Journal of Big Data

RESEARCH Open Access

Text summarization based on semantic graphs: an abstract meaning representation graph-to-text deep learning approach

Panagiotis Kouris[1*], Georgios Alexandridis[1] and Andreas Stafylopatis[1]

*Correspondence:
pkouris@islab.ntua.gr

[1] School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece

Abstract

Nowadays, due to the constantly growing amount of textual information, automatic text summarization constitutes an important research area in natural language processing. In this work, we present a novel framework that combines semantic graph representations along with deep learning predictions to generate abstractive summaries of single documents, in an effort to utilize a semantic representation of the unstructured textual content in a machine-readable, structured, and concise manner. The overall framework is based on a well defined methodology for performing semantic graph parsing, graph construction, graph transformations for machine learning models and deep learning predictions. The employed semantic graph representation focuses on using the model of abstract meaning representation. Several combinations of graph construction and graph transformation methods are investigated to specify the most efficient of them for the machine learning models. Additionally, a range of deep learning architectures is examined, including a sequence-to-sequence attentive network, reinforcement learning, transformer-based architectures, and pre-trained neural language models. In this direction, a semantic graph representation of an original text is extracted, and then the present framework formulates the problem as a graph-to-summary learning problem to predict a summary of an original text. To the best of our knowledge, this formulation of graph-to-summary predictions in abstractive text summarization, without other intermediate steps in the machine learning phase, has not been presented in the relevant literature. Another important contribution is an introduction of a measure for assessing the factual consistency of the generated summaries in an effort to provide a qualitative evaluation. To assess the framework, an extensive experimental procedure is presented that uses popular datasets to evaluate key aspects of the proposed approach. The obtained results exhibit promising performance, validating the robustness of the proposed framework.

Keywords: Abstractive text summarization, Semantic graph, AMR graph summarization, Graph-to-text, Deep learning, Factual consistency

Introduction

The domain of automatic text summarization (TS) [1, 2] remains an active research field for more than half a century [3, 4] due to the positive perspectives, primarily in reducing the reading time or minimizing the processing effort of the large and constantly increasing amount of textual data that is available today. Automatic TS is considered a challenging task because it aims at representing an original text in an abridged form while retaining the important information of the initial content. The purpose of automatic TS is to generate a short version of the original text that is informative and also coherent, covering the topics of the initial content and eliminating redundancy.

Some of the initial approaches in abstractive TS represented the original text into a graph and then composed a summary from the graph representation [5, 6]. These approaches construct summaries by merging or discarding nodes, retaining the most important of them, to obtain a more concise version of the original graph (i.e., a graph with smaller dimensionality than the original one). This graph constitutes an intermediate step that is used to obtain the final summary in textual form. As a result, the graph representation encodes the sequence of words of a text, with the common words among sentences to be represented by common nodes in the graph (i.e., nodes that are common for more than one sentence). In such cases, transforming in-between graph and textual form is a straightforward task. The drawback, however, lies within the fact that a large initial graph is created, which is difficult to be processed for computing the final summary. Also, these graphs do not incorporate any semantic representation other than syntactic and grammatical relationships. This constitutes a weakness of these approaches, as they ignore the semantic aspects of a text.

To overcome the aforementioned problem, an alternative graph representation of text has been proposed, which incorporates semantic relations and uses a kind of subtraction in order to reduce the size of the original graph. Such a perspective could be achieved by using semantic graphs, such as the abstract meaning representation (AMR) graphs [7, 8] that focus on a semantic representation of a text in a concise form. AMR graphs, which are described in "Preliminaries for semantic graph-based representation", strive for more semantic representation and less syntactic or grammar structure of a sentence.

The present framework constitutes an attempt of combining machine learning and semantic-based techniques in the field of automatic TS. Such a combination should be examined because most research focuses either on machine learning approaches or on knowledge-based and semantic-based techniques, treating them as separate methodologies [9]. In the direction of combining aspects and characteristics of the aforementioned domains, some recent work has been done [10–14]. The most relevant of them, such as [10, 11, 13], which are based on AMR graph representation, are described in "Preliminaries and related work". To further investigate such a combination, the proposed framework tries to utilize machine learning and semantic graph representation for contributing to the domain of abstractive TS.

More specifically, the present work focuses on improving the semantic graph-based approaches of single-document abstractive TS, because of the positive perspectives these approaches present, such as semantic representation in a structured, machine-readable and concise version of the content. Furthermore, these methodologies detect redundant information and produce semantically related sentences in the summary. Recent works in this domain utilizes AMR graphs, forming a new field [15, 16] that requires further research, as this aspect of utilizing semantic-based representations in abstractive TS has not been studied sufficiently. It should be noted that the AMR graph-based approaches still exhibit reduced performance when compared to deep learning-based ones [17] (i.e., deep learning techniques that do not employ any semantic-based representation). However, the semantic graph-based approaches are constantly improving, according to the relevant literature [11, 13, 18–22]. In this respect, the methodology described in the current work falls in the category of AMR graph-based approaches, as it improves the performance of a relevant system. In "Preliminaries and related work" that follows, we overview the relevant literature, outlining the differences and merits of our proposed approach.

Additionally, a framework based on semantic graph-to-text machine learning predictions is presented and valuable features and aspects of semantic graphs are examined, with respect to their suitability for abstractive TS. In particular, we investigate if a semantic graph representation of an original text, as an intermediate step, could be an efficient factor for estimating a summary. More specifically, the proposed approach constructs a summary in two steps; (*i*) retrieving a semantic graph of an original text, and (*ii*) using an obtained semantic graph as input to predict a summary in textual form. In the first step, an efficient AMR parser could be used in order to make the semantic graphs available for the next step. In the second step, we formulate the problem in a graph-to-text learning form, utilizing machine learning techniques for estimating a summary of an original text.

Within the above framework, the main contribution of this work is four-fold; (*i*) the graph-to-summary formulation of the problem of abstractive TS using deep learning techniques, (*ii*) the examination of a range of deep learning models, (*iii*) the investigation of semantic graph-based representation schemes and (*iv*) the introduction of a set of evaluation metrics, aiming at a qualitative evaluation in automatic TS.

To the best of our knowledge, none of the aforementioned points has been studied in the relevant literature so far. Graph-to-text or semantic graph-to-summary predictions with deep learning models (i.e., models whose input is only a graph representation and they predict a summary in textual form, without other intermediate steps) have yet to be considered in abstractive TS. However, as it is described in more detail in "Preliminaries and related work", some approaches have been proposed in the relevant literature that are based on a pipeline of converting a semantic graph of an input text into a semantic graph of its summary and then, the summary graph is used for obtaining the respective summary in textual form [18–21]. Moreover, other approaches use the input text along with its semantic representation to produce a summary, incurring an additional computational cost [10, 11]. In contrast, the present work introduces a semantic graph-to-text solution which outperforms the aforementioned approaches ("Results"), avoiding the additional computational cost of using the input text in the machine learning phase and limiting the information loss of the successive conversions within the pipeline. Although AMR graph-to-text generators have already been implemented [23, 24], graph-to-summary approaches for obtaining a summary from a semantic graph have not been studied yet in the field of abstractive TS. Additionally, we propose and examine a range of deep learning architectures, investigating the versatility of the methodology and the merits or drawbacks of the different neural-based models. The deep learning approaches include recurrent neural networks in encoder-decoder architecture, reinforcement learning (RL), transformers, and pre-trained language models. Moreover, we investigate various semantic graph representations as input for a deep learning model, identifying the most efficient of them. The employed deep learning models along with the proposed semantic graph representations, have not been studied yet in the framework of AMR graph-based approaches.

Another novelty of the current work is a set of evaluation metrics proposed for automatic TS. In particular, we introduce a set of metrics for assessing the factual consistency of the predicted summaries, in an effort to provide a qualitative evaluation ("Factual consistency"). More specifically, an extensive experimental procedure has been conducted, estimating various and significant aspects of the proposed framework, using two popular datasets. The experimental results and the performance comparison between the proposed approach and relevant works validate the efficiency of the proposed methodology.

The rest of this paper is organized as follows; "Preliminaries and related work" introduces the background along with the notation of the semantic graphs and provides an overview of the relevant literature. "The proposed framework" presents the proposed framework including the components of semantic graph parsing ("Semantic graph parsing"), semantic graph construction ("Semantic graph construction"), semantic graph transformations ("Graph transformations for machine learning") and deep learning prediction ("Deep learning prediction"). "Experiments" describes the experimental procedure, where the evaluation metrics are presented and the measures for assessing the factual consistency are introduced. "Results" presents the obtained results, which are discussed in "Discussion". Finally, "Conclusion" concludes this work, with some final remarks and future work directions.

Preliminaries and related work

Semantic graph-based text summarization

Preliminaries for semantic graph-based representation

Semantic graph-based representation Since the standard input in single-document TS are unstructured textual data, their transformation into a graph representation may be beneficial for handling the system input to estimate a summary. A graph-based text representation captures the content of a text in a structured, concise and machine-readable format. In this work, as it shall be explained below, we assume semantic graphs that represent the unstructured input text. Firstly, we examine the conceptual graphs as a general formalism of semantic graphs. Secondly, we focus on a particular formalism of AMR that, also, constitutes a semantic graph representation, which is employed for evaluating our framework ("Experiments"). In the context of knowledge and semantic representation, and reasoning in the artificial intelligence domain [25, 26], we make an introduction to the semantic graphs, as our framework focuses on representing textual information in this form.

Conceptual graphs Conceptual graphs, which are based on first-order logic, facilitate a mapping either from natural language to graphs or from graphs to natural language [26, 27]. As it is described in Definition 1, a conceptual graph *CG* is comprised of concept

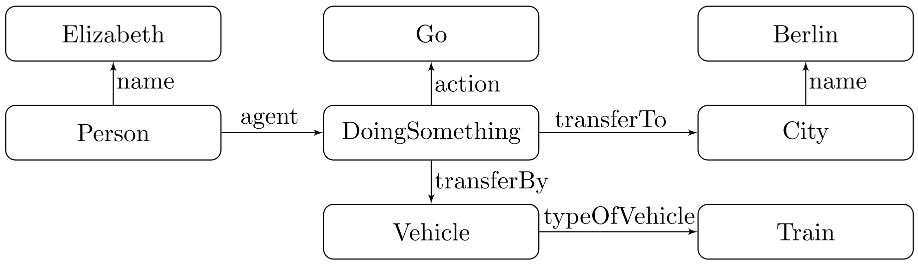

Fig. 1 The conceptual graph of the sentence *"Elizabeth is going to Berlin by train"*

types *C*, relations *R* and individuals *I*, giving semantics of natural language. The concept types and individuals are represented by nodes and the relations denote the edges among nodes. More specifically, concept types may be abstract classes, objects or entities. It is obvious that the individuals are specific values or names that correspond to concept types, such as *yellow* may be an individual of concept type *color* or *John* may constitute an individual of concept type *person*.

Definition 1 [Conceptual Graph] A conceptual graph is a semantic mapping from a text T to a graph $CG = \{N, E\}$ that is comprised of a set of nodes $N = C \cup I$ and a set of edges $E = R$ among nodes, where C denotes a set of concept types, R is a set of relations and I is a set of individuals.

A conceptual graph may represent the content of a text by using two notations; logic, or graph representation. To better illustrate this point, Example 1 represents a mapping of a sentence to a conceptual graph in the aforementioned two formats.

Example 1 (Conceptual Graph representation) Given the sentence *"Elizabeth is going to Berlin by train"*,

(i) The conceptual graph is represented in Fig. 1, where the classes *Person, DoingSomething* and *City* correspond to the specific individuals *Elizabeth, Go* and *Berlin*, respectively. Also, the nodes of the graph are connected by particular relations and the class *Train* may be assumed as a sub-class of *Vehicle* that is a leaf of the graph, which is not related to an individual.

(ii) The conceptual graph in logic format may be as following:

$$\exists p, d, c, e, g, b, v, t : instance(p, Person) \wedge instance(d, DoingSomething) \wedge instance(c, City) \wedge instance(e, Elizabeth) \wedge instance(g, Go) \wedge instance(b, Berlin) \wedge instance(v, Vehicle) \wedge instance(t, Train) \wedge name(p, e) \wedge agent(p, d) \wedge action(d, g) \wedge transferTo(d, c) \wedge name(c, b) \wedge transferBy(d, v) \wedge typeOfVehicle(v, t)$$

As we can see above, in logic format, we utilize variables for denoting the entities or individuals.

Abstract meaning representation AMR constitutes a semantic representation language, which mainly captures the question *who is doing what to whom* in a sentence [7]. This is a question that may be replied easily by humans, but it is very difficult for machines to analyze. Moving from the analysis of the syntactic structure to the semantics of a sentence, AMR focuses on representing it in a particular semantic graph. More specifically, AMR is a rooted, directed acyclic graph (DAG) with labelled edges and nodes, which represents the meaning of a sentence. AMR graphs are both machine-readable format and easy for people to understand. Nevertheless, AMR is closer to English than other languages and it is not an interlinguistic format. Moreover, the same AMR may be assigned to different sentences with the same meaning (e.g., the sentences "John wants his friend to believe him" and "John has a desire to be believed by his friend" may have the same AMR representation). In other words, an AMR graph may be translated into different sentences, all of which have the same meaning. Furthermore, AMR includes more logical than syntactic representation, dropping inflectional morphology for tenses, as well as omitting articles and punctuation.

Example 2 illustrates the usage of AMR, representing a sentence in three notations; logic, graph and AMR format that is based on PENMAN project [28]. An AMR graph includes nodes denoted by variables (e.g. a, b), which represent semantic concepts for entities, events, properties and states. The nodes are connected by relations forming a rooted DAG. AMR concepts may be English words (e.g., person), PropBank framesets [29] (e.g., want-01) or special keywords that include entity types (e.g., date-entity, world-region), quantities (e.g., distance-quantity) and logical conjunctions (e.g. and). AMR uses relations that follow the conventions of the PropBank project (e.g., :ARG0, :ARG1, :ARG2). Moreover, it uses approximately 100 additional relations for quantities (e.g., :quant, :unit), date-entities (e.g., :day, :month, :year, :time), lists (e.g., :op1, :op2, :op3) and general semantic relations (e.g., :age, :cause, :compared-to, :consist-of, :domain, :location, :name, :polarity, :topic, etc.). Also, AMR includes an inverse for each relation (e.g., :ARG0-0f, :location-of). Details and guidelines about the AMR project are provided in [30] or in the revised version of the AMR specifications[1]. Finally, a set of annotated AMR examples of natural language sentences in the English language is available in [31].

Example 2 (AMR representation)

Given the sentence *"Mary wants Jennifer to believe her"*,

(i) the AMR graph is presented in Fig. 2.

(ii) The AMR representation in logic format is:

$$\exists w, b, p1, p2, n1, n2 : instance(w, want\text{-}01) \wedge instance(p1, person) \wedge instance(n1, name) \wedge instance(b, believe\text{-}01) \wedge instance(p2, person) \wedge instance(n2, name) \wedge ARG0(w, p1) \wedge name(p1, n1) \wedge op1(n1, \text{"}Mary\text{"}) \wedge ARG1(w, b) \wedge ARG0(b, p1) \wedge ARG1(b, p2) \wedge name(p2, n2) \wedge op1(n2, \text{"}Jennifer\text{"})$$

[1] AMR 1.2.6 Specification: https://github.com/amrisi/amr-guidelines/blob/master/amr.md

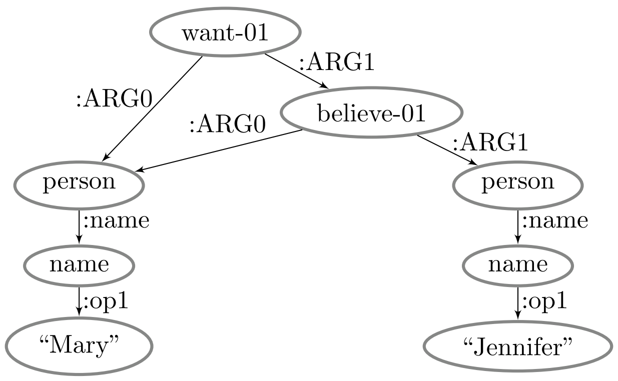

Fig. 2 The AMR graph of the sentence *"Mary wants Jennifer to believe her"*

(iii)The AMR in text format is as follows:

$$\begin{aligned}
&(w\ /\ want\text{-}\ 01\\
&\quad :ARG0\ (p1\ /\ person\ \ :name\ (n1\ /\ name\ \ :op1\ "Mary"))\\
&\quad :ARG1\ (b\ /\ believe\text{-}\ 01\\
&\qquad :ARG0\ (p2\ /\ person\ \ :name\ (n2\ /\ name\ \ :op1\ "Jennifer"))\\
&\qquad :ARG1\ p1))
\end{aligned}$$

This work focuses on the AMR semantic formalism, where its utilization constitutes a perspective that is examined in the research fields of natural language processing and especially in TS.

Semantic graph-based related work

Previous research in the field of single-document TS has shown progress in utilizing semantic representations of text. In [32], authors propose a method that is based on conceptual graphs [27], which focuses on ranking and pruning nodes in order to select the most important of them. Also, they use weighted conceptual graphs and semantic patterns from WordNet [33, 34] and VerbNet [35] to keep coherent structures, identifying the particular concepts that correspond to a summary. In [36], authors employ AMR and other lexical resources of WordNet and PropBank [29] in order to select the most important concepts that generate an abstractive summary. This approach combines discursive information of rhetorical structure theory (RST) [37] and the well-known PageRank algorithm [38] to identify the most relevant concepts, composing a summary by using the SimpleNLG tool [39] for natural language generation. Instead of the aforementioned approaches, which use heuristic algorithms for reducing the dimensionality of an initial graph to compose a summary, our work employs conceptual graphs that are given as input to machine learning models in order to generate summaries.

Some approaches form a pipeline by using the AMR representation as an intermediate step, where an input text is transformed into an AMR graph which is used for estimating a summary graph that, in turn, generates the final summary in textual form [18–21]. More specifically, in [19], the authors propose a framework for creating an AMR graph of a document by combining the sub-graphs of each sentence and transforming them into a single graph. The approach focuses on graph-to-graph prediction, formulating this problem with integer linear programming along with supervised learning for parameter estimation in order to predict the summary graph. This approach does not include any AMR-to-text generation of the summary in textual form. In a similar approach [18], the authors propose a pipeline for selecting the most important sentences of a text, extracting an AMR sub-graph for each selected sentence. Moreover, in [20], the authors propose an algorithm that extracts a semantic graph of a document by using a co-reference resolution to merge the sub-graphs of the sentences. Then, a summary graph is created by finding the most important relations between nodes, capturing the surrounding information of each path in the graph. Furthermore, in [21], a co-reference resolution is performed before retrieving the AMR graphs of the sentences. Also, an algorithm for merging the AMR graphs is proposed to form the summary graph. To generate the summary, the last three approaches described above, [18, 20] and [21], use an available AMR-to-text generator [23, 24]. All of the aforementioned approaches employ a series of transformations; from text to text graph, from text graph to summary graph, and from summary graph to summary, increasing the overall computational load. Also, these transformations may incur a significant information loss between each step. In contrast to the pipeline techniques, the proposed solution in this work utilizes the benefits of a concise semantic representation of an original text, as the above approaches do in the first pipeline step, to generate the summary in textual form directly from a conceptual graph, without any other transformation. In particular, our approach copes with the problem of graph-to-summary predictions, avoiding further transformations that may lead to a reduced performance.

Since AMR graph construction and transformations are required, in [22], the authors present an analysis and methods of constructing AMR graphs for summarization. The authors examine merge-based strategies and the performance of content selection methods. Also, they propose a method of merging the nodes by combining co-reference resolution with concept merging. In a similar direction [40], authors propose an approach for merging AMR sentence graphs into a single document graph, utilizing named entities and pronominal nodes. In this work, in the case of multi-sentence document summarization, we examine particular techniques of merging the semantic graphs of the sentences of a document in order to obtain an overall document graph.

Semantic graph-based machine learning text summarization

Preliminaries for semantic graph-based machine learning

Machine learning models have been developed for generating summaries of text [41]. More specifically, such models predict an output sequence of tokens $Y' = (y'_1, y'_2, ...)$ (the summary), given an input sequence of tokens $X = (x_1, x_2, ...)$ (the candidate text for summarization). In the case of combining semantic graph-based representation with machine learning predictions, a typical procedure is as follows; given an input sequence of tokens $X = (x_1, x_2, ...)$ (i.e., an input text) a semantic graph representation $G = (g_1, g_2, ...)$ is obtained (representing the original text). Then a deep learning model, which is trained on semantic graph-summary pairs, is used for predicting a sequence of tokens $Y' = (y'_1, y'_2, ...)$ that corresponds to a summary. An overview of the relevant literature about semantic graph-based machine learning approaches follows next. The particular methodology, used in the current work, that combines semantic representation with deep learning predictions is introduced in "The proposed framework", where the proposed framework is presented. Additionally, the deep learning models employed to generate a summary of a given semantic graph, are described in detail in "Deep learning prediction".

Semantic graph-based machine learning related work

In the area of combining semantic representation with machine learning predictions, which constitutes the field of the present work, authors in [10] extend the attention-based neural approach of [42] for abstractive TS, which is used as a baseline model, by incorporating an AMR encoder based on a tree-LSTM architecture [43]. In this approach, a two-step training scheme is applied. In the first step, the baseline model is trained using text-summary pairs, and in the second phase, the obtained AMR graphs of text are used for further training and adapting the parameters of the employed AMR encoder, which improves the performance of the initial model. This approach combines semantic and syntactic aspects of the original text, improving the generated summaries. In [11], the authors propose an approach to guide the natural language generation process using particular information from an original text, in order to generate its summary from its AMR graph. The approach is based on a standard sequence-to-sequence model [44] to estimate a summary of an AMR graph. More precisely, information from an original text that does not exist in an AMR graph is utilized so as to improve the quality of a summary. Additionally, in [13], authors propose an encoder-decoder architecture that takes as input an original text along with its semantic graph to predict a summary. The aforementioned approaches are in line with our work because they are based on machine learning predictions to estimate a summary based on the AMR representation of an original text. In this direction, our approach examines a range of machine learning models, including sequence-to-sequence learning, reinforcement learning (RL), and transformers, in an effort to further improve the produced summaries, as explained. Finally, we note that in contrast to [11] and [13], our deep learning models receive as input only a graph representation of an original text (i.e. the source text is not required in the machine learning process) to generate a summary.

The proposed framework

The proposed framework is illustrated in Fig. 3. The input is comprised of a single-document *text*, while the output is a *summary* of the original text. Its main components are four, starting with semantic graph parsing, whose purpose is to retrieve a conceptual graph for each sentence of the original text. The set of semantic graphs is used in the second component for constructing a graph of the overall text. The third component transforms the text graph into an appropriate form for using in machine learning predictions. The last step includes machine learning predictions, where a deep learning model, having been trained on a corpus of graph-summary pairs, predicts a summary for a new input that has been given. In this section, we introduce the task of semantic graph parsing, define the semantic graph construction and graph transformation processes and present the phase of deep learning prediction.

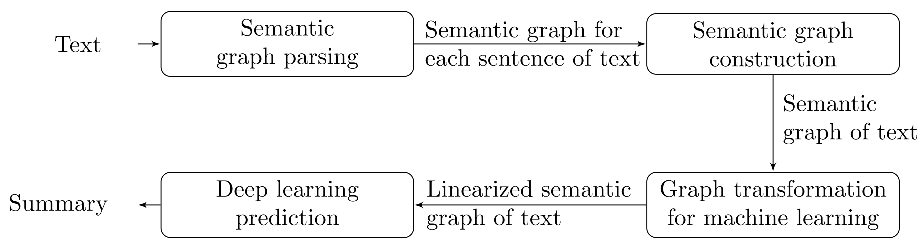
Text
Semantic graph parsing
Semantic graph for each sentence of text
Semantic graph construction
Semantic graph of text
Graph transformation for machine learning
Linearized semantic graph of text
Deep learning prediction
Summary

Fig. 3 The workflow of the proposed framework

Semantic graph parsing

According to the task of semantic graph parsing, given a sentence $S = (w_1,\ w_2,\ \dots\ w_n)$, this task aims at extracting a semantic representation of this sentence to a graph $G = (N, E)$, where N denotes a set of concepts that correspond to the nodes of the graph and E is a set of edges among the nodes. Such a graph may be represented by a set of triplets $G = \{(u_1, u_2, e_{12}),\ \dots,\ (u_i, u_j, e_{ij})\}$, where $u_i \in N$ and $u_j \in N$ represent semantic nodes and $e_{ij} \in E$ denotes an edge between two nodes u_i and u_j. Essentially, the process of semantic graph parsing is based on a function (f_p) that performs a mapping between a text (T) and its particular graph ($f_p : T \to G$).

To demonstrate the process of semantic graph parsing, we assume that we have a text T that includes a sequence of sentences $T = (S_1,\ S_2,\ \dots,\ S_k)$, where each sentence is comprised of a sequence of words $S_i = (w_1,\ w_2,\ \dots,\ w_n)$. The process aims at extracting a semantic graph for each sentence of the text. Therefore, given a sequence of sentences of a text, the process returns a sequence of semantic graphs $\mathsf{G}_\mathsf{T} = (G_1,\ G_2,\ \dots,\ G_n)$. This step of graph parsing retrieves a sequence of semantic graphs for each text that is given to the next component of the framework, which constructs a particular graph for each text, as we explain in detail below.

To represent a text to a semantic graph, we adopt the semantic formalism of AMR graphs ("Preliminaries for semantic graph-based representation"). For performing AMR parsing, several parsers are available to be used [23, 24, 45, 46]. Therefore, in the experimental procedure that is conducted to evaluate the present framework, we use an already developed AMR parser as it is described in "Experiments".

Semantic graph construction

The process of parsing, which has been discussed above, aims at retrieving a semantic graph for each sentence of a text. Having these sub-graphs of a text, we examine two strategies for constructing the semantic graph of the text. In the first strategy, we assume the semantic graph as a sequence of the sub-graphs, and in the second strategy, we assume the semantic graph as a combination of the sub-graphs.

Semantic graph as a sequence of sub-graphs

In this strategy, the overall graph of a text G_T is comprised of a tuple of the sequence of sub-graphs G_i of a text so that $\mathsf{G}_\mathsf{T} = (G_1,\ G_2,\ \dots,\ G_n)$, where G_i with $i \in (1,\ 2,\ \dots\ n)$ represents the semantic sub-graph that correspond to sentence S_i of the text. The sub-graphs remain independent of each other (i.e., individual graphs without any edge among their nodes) and all together represent the overall text in a sequence of sub-graphs.

Semantic graph as a combination of sub-graphs

According to this strategy, we combine the sub-graphs, which represent the sentences of a text, in order to construct a rooted semantic graph of this text. This strategy achieves compression of the content reducing or eliminating the redundancy before estimating and producing a summary in text form, utilizing machine learning predictions. We could express the resulting combined graph as a union of the sub-graphs as follows $\mathsf{G_C} = G_1 \cup G_2 \cup ... \cup G_n$.

More specifically, to combine two or more semantic sub-graphs, we follow a similar approach of [19] and in addition, we utilize the resource description framework (RDF) [47] as a graph representation to design the Algorithm 1. This Algorithm provides a systematic way of achieving a combination of sub-graphs to produce a semantic graph of an overall text with multi-sentences.

Algorithm 1 Combination of sub-graphs of a multi-sentence text to produce a semantic graph

```
Require: $G_T$
1: $RDF_C \leftarrow \{\}$
2: for all $G_i \in G_T$ do
3:     $G_i \leftarrow$ Add root node to $G_i$
4:     $RDF_i \leftarrow$ RDF triplets of $G_i$
5:     $RDF_C \leftarrow RDF_C \cup RDF_i$
6: end for
7: $G_C \leftarrow$ Construction of the combined semantic graph using $RDF_C$
8: return $G_C$
```

In particular, Algorithm 1 takes as input a tuple of the individual semantic sub-graphs of a text ($\mathsf{G_T} = (G_1, G_2, ..., G_n)$, where G_i is a sub-graph that represent the sentence S_i of a text). The procedure starts by initializing the set of the combined RDF triplets (RDF_C) to an empty set (line 1). Then, the algorithm, in the loop of lines 2-6, examines the sub-graphs in G_T by adding a root node to each sub-graph (line 3) and retrieving the RDF triplets for each sub-graph G_i assigning them to the set RDF_i (line 4), which represent the set of RDF triplets of the current sub-graph. The combined set of RDF triplets, which is denoted by RDF_C, results from the union of the current sets RDF_C and RDF_i (line 5). In line 7, the algorithm constructs the combined semantic graph G_C by using the RDF triplets of the set RDF_C. Finally, the algorithm returns the overall semantic graph G_C as a combination of sub-graphs for an input text (e.g., in *AMR* format).

It should be clarified that the obtained semantic sub-graphs of a text may not include any connection among them (i.e., without any common node due to a lack of overlapping information among sentences). For this reason, we add a root node to each sub-graph which, also, constitutes the root node of the overall combined semantic graph that is created, as it is illustrated in Example 3 bellow.

Example 3, which follows, further illustrates the methodology of constructing a semantic graph. In this example, we suppose that we have a text with two sentences retrieving from them a semantic graph that represents the content of this text.

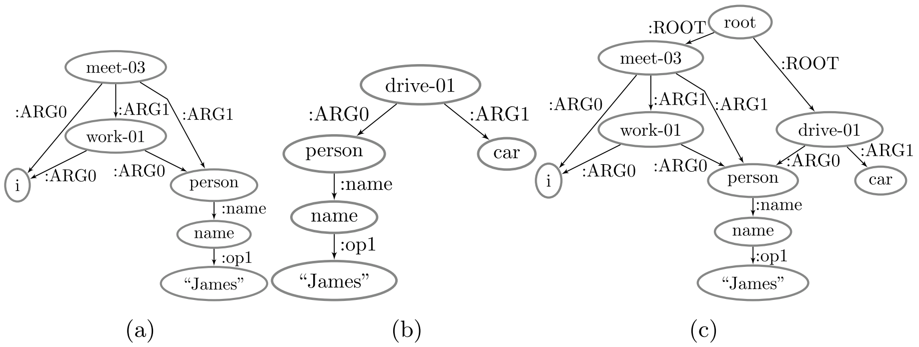

Fig. 4 **a** The AMR graph of the sentence *"I met James, who was going to work"*, **b** the AMR graph of the sentence *"James was driving his car"*, and **c** the combined AMR graph of the sentences *"I met James, who was going to work"* and *"James was driving a car"*

Example 3 (Semantic graph as a combination of sub-graphs)

Given the sentences S_1 and S_2 of a text:

S_1: *I met James, who was going to work.*
S_2: *James was driving a car.*

The AMR graphs of the sentences S_1 and S_2 are presented in Fig. 4a and b, respectively.

According to algorithm 1, the RDF triplets of the first sentence (RDF_{S_1}) and the second sentence (RDF_{S_2}) are as follows.

$$\begin{aligned} RDF_{S_1} = \{ & (root,\ :ROOT,\ meet-03), \\ & (meet-03,\ :ARG0,\ i), \\ & (meet-03,\ :ARG1, work-01), \\ & (meet-03,\ :ARG1,\ person), \\ & (work-01,\ :ARG0,\ i), \\ & (work-01, :ARG0,\ person), \\ & (person,\ :name,\ name), \\ & (name,\ :op1,\ ''James'')\} \end{aligned}$$

$$\begin{aligned} RDF_{S_2} = \big\{ & (root,\ :ROOT,\ drive-01), \\ & (drive-01,\ :ARG0,\ person), \\ & (drive-01, :ARG1,\ car), \\ & (person,\ :name,\ name), \\ & (name,\ :op1,\ ''James'')\big\} \end{aligned}$$

The combined set of RDF triplets (RDF_C) is provided by the union of both sets, RDF_{S_1} and RDF_{S_2}, as follows.

$$RDF_C = RDF_{S_1} \cup RDF_{S_2} = \{(root,\ :ROOT,\ meet\text{-}03),\ (root,\ :ROOT,\ drive\text{-}01),\ (meet\text{-}03,\ :ARG0,\ i),\ (meet\text{-}03,\ :ARG1,\ work\text{-}01), (meet\text{-}03,\ :ARG1,\ person),\ (work\text{-}01,\ :ARG0,\ i),\ (work\text{-}01,\ :ARG0,\ person),\ (person,\ :name,\ name),\ (name,\ :op1,\ \text{“}James\text{"}),\ (drive\text{-}01,\ :ARG0,\ person), (drive\text{-}01,\ :ARG1,\ car)\}$$

According to the combined RDF triplets (RDF_C), the semantic graph of these two sentences is illustrated in Fig. 4c.

The obtained semantic graphs, in turn, should be transformed into an appropriate format to be given as input to a machine learning model. In the next section, we present the methods of transforming the semantic graphs for using them in the deep learning phase, where a model is trained on semantic graph-summary pairs to predict a summary of a new instance.

Graph transformations for machine learning

The machine learning models that we examine in this approach are based mainly on sequence-to-sequence (seq2seq) architectures. These architectures take as input a sequence of tokens that represent a semantic graph of an input text to predict, also, a sequence of tokens forming a summary. Therefore, we should transform the semantic graph representation of an input text into a sequence of tokens (i.e., tokens that describe a graph). For this purpose, due to the fact that we adopt the AMR formalism to represent semantic graphs, we use the AMR graphs in text format because this representation is transformed into a sequence of tokens. In this direction, we examine some alternatives for representing a semantic graph to a sequence of tokens that are as follows.

Original semantic graph: This is a linearized version of an AMR graph in text format. To illustrate, the text format of the AMR graph of Example 2 is linearized to a sequence of tokens as follows.

$$(\ a\ /\ want\text{-}01\ \ :ARG0\ (\ b1\ /\ person\ \ :name\ (\ n1\ /\ name\ \ :op1\ \text{“}Mary\text{"}\ )\ )\ \ :ARG1\ (\ c\ /believe\text{-}01\ \ :ARG0\ (\ b2\ /\ person\ \ :name\ (\ n2\ /\ name\ \ :op1\ \text{“}Jennifer\text{"}\ )\ )\ \ :ARG1\ b1\ )\ )$$

Original semantic graph without sense numbers: In this representation, we have removed the sense numbers of the AMR concepts in order to reduce the size of the vocabulary (e.g., *want-01* is changed to *want*). For example, the AMR graph of Example 2 is linearized as follows.

$$(\ a\ /\ want\ \ :ARG0\ (\ b1\ /\ person\ \ :name\ (\ n1\ /\ name\ \ :op1\ \text{“}Mary\text{"}\ )\ )\ \ :ARG1\ (\ c\ /\ believe\ :ARG0\ (\ b2\ /\ person\ \ :name\ (\ n2\ /\ name\ \ :op1\ \text{“}Jennifer\text{"}\ )\ )\ \ :ARG1\ b1\ )\ )$$

Simplified semantic graph with sense numbers: In this linearized version, we remove the variables of the concepts and the symbol “/" after a variable (e.g., *b1 / person* is transformed to *person*) or replace the variables with the respective concepts (e.g., *b1* is replaced by *person*) as well as the first and the last brackets are removed. This transformation reduces the length of the linearized graph representation. For instance, the AMR graph of Example 2 is transformed as follows.

$$want\text{-}01\ :ARG0\ (\ person\ :name\ (\ name\ :op1\ "Mary"\ )\ )\ :ARG1\ (\ believe\text{-}01\ :ARG0\ (person\ :name\ (\ name\ :op1\ "Jennifer"\ )\ )\ :ARG1\ person$$

Simplified semantic graph without sense numbers: In addition to the previous representation, we remove the sense numbers changing the linearized AMR graph of the example 2 as follows.

$$want\ :ARG0\ (\ person\ :name\ (\ name\ :op1\ "Mary"\ )\ )\ :ARG1\ (\ believe\ :ARG0\ (\ person\ :name\ (\ name\ :op1\ "Jennifer"\ )\ )\ :ARG1\ person$$

The simplified versions of the above alternatives provide a concise representation that leads to a compression of data, reducing the computational load of the overall approach. The deletion of sense numbers in AMR representation leads to a decreasing size of the vocabulary that also reduces the search space of the machine learning predictions. The effect of each AMR representation is examined in the experimental procedure.

In the case of a multi-sentence text that correspond to a multi-graph representation, the linearized version of the semantic graph is composed of a sequence of linearized sub-graphs that are separated by a predefined symbol or token (e.g., *[linearized AMR of sentence 1] [EOG] [linearized AMR of sentence 2],...,[EOG] [linearized AMR of sentence n]*, where *[EOG]* denotes the end of a graph or sub-graph), following the graph as sequence of sub-graphs construction method mentioned above. On the other hand, in the case of combined multi sub-graphs to obtain the semantic graph of a text, the root node of an obtained graph (which mentioned above in "Semantic graph as a combination of sub-graphs") is used for extracting the paths of a graph (e.g., using the DFS algorithm) that forms the linearized version of the semantic graph. Then, the root node is removed from the linearized version of the semantic graph because this additional node (i.e., the node that has been added to represent the root of a graph) does not convey any useful information and also this burdens the machine learning models with additional computational cost.

Deep learning prediction

Graph to text generation

The proposed framework is based on a graph-to-text generation process, as it obtains a semantic graph representation for each text to generate a summary. As it has been explained above, after applying semantic graph parsing ("Semantic graph parsing"), semantic graph construction of an original text ("Semantic graph construction") and graph transformation ("Graph transformations for machine learning"), the semantic representation of the initial text corresponds to a linearized representation of the semantic graph. A linearized semantic graph constitutes a sequence of tokens (out of a particular set of tokens) that represents an obtained semantic graph. For example, the linearized versions of the semantic graph of Fig. 2 (Example 2) have been presented in "Graph transformations for machine learning" for each graph transformation method. Then, a sequence of tokens that represents a semantic graph (i.e., a semantic graph representation) is given as input to a machine learning model to predict a sequence of tokens that correspond to the generated summary. The particular steps of obtaining a semantic graph representation of an original text have been explained in detail above ("Semantic graph parsing", "Semantic graph construction" and "Graph transformations for machine learning"). The employed deep learning models and the process of deep learning predictions are described in detail in this section ("Deep learning prediction") below.

It should be noted that the process of converting an original text to its semantic graph representation may lead to information loss. In the case of AMR graph parsing, this issue has been studied in the literature [23, 24, 45, 46] in an attempt to specify the accuracy of semantic graph parsing. In our case, we focus on evaluating the generated summaries because this is a TS framework. Also, it is not possible to examine directly the accuracy of the semantic graph representation due to the lack of usage examples that contain reference semantic graphs for the employed and widely used text summarization datasets. The experimental results along with a comparison to other similar methodologies based on graph-to-summary generation, are presented and discussed in "Results" and "Discussion", respectively.

The computational load of the proposed framework depends on its individual components. Assuming that we have extracted an AMR graph from the initial text, the proposed framework performs semantic graph construction, graph transformation and machine learning inference. The computational complexity of the graph construction process depends on the particular methodology that we use. In the worst case of using a semantic graph as a combination of sub-graphs ("Semantic graph as a combination of sub-graphs"), a task performed by Algorithm 1, the computational complexity is $\mathcal{O}(k \cdot n \cdot \log n)$, where k represents the number of the sub-graphs and n the number of examined *RDF* triplets (node-edge-node) of the semantic graph. Since $k << n$, the computational complexity of the algorithm is assumed to be $\mathcal{O}(n \cdot \log n)$ [48], as the algorithm combines each set of *RDF* triplets (node-edge-node) of a sub-graph with the remaining set of *RDF* triplets. Although the computational complexity of the abovementioned case is equal to $\mathcal{O}(n \cdot \log n)$, in practice, the process is performed in almost negligible time because the number of n is limited to a few decades or few hundreds, according to the length of the text. The same conclusion is valid for the rest of the computational processes (i.e. graph transformation and machine learning inference), as the particular algorithms are executed on human-readable texts, which have a limited length of textual content.

Semantic graph-based machine learning phase

Given a semantic graph in a particular format as it has been explained in "Graph transformations for machine learning", the machine learning phase of the proposed framework is performed in two steps. Firstly, the tokens of the training set are represented in a continuous vector space, by using either context-independent (e.g. word2vec, glove) [49] or context-dependent (e.g., ELMO, BERT) [50] embeddings. Then, the retrieved vectors are provided as input to a deep learning model. In the second step, a deep learning model is trained to predict a summary of an original text ("Deep learning models"). It is noted that during the phase of training of a machine learning model, the vectors of word representation are not assumed as fixed vectors but they may be further adapted according to the employed usage examples of the training set.

Word representation

The context-independent word representation techniques map each token to a vector of real values of a dimensionality *D*. These vectors are known as *word embeddings* [49, 51]. The main advantage of word embeddings is that they retain the semantic relationship between words (i.e., words with similar meanings are placed close to each other in the embedding vector space) that is in line with the proposed framework. However, a drawback of this representation is that each word is represented by a single global vector ignoring its context (i.e., the same vector is assigned to a word without taking into account its meaning in a text).

On the other hand, a context-dependent word representation, which is also known as contextual embeddings, maps each word to a vector-based representation according to its context [50]. Therefore, contextual embeddings capture syntactic and semantic properties of words (or sub-words) that are based on a sequence of words that include the target word. In several tasks of natural language processing (e.g., sentiment analysis, question answering, machine translation, TS, etc.), contextual embeddings that are based on pre-trained models on large-scale corpora achieve state-of-the-art performance [52].

The proposed framework makes use of both, non-contextual and contextual embeddings in the phase of training machine learning models ("Deep learning models"), where the tokens in graph-summary pairs are mapped to their particular vectors. The framework may employ any word embedding methodology provided that the vectors retain the semantic relationships of words in the vector space. In the experimental procedure of this work ("Experiments"), we make use of Word2Vec [53, 54] and *BERT* model [55] for non-context and contextual embeddings, respectively, according to the employed deep learning models that are presented below.

To obtain non-contextual embeddings, we train a particular model (e.g., *word2vec*) from scratch on the usage examples of an employed training set (e.g., *Gigaword*) that contains semantic graph-summary pairs. A semantic graph is represented by a sequence of tokens according to the employed linearized graph representation ("Graph transformations for machine learning") and a summary, which is in textual form and also represented by a sequence of tokens. Both of the aforementioned sequences of tokens are concatenated to create a usage example for training a word embedding model. The sequence of tokens that represents a semantic graph of the original text contains words that are common with the vocabulary of summaries (e.g., boy, train, person, etc.). The vectors of these words are obtained by using the aforementioned pairs of the semantic graph - summary in the phase of training the model of word embeddings. Also, the semantic graphs contain words that carry their senses (e.g., want-01, say-01, etc.) and keywords for denoting the relations (e.g., :ARG0, :ARG1, :name, etc.), both of them appeared only on the semantic graph of the original text. Therefore, the senses and the relations are represented by vectors that are obtained by using only the semantic graphs of the original texts in training a model of word embeddings. It should be noted that the obtained word embeddings should be further adapted during training end-to-end a deep learning model ("Deep learning models") for estimating a summary. Also, we may omit this step of training word embeddings by initializing randomly the vector of each token that will be adapted accordingly during the phase of training a model for automatic

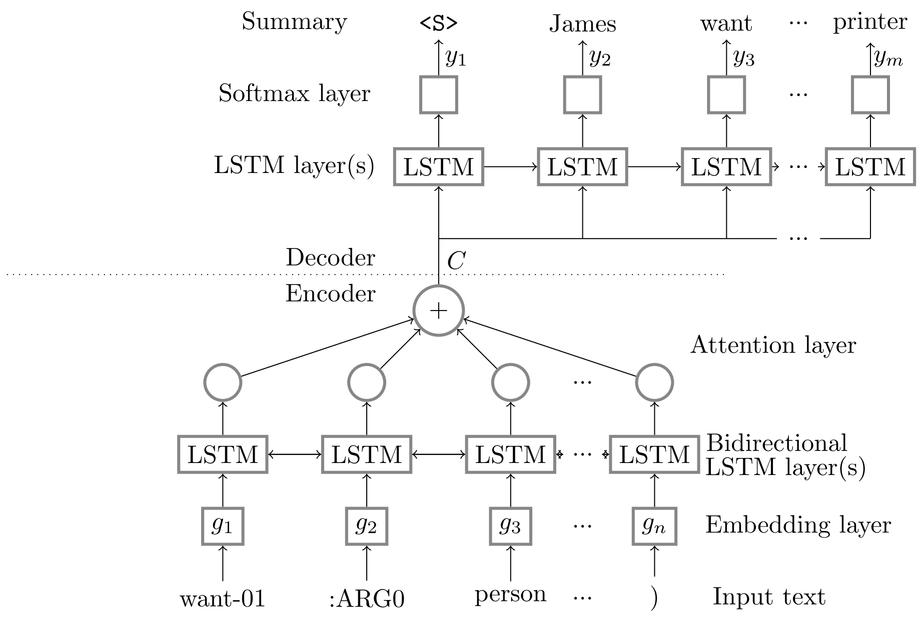

Fig. 5 The attentive sequence to sequence deep learning model

TS. However, having trained word embeddings, the process of training a model, which learns to predict summaries, is accelerated.

For the contextual embeddings, we make use of the *BERT* (Bidirectional Encoder Representations from Transformers) model [55], which uses sub-word embeddings (i.e., a word may be split into sub-words according to the vocabulary of the pre-trained model). To incorporate the keywords that denote the relations of the semantic graphs to the pre-trained model, we add these tokens to the vocabulary of the pre-trained model as tokens that are never split into sub-words. The vectors of these tokes are initialized randomly and adapted accordingly during training a particular deep learning model ("Deep learning models").

Deep learning models

Given a sequence of elements of a semantic graph $G = (g_1, g_2, ...)$ that represents an original text, a deep learning model, which is trained on semantic graph-summary pairs, is used for predicting a sequence of tokens $Y' = (y'_1, y'_2, ...)$ that corresponds to a summary. For this purpose, we examine five deep learning models; (*i*) an attentive sequence to sequence with pointer-generator, (*ii*) a reinforcement learning (*iii*) a transformer, (*iv*) a transformer with contextual embeddings, and (*v*) a pre-trained encoder transformer. These architectures of neural networks are described in detail below.

Attentive sequence to sequence with pointer generator Figure 5 illustrates an encoder-decoder deep learning architecture with an attention mechanism, where the encoder takes the embeddings of a semantic graph representation, and the decoder learns to predict the respective summary in text format. This network is extended to incorporate a pointer generator [56] and a coverage mechanism as we describe in detail below.

Embedding layer: The embedding layer receives an element from the source semantic graph, assigning to it a vector of the used embedding space and subsequently forwards it to the next layer of the encoder. The embedding vectors may be available in training phase and they are learnable during training the model, as we have explained in "Graph to text generation" above.

Bi-directional LSTM layer: The second layer of the model consists of one or more bi-directional LSTM units [57, 58], which receive the embedding vectors of a sequence of semantic graph elements $G = (g_1, g_2, \ldots, g_n)$ (one vector for each time step) in forward and reverse order (as it implements a bi-directional structure), producing a hidden state $H_t = bi\_lstm(g_t, H_{t-1})$ at their output. According to Equation 1, this hidden state is formed by the concatenation of the hidden state vectors ($\overrightarrow{h_t}$ and $\overleftarrow{h_t}$) of both directions of the bi-directional LSTM.

$$\begin{aligned}
\overrightarrow{h_t} &= lstm(g_i, \overrightarrow{h_{t-1}}),\ g_i \in (g_1, g_2, ..., g_{n-1}, g_n) \\
\overleftarrow{h_t} &= lstm(g_j, \overleftarrow{h_{t-1}}),\ g_j \in (g_n, g_{n-1}, \ldots, g_2, g_1) \\
H_t &= [\overrightarrow{h_t}; \overleftarrow{h_t}]
\end{aligned} \tag{1}$$

Attention Layer: The encoder is equipped with an attention mechanism [56, 59], which enhances the accuracy of the predictions by focusing on relevant words of the input semantic graph. Since there are English words among the elements of a semantic graph, as we have already explained in "Preliminaries for semantic graph-based representation", and only these words may be presented in the summary (i.e., special keywords such as relations do not appear in summary), the attention mechanism focuses on these words that are possible to appear in the summary. This mechanism computes a context vector c_t, as a weighted sum of the encoder hidden states H_t, according to Equation 2.

$$\begin{aligned}
c_t &= \sum_{i=1}^{|G|} a_{t,i} \cdot H_i \\
a_{t,i} &= softmax(e_{t,i}) \\
e_{t,i} &= V_i \cdot tanh(W_h \cdot H_i + W_s \cdot s_{t-1} + b_i)
\end{aligned} \tag{2}$$

where a_{ti} is the weight of each time step t of the encoder's state H_i, $e_{t,i}$ indicating the degree of fitting the output of step t with the input around the word i, and s_{t-1} is the previous state of the encoder. V_i, W_h and W_s are network weights and b_i represents bias; these parameters are adapted during training.

LSTM layer (decoder): The decoder is comprised of unidirectional LSTM units. Their purpose is to predict the next word y_t in the summary, based on their hidden state $h_{d,t}$ at time t, the context vector c_t and the previous hidden state $h_{d,t-1}$ of the decoder. During training, the target sequence of word vectors $Y = (y_1, y_2, \ldots, y_m)$ is also made available to the decoder (one-by-one word embedding of a reference summary, at each time step t) and the decoder learns to predict the next one, shaping the final summary.

Softmax layer: The softmax layer is used for generating a probability distribution of the next word over the set of candidate words. In particular, at each time step t, the softmax layer computes the probability of each candidate word y_i of the vocabulary Y for a predicted summary, according to Equation 3.

$$p_t(y_i|G, y_{t-1}) = \frac{e^{h_i^T w_i + b_i}}{\sum_{j=1}^{k} e^{h_t^T w_j + b_j}} \tag{3}$$

where G, y_{t-1} and h_t are the input semantic graph, the previous estimated word, and the decoder hidden state, respectively. The parameters w and b correspond to the weights and bias that are adapted during the learning process. The sum of the probabilities in the set of candidate words is equal to one (Equation 4).

$$\sum_{i=1}^{k} p_t(y_i|G, y_{t-1}) = 1 \tag{4}$$

Pointer-generator: The above described network is also equipped with a Pointer-generator [56], which allows out-of-vocabulary (OOV) words to be copied from the source to the summary. This network generates words by either sampling them from a fixed vocabulary of the training set or from the words that appear in the source graph. Therefore, the vocabulary is extended to include OOV words that may be appeared in the source graph, as OOV words can be assumed only the English words that are appeared in a semantic graph. The special keywords that constitute elements of a semantic graph can not be considered as new words because they constitute predefined tokens and as a result, the pointer-generator does not take them into account as they never appear on the output. In particular, at each time step t, the model samples a word either from the vocabulary distribution of the training set or from the attention distribution of the current usage example (the attention distribution is based on the tokens of the original text, some of which may not be included in the vocabulary of the training set as OOV words). The probability distribution of the generator is computed by the Equation 5.

$$P_{g,t} = \sigma(w_c \cdot c_t + w_s \cdot s_t + w_g \cdot g_t + b_p) \tag{5}$$

where σ denotes the sigmoid function, c_t is the context vector as calculated by the attention mechanism (Equation 2), s_t is the decoder state and g_t is the input vector. The weights w_h, w_s, w_g are adapted during training and b_p is the bias of the pointer generator. The probability $P_t(i)$ of the word i appearing in step t in the estimated summary is calculated according to Equation 6, which gives the probability distribution of the extended vocabulary.

$$P_t(i) = P_{g,t} \cdot P_v(i) + (1 - P_{g,t}) \cdot a_{i,t} \tag{6}$$

where $P_v(i)$ is the probability of presenting a word i to the output sequence. P_v is the probability distribution of words of the fixed vocabulary of the training set as calculated from the softmax layer (Equation 3). The parameter $a_{i,t}$ represents the weight of the word i in time step t, as calculated by the attention mechanism. If the word i is a OOV word, then $P_V(i) = 0$. Otherwise, if a word does not appear in the input graph, then $a_{i,t} = 0$. In this way, the network estimates the probability distribution on the extended vocabulary or on the fixed vocabulary in case there are not any OOV words.

Coverage mechanism: In addition, the model incorporates a coverage mechanism to avoid repetition of the same words at the output, adapting the solution suggested in [60]. For this purpose, a coverage vector cov_t is calculated at each time-step t of the

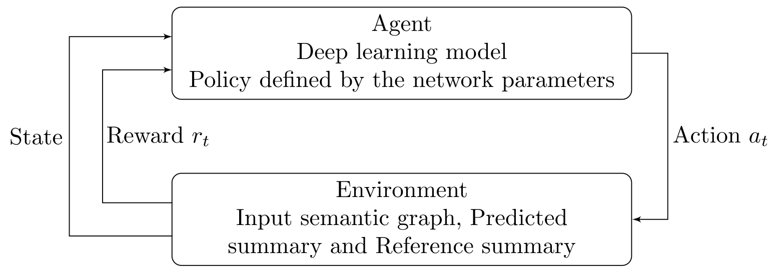
Agent
Deep learning model
Policy defined by the network parameters
State
Reward r_t
Action a_t
Environment
Input semantic graph, Predicted summary and Reference summary

Fig. 6 Reinforcement-learning model

recurrent neural network, which is equal to the sum of the weights of the attention mechanism according to Equation 7 for all previous steps t'.

$$cov_t = \sum_{t'=0}^{t-1} a_{t'} \tag{7}$$

The vector cov_t may be assumed as a coverage distribution of the words in the input text. This vector indicates how well these words are covered by the attention mechanism. It should be noted that in the first time-step, where no element of the input graph has been examined, $cov_0 = 0$. The cover vector is given as an additional input to the attention mechanism by modifying the Equation2 of the $e_{t,i}$ presented above according to Equation 8.

$$e_{t,i} = tanh(w_h \cdot H_i + w_s \cdot s_{t-1} + w_{cov} \cdot cov_t + b) \tag{8}$$

where w_{cov} is a parameter that is adapted during the training process. This modification prevents the attention mechanism from paying attention to the same words, avoiding the repetition of the same words in the output.

The deep learning model is trained end-to-end using supervised learning on a training set of semantic graph-summary pairs, using stochastic gradient descent and minimizing the negative log-likelihood of the target word y_t (Equation 9), which is used as loss function

$$Loss = -\sum_{t=1}^{T} logP(y_t|G) \tag{9}$$

where $P(y_t|G)$ is the likelihood of the target word at time-step $t \in [1, 2, \ldots, T]$ of T words in the summary, given an input semantic graph G.

Additionally, the technique of *dropout* is also used [61, 62], which randomly drops connections of units from the neural network during training in order to avoid overfitting. Moreover, to predict an optimal summary we use the *beam search* algorithm [63, 64], where, at each time step of the decoder, it retains the w (beam-width) candidate tokens of the highest log-probability to determine the best output summary according to the beam search algorithm.

Reinforcement-learning model In contrast to the above described model (attentive sequence to sequence with pointer generator model) that minimizes a loss function, which does not constitute a particular measure for evaluating TS, a reinforcement learning (RL) approach learns a policy that maximizes a specific metric, such as one of the *Rouge* measures. The employed RL model extends a seq2seq network, according to the approach of [65].

More specifically, in a RL architecture, an encoder-decoder neural network (e.g., the attentive sequence to sequence with pointer generator model described above) is used as an agent that interacts with a given environment aiming at maximizing a reward. Figure 6 depicts the basic elements of a RL model for automatic TS. The parameters of the network (θ) define a policy, which determines the actions (a_t) for each time-step t to estimate a sequence of words that constitutes a summary of a given semantic graph. After each action, the agent (the deep learning model) receives a reward r_t in order to update the parameters θ. The reward is calculated by comparing the predicted summary with the reference summary according to a specific metric.

The process of training uses the self-critical sequence training algorithm [66]. According to this algorithm, the machine learning model produces two sequences of words in each iteration of training. The first sequence of words $Y_s = (y_{s,1}, y_{s,2}, \ldots, y_{s,t}, \ldots, y_{s,T})$ retrieved by sampling from the probability distribution $p(y_{s,t}|y_1, \ldots, y_{t-1}, G)$ of the model, according to the probabilities of the softmax layer, with respect to an input semantic graph G. Where $y_{s,t}$ represents a sampling word in the time-step t. The second sequence of words $\hat{Y} = (\hat{y_1}, \hat{y_2}, \ldots, \hat{y_t}, \ldots, \hat{y_T})$ constitutes a predicted output sequence (i.e., as it is predicted in the phase of testing using the beam search algorithm).

During training, the recurrent neural network (attentive sequence to sequence with pointer generator), which is used as an agent, estimates the sampling sequence Y_s of the output words given the previous word of the target sequence at each time step t. In inference, on the other hand, the model takes into account the previously estimated words to generated the predicted sequence $\hat{Y}$, without knowing the previous words of the target sequence. This means that the target sequence of words is known to the model during training but not in inference. As a result, the sampling sequence is more accurate than the estimated one. This constitutes a training bias problem, which is known as *exposure bias*. To mitigate its effect, the model aims at maximizing the reward function by computing both the reward of the sampling sequence $r(\hat{Y})$ and the reward of the predicted sequence $r(Y_s)$ during training.

More specifically, following the approach of [66], the *RL* model aims at maximizing the reward function of the sampling sequence by minimizing the error function of Equation 10.

$$L_{RL} = -(r(Y_s) - r(\hat{Y})) \sum_{i=1}^{T} log(p(y_{s,t}|y_{s,1}, y_{s,2}, \ldots, y_{s,t-1}; \theta, G)) \tag{10}$$

where $r(\hat{Y})$ is subtracted from $r(Y_s)$ in order to reduce the variance of the reward of the sampling sequence. It is obvious that minimizing L_{RL} corresponds to a maximization of the conditional probability $p(y_{s,t}|y_{s,1}, y_{s,2}, \ldots, y_{s,t-1}; \theta, G)$ of the sampling sequence. Therefore, the reward the model receives increases during training (more details about this model are provided in [65, 66]).

Transformer-based models Since the models that are based on transformers [67–71] are considered as the state-of-the-art approaches in natural language processing [72], we employ such architectures in an effort to examine the effectiveness of transformer-based models in the proposed framework. In this direction, we are based on the methodology of [70] employing three transformer-based approaches; (i) a transformer network (TR), (ii) a transformer network with contextual embeddings (TRCE) and (iii) a pre-trained encoder transformer model (PETR).

TR: The first model is a transformer network of an encoder-decoder architecture with self-attention layers, as it is described in detail in [67]. This model utilizes parallelism of computations to identify dependencies between input and output, avoiding the high computational cost of a recurrent or convolutional network. The TR model is trained from scratch on the training set, adapting its weights without using pre-trained embeddings (e.g., word2vec, glove, BERT, etc.).

TRCE: The second alternative is an identical model to the aforementioned TR model, which uses contextual embeddings (e.g., BERT) in order to initialize the embedding vectors that are given as input to the network in the training process. This alternative aims at accelerating the convergence time of the machine learning model. Also, the proper initialization of the embedding vectors may lead to an improvement in the accuracy of the model predictions. The aforementioned assumptions are investigated in the experiments ("Experiments").

PETR: The third model employs a pre-trained BERT [55] encoder with a transformer decoder, as it is described in [70]. This model, during the process of training, performs a fine-tuning of the weights of the pre-trained encoder and also, it adapts the weights of the decoder which is trained from scratch. Currently, the pre-trained language models have dominated the field of natural language processing. The specific model employed in this work (BERT), is a language representation model, which has been trained on a large amount of textual data (i.e., *BookCorpus* consists of 800 million words [73] and the English *Wikipedia* contains about 2.5 billion words). The BERT model outputs a contextual vector t_i for each word i.

Experiments

In this section, we describe the experimental procedure[2], which aims at evaluating the proposed framework by examining various aspects of the overall methodology. "Datasets" describes the employed datasets, "Evaluation metrics" provides a description of the evaluation measures introducing metrics for measuring the factual consistency of the generated summaries. "Experimental procedure and parameter tuning" describes the experimental procedure and finally, "Competitive approaches" presents other relevant approaches that are used for comparison reasons.

Datasets

To evaluate the proposed framework, we use two popular datasets of TS; the annotated *Gigaword* [74], and the *CNN/DailyMail* [75]. The particular version of the *Gigaword*

[2] Source code used in the experimental procedure: https://github.com/pkouris/semgraphtextsum

Table 1 The combinations of graph construction and transformation methods (data schemes)

Graph construction	Graph transormation	Abbreviation of data scheme
Sequence of sub-graphs	Original AMR	S-OAMR
	Original AMR without sences	S-OAMRWS
	Simplified AMR	S-SAMR
	Simplified AMR without sences	S-SAMRWS
Combination of sub-graphs	Original AMR	C-OAMR
	Original AMR without senses	C-OAMRWS
	Simplified AMR	C-SAMR
	Simplified AMR without senses	C-SAMRWS

dataset used is that of [42], which has been widely adopted in TS [76]. It contains about 3.8 million article-summary pairs with 123 million tokens of a vocabulary of 119, 000 distinct tokens. The average length of an article and a summary is 31.4 and 8.3 tokens, respectively. The test set and the validation set are composed by randomly sampling 2, 000 pairs of instances for each of these sets; a common practice when this dataset is used in TS [42, 56, 77].

The second dataset, *CNN/DailyMail*, is a document-level dataset, which also is widely used in TS. It is comprised of articles or stories and their summaries of multi-sentences. To obtain the non-anonymized version of this dataset, we follow the methodology of preprocessing that is proposed by the creators of the dataset [77]. This version of the dataset contains a training set of 287, 227 article-summary pairs, a test set of 11, 490 instances, and a validation set of 13, 368 usage examples. Following the typical procedure when this dataset is used in deep learning models [56, 78, 79], we limit the length of the articles to 400 tokens and the summaries to 100 tokens. After preprocessing, the average length of a text and a summary are 386.42 and 61.08 tokens, respectively. This version of the dataset contains about 128 million tokens of a vocabulary of 510, 607 distinct tokens.

For both datasets described above, we obtain the AMR graphs of their texts by performing AMR parsing [80]. The AMR parser returns an AMR graph for each sentence of a text. After parsing, we commence with graph construction and transformations, as described in "Semantic graph construction" and "Graph transformations for machine learning". According the graph construction process, we assume a semantic graph as (*i*) a sequence of sub-graphs or (*ii*) a combination of sub-graphs (i.e., merging the sub-graphs to generate a graph representation). Additionally, applying the graph transformations to AMR graphs, we obtain four alternatives; (*i*) Original AMR (OAMR), (*ii*) original AMR without sences (OAMRWS), (*iii*) simplified AMR (SAMR), and (*iv*) simplified AMR without senses (SAMRWS). Table 1 outlines the overall alternatives of the semantic graphs, which correspond to the versions of the datasets that we examine in the experimental procedure. These eight alternatives are applied on the *CNN/DailyMail* dataset because it contains texts of multi sentences, and as a consequence multi semantic graphs, where we apply both graph constructions and transformations. In the case of *Gigaword* dataset, we examine only the four alternatives of the graph transformations because this dataset contains texts of one sentence and the AMR graph that is obtained

Table 2 The vocabulary size and the length of the semantic graph-summary pairs of the Gigaword and CNN/DailyMail (CNN/DM) datasets for the data approaches according to the employed graph construction and transformation technique (data scheme)

Dataset	Data scheme	Average length		Number of tokens		Number of distinct tokens	
		Graph	Summary	Graph	Summary	Graph	Summary
Gigaword	OAMR	128.4	8.2	441.7*M*	28.2*M*	84, 627	68,882
	OAMRWS	128.4	8.2	441.7*M*	28.2*M*	82, 029	68,882
	SAMR	66.1	8.2	227.3*M*	28.2*M*	84, 400	68,882
	SAMRWS	66.1	8.2	227.3*M*	28.2*M*	81, 806	68,882
CNN/DM	S-OAMR	849.1	61.08	277.4*M*	17.5*M*	253, 961	195,208
	S-OAMRWS	849.1	61.08	277.4*M*	17.5*M*	245, 296	195,208
	S-SAMR	494.4	61.08	142.0*M*	17.5*M*	253.165	195,208
	S-SAMRWS	494.4	61.08	142.0*M*	17.5*M*	244, 388	195,208
	C-OAMR	830.7	61.08	272.5*M*	17.5*M*	253, 961	195,208
	C-OAMRWS	830.7	61.08	272.5*M*	17.5*M*	245, 296	195,208
	C-SAMR	478.6	61.8	140.3*M*	17.5*M*	253.165	195,208
	C-SAMRWS	478.6	140.3*M*	17.5*M*	244,388	195, 208	

for each text does not require any process of constructing a semantic graph for the overall text.

According to the employed data alternatives, Table 2 reports statistics about the vocabulary size and the length of the semantic graph-summary pairs of both datasets. The average length of the semantic graphs and summaries on a particular dataset and data scheme corresponds to the average number of elements or tokens of the linearized graphs and summaries, respectively. Also, we have computed the overall number of tokens and the number of the distinct tokens (i.e., vocabulary size) for each data scheme. The *SARMWS* data scheme minimizes the vocabulary size and reduces the length of a semantic graph, while the data schemes that retain the sense numbers (*OAMR* and *SAMR*) increase the vocabulary size and the length of a graph representation. The original text has the minimum length but the maximum vocabulary size, compared to the linearized AMR data schemes.

Evaluation metrics

Rouge metrics

To evaluate the performance of the proposed framework, we use the official *Rouge* package [81]. In particular, we calculate the average *F-1 score* of *Rouge-1* (word overlap), *Rouge-2* (bigram overlap) and *Rouge-L* (longest common sequence) on the test set of the datasets, as it is a typical practice in evaluating TS systems in the relevant literature [77, 82, 83].

Blue metric

In addition to the Rouge metrics, we also employ the *Blue* metric [84] to evaluate *n*-gram overlapping between system summaries and the respective reference summaries. In particular, the *Blue* metric is a precision-based measure, which calculates a weighted score, taking into account the proportion of n-grams ($n \in \{1, 2, 3, 4\}$) of a generated summary that are included to the reference summary. The weight for each n is assumed to be equal (e.g., 0.25), as is the typical use of this metric [85]. Also, the *Blue* measure includes a brevity penalty, which is applied when the length of a system summary is shorter than that of the reference summary. Finally, we note that the *Blue* metric is the primary evaluation measure for machine translation and this is also used for assessing the generated summaries in the field of automatic TS [86].

Factual consistency

Factual Consistency has been presented in [87, 88], where the authors investigated the correlation between this measure and human evaluation. Especially in [88], the authors conclude that *Factual Consistency* is the measure presenting the highest correlation with human evaluation scores compared to any other measure, like the *Rouge* metrics. In an effort to provide a qualitative assessment for evaluating the *Factual Consistency* of the predicted summaries, we extend the approach of [14] computing the *precision*, *recall* and f_β scores of factual consistency. More specifically, triplets, such as *(subset, relation, object)*, are retrieved from a predicted summary and the respective original text that constitutes the facts. The triplets of facts have been obtained using the open information extraction *(OpenIE)* approach [88, 89]. The overlap between the retrieved facts of a system summary and the source text determines the factual consistency that is based on *precision* (FC_p), *recall* (FC_r) and f_β (FC_{f_β}), which are computed according to Equations 11.

$$\begin{aligned} FC_p &= \frac{|F_t \cap F_s|}{|F_s|} \\ FC_r &= \frac{|F_t \cap F_s|}{|F_t|} \\ FC_{f_\beta} &= \frac{(1+\beta^2)\cdot FC_p \cdot FC_r}{\beta^2 \cdot FC_p + FC_r} \end{aligned} \tag{11}$$

where F_t is a set of facts from a source text, F_s is a set of facts of the respective predicted summary and β is the factor that indicates how much more important *recall* (FC_r) is than *precision* (FC_p). To clarify the usage of f_β score, in the case that the source text is much longer than the summary (e.g., *CNN/DailyMail* dataset), the *recall*-based factual consistency is expected to take very small values because, in this case, the source text contains much more triplets of facts than those of its summary. This is the reason that we use the f_β measure, which provides a weighted score of precision and recall, according to the relative length among a source text and its summary. To specify, the fraction of summary length (S_L) to source text length (T_L) gives the value of β, according to Equation 12.

$$\beta = \frac{S_L}{T_L} \tag{12}$$

For example, if the number of tokens of a text and its summary are 400 and 100, respectively, then $\beta = 100/400 = 0.25$. Also, β value may be fixed according to the average length of texts and summaries of the usage examples of a dataset. Therefore, the f_β measure provides a weighted value of factual consistency which indicates the coverage of salient information between a text and its summary. Finally, we note that the measures of factual consistency aim at a more qualitative than quantitative evaluation, while the *Rouge* metrics perform a quantitative assessment.

New tokens rate

Since the proposed framework constitutes an abstractive TS approach, we use a metric to investigate the ability of the examined models (i.e. the variations of deep learning architectures and data schemes) to generate new tokens in a predicted summary of an original text. Therefore, the last metric is that of *New Tokens Rate* (*NTR*), which corresponds to the percentage of tokens appearing in the generated summary, but not included in the input text (a similar version of this measure has been used in [42, 90]).

Experimental procedure and parameter tuning

After performing the tasks of AMR parsing, graph construction and graph transformation on the employed datasets, several versions of semantic graph-summary pairs are obtained for each dataset (see in "Datasets") to be used in the training and evaluation phases. The particular data schemes and the performance of the respective models are further discussed in Sects. "Results" and "Discussion".

Word representation: We use *word2vec* representation as non-contextual embeddings for the seq2seq attentive and RL model, while for the transformer-based models we use the pre-trained *BERT* model as contextual embeddings (see in "Graph to text generation"). A *word2vec* model of CBOW architecture [53] is trained to obtain word embeddings of 300 dimensions for each version of the training data. Each Word2vec model, which corresponds to each data scheme, has been trained for 10 epochs with a decaying learning rate from 0.02 to 0.001 and a window size equal to 5.

Training the attentive seq2seq with pointer generator (AS2SP) model ("Deep learning models"): This network is trained on the different versions of the training data obtaining an equal number of trained models. The parameters of this model have been optimized by using the validation set of the datasets. The encoder contains two layers of bi-directional LSTM units of dimensionality 256 and the decoder uses one layer of LSTM units of 512 dimensions each. The training data for each epoch are randomly shuffled and the batch size is set to 64 for the *Gigaword* and 32 for the *CNN/DailyMail* dataset. Also, the learning rate is set to 0.001. Moreover, the *Adam* optimizer [91] with gradient norm clipping [92] is employed with the loss function of negative conditional log-likelihood [93]. Additionally, *dropout* with $p = 0.2$ is also used. The vocabulary is limited to 150,000 words, using the most frequent tokens of the training set. We have trained the models (on NVidia K40 GPUs) for about 15 epochs until the models converged sufficiently.

Training the RL model ("Deep learning models", **reinforcement-learning model)**: Since this model uses the AS2SP network as an agent, the dimensionality of the LSTM layers is the same as the model mentioned above. The learning rate is set to 10^{-4}, and the batch size is set to 32 and 16 for the *Gigaword* and *CNN/DailyMail* datasets, respectively. For the rest of the parameters, we have made the same assumptions as the above-described model.

Training the TR model ("Deep learning models", **transformer-based models)**: The encoder and the decoder of the transformer architecture consist of 6 layers each. The hidden size (model dimensionality) of the encoder and the decoder is set to 512 and the inner layer has 2048 dimensions. Moreover, the number of heads is set to 8 (i.e., the model implements 8 parallel attention layers reducing the model dimensionality of each attention layer to $512/8 = 64$). The batch size is set to 64 for both datasets, *Gigaword* and *CNN/DailyMail*, and the dropout probability is set to $p = 0.1$. The Adam optimizer is employed with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and the learning rate is adapted during training, as shown in Equation 13, where $warmupSteps = 10,000$ and $a = 0.05$. According to this equation, the learning rate is increased for the first *warmupSteps* steps and then, it is decreased.

$$lr = a \cdot \min\{step^{-0.5}, \; step \cdot warmupSteps^{-1.5}\} \tag{13}$$

Training the TRCE model ("Deep learning models", transformer-based models): The difference between this model with the aforementioned TR model is that the TRCE model uses *BERT* embeddings, as an input to the embedding layer to initialize the corresponding vectors, while the TR model is trained from the scratch. During training, these embeddings further adapt their weights.

Training the PETR model ("Deep learning models", **transformer-based models)**: The PETR model uses a pre-trained *BERT* encoder and a transformer decoder of 6 layers identical to the decoder of TR model mentioned above. The model size of the pre-trained encoder and the decoder is set to 768. Moreover, the model uses two *Adam* optimizers, one for the encoder and one for the decoder to perform a stable fine-tuning. The parameters of both optimizers are $\beta_1 = 0.9$ and $\beta_2 = 0.99$ and these optimizers have different learning rates according to Equation 13, where $a_{enc} = 0.002$, $a_{dec} = 0.1$, $warmupSteps_{enc} = 20,000$ and $warmupSteps_{dec} = 10,000$. The different learning rates aim at a stable fine-tuning of the model weights, as the *BERT* encoder will be fine-tuned with a smaller learning rate and a smoother decay than the decoder, avoiding the overfitting or underfitting of the encoder or decoder [70]. The rest of the parameters remain the same as the TR and TRCE models.

Generating system summaries: To optimize the predicted summary, we use the beam search algorithm [63, 64] with a width equal to 5.

Competitive approaches

Our framework, which focuses on the field of semantic graph-based approaches for abstractive TS, is compared with other relevant approaches [10, 13, 18, 22]. In the next section, we report the performance of the most relevant systems. As relevant approaches, we assume those TS systems that are based on AMR representation and also are capable of coping with a dataset of a large size, such as the *CNN/DailyMail* or *Gigaword*. Since our methodology is based on deep learning models, these models require a large amount of data to be trained sufficiently. Therefore, we exclude some approaches [11, 19, 20], which have been mentioned in the related work section ("Preliminaries and

Table 3 *Rouge* scores on the *Gigaword* dataset of the attentive seq2seq with pointer-generator (AS2SP), reinforcement learning (RL), transformer (TR), transformer with contextual embeddings (TRCE) and pre-trained encoder transformer (PETR) networks for the data schemes: (*i*) original AMR (OAMR), (*ii*) original AMR without senses (OAMRWS) (*iii*) Simplified AMR (SAMR) and (*iv*) simplified AMR without senses (SAMRWS) ($p_{value} < 0.01$)

Model	Data scheme	Rouge-1	Rouge-2	Rouge-L	NTR (%)
AS2SP	OAMR	34.62	12.69	31.89	37.48
	OAMRWS	35.24	13.36	32.81	36.81
	SAMR	36.84	14.12	33.73	35.83
	SAMRWS	37.93	14.36	34.84	35.68
RL	OAMR	35.14	13.31	32.65	36.75
	OAMRWS	35.44	13.63	32.87	37.21
	SAMR	37.18	14.12	34.22	34.07
	SAMRWS	38.17	15.24	35.40	34.11
TR	SAMRWS	36.14	13.02	33.04	41.35
TRCE	SAMRWS	36.68	12.85	33.57	**41.46**
PETR	SAMRWS	**38.97**	**15.87**	**36.17**	41.17
ABS+AMR [10]		31.64	12.94	28.54	–
SemSUM [13]		38.78	19.75	36.09	–

related work"), that require a small amount of data, and also, they have been evaluated in small datasets (e.g. *proxy report* section of the *AMR Bank* [94]), because our approach is not appropriate for such data. As we can see in our analysis below, this comparison is based on *Rouge* scores that have been reported in the respective papers of the competitive approaches. We assume that the experimental results of the other approaches are directly comparable with ours because these approaches have obtained the train, test and validations sets of both datasets, *Gigaword* and *CNN/DailyMail*, following the same methodology as we have done (i.e. following specific preprocessing steps for *Gigaword* and *CNN/DailyMail*, "Datasets").

Results

Table 3 outlines the performance of the AS2S, RL, TR, TRCE and PETR models in terms of the *Rouge* scores on the *Gigaword* dataset for the four alternatives of the data that are based on the graph transformation methods ("Graph transformations for machine learning"), which are applied to create pairs of semantic graph-summary forming the usage examples of the training, testing and validation sets. Additionally, the percentage of *NTR* ("Evaluation metrics") is reported for the *Gigaword* dataset. Since, the best *Rouge* scores are obtained for the SARMWS graph transformation method, in the case of the transformer-based models (TR, TRCE, PETR), we report the *Rouge* scores for this data scheme. Also, for the same reason, in the rest of the experiments, we use the SAMRWS as a graph transformation method. The last rows of Table 3 include the performance of the baseline models ("Competitive approaches") that are used for comparison reasons. As we observe, the RL and PETR models obtain the highest *Rouge* scores.

Additionally, we assess the statistical significance of the *Rouge* scores for each model by the Welch's *t*-test [95]. To compute the statistical significance, we use the *Rouge* scores for each instance of the test set. The *t*-test is calculated on each pair

Table 4 *Rouge* scores on the *CNN/DailyMail* dataset of the AS2SP, RL, TR, TRCE and PETR deep learning models for the graph construction methods: (*i*) sequence of sub-graphs and (*ii*) combination of sub-graphs, and the simplified AMR without senses (SAMRWS) as graph transformation technique ($p_{value} < 0.01$)

Model	**Sequence of sub-graphs**				**Combination of sub-graphs**			
	R-1	**R-2**	**R-L**	**NTR (%)**	**R-1**	**R-2**	**R-L**	**NTR (%)**
AS2SP	36.44	11.25	29.98	25.26	37.69	11.21	30.52	27.34
RL	37.11	**11.49**	**31.20**	26.16	**39.83**	**11.95**	**31.65**	28.30
TR	36.32	7.00	25.53	45.28	31.57	4.88	22.73	44.16
TRCE	39.49	8.70	26.61	44.81	34.36	6.57	23.91	45.15
PETR	**40.75**	11.18	27.67	45.38	35.50	8.46	24.52	44.92
Lead-3-AMR [18]	31.70	5.80	16.80	–	–	–	–	–
Unmerged [22]	33.90	9.80	23.00	–	–	–	–	–

Table 5 *Blue-scores* on the *Gigaword* and *CNN/DailyMail* datasets of the employed deep learning models using the simplified AMR without senses as semantic graph transformation and the semantic graph construction methods that are applied to the *CNN/DailyMail* dataset ($p_{value} < 0.012$)

Model	**Gigaword**	**CNN/DailyMail**	
		Sequence of sub-graphs	**Combination of sub-graphs**
	Blue-score	**Blue-score**	**Blue-score**
AS2SP	14.65	15.81	16.45
RL	19.01	18.73	**18.82**
TR	14.97	17.05	14.68
TRCE	13.79	17.67	15.37
PETR	**20.05**	**19.86**	18.59

of data alternatives and for each *Rouge* metric (*Rouge-1, Rouge-2* and *Rouge-L*). The obtained values of statistical significance (p_{value}), which are reported in the caption of Table 3, prove that the results are statistically significant in all examined cases. The same methodology is followed to calculate the statistical significance in the rest of the measurements that follow (i.e., results in Tables 4, 5, 6, 7).

Similar to the above-described results, Table 4 reports the performance of the employed models in terms of *Rouge* scores on *CNN/DailyMail* dataset for the two methods of graph construction ("Semantic graph construction") and the simplified AMR without senses (SAMRWS) as graph transformation scheme. Additionally, *NTR* percentage and the assessment of statistical significance (p_{value}) are reported. Also, the last row of this table include the performance of the competitive approach that uses the same dataset. RL and PETR are those deep learning models that achieve the best Rouge scores.

Additionally, Table 5 includes the performance of the employed models in terms of *Blue* scores on *Gigaword* and *CNN/DailyMail* datasets using simplified AMR without senses (SAMRWS) as a graph transformation scheme. Moreover, in case of *CNN/DailyMail* dataset, the two proposed methods of graph construction ("Semantic graph construction") are examined. Also, the assessment of statistical significance (p_{value}) is

Table 6 Factual consistency on the *Gigaword* dataset of the employed deep learning models and the graph transformation methods as data schemes, setting $\beta = 0.264$ for computing the FC_{f_β} metric ($p_{value} < 0.02$)

Model	Data scheme	FC_p (%)	FC_r (%)	FC_{f_β} (%)
AS2SP	OAMR	74.03	24.93	65.60
	OAMRWS	77.92	28.93	70.16
	SAMR	81.93	26.60	72.13
	SAMRWS	83.23	24.67	72.06
RL	OAMR	80.11	24.48	69.75
	OAMRWS	81.67	25.21	71.25
	SAMR	81.12	25.58	71.04
	SAMRWS	**83.48**	25.96	72.92
TR	SARMWS	79.17	27.69	70.60
TRCE	SARMWS	61.55	32.13	58.08
PETR	SARMWS	79.01	**39.38**	**74.14**
Reference summaries		76.87	32.65	70.63

reported. Similar to measurements that are based on Rouge metrics (Tables 3 and 4), RL and PETR models achieve the highest *Blue* scores.

Since factual consistency concerns the domain of automatic TS [14, 87, 88, 96], Tables 6 and 7 report the factual consistency ("Evaluation metrics") of the experiments on *Gigaword* and *CNN/DailyMail* datasets, respectively. To compute the f_β score, we set $\beta = 0.264$ and $\beta = 0.158$ for the Gigawornd and *CNN/DailyMail* datasets, respectively, as these values represent the fractions of the average length of a summary to the average length of a text in the test sets of the datasets (the usage of β coefficient is explained in detail in "Evaluation metrics"). The reported values correspond to the average factual consistency of the usage examples of the test set for each dataset for the employed machine learning models and data schemes. Moreover, the last row of these Tables includes the factual consistency of the reference summary. Similar to *Rouge* scores, in both cases, in short-level TS (Gigaword dataset) and document-level TS (CNNDailyMail dataset), the RL and PETR deep learning models exhibit the highest scores in terms of factual consistency.

Table 7 Factual consistency on the *CNN/DailyMail* dataset of the employed deep learning models, the two graph construction methods and the simplified AMR without senses (SAMRWS) as graph transformation scheme, setting $\beta = 0.158$ for computing the FC_{f_β} metric ($p_{value} < 0.025$)

Model	Sequence of sub-graphs			Combination of sub-graphs		
	FC_p (%)	FC_r (%)	FC_{f_β} (%)	FC_p (%)	FC_r (%)	FC_{f_β} (%)
AS2SP	64.96	2.68	41.45	58.97	2.71	39.16
RL	**68.79**	**3.00**	**44.82**	**63.87**	2.66	40.91
TR	57.61	2.65	38.26	61.09	1.83	34.14
TRCE	62.72	2.65	40.38	54.55	2.28	35.00
PETR	65.45	2.97	43.27	58.89	**3.33**	**41.86**
Reference summaries	65.44	4.8	50.05	–	–	–

Table 8 Examples of short-level TS from the input text to the output summary for variations of graph transformation methods

Input text: more than one million chinese have studied abroad over the last decade, an official with the ministry of education said here on monday .	
OAMR	**Linearized semantic graph:** (s / say-01 :arg0 (p / person :arg0-of (h2 / have-org-role-91 :arg1 (m3 / ministry) :arg2 (o / official))) :arg1 (s3 / study-01 :arg0 (p2 / person :mod (c / country :name (n / name :op1 " china ") :wiki " china ") :quant (m4 / more)) :arg1 (e / educate-01) :location (a / abroad) :time (s2 / since :time (l / late))) :arg2 (m / monday) :location (h / here))
	System Summary: more than chinese students studying abroad
OAMRWS	**Linearized semantic graph:** (s / say :arg0 (p / person :arg0-of (h2 / have-org-role :arg1 (m3 / ministry) :arg2 (o / official))) :arg1 (s3 / study :arg0 (p2 / person :mod (c / country :name (n / name :op1 " china ") :wiki " china ") :quant (m4 / more)) :arg1 (e / educate) :location (a / abroad) :time (s2 / since :time (l / late))) :arg2 (m / monday) :location (h / here))
	System Summary: more chinese students studying abroad
SAMR	**Linearized semantic graph:** say-01 :arg0 (person :arg0-of (have-org-role-91 :arg1 ministry :arg2 official)) :arg1 (study-01 :arg0 (person :mod (country :name (name :op1 china) :wiki china) :quant more) :arg1 (educate-01) :location abroad :time (since :time late)) :arg2 monday :location here
	System Summary: more chinese students abroad study
SAMRWS	**Linearized semantic graph:** say :arg0 (person :arg0-of (have-org-role :arg1 ministry :arg2 official)) :arg1 (study :arg0 (person :mod (country :name (name :op1 china) :wiki china) :quant more) :arg1 (educate :arg1 (rrb :mod moe)) :location abroad :time (since :time late)) :arg2 monday :location here
	System Summary: more than one million chinese students studying abroad
	Reference summary: number of chinese students abroad exceeds one million

Table 9 Examples of document-level TS from the input text to the output summary for two graph construction methods, graph as a sequence of sub-graphs (S-SAMRWS) and combined sub-graphs (C-SAMRWS), and simplified AMR without senses (SAMRWS) as data transformation scheme

Input text: a family trip to a nebraska zoo turned terrifying for one family after the gorilla they were looking at leaped toward the exhibit window , cracking it . kevin cave caught the incident on video that he posted on his reddit page . it has already been viewed more than 1 million times . cave said when his family first arrived at omaha 's henry doorly zoo gorilla exhibit , he noticed one of the gorillas had a cut below his eye that was " bleeding a little bit . [...] even with the crack , the public was never in danger , he said , because the window has multiple layers of both glass and acrylic . kijoto is a 20 year old western lowland gorilla , according to a release on the zoo 's website . he weighs 375 pounds .	
S-SAMRWS	**Linearized semantic graph:** nebraska :arg0-of (trip :arg1 family :arg1-of (crack :arg0 (gorilla :arg0-of (turn :arg3 (terrify :arg1 (family :arg0-of (exhibit :arg1 window))) :time (after :op1 (leap))) :arg1-of (look :arg0 they)))) :mod-of zoo [EOG] reddit :poss (he :arg1-of (post :arg0 (cave :arg0-of (catch :arg1 (incident :location video)) :mod kevin) :arg2 page [EOG] view :arg0 1 :arg1 more :time already [EOG] [...] say :arg1 (release :arg1 (website :poss zoo :arg2 (gorilla :domain kijoto :mod lowland :mod (world-region :name (name :op1 west) :wiki western_world) :mod (old :quant (temporal-quantity :quant 20 :unit year [EOG] weigh :arg0 he :arg1 375 [EOG]
	System Summary: kevin cave captured the incident in henry doorly zoo in nebraska . the gorilla was captured in a video of a cave gorillas leaped toward the exhibit window. the video has already been more than 1 million views .
C-SAMRWS	**Linearized semantic graph:** nebraska :arg0-of (trip :arg1 (family :arg0-of (exhibit :arg1 window)) :arg1-of (crack :arg0 (gorilla :arg0-of (turn :arg3 (terrify :arg1 family) :time (after :op1 leap)) :arg1-of (look :arg0 they) :domain kijoto :mod lowland :mod (world-region :name (name :op1 west) :wiki western_world) :mod (old :quant (temporal-quantity :quant (20 :unit year)))))) :mod-of (zoo :arg2 (gorilla)) (say :arg1 (release :arg1 (website :poss zoo))) [...] (reddit :poss (he :arg1-of (post :arg0 (cave :arg0-of (catch :arg1 (incident :location video)) :mod kevin) :arg2 page)))(view :arg0 1 :arg1 more :time already [EOG]
	System Summary: kevin cave captured the incident in henry doorly zoo in nebraska . the gorilla was spotted terrifying a family at the henry zoo in nebraska . the gorilla was overheard fighting with one another .
	Reference summary: gorilla leaps toward exhibit window and hits it , sending family running . zoo says patrons were never in danger .

Case study

In an attempt to further illustrate the workflow and the main aspects of the proposed approach for predicting the final summaries, Tables 8 and 9 present examples of producing system summaries on the sentence level and on the document level TS, respectively. In particular, after performing AMR parsing for obtaining the particular AMR graphs ("Semantic graph parsing"), the input text has been transformed to a linearized semantic graph representation ("Graph transformations for machine learning") taking into account a graph construction technique that has been applied ("Semantic graph construction"). The employed machine learning model (in this case, the AS2SP model of Sect. "Deep learning models" has been used) is trained end-to-end to predict a system summary. Then, the deep learning model generates a summary of a given input text.

In the example of short TS (Table 8), we can see that the system summaries vary among the different data schemes. We notice that the summary gradually improved, taking its most informative form for the SAMRWS data transformation scheme. Additionally, the predicted summaries include words that do not appear in the input text. For example, in the SAMRWS data scheme, the words *number* and *exceeds* don't exist either in the original text or in the input semantic graph. However, these words are in line with the semantics and the overall content of the particular instance. This shows the potential of the present framework to generate new text or paraphrase the original content, as it is an abstractive TS approach.

In the case of document level TS (Table 9), we provide examples of predicting system summaries for both graph construction methods and the simplified AMR without senses (SAMRWS) data transformation scheme. In the instance of the semantic graph as a sequence of sub-graphs (S-SAMRWS), we observe that the successive representations of the sub-graphs are separated from each other by the token [EOG] (end of graph or sub-graph). While in the case of the combined sub-graphs (C-SAMRWS), we have a single representation of the overall graph of the whole text. The generated summaries differ from each other, and in both cases, S-SAMRWS and C-SAMRWS, the summaries contain three sentences. Also, the summaries contain words or phrases that do not exist in the original text, rephrasing the original content. As we can see, despite the few grammatical and syntactical errors, the summaries produced may be considered informative as, they capture a part of the content of the original text with salient information.

Discussion

The experimental procedure aims at examining various aspects of the proposed framework. In particular, the effect of various deep learning models along with graph construction methods and data transformation techniques, in the context of semantic graph-to-summary predictions, are discussed in "The effect of the deep learning models", "The effect of graph construction methods" and "The effect of graph transformation techniques", and that follow. Moreover, "New tokens rate in generated summaries" describes the obtained results on the *NTR* metric. Additionally, "Factual consistency" discusses on the factual consistency of the generated summaries as a metric that focuses on a qualitative evaluation. Finally, in "Summarizing the results", we made some concluding remarks concerning the framework as a whole.

The effect of the deep learning models

In this work, we examine five deep learning models ("Deep learning models") ranging from an attentive encoder-decoder network, RL model and transformer-based architectures. The experimental results validate that all of these models achieve state-of-the-art performance in the field of the AMR-based TS that we examine. The RL approach, which uses as an agent the AS2SP network, outperforms the AS2SP model. The positive results of RL may be attributed to the fact that it tries to optimize the *Rouge-L* score, which is a particular metric for TS, as opposed to the AS2SP model that minimizes an error function (i.e., negative log-likelihood loss function), which is not an evaluation metric for TS. The TRCE network, which initializes its embedding vectors by using the *BERT* model, obtains better results than the TR model, which uses random initialization for its embedding vectors. The PETR model, which is based on a *BERT* pre-trained encoder, achieves better results than the other transformer-based architectures and the AS2SP model. Finally, the RL and PETR approaches are those that stand out compared to the other examined architectures by achieving the best performance, in terms of the employed metrics (*Rouge* & *Blue* scores, *Factual Consistency*) on both datasets (*Gigaword* and *CNN/DailyMail*).

The effect of graph construction methods

The proposed graph construction methods ("Semantic graph construction") are applied in the cases that the source text contains more than one sentence, aiming at creating the semantic graph of the overall text, according to the obtained sub-graphs that correspond to the individual sentences. In the case of transformer-based deep learning models, the sequence of sub-graphs as a graph construction method is more effective than the combination of sub-graphs. On the other hand, the architectures that are based on recurrent neural networks (AS2SP and the agent of RL model) tend to obtain better evaluation scores in the case of using the combination of sub-graphs. In particular, according to the obtained *Rouge* & *Blue* scores, as well as *Factual Consistency* (Tables 4, 5 and 7), we may observe that the performance of the two graph construction methods is comparable. Therefore, the sequence of sub-graphs should be preferred to be combined with the proposed framework, as it constitutes the simplest method to create a semantic graph. This is a preferable technique because it does not require any methodology for merging sub-graphs, avoiding the additional computational cost that is created by the second method.

The effect of graph transformation techniques

We have examined four graph transformation techniques ("Graph transformations for machine learning") that correspond to four alternatives of the dataset ("Datasets"), which aims at creating an appropriate representation of a semantic graph for using it as input to a machine learning model. Ranging from a linearized version of an original semantic graph (OAMR) to a simplified version of a semantic graph without sense numbers (SAMRWS), the length of the graph representation and the number of distinct tokens that represent a semantic graph are decreased (Table 2). The data scheme with the minimum length of the input sequence and also the minimum vocabulary size (SAMRWS) achieve the best performance, according to the experimental results. This may be due to the fact that a seq2seq machine learning model is trained more efficiently by using a sequence of tokens with a reduced length and a decreased vocabulary size of the training set. In this case of the reduced vocabulary size, each input token corresponds to more usage examples for training, improving the model predictions. Also, the data schemes that use the senses of the words (e.g., say-01) appear to have more distinct tokens for representing a graph than those without sense numbers (e.g., say). The experimental results validate that the most effective graph transformation method is that of SAMRWS which is proposed to be used in the present framework.

New tokens rate in generated summaries

NTR captures the level of abstraction of the generated summaries, as it indicates the percentage of tokens in a predicted summary that do not appear in the respective original text. As we can see in Tables 3 and 4, for the *Gigaword* and *CNN/DailyMail* datasets, respectively, *NTR* tends to be inversely proportional to the vocabulary size. Data schemes with more distinct tokens (e.g., OAMR) in representing semantic graphs achieve higher *NTR* than those of reduced vocabulary size (e.g., SAMRWS). Also, the level of *NTR* varies among deep learning models for the same data scheme. The AS2SP and RL approaches seem to have slight differences between *NTR* values, while the transformer-based models appear to have the most abstraction when compared for the same data scheme.

Factual consistency

We measure the factual consistency ("Evaluation metrics") of the generated summaries, investigating the degree that the content of an original text is covered by a summary, as an effort to provide a qualitative assessment. According to the obtained results (Tables 6 and 7), the generated summaries in short level TS (*Gigaword* dataset) exhibit greater factual accuracy than those of the document level TS (*CNN/DailyMail* dataset). This may be attributed to the fact that in short level TS, the predicted summaries cover more facts of the original text, while in document level TS, the summaries are not capable of reflecting the facts of the original text sufficiently. To clarify, in the case of the *Gigaword* dataset, the average length of a text is about four times longer than that of a summary, while in *CNN/DailyMail* dataset, the summary is almost seven times shorter than the text. Therefore, when a text has much more length than the length of its summary, the factual consistency is expected to obtain decreased score due to the limited number of facts that the summary is capable of covering.

Similarly to the *Rouge* scores mentioned above, the scores of factual consistency are maximized for the SAMRWS graph transformation method. Additionally, in a comparison of graph construction methods, we obtain the best scores for using the sequence of sub-graphs as a method of creating a semantic graph of a text. Also, the experimental results validate that the PETR and RL exhibit the best performance, in terms of factual consistency.

Moreover, for some models and data schemes, the factual consistency of the generated summaries appears to be improved compared to that of the reference summaries. This is not assumed as an implication that the predicted summaries are better than the reference ones because the human-written summaries may capture the facts of an initial text with rephrased content. On the contrary, a system tends to copy words or phrases from an original text to the machine-produced summary, allowing the factual consistency to achieve high scores. Therefore, the limited score in the case of the reference summaries is not an implication that they include weaknesses. On the other hand, the high scores, especially in the case of short level TS (*Gigaword* dataset), may be assumed as a strong indication that the proposed framework generates factual consistent summaries.

Summarizing the results

The analysis that has been presented so far indicates that the proposed framework may be an efficient solution that utilizes semantic graphs in the field of abstractive TS, outperforming other state-of-the-art semantic graph-based systems, especially in the case of using reinforcement learning or transformer-based approaches. To construct a semantic graph of a text, the sequence of sub-graphs is the simplest graph construction method, which exhibits sufficient performance compared with that of the combination of sub-graphs. This is the method that is proposed for graph construction to avoid merging of the sub-graphs that require complex procedures creating additional computational cost. To create a linearized representation of a semantic graph, which is required as an input to a machine learning model, the graph transformation method that produces the most concise version of a semantic graph representation and also leads to the minimum vocabulary size achieves the best performance. The particular graph transformation method is that of the simplified AMR without senses (SAMRWS), which is proposed, as this is the most efficient one, according to the experimental results. Furthermore, *NTR* is affected by the employed data scheme and the machine learning model. *NTR* is decreased when the vocabulary size is reduced (e.g., in the SAMRWS data scheme) and this metric is maximized when the transformer-based models are used. Finally, in evaluating the factual consistency, the short level TS achieves higher factual consistency than that of document level TS. The experimental results of the factual consistency validate that a generated summary exhibits sufficient coverage of the facts of its source text.

Conclusion

In this work, a novel framework for abstractive TS that combines semantic representation of the input text along with deep learning predictions has been proposed. The framework is based on a well-defined model for constructing a semantic graph and transforming it to be utilized by a deep learning model. The workflow of the framework includes semantic graph parsing, text graph construction, graph transformations for deep learning, and deep learning predictions. In this direction, deep learning architectures were examined for predicting a summary, given an input semantic graph of a source text. As it has already been mentioned in detail, the machine learning models include an attentive encoder-decoder with a pointer generator network, RL, transformer-based architectures, and pre-trained neural language models. Overall, the methodology copes with the problem of semantic graph-to-summary learning, investigating several data schemes along with a range of deep learning models to specify the aspects and the methodology that improve the performance of a semantic graph-based abstractive TS system.

An extensive experimental procedure was conducted by using two popular datasets (*Gigaword* and *CNN/DailyMail*) in order to evaluate the performance of the proposed framework, examining various aspects of the approach. To assess the factual consistency of the generated summaries, we have introduced an extension of a particular metric in an effort to provide a qualitative evaluation. The experimental results obtained are considered promising, as they were better or comparable with other relevant approaches (i.e., approaches that are based on semantic graphs). The positive results, which have been discussed in "Discussion", may be attributed to the appropriate semantic-based data transformations and model optimization.

Although the proposed framework exhibits satisfactory performance when compared with other related semantic graph-based approaches, it can still be further enhanced. The semantic representation of input text needs to be studied more, as the the generated summaries depend on the quality of the semantic formulation of the original content. Also, further investigation in deep learning architectures, especially pre-trained neural language models or RL models with transformer-based networks as an agent, may yield better results for automatic TS and, more specifically, for semantic-based abstractive TS. Finally, automatic evaluation for TS may be further studied with respect to investigating the correlation between the proposed extension of *Functional Consistency* and human evaluation, as it may provide useful insights into improving automatic TS.

Acknowledgements
This work was supported by computational time granted from the National Infrastructures for Research and Technology S.A. (GRNET S.A.) in the National HPC facility - ARIS - under project ID pa181003-islab-med.

Author contributions
P.K. studied the related literature, designed and implemented the proposed framework, conducted the experiments and prepared the manuscript. A.S. and G.A. contribute designing the research, reviewing and preparing the manuscript. All authors read and approved the final manuscript.

Funding
Not Applicable.

Availability of data and materials
The evaluation procedure has been based on two datasets, which are widely adopted in TS. References to these datasets are included in "Datasets'.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 10 November 2022 Accepted: 21 June 2024
Published online: 14 July 2024

References
1. Gambhir M, Gupta V. Recent automatic text summarization techniques: a survey. Artif Intell Rev. 2017;47(1):1–66.
2. Widyassari AP, Rustad S, Shidik GF, Noersasongko E, Syukur A, Affandy A, et al. Review of automatic text summarization techniques & methods. J King Saud Univ-Comput Inf Sci. 2020;

3. Luhn HP. The automatic creation of literature abstracts. IBM J Res Dev. 1958;2(2):159–65.
4. Edmundson HP. New methods in automatic extracting. J ACM (JACM). 1969;16(2):264–85.
5. Filippova K, Strube M. Sentence fusion via dependency graph compression. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2008; 177–185. Association for Computational Linguistics.
6. Filippova K. Multi-sentence compression: finding shortest paths in word graphs. In: Proceedings of the 23rd International Conference on Computational Linguistics, 2010;322–330 . Association for Computational Linguistics.
7. Banarescu L, Bonial C, Cai S, Georgescu M, Griffitt K, Hermjakob U, Knight K, Koehn P, Palmer M, Schneider N. Abstract meaning representation for sembanking. In: Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, 2013; 178–186.
8. Tohidi N, Dadkhah C. A short review of abstract meaning representation applications. Model Simul Electr Electron Eng. 2022;2(3):1–9.
9. Gupta S, Gupta SK. Abstractive summarization: an overview of the state of the art. Expert Syst Appl. 2019;121:49–65.
10. Takase S, Suzuki J, Okazaki N, Hirao T, Nagata M. Neural headline generation on abstract meaning representation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 2016; 1054–1059.
11. Vlachos A, et al. Guided neural language generation for abstractive summarization using abstract meaning representation. arXiv preprint arXiv:1808.09160 2018;
12. Kouris P, Alexandridis G, Stafylopatis A. Abstractive text summarization based on deep learning and semantic content generalization. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019; 5082–5092.
13. Jin H, Wang T, Wan X. Semsum: semantic dependency guided neural abstractive summarization. In: Proceedings of the AAAI Conference on Artificial Intelligence. 2020;34:8026–33.
14. Kouris P, Alexandridis G, Stafylopatis A. Abstractive text summarization: enhancing sequence-to-sequence models using word sense disambiguation and semantic content generalization. Comput Linguist. 2021;47(4):813–59.
15. El-Kassas WS, Salama CR, Rafea AA, Mohamed HK. Automatic text summarization: a comprehensive survey. Expert Syst Appl. 2021;165: 113679.
16. Sindhu K, Seshadri K. Text summarization: a technical overview and research perspectives. Handbook of Intelligent Computing and Optimization for Sustainable Development. 2022; 261–286.
17. Suleiman D, Awajan A. Deep learning based abstractive text summarization: approaches, datasets, evaluation measures, and challenges. Math Probl Eng. 2020; 2020.
18. Dohare S, Karnick H, Gupta V. Text summarization using abstract meaning representation. arXiv preprint arXiv:1706.01678 2017;
19. Liu F, Flanigan J, Thomson S, Sadeh N, Smith NA. Toward abstractive summarization using semantic representations. arXiv preprint arXiv:1805.10399 2018;
20. Dohare S, Gupta V, Karnick H. Unsupervised semantic abstractive summarization. In: Proceedings of ACL 2018, Student Research Workshop, 2018; 74–83.
21. Mishra R, Gayen T. Automatic lossless-summarization of news articles with abstract meaning representation. Proc Comput Sci. 2018;135:178–85.
22. Lee F-T, Kedzie C, Verma N, McKeown K. An analysis of document graph construction methods for amr summarization. arXiv preprint arXiv:2111.13993 2021;
23. Flanigan J, Dyer C, Smith NA, Carbonell JG. Generation from abstract meaning representation using tree transducers. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016; 731–739.
24. Konstas I, Iyer S, Yatskar M, Choi Y, Zettlemoyer L. Neural amr: Sequence-to-sequence models for parsing and generation. arXiv preprint arXiv:1704.08381 2017;
25. van Harmelen F, van Harmelen F, Lifschitz V, Porter B. Handbook of knowledge representation. San Diego: Elsevier Science; 2007.
26. Trentelman K. Survey of knowledge representation and reasoning systems. Defence Science and Technology Organisation EDINBURGH (AUSTRALIA) 2009;
27. Sowa JF. Conceptual graphs. Found Artif Intell. 2008;3:213–37.
28. Bateman JA, Kasper RT, Moore JD, Whitney RA. A general organization of knowledge for natural language processing: the penman upper model. Technical report, USC/Information Sciences Institute, Marina del Rey, CA: Technical report; 1990.
29. Palmer M, Gildea D, Kingsbury P. The proposition bank: an annotated corpus of semantic roles. Comput Linguist. 2005;31(1):71–106.
30. Banarescu L, Bonial C, Cai S, Georgescu M, Griffitt K, Hermjakob U, Knight K, Koehn P, Palmer M, Schneider N. Abstract meaning representation (amr) 1.0 specification. In: Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL, 2012; 1533–1544.
31. Knight K, Badarau B, Baranescu L, Bonial C, Bardocz M, Griffitt K, Hermjakob U, Marcu D, Palmer M, O'Gorman T, et al. Abstract meaning representation (amr) annotation release 3.0 ldc2020t02. Web Download. Philadelphia: Linguistic Data Consortium 2020; https://doi.org/10.35111/44cy-bp51.
32. Miranda-Jiménez S, Gelbukh A, Sidorov G. Summarizing conceptual graphs for automatic summarization task. In: International Conference on Conceptual Structures, 2013; 245–253. Springer.
33. Miller GA. Wordnet: a lexical database for English. Commun ACM. 1995;38(11):39–41.
34. Fellbaum C. WordNet: an electronic lexical database. MIT press, 1998.
35. Schuler KK. VerbNet: A Broad-coverage, Comprehensive Verb Lexicon. University of Pennsylvania, 2005.
36. Vilca GCV, Cabezudo MAS. A study of abstractive summarization using semantic representations and discourse level information. In: International Conference on Text, Speech, and Dialogue, 2017; 482– 490. Springer.
37. Mann WC, Thompson SA. Rhetorical structure theory: toward a functional theory of text organization. Text-interdisciplinary J Study Discourse. 1988;8(3):243–81.

38. Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Comput Netw ISDN Syst. 1998;30(1–7):107–17.
39. Gatt A, Reiter E. Simplenlg: A realisation engine for practical applications. In: Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009), 2009;90–93.
40. Naseem T, Blodgett A, Kumaravel S, O'Gorman T, Lee Y-S, Flanigan J, Astudillo RF, Florian R, Roukos S, Schneider N. Docamr: Multi-sentence amr representation and evaluation. arXiv preprint arXiv:2112.08513 2021.
41. Alomari A, Idris N, Sabri AQM, Alsmadi I. Deep reinforcement and transfer learning for abstractive text summarization: a review. Comput Speech Lang. 2022;71:101276.
42. Rush AM, Chopra S, Weston J. A neural attention model for abstractive sentence summarization. arXiv preprint arXiv:1509.00685 2015.
43. Tai KS, Socher R, Manning CD. Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075 2015.
44. Luong M-T, Pham H, Manning CD. Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 2015.
45. Damonte M, Cohen SB, Satta G. An incremental parser for abstract meaning representation. arXiv preprint arXiv:1608.06111 2016.
46. Foland W, Martin JH. Abstract meaning representation parsing using lstm recurrent neural networks. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017; 463–472.
47. Pan JZ. In: Staab, S., Studer, R. (eds.) Resource Description Framework, Springer, Berlin, Heidelberg 2009;71–90.
48. Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms, edn. The MIT Press, 2022.
49. Almeida F, Xexéo G. Word embeddings: a survey. arXiv preprint arXiv:1901.09069 2019.
50. Liu Q, Kusner MJ, Blunsom P. A survey on contextual embeddings. arXiv preprint arXiv:2003.07278 2020.
51. Li Y, Yang T. In: Srinivasan, S. (ed.) Word embedding for understanding natural language: A Survey. Springer, Cham 2018;. 83–104.
52. Qiu X, Sun T, Xu Y, Shao Y, Dai N, Huang X. Pre-trained models for natural language processing: a survey. Sci China Technol Sci. 2020;63(10):1872–97.
53. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 2013.
54. Rong X. word2vec parameter learning explained. arXiv preprint arXiv:1411.2738 2014.
55. Devlin J, Chang M-W, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 2018.
56. See A, Liu PJ, Manning CD. Get to the point: summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368 2017.
57. Graves A, Jaitly N, Mohamed A-r. Hybrid speech recognition with deep bidirectional lstm. In: Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop On, 2013; 273–278. IEEE.
58. Lipton ZC, Berkowitz J, Elkan C. A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:1506.00019 2015.
59. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 2014.
60. Tu Z, Lu Z, Liu Y, Liu X, Li H. Modeling coverage for neural machine translation. arXiv preprint arXiv:1601.04811 2016.
61. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
62. Watt N, du Plessis MC. Dropout algorithms for recurrent neural networks. In: Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists, 2018;72–78. ACM.
63. Graves A. Sequence transduction with recurrent neural networks. arXiv preprint arXiv:1211.3711 2012.
64. Boulanger-Lewandowski N, Bengio Y, Vincent P. Audio chord recognition with recurrent neural networks. In: ISMIR, 2013; 335–340. Citeseer.
65. Paulus R, Xiong C, Socher R. A deep reinforced model for abstractive summarization. In: International Conference on Learning Representations 2018;.
66. Rennie SJ, Marcheret E, Mroueh Y, Ross J, Goel V. Self-critical sequence training for image captioning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017; 7008–7024.
67. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. In: Advances in Neural Information Processing Systems, 2017; 5998–6008.
68. Zhang H, Xu J, Wang J. Pretraining-based natural language generation for text summarization. arXiv preprint arXiv:1902.09243 2019.
69. You Y, Jia W, Liu T, Yang W. Improving abstractive document summarization with salient information modeling. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019; 2132–2141.
70. Liu Y, Lapata M. Text summarization with pretrained encoders. arXiv preprint arXiv:1908.08345 2019.
71. Xu S, Li H, Yuan P, Wu Y, He X, Zhou B. Self-attention guided copy mechanism for abstractive summarization. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020; 1355–1362.
72. Wolf T, Chaumond J, Debut L, Sanh V, Delangue C, Moi A, Cistac P, Funtowicz M, Davison J, Shleifer S, et al. Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020; 38–45.
73. Zhu Y, Kiros R, Zemel R, Salakhutdinov R, Urtasun R, Torralba A, Fidler S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In: Proceedings of the IEEE International Conference on Computer Vision,2015; 19–27.
74. Napoles C, Gormley M, Van Durme B. Annotated gigaword. In: Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction. Association for Computational Linguistics, 2012; 95–100.

75. Hermann KM, Kocisky T, Grefenstette E, Espeholt L, Kay W, Suleyman M, Blunsom P. Teaching machines to read and comprehend. In: Advances in Neural Information Processing Systems, 2015;1693–1701.
76. Joshi A, Fernández E, Alegre E. Deep learning based text summarization: approaches databases and evaluation measures. In: International Conference of Applications of Intelligent Systems 2018.
77. Nallapati R, Zhou B, dos Santos C, Gulcehre C, Xiang B. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In: Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning, pp. 280–290. Association for Computational Linguistics, Berlin, Germany. 2016; 280–290 https://doi.org/10.18653/v1/K16-1028.
78. Shi T, Keneshloo Y, Ramakrishnan N, Reddy CK. Neural abstractive text summarization with sequence-to-sequence models. arXiv preprint arXiv:1812.02303 2018.
79. Cohan A, Dernoncourt F, Kim DS, Bui T, Kim S, Chang W, Goharian N. A discourse-aware attention model for abstractive summarization of long documents. arXiv preprint arXiv:1804.05685 2018.
80. Flanigan J, Thomson S, Carbonell JG, Dyer C, Smith NA. A discriminative graph-based parser for the abstract meaning representation. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014; 1426–1436.
81. Lin C-Y. Rouge: A package for automatic evaluation of summaries. Text Summarization Branches Out 2004.
82. Chopra S, Auli M, Rush AM. 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,2016; 93–98.
83. Gao Y, Wang Y, Liu L, Guo Y, Huang H. Neural abstractive summarization fusing by global generative topics. Neural Comput Appl. 2020;32(9):5049–58.
84. Papineni K, Roukos S, Ward T, Zhu W-J. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002; 311–318.
85. Celikyilmaz A, Clark E, Gao J. Evaluation of text generation: a survey. arXiv preprint arXiv:2006.14799 2020.
86. Fabbri AR, Kryściński W, McCann B, Xiong C, Socher R, Radev D. SummEval: re-evaluating summarization evaluation. Trans Assoc Comput Linguist. 2021;9:391–409. https://doi.org/10.1162/tacl_a_00373 (https://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00373/1923949/tacl_a_00373.pdf).
87. Kryściński W, McCann B, Xiong C, Socher R. Evaluating the factual consistency of abstractive text summarization. arXiv preprint arXiv:1910.12840 2019.
88. Goodrich B, Rao V, Liu PJ, Saleh M. Assessing the factual accuracy of generated text. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019; 166–175.
89. Angeli G, Premkumar MJJ, Manning CD. Leveraging linguistic structure for open domain information extraction. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2015; 344–354.
90. Nallapati R, Xiang B, Zhou B. Sequence-to-sequence rnns for text summarization. 2016; arXiv:1602.06023.
91. Kingma DP, Ba J. Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 2014.
92. Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks. In: International Conference on Machine Learning, 2013; 1310–1318.
93. Golik P, Doetsch P, Ney H. Cross-entropy vs. squared error training: a theoretical and experimental comparison. In: Interspeech. 2013;13:1756–60.
94. Knight K, Baranescu L, Bonial C, Georgescu M, Griffitt K, Hermjakob U, Marcu D, Palmer M, Schneider N. Abstract meaning representation (amr) annotation release 1.0 ldc2014t12. Web Download. Philadelphia: Linguistic Data Consortium 2014.
95. Sakai T. Two sample t-tests for ir evaluation: Student or welch? In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2016; 1045–1048.
96. Zhang Y. Evaluating the factual correctness for abstractive summarization. CS230 Project 2019.