorithm), the predicted generalized summary and the output summary (after post-processing the predicted summary). The underlined words are those that have been generalized and vice versa. Similarly, Table 2 outlines an example LG approach.

6.5 Results

Table 3 illustrates the ROUGE scores on the `Gigaword` dataset for both generalization strategies (NEG, LG) and various thresholds of word frequency θ_f. Similarly, Table 4 contains the ROUGE scores on the `DUC 2004` dataset. Apart from the NEG-infinity and LG-infinity configurations (which over-generalize), the other configurations of our model outperform the baseline approach on both datasets.

[2]Source code: https://github.com/pkouris/abtextsum

Intuitively, improved results were expected especially in the generalization of low-frequency words, as machine learning approaches typically require a sufficient number of samples in order to be trained properly. This is exactly the case for the LG strategy, as the best results are obtained when generalizing words that have at most 100 occurrences ($\theta_f = 100$) in the `Gigaword` dataset. Similarly, the best ROUGE-1 and ROUGE-2 scores for the LG strategy in the `DUC 2004` dataset are also obtained when $\theta_f = 100$. However, the NEG strategy exhibits its best performance at $\theta_f = 500$ on the `Gigaword` dataset and at $\theta_f = 1000$ on the `DUC 2004` dataset, with the exception of the ROUGE-2 metric which is maximized at $\theta_f = 500$.

Therefore, the LG strategy seems to be more fit in improving the performance of the deep learning system when generalizing low-frequency words. On the other hand, the NEG strategy has a positive effect on system performance, even though frequent words ($\theta_f \geq 500$) are generalized to the predefined named entities. This may be happening because most words describing named entities (especially those in E) have a specific function within the text and the reduction of their number (through the generalization to named entities) may lead to a more accurate prediction.

In both strategies, the configurations that generalize all concepts regardless of their frequency ($\theta_f = \infty$), exhibit the worst performance. In these cases of over-generalization, the deep learning model fails to learn the particular function of each word, as the generalized terms have a wide range of uses in the text. Another possible explanation of this failure is that the post-processing task of producing the final summary is not able

Model	θ_f	ROUGE-1	ROUGE-2	ROUGE-L
NEG-100	100	27.85	9.74	25.79
NEG-200	200	27.80	9.57	25.23
NEG-500	500	28.50	**10.07**	26.11
NEG-1k	1000	**28.73**	9.87	**26.12**
NEG-infinity	∞	27.33	9.01	24.41
LG-100	100	**28.89**	**10.10**	24.46
LG-200	200	28.68	9.84	25.76
LG-500	500	28.66	9.32	**25.77**
LG-1k	1000	28.40	9.21	25.43
LG-infinity	∞	26.49	7.89	23.72
Baseline	-	27.56	8.90	25.20
ABS+	-	28.18	8.49	23.81
RAS-Elman	-	28.97	8.26	24.06
words-lvt5k-1sent	-	28.61	9.42	25.24
GLEAM	-	29.51	9.78	25.60

Table 4: ROUGE scores on the DUC 2004 dataset.

to accurately match the generalized concepts with specific words, due to a large amount of the former. Obviously, a trade-off exists between θ_f and the obtained performance.

The last lines of Table 4 also exhibit that the best NEG and LG configurations outperform the other systems in terms of the ROUGE-2 and ROUGE-L scores and demonstrate a near-optimal performance when the ROUGE-1 score is considered, thereby indicating the robustness of the proposed methodology on the DUC 2004 dataset. In case of the Gigaword dataset, the further preprocessing of data has led to a significant performance improvements, especially in comparison to previous work (Chopra et al., 2016; Gao et al., 2019). Even though the aforementioned steps have resulted in more informative and accurate summaries, they do not permit a direct comparison with previously reported results.

7 Conclusion and Future Work

Even though deep learning approaches have been widely used in abstractive TS, it is evident that their combination with semantic-based or structure-based methodologies needs to be more thoroughly studied. In this direction, the proposed novel framework combines deep learning techniques with semantic-based content methodologies so as to produce abstractive summaries in generalized form, which, in turn, are transformed into the final summaries. The experimental results have demonstrated that the followed approach enhances the performance of deep learning models.

The positive results may be attributed to the optimization of the parameters of the deep leaning model and the ability of the method to handle OOV and very low frequency words. The obtained results show that the proposed approach is an effective methodology of handling OOV or rare words and it improves the performance of text summarization.

Of course, certain aspects of the proposed methodology could be extended. Since currently only nouns are considered for generalization, an expansion to verbs could result in additional improvement. Moreover, as the ambiguity is a challenging problem in natural language processing, it would be interesting to capture the particular meaning of each word in the text so that our methodology manages to uncover the specific semantic meaning of words. Finally, the distinct semantic representation of each word could further enhance the performance of the deep learning model.

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D Trippe, Juan B Gutierrez, and Krys Kochut. 2017. Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*.

S Alshaina, Ansamma John, and Aneesh G Nath. 2017. Multi-document abstractive summarization based on

predicate argument structure. In *Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2017 IEEE International Conference on*, pages 1–6. IEEE.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Riadh Belkebir and Ahmed Guessoum. 2016. Concept generalization and fusion for abstractive sentence generation. *Expert Systems with Applications*, 53:43–56.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2013. Audio chord recognition with recurrent neural networks. In *ISMIR*, pages 335–340. Citeseer.

Qian Chen, Xiao-Dan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling document. In *IJCAI*, pages 2754–2760.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98.

Christiane Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mahak Gambhir and Vishal Gupta. 2017. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66.

Yang Gao, Yang Wang, Luyang Liu, Yidi Guo, and Heyan Huang. 2019. Neural abstractive summarization fusing by global generative topics. *Neural Computing and Applications*.

Pierre-Etienne Genest and Guy Lapalme. 2011. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 64–73. Association for Computational Linguistics.

Pavel Golik, Patrick Doetsch, and Hermann Ney. 2013. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech*, volume 13, pages 1756–1760.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

Som Gupta and S. K Gupta. 2019. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49 – 65.

Fadi Hassan, Josep Domingo-Ferrer, and Jordi Soria-Comas. 2018. Anonymization of unstructured data via named-entity recognition. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 296–305. Springer.

Monika Joshi, Hui Wang, and Sally McClean. 2018. Dense semantic graph and its application in single document summarisation. In *Emerging Ideas on Information Filtering and Retrieval*, pages 55–67. Springer.

Atif Khan, Naomie Salim, Haleem Farman, Murad Khan, Bilal Jan, Awais Ahmad, Imran Ahmed, and Anand Paul. 2018. Abstractive text summarization based on improved semantic graph approach. *International Journal of Parallel Programming*, pages 1–25.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, pages 957–966.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. *arXiv preprint arXiv:1805.03989*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Ibrahim F Moawad and Mostafa Aref. 2012. Semantic graph reduction approach for abstractive text summarization. In *Computer Engineering & Systems (ICCES), 2012 Seventh International Conference on*, pages 132–138. IEEE.

N Moratanch and S Chitrakala. 2016. A survey on abstractive text summarization. In *Circuit, Power and Computing Technologies (ICCPCT), 2016 International Conference on*, pages 1–7. IEEE.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

Shengli Song, Haitao Huang, and Tongxiao Ruan. 2018. Abstractive text summarization using lstm-cnn based deep learning. *Multimedia Tools and Applications*, pages 1–19.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1091–1095.

Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2016. Abstractive cross-language summarization via translation model enhanced predicate argument structure fusing. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(10):1842–1853.