Feasibility of Deep Learning and Knowledge-Based Approaches for the Generation of Fictional Stories from Historical Data

Miguel García García
garciamiguel@posgrado.uimp.es

Master's Thesis
Master's Degree in Artificial Intelligence Research
Menéndez Pelayo International University, Madrid, Spain

Supervisor: Jesús Vilares Ferro, Universidade da Coruña, *jesus.vilares@udc.es*

***Abstract* — The demand for entertainment products is growing. To a greater or lesser extent, these products commonly revolve around a story, but writing one is usually hard and time consuming. Aiming to support writers with this task, we propose three different approaches for automatic story generation from historical data. Our first solution is based on automatic summarization. The second approach relies on a GPT-2 model to generate new text. Finally, our third proposal consists of a transparent, customizable knowledge-based method able to generate novel sentences. The output generated by any of them can be personalized with entities provided as input by the user. We analyze the feasibility of each of these approaches and discuss their advantages and disadvantages.**

***Index Terms* — Natural language processing, deep learning, language models, summarization, text generation.**

I. Introduction

THE demand for entertainment products that can provide us a time of enjoyment and disconnection from reality, be they books, movies or videogames, is at an all-time high and has been boosted by the recent sanitary crisis. These kinds of products usually take a story, either real or fictional, to serve as their backbone or to make their context richer and more attractive. However, writing a story can be an arduous, time-consuming process, especially for non-professional writers, as in the case of indie developers, for example. Our objective is to explore different approaches for automated generation of stories that can serve as the common thread of a videogame or as an initial basis for writers, scriptwriters or developers in the process of kickstarting a more complex, human-written story.

In this paper we propose three different approaches for automated storytelling, namely machine generation of written stories, taking as basis historical data. These solutions can be classified into two different categories: language model and knowledge-based approaches.

Language model-based approaches correspond to statistical or machine learning tools that analyze the patterns in human language in order to recognize, predict or generate words. In this category we will explore two different solutions: the first one based on automatic summarization, and the second one relying on a GPT-2 deep learning model.

On the other hand, our knowledge-based approach is focused on extracting, analyzing, and storing information from history documents. The information acquired during this learning process will be used later to generate new comprehensible stories in a transparent, customizable and extensible way.

The rest of this paper is structured as follows. Section 2 provides the reader with an overview of some related work on the topic of study and related techniques. Next, our three proposals are described in Section 3, while we analyze their performance in Section 4. Finally, Section 5 presents our conclusions and future work.

II. Related Work

The literature includes several works about knowledge extraction and generation of new stories from a corpus of data. Some of these works focus on the generation of timelines, which can be defined as ordered sequences of events, while other works focus on automated storytelling.

A first interesting publication is [1], which can be classified into the timeline generation category and whose approach to entity extraction has attracted our attention. Its authors propose a simple approach that takes as input a set of chapters from history books for their labelling, focusing on relevant words such as *kill*, *die*, or *defeat*, for example. Next, in order to generate the timelines corresponding to the events described in a document, the system extracts the named entities (e.g. names of people, places, etc.) present in the text by using Named Entity Recognition (NER) techniques [2], resolves their coreferences (e.g. to identify what entity is being represented by a pronoun)

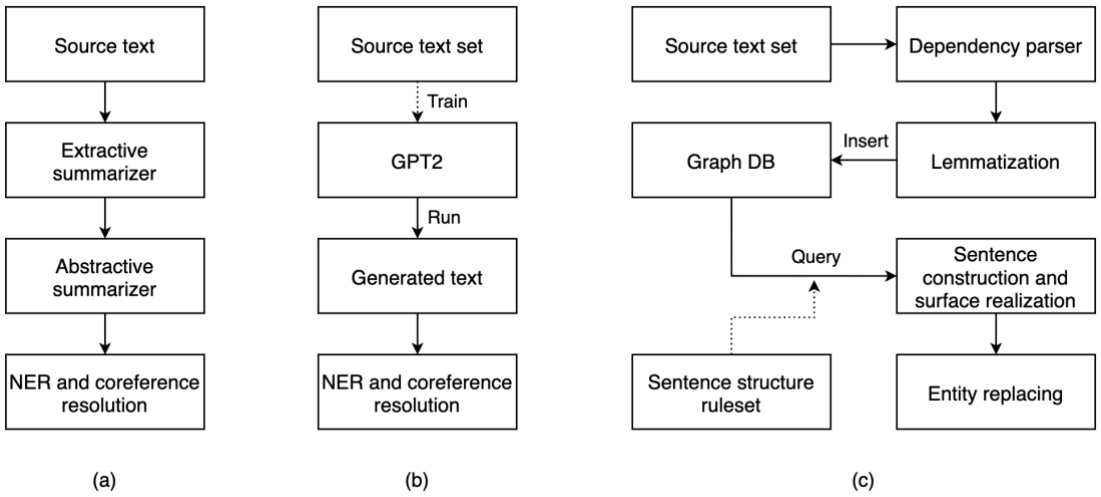

Fig. 1. General schematics of the proposed approaches: (a) summarization-based; (b) deep language model-based; and (c) knowledge-based.

through coreference resolution [3] and, finally, extracts the related temporal expressions.

In [4], a system especially dedicated to knowledge workers is proposed. It consists of a semantic layer over the document collection that facilitates the extraction of key concepts. It also relies on event templates customized by the user with information about relevant dates, people and places. In a similar way to [1], the system identifies the entities in the text through a NER process and then applies syntactic dependency parsing to identify binary relationships between them.

With regard to storytelling proposals, these are varied and usually rely on machine learning and big data techniques. Precisely for this reason, they have recently become more relevant with the surge of deep learning architectures and the advancements in hardware infrastructure.

The authors of [5] describe a storytelling unsupervised system based on neural networks. It translates the problem into an event-to-event phase based on Long Short-Term Memory networks (LSTMs) for generating the sequence of events, and an event-to-sentence phase to translate it into a natural language sentence by means of an ensemble of four different models.

In [6], the researchers describe an approach for story generation consisting of two steps: story generation and surface realization. This way, they first generate an internal representation of the plot and then they use Bidirectional Gated Recurrent Unit (BiGRU) neural network models to translate such abstract representation into actual natural language.

Finally, [7] describes two different approaches for story generation from short descriptions. The first of them interprets the task as a translation problem, while the second one interprets it as a sequence-to-sequence problem using bidirectional encoders. Both methods are found to form relatively coherent and legible stories, although they also tend to deviate from the input description.

TABLE I
SENTENCE SUBSTRUCTURES

Type	Substructure as a Neo4J query
Base (active)	(s:PROPN) <-[:nsubj]- (v:VERB) -[:dobj]-> (d)
Base (passive)	(s:PROPN) <-[:nsubjpass]- (v:VERB) -[:agent]-> (a:ADP) -[:pobj]-> (p), (v) -[:auxpass]-> (x)
Prep	(v) -[:prep]-> (pr) -[:pobj]-> (p)
Advmod	(v) -[:advmod]-> (a:ADV)
Conj	(v) -[:conj]-> (cj), (v) -[:cc]-> (c:CCONJ)
Neg	(v) -[:neg]-> (ng)

TABLE II
SENTENCE RECONSTRUCTION PRIORITIES

Part of the sentence	POS tag	Dependency type priorities
Left	VERB	nsubj, aux, nsubjpass, auxpass, neg
	PROPN	cc, conj, prep
Right	VERB	agent, cc, conj, advmod, dobj, prep
	NOUN	cc, conj, prep
	PROPN	cc, conj, prep
	ADP	pobj

III. OUR PROPOSALS

In this section, we describe each of the three approaches proposed in this work. Basic schematics of the three respective processes involved are shown in Figure 1.

A. Summarization-Based Approach

Our first proposal is based on automatic summarization [8]. The process consists of taking an input document that will act as basis, applying an abstractive summarizer on it, and then replacing the entities originally contained in the resulting summary with those provided by the user as input. Thus, we obtain a story that, although it is based on the source document,

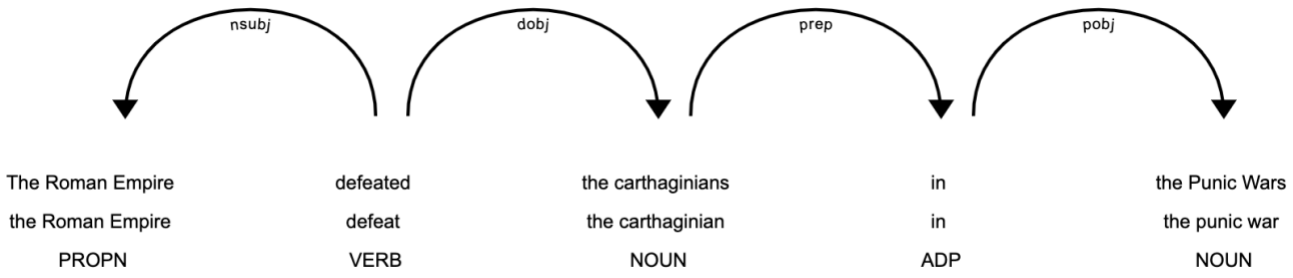

Fig. 2. Example of dependency parsing and lemmatization

it is also adapted to our needs and has a different structure than the base text.

This type of summarizer, instead of simply returning a concatenation of the most relevant sentences of a text, attempts to generate summaries more similar to those that humans would create by interpreting the original text and producing a shorter paraphrasing. In this case, we have used an abstractive summarizer based on BART [9]. It is a deep learning model that uses a standard sequence-to-sequence translation architecture with a bidirectional encoder and a left-to-right encoder. This tool achieves state-of-the-art results not only in summarization, properly speaking, but also when applied to other tasks such as question answering, for example.

However, we also need to deal with an important drawback of most deep learning models, their limited input sequence length. This summarizer in particular only supports a maximum of 1,024 tokens as input, so we need to reduce the dimensions of the text before running it. This could be done by arbitrarily discarding part of the input text, a risky procedure since it could result in the loss of important information. To avoid this, we have chosen to apply an extractive summarization [8] process instead. In contrast with abstractive summarization, extractive one does not have any restriction on its input size, thus allowing us to reduce the length of the input while preserving the relevant information it contains. This type of summarizer first identifies the most common words of the input text and selects its most relevant sentences based on the total frequency of the words it contains. As explained before, the resulting summary is then used as input for the abstractive summarizer for the generation process. However, since the intermediate summary is, ultimately, a concatenation of individual sentences from the text, this may result in a loss of context that could affect performance when generating the final abstractive summary.

Inspired by [1], the last step of this process consists of the identification and subsequent replacement of the entities present in the abstractive summary by those entities previously provided by the user as input context. For example, in the case of a text about the Punic wars, those references to *Hannibal Barca* and *Carthage* could be substituted by *Darth Vader Jr.* and *Lollipop Republic*. To do this, NER and coreference resolution techniques are applied. Both of these tasks are critical since they allow us to perform this process without losing the coherence of the text.

The advantages of this method are its conceptual simplicity and that it does not require to train any machine learning model. However, the resulting storyline will be logically similar to that of the source article selected for the summarization process.

B. Deep Language Model-Based Approach

Our second proposal is aimed at generating novel text by relying on a deep learning-based language model followed, as before, by an entity recognition and replacement phase. For this task, we use the well-known GPT-2 (for Generative Pre-trained

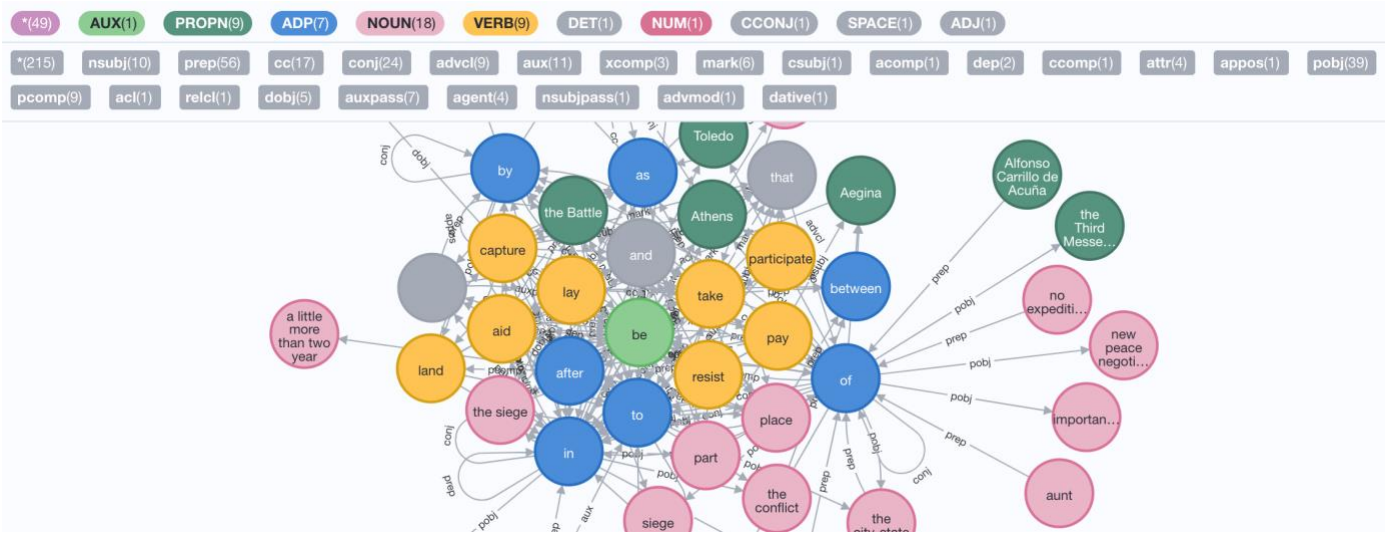

Fig. 3. Graphical representation of part of the graph database used in our knowledge-based approach. We can see some of the nodes, colored by type, and their labeled relationships.

Transformer 2) model [10] since it constitutes the state-of-the-art for text generation applications.[1]

We start with an input dataset consisting of a collection of text documents about the topic of interest, in this case Wikipedia articles from a given category (*Wars by period*) and its subcategories. As in our first proposal, an extractive summarization process has been applied on these documents to reduce them so that they can fit the maximum size allowed by the GPT-2 model. Their corresponding subcategories have been preserved so that we can teach the model to generate different text depending on the subcategory selected. A portion of the source dataset is then used to fine-tune the model to adapt it to our specific domain.

The resulting model is able to generate text both with no input at all or starting with any free input text and, thanks to its fine-tuning, the sentences generated will correspond to stories within the domain of interest (historical wars). We can also provide one of the previously mentioned Wikipedia categories as input in order to generate stories with characteristics of that specific category (e.g., *Wars of the Middle Ages*).

Moreover, if the user requires the presence of specific entities in the output stories, we rely again on a NER and coreference resolution process to replace their occurrences in the output text by those entities supplied by the user.

This second approach provides innovative stories by generating new text, leveraging the knowledge acquired by the model from a collection of documents about the desired topic.

C. Knowledge Graph-Based Approach

Our last proposal gets rid of any language model and, instead, it uses a knowledge-based approach. This way, the system feeds a knowledge database with grammatical information extracted from the texts of a source document collection about the topic of interest and then, when needed, builds new sentences by combining those pieces of information according to a guiding grammar.

So, the first step consists of building a knowledge database about the desired topic. In this case, we have taken as our information source a subset of articles from Wikipedia (in particular from the category *Wars by period*). These documents were processed using a dependency parser [11] to obtain the grammatical structure of their sentences based on the relationships between their words. This way, we lemmatize the text, associate a Part-of-Speech (POS) tag (e.g., noun, verb, adjective, etc.) to every word and identify the syntactic relationships between them, as shown in Figure 2.

All the information obtained from the source texts (lemmas, POS tags and syntactic dependencies) is collected into a graph database, in our case Neo4J [12]. The usage of a graph database enables us to easily store the words and dependencies as nodes and edges, to graphically visualize the relationships between them (as seen in Figure 3) and to query the database to obtain the desired information conveniently structured. The reason behind the use of lemmatization is to better generalize the information held into the database, since it allows us to normalize different forms of the same verb into a single node. Consequently, more node combinations are possible, thus enabling the generation of more varied sentences.

Once the knowledge database about the topic of interest is available, we can proceed with the next phase, the text generation, properly speaking. To do this, we rely on a set of predefined sentence substructures, as shown in Table I. Using this example, we randomly combine one of the two base substructures with up to four of the other substructures, and then query the database for a set of nodes and edges matching the desired skeleton. However, it must be noted that, at this point, the result obtained is not yet a sentence, but just a set of nodes and relations between them. To build an actual sentence, we must sort these elements according to the relations and a set of priorities for each dependency, as it can be seen in Table II. This process takes the verb as its starting point and performs a depth-first exploration of the relationships which correspond to those grammatical elements usually placed to the left of the main verb. Then, starting again at the verb, we repeat a similar recursive process for those elements to the right of the verb.

Moreover, since our database stores lemmas, not word forms, we also need to apply a surface realization process in order to generate understandable, syntactically correct sentences. For example, *Hannibal not kill the enemy* should be transformed into *Hannibal did not kill the enemy*. We rely on the SimpleNLG [13] Java library for this purpose.

As with our previous approaches, the last step consists of replacing the entities of the output text with those previously provided by the user. However, this time there is no need of a NER process since that information is already available through the POS tags associated to each word.

This lack of need of a NER process, together with the high level of personalization in terms of sentence structures and reconstruction strategies, are the most remarkable advantages of this knowledge-based approach. Its transparency and explainability, in contrast with the black-box nature of our previous proposals, are also relevant features to be taken into account.

IV. Experiments and Results

A. Experimental Setup

The topic chosen for testing our proposals is *Ancient Wars*. Taking this into account, the source document dataset we have used is composed of 810 articles from the *Wars by Period* Wikipedia category. Their contents were previously scraped and the irrelevant information, such as metadata, references or notes, was discarded.

In the case of our knowledge graph-based approach, and in order to not compromise the resources of our machine during the experiments, our prototype used only a subset of 50 of these articles to generate its knowledge database.

The GPT-2 model was trained on an Nvidia Tesla K80 GPU on Google Colab.[2] The rest of the experiments were performed on a standard PC with a 2.0 GHz dual-core Intel Core i5 and 8 GB of RAM.

[1] Although a more powerful model, GPT-3, is now available, it is not for public use, thus limiting its utilization.

[2] https://colab.research.google.com/

TABLE III
SAMPLE OUTPUTS

Approach	Generated text before entity replacing	Generated text after entity replacing
Summarization-based	The ottomans took advantage of the byzantine civil war of 1321–1328 to capture bursa . The city was the birthplace of ottoman architecture . After the fall of the city, the sultan's son and successor orhan built a mosque and medrese in the hisar district . After his death, orhan was buried there in his türbe (mausoleum) next to his father . The siege of prusa occurred from 1317 until the capture on 6 april 1326 . The capture established osman i (othman) and his successors as the major power in . asia minor, beginning the ottomen empire . The muslim traveler ibn battuta was impressed by the . sultan and found bursi an enjoyable city "with fine bazaars and wide streets, surrounded on all sides by . gardens and running springs" The musmuslim traveler was impressed with the city in 1331 was impressed by the sultry traveler ibn Battuta who visited bursA in . 1331 and found it was "a fine city, surrounded by running springs and gardens and gardens" The city fell only after six or nine years of the siege . The lack of expertise and . adequate siege equipment at this stage of the war meant that the city fell after six and nine years. The city of burs. was besieged, starved out of the . ottmars was besieged .	[NORP-0] took advantage of the byzantine civil war of 1321–1328 to capture bursa . [NORP-0] was the birthplace of ottoman architecture . After the fall of [NORP-0], [PERSON-0] built a mosque and [PERSON-2] in [LOC-0] . After [PERSON-0] death, [PERSON-1] was buried there in [PERSON-1] türbe (mausoleum) next to [PERSON-1] father . The siege of prusa occurred from [DATE-1] until [DATE-0] . [DATE-0] established osman i ([PERSON-3]) and his successors as the major power in . [LOC-1] minor, beginning the ottomen empire . [NORP-1] was impressed by the . sultan and found [PERSON-4] an enjoyable city "with fine bazaars and wide streets, surrounded on all sides by . [PERSON-5] and running springs" [NORP-1]was impressed with [NORP-0] in [DATE-2] was impressed by [NORP-1] in . [CARDINAL-0] and found it was "a fine city, surrounded by running springs and gardens and gardens" [NORP-0] fell only after [DATE-3] of the siege . The lack of expertise and . adequate siege equipment at this stage of the war meant that [NORP-0] fell after [DATE-4]. The city of burs. was besieged, starved out of the . [PERSON-6] was besieged .
Deep language model-based	The battle of dorya was fought at the battle of lancia in 278 bc between the forces of rome's greek senate and the forces of the sretan army. The battle was fought in a swamp, between rome's forces, led by gaius adolphus ii of labania and the forces of the greek senate in a swamp. During the course of the battle, the senate led by rome and its allies took the initiative, taking the initiative against the sretan army, which was headed by gaius adolphus ii of labania and his sons. During the course of the battle, the senate led by gaius adolphus ii of labania and his allies took the initiative against the sretan army, which was headed by gaius adolphus ii of labania and his sons. Rome sent a fleet to dorya, but the navy was defeated and rome was forced to abandon its base at labania due to the destruction of its main fleet. The battle of dorya took place in 278 bc between the forces of rome and its allies. Rome sent a fleet to dorya, but the navy was defeated and rome was forced to abandon its base at labania	[GPE-0] was fought at the battle of [ORG-5] in [CARDINAL-0] [ORG-6] between the forces of [GPE-1]'s [ORG-7] and the forces of [ORG-0]. The battle was fought in a swamp, between [GPE-1]'s forces, led by [PERSON-0] of [PERSON-5] and the forces of [ORG-1] in a swamp. During the course of [GPE-0], the [ORG-2] led by [GPE-2] took the initiative, taking [ORG-3]. During the course of [GPE-0], [ORG-1] led by [PERSON-2] and [PERSON-2] allies took [ORG-3]. [GPE-1] sent a fleet to [GPE-3], but [ORG-4] was defeated and [GPE-1] was forced to abandon [ORG-4] base at [PERSON-3] due to the destruction of [ORG-4] main fleet. [GPE-4] took place in [CARDINAL-1] bc between the forces of [GPE-2]. [GPE-1] sent a fleet to [GPE-3], but [ORG-4] was defeated and [GPE-1] was forced to abandon [ORG-4] base at [PERSON-3]
Knowledge graph-based	the Eretrians was not harmed by August in a public ceremony	[PROPN] was not harmed by [PROPN] in [NOUN]
	Simeon sent four thousand horseman at the sudden assault	[PROPN] sent [NOUN] at [NOUN]
	Telets was defeated by the tribesman's number	[PROPN] was defeated by [NOUN]
	the panicked Senate gave 300,000 for Rehoboam	[PROPN] gave [NUM] for [PROPN]

B. Model Details

As explained before, when abstractive summarization is applied, the BART [9] implementation from Huggingface Transformers [14] is used. It is a popular model, widely used by the community for document summarization tasks.

The GPT-2 model [10] is provided pretrained on the WebText corpus, a dataset composed of texts obtained by following 45 million links posted by users on the *Reddit* social network. We use Huggingface's implementation, which we have fine-tuned with our source dataset following a 70%-20%-10% train-validation-test subset scheme. The training process consisted of 5 epochs with a small 1e-5 learning rate in order to avoid overfitting and to preserve the base capabilities of the original model. When evaluating the model on the test dataset a perplexity value of 10.44 was achieved.

C. Evaluation Criteria

Measuring the quality of automatically generated sentences is a complex task with a non-minor degree of subjectivity that should be performed by a group of expert evaluators. The procedures to follow are varied. For example, by assessing aspects of the output such as its coherence, utility, etc. using a 5-point scale [15], or taking a Turing test approach [16], by showing the evaluators a mix of human and machine-written stories and asking them to classify them. Unfortunately, in our case, these procedures are not feasible, since, at this point, we lack the necessary resources. Instead, we have been forced to opt for a simpler, low-resourced procedure.

Special attention has been paid to named entities. In order to evaluate this aspect, we have manually identified the entities present in a set of 10 automatically generated texts: 5 using the summarization-based approach and the other 5 using the generation-based approach. Then, we calculated which percentage of those entities could be correctly recognized by the system. For that purpose, we considered that an entity is correctly recognized if it has the correct type (e.g., in *Cornelius was victorious*, *Cornelius* should be detected and labeled as PERSON). In the case of embedded entities such as *the battle of dorya*, we considered as valid entities both the entire string and *dorya* alone, since for our task the two solutions would be adequate. Our experiments have shown that around 80% of the entities are correctly recognized. Note that this test does not

include our knowledge graph-based approach since it does not rely on any NER process.

D. Results

Table III includes some examples of the texts generated with the three proposed approaches for the topic *Ancient Wars*. Because of their different nature, the summarization and deep language model-based approaches return long texts directly, while the knowledge graph-based approach generates sentences.

In the case of our summarization-based approach (top row), we can see that it generates a relatively coherent text, although there are also some issues related with punctuation, the unnecessary repetition of expressions such as *was impressed by*, as well as the use of two different names to denote the same city: *bursa* and *bursi*. Regarding named entity management, we can see that some entities are not properly identified; for instance, *bursa* is not detected. Moreover, some coreferences have not been accurately resolved; for example, PERSON-0 is actually the same as PERSON-1, and *the ottomans* has the same tag as the entities and pronouns referring to *the city*, *bursa*. On the other hand, the system has been able to successfully resolve references to distant entities, as in the case of *the city*, referenced at the beginning and at the end of the text.

With respect to our deep language model-based approach (mid row), we can see a well written story although also containing some incoherencies, as in the case of the opposing forces, which are not the same throughout the story. For example, at first *rome's greek senate* fights *the sretan army* but, later, *rome's forces* are said to fight *the Greek senate*. We can also find some repetitions, like *during the course of the battle*, *the senate led by*, or the fact that the date of the battle is mentioned both at the beginning and at the end of the text. When it comes to entity processing, this second approach commits some errors, like identifying *bc* as an ORG, *labania* as a PERSON or a whole phrase, *the initiative against the sretan army, which was headed by gaius adolphus ii of labania and his sons*, as an ORG. Regarding coreference resolution, it does a remarkable job, being able to trace and connect concepts throughout the whole text. There is a punctual confusion between *rome* and *the navy* when resolving the possessive determiner *its*, but it is able to resolve other determiners of the same type in other parts of the text.

As expected, our last proposal, the knowledge graph-based approach (bottom row), generates individual sentences that are correct in structure, although we can find some inaccuracies when performing the surface realization process (e.g., *the Eretrians was not harmed* does not have the proper verb conjugation). The entity recognition process is precise since it does not rely on the detection made by other tools but, instead, it uses the POS tags linked to each word. However, the biggest issue of this approach, due to the fact that it is not a proper story generator but a sentence generator, is the lack of continuity among the sentences.

V. Conclusions and Future Work

Throughout this paper we have proposed and analyzed the performance of three different approaches for the automatic generation of new stories taking as basis a collection of documents about the topic of interest, in this case historical data. The first approach relies on automatic summarization while the second one integrates text generation and deep learning techniques, including a GPT-2 deep language model. Both of them rely on NER and coreference resolution processes to identify and replace the original entities of the text with those entities eventually provided by the user as input. Our third and last proposed solution opts for a completely different approach. The first stage consists of building a knowledge database of the desired topic using a document source. Such database is then used for extracting and combining different pieces of information to create new sentences. Although our experiments have shown that the three methods yield promising results that confirm their feasibility, we believe that the higher explainability and extensibility of the knowledge graph-based approach provide it with a great potential for further exploitation.

As future work, it would be desirable to recruit a team of evaluators to extend our evaluation experiments. Another possibility would be to explore other abstractive summarization approaches that could handle larger input texts directly. Regarding the knowledge-based approach, it should be easy to add more varied sentence structures to generate richer texts. However, we consider of greater interest to study how to connect the generated sentences to form longer, coherent stories.

References

[1] B. Harsimran, P. Sangameshwar and H. Swapnil, "Event Timeline Generation from History Textbooks," in *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, Taipei, Taiwan, 2017.

[2] S. Sekine and E. Ranchhod, Eds., Named entities: recognition, classifiation and use, John Benjamins Publishing Company, 2009.

[3] R. Mitkov, Anaphora Resolution, London: Pearson Education, 2002.

[4] P. Bourgonje, J. Schneider, R. Georg and F. Sasaki, "Processing Document Collections to Automatically Extract Linked Data: Semantic Storytelling Technologies for Smart Curation Workflows," in *Proceedings of the 2nd International Workshop on Natural Language Generation and the Semantic Web (WebNLG 2016)*, Edinburgh, Scotland , 2016.

[5] P. Ammanabrolu, E. Tien, W. Cheung, Z. Luo, W. Ma, L. Martin and M. Riedl, "Guided Neural Language Generation for Automated Storytelling," in *Proceedings of the Second Workshop on Storytelling*, Florence, Italy, 2019.

[6] L. Yao, N. Peng, R. Weischedel, K. Knight, D. Zhao and R. Yan, "Plan-And-Write: Towards Better Automatic Storytelling," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, Honolulu, Hawaii, USA, 2019.

[7] P. Jain, P. Agrawal, A. Mishra, M. Sukhwani, A. Laha and K. Sankaranarayanan, "Story Generation from Sequence of Independent Short Descriptions," *ArXiv*, vol. abs/1707.05501, 2017.

[8] M. Gambhir and V. Gupta, "Recent automatic text summarization techniques: a survey," *Artificial Intelligence Review*, vol. 47, 2017.

[9] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," *ArXiv*, vol. abs/1910.13461, 2019.

[10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, "Language Models are Unsupervised Multitask Learners," OpenAI, 2019. [Online]. Available: https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[11] M. Honnibal and M. Johnson, "An Improved Non-monotonic Transition System for Dependency Parsing," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[12] J. Webber, "A programmatic introduction to neo4j," in *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*, Tucson, Arizona, 2012.

[13] A. Gatt and E. Reiter, "SimpleNLG: A realisation engine for practical applications," in *Proceedings of the 12th European Workshop on Natural Language Generation*, Athens, 2009.

[14] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu and C. Xu, "Transformers: State-of-the-Art Natural Language Processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online, 2020.

[15] A. Celikyilmaz, E. Clark and J. Gao, "Evaluation of Text Generation: A Survey," *ArXiv*, vol. abs/2006.14799, 2020.

[16] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433-460, 1950.