From Text to Knowledge Graph: Comparing Relation Extraction Methods in a Practical Context

Roos M. Bakker[1,2], Daan L. Di Scala[1]

[1]TNO Netherlands Organisation for Applied Scientific Research, Department Data Science, Kampweg 55, 3769 ZG Soesterberg, The Netherlands; roos.bakker@tno.nl, daan.discala@tno.nl.

[2]Leiden University Centre for Linguistics, Leiden University, 2311 BE Leiden, Reuvensplaats 3-4, the Netherlands

Abstract

Knowledge graphs provide structure and semantic context to unstructured data. Creating them is labour intensive: it requires a close collaboration of graph developers and domain experts. Therefore, previous work has made attempts to automate (parts of) this process, utilising information extraction methods. This paper presents a comparative analysis of methods for extracting relations, with the goal of automated knowledge graph extraction. The contributions of this paper are two-fold: 1) the creation of a small dataset containing different versions of a news message annotated with triples, and 2) a comprehensive comparison of relation extraction methods within the context of this dataset. The primary objective of this paper is to assess these methods within a real-life use case scenario, where the resulting graph should aspire to the quality standards achievable through manual development. Prior methodologies often relied on automatically extracted datasets and a limited range of relation types, consequently constraining the expressivity and richness of resulting graphs. Furthermore, these datasets typically feature short or simplified sentences, failing to reflect the complexity inherent in real-world texts like news messages or research papers. The results show that GPT models demonstrate superior performance compared to the other relation extraction methods we tested. However, in the qualitative analysis performed additionally to the evaluation metrics, it was noted that alternative approaches like REBEL and KnowGL exhibit strengths in leveraging external world knowledge to enrich the graph beyond the textual content alone. This finding underscores the importance of considering a variety of methods that not only excel in extracting relations directly from text but also incorporate supplementary knowledge sources to enhance the overall richness and depth of the resulting knowledge graph.

Keywords

Knowledge Graph Extraction, Relation Extraction, Ontology Learning, Knowledge Graphs

1. Introduction

In the digital age, the abundance of textual data presents a chance and a challenge. Knowledge graphs can aid in structuring and utilising this knowledge, leading to an increasing demand for methodologies to extract knowledge from textual sources. Knowledge graph extraction has become a popular technique to answer this demand because of their ability to organise and connect information. By representing knowledge and data in a structured graph format,

GeNeSy'24: First International Workshop on Generative Neuro-Symbolic AI, co-located with ESWC 2024, May 26, 2024, Hersonissos, Crete, Greece

*Corresponding author.

roos.bakker@tno.nl (R. M. Bakker); daan.discala@tno.nl (D. L. Di Scala)

0000-0002-1760-2740 (R. M. Bakker); 0000-0003-1548-6675 (D. L. Di Scala)

relationships between entities become explicit, allowing for reasoning, interoperability, efficient retrieval, and other downstream applications.

Creating knowledge graphs from scratch is a labour intensive task. Domain specialised models require time from domain experts and graph developers to ensure its quality. Information extraction techniques can support this process by extracting entities and relations from text. Existing fields of study that are often used in the process of knowledge graph extraction include, but are not limited to, relation extraction, Named Entity Recognition (NER), keyword extraction, and link prediction [1]. Additionally, end-to-end approaches have been suggested, which classify relations in texts utilising language models [2]. The multi-step approach suffers from its dependence on individual parts. As Jaradeh et al. [1] discuss, their text triple extractor is a weakness in their architecture due to low quality of its output. End-to-end relation extraction models are often task specific, with set relation types and do not have the flexibility to work on a range of texts [2]. Additionally, many datasets for this task are created distantly supervised, which impacts the quality.

In this paper, we compare different knowledge graph extraction techniques with the goal to create a knowledge graph that represents the text from which it is extracted in a way a graph developer would manually create it for an applied use case. For this purpose, a news message and a simple baseline text are annotated, identifying subjects, objects, and their relation. We run a collection of relation extraction methods on this dataset, and evaluate their performance. The contributions of this paper are as follows: 1) We introduce a small annotated dataset containing different versions of a news message, and 2) We carry out a comprehensive comparison of relation extraction techniques within the context of this dataset. With the dataset, containing simple and complex versions of a news message, we demonstrate how complexity affects performance. Additionally to using standard evaluation metrics (precision, recall, F1), we include a clustering coefficient to show the density of the knowledge graph. To illustrate the strengths and weaknesses of the methods that are not reflected in the metrics, we provide a qualitative analysis. Additional results and materials are included in our open source repository.[1]

In the next section, we will give an overview of knowledge graph extraction, including information extraction techniques. In Section 3, we introduce our data and annotation method and discuss the various methods. In Section 4, we will present our findings and discuss their implications in Section 5. Finally, we will conclude with a summary of our work and suggestions for future research directions in Section 6.

2. Related Work

Several fields are involved or related to the extraction of knowledge graphs, such as information extraction and ontology learning. In this section, we first elaborate on the definition of knowledge graphs, followed by a short overview of knowledge graph extraction and related fields. Afterwards, we will discuss the field of information extraction and techniques related to knowledge graph extraction.

Knowledge graphs and similar concepts such as ontologies have been around since the originating of the field of philosophy. The computer science interpretation of knowledge

[1]https://gitlab.com/genesysubmission/text2kg

modelling, with terms such as knowledge graphs, appears in literature as early as the 1970s [3], with early research on extraction from text and other sources starting a little later [4]. Knowledge graphs as a term and technique have become more popular since Google announced their implementation [5].

The knowledge graph as it is known today is a powerful representation framework that organises information in the form of interconnected nodes and edges [6]. In this graph-based structure, nodes typically represent entities (such as people, places, or concepts), while edges denote relationships between these entities. The combination of the two nodes and its relation is called a triple. This structure allows for the creation of a rich network that captures the context and connections within a dataset. Knowledge graphs enable a more nuanced and context-aware understanding of information, facilitating effective data exploration and retrieval [7]. They are therefore increasingly more used in applications where understanding and transparency of the data is essential, such as the safety or the medical domain.

2.1. Knowledge Graph Extraction

Knowledge Graph Extraction is a task that aims to extract knowledge graphs from different sources, using a variety of techniques. It is closely related to ontology learning and information extraction. Ontology learning, an established field, focuses on automatically or semi-automatically learning ontologies, including complex elements like rules and hierarchies. Information Extraction techniques are often used for extracting knowledge graphs. We will discuss often used techniques in Section 2.2. Throughout this work, we adopt the term “knowledge graph extraction”. Our objective is to extract structured information from unstructured texts, with the goal to create a knowledge graph. This term underscores our focus on this specific task, distinct from related tasks like relation extraction or specialised areas such as ontology learning.

2.1.1. Ontology Learning

Ontologies have been used as knowledge bases, reasoning tools, and schematic tools in the context of information science. Consensus is that they are stricter than knowledge graphs; in ontology terms they could be considered a subclass of knowledge graph [8]. An ontology is a formal specification of concepts in the world [9]. In other words, an ontology can represent knowledge about part of our world. For instance, an ontology about coffee can include different types of coffee, coffee machines, types of preparation, etc.

Creating ontologies is an extensive task, which involves the time of a domain expert and a modeller and requires maintenance. Therefore, automatically creating ontologies or part of them has been a fruitful field of research. Multiple overviews have been published, for instance earlier overviews based on rule-based approaches from Buitelaar et al. [10] to recent surveys including machine learning approaches from Khadir et al. [11]. Buitelaar et al. [10] give an extensive overview of the field as it was until 2005. They divide the task into complexity levels: starting with the learning of just terms and ending at the top with hierarchies, relations, and finally rules. State-of-the-art techniques were rule-based and focused on lexico-syntactical patterns. Such patterns could not consistently be identified, and the recall was low [10]. For all approaches on all levels, manual work was necessary to produce a coherent ontology.

As statistical approaches gained in popularity due to the increase of computing power and successful machine learning applications, the field changed. In their survey, Asim et al. [12] include more recent statistical approaches such as co-occurrences, hierarchical clustering, and shortly touch upon transforming ontological concepts and relations into vectors. They make the distinction between linguistic and statistical approaches, and recognise the difference between term extraction and relation extraction, with the second being the more complex task [12].

Recent advancements in ontology learning include the use of natural language processing techniques for more efficient and scalable ontology learning [13, 11]. The term ontology learning is used less, and the focus seems to have shifted to knowledge graphs. Information extraction techniques are often combined, for example in the multi-tool Plumber [1], that tries to optimise the combination of different approaches. Jaradeh et al. [1] state that current approaches are not viable for complete knowledge graph construction from unstructured text, because tasks such as keyword extraction are not enough by themselves to produce a knowledge graph. However, within the field of information extraction, the relation extraction task has been approached as an end-to-end task, which might produce a knowledge graph with its combined relations. In the next section, we discuss the field of information extraction and relevant tasks and techniques for ontology learning and knowledge graph extraction.

2.2. Information Extraction Techniques

The field of Information Extraction is closely associated with the extraction of knowledge graphs. A specific area within information extraction dedicated to knowledge graphs is Open Information Extraction (OpenIE) [14]. This technique involves generating triples, comprising a subject, relation, and object, from textual data. Multiple techniques are often combined, in a similar approach to Jaradeh et al. [1] as described above. Information Extraction as a field underwent a surge with the implementation of word embeddings [15]. These first models lead to more complex architectures such as long short-term memory models (LSTMs) and the current state-of-the-art, transformers [16], of which BERT [17] is widely used for a variety of tasks.

Currently, decoder-only models such as GPT [18] have gained prominence. They are known as generative Large Language Models (LLMs), due to the vast amounts of texts and parameters with which they are trained. They excel at absorbing and producing factual information in natural text [18], with their main purpose being language generation. However, they also show emergent behaviour on other tasks such as semantic entailment [19]. Albeit powerful models, they show limitations such as factual consistency and hallucinations [20].

Node Extraction For Knowledge Graph Extraction, multiple techniques can be combined in steps to produce a graph. A first step is the extraction of nodes. Named Entity Recognition, where entities are identified in texts, can serve as an initial step in knowledge graph extraction, although its application has been limited thus far [21]. Similarly to NER, keyword extraction can provide subjects and objects to the graph. A popular technique, TF-IDF, is based on frequency analysis together with the uncommonness of a word [22]. Recent approaches are often based on language models which are trained on this task, such as the BERT-based method

KeyBERT [17, 23]. This approach demonstrates high performances in producing accurate keywords for respective texts.

Relation Extraction Beyond extracting terms or concepts from text, determining the relationships between these concepts is crucial for creating graphs. This task is often called Relation Extraction. Similarly to early approaches of keyword extraction, early approaches of relation extraction were based on frequencies. For knowledge graph extraction, such approaches have been demonstrated by [24], with the side note that manual filtering and other steps are necessary to produce a high quality graph. Nowadays, statistical approaches are prevalent, with architectures such as graph LSTMs [25] and transformers.

Relation extraction can be done on sentences, or on paragraphs or documents. On a document level, this is still a challenging task, but the results might be more representative to the domain and comparable to manual development because relations outside sentence boundaries are also included [25]. Wang et al. [26] pre-train language models to recognise triples using relation datasets, demonstrating state-of-the-art performance. However, with state-of-the-art F1 scores going up until 0.67 for models such as [27], extracting relations on a document level is still a challenging task with room for improvement.

Extracting relations on a sentence level has been approached traditionally as a multi-step problem, similarly to ontology learning approaches described above. Recently, more end-to-end solutions have been proposed [2, 28]. Huguet Cabot and Navigli [2] introduce the REBEL model and dataset, where an encoder-decoder transformer is trained on their dataset for relation extraction. The distantly supervised dataset is created by expanding on T-REx [29]. 220 types of relations are extracted from Wikipedia abstracts, which are combined with extracted relations from Wikipedia texts using a Natural Language Inference model. The REBEL model has shown state-of-the-art performance on relation classification benchmarks. With the KnowGL model, Rossiello et al. [30] extend the REBEL dataset by adding entity labels and types, with the goal to generate a set of facts relevant for generating a knowledge graph.

Early work on generative large language models for relation extraction shows potential. Bakker et al. [31] perform a qualitative comparison of methods among which GPT-3.5 Turbo and propose a multi-step approach. However, this works lacks a quantitative analysis. Wan et al. [32] use a prompt engineering approach in a multi-step architecture for relation extraction and demonstrate that such an approach shows promise for relation classification. Allen et al. [33] outline the diverse roles of LLMs in knowledge engineering, and propose research questions, among which are questions regarding how LLMs can support the engineering of knowledge systems. Our work implements and tests one possible answer to this question: the automatic extraction of a knowledge graph from text.

3. Method

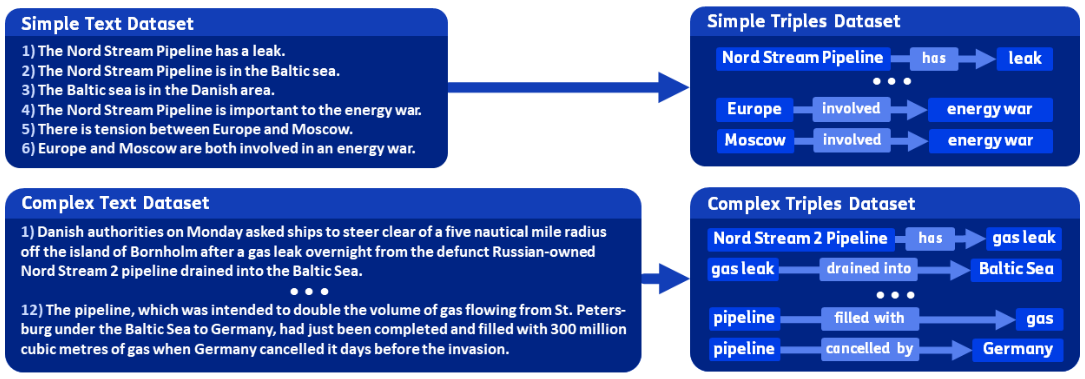
In this paper, we compare different extraction techniques with the goal to create a knowledge graph that represents the text in a way a knowledge graph developer would manually create it. For this purpose, a small annotated dataset was created, which is described in Section 3.1. We run a collection of relation extraction methods on the dataset, they are described in Section 3.2.

Finally, we evaluate their performance with the metrics described in Section 3.3.

3.1. Dataset

We study a real-life use case and compare performance of different relation extraction methods. An example of a domain where knowledge graph extraction is valuable is the Safety domain. Keeping track of multiple sources of information is critical for effective decision-making and response to emerging threats. News messages are an important source of information. By extracting structured knowledge from news messaged, safety organisations can swiftly identify and analyse relevant entities and relations. Therefore, we chose to annotate a news message that is relevant for this domain: the first news message about the Nord Stream Pipeline incident by Reuters[2]. On September 26, 2022, a series of underwater explosions and subsequent gas leaks struck the two Nord Stream natural gas pipelines. Both pipelines were inactive due to the Russian invasion of Ukraine. The leaks occurred in international waters prompting separate investigations by Denmark, Germany, and Sweden. As of January 2024, investigations continue, with the explosions characterised as sabotage and the perpetrators yet to be officially identified. The news message does not include details on the cause or involved parties.

This news message consists of two parts: 1) a report of the incident and involved parties, and 2) a collection of statements on the incident that the writer has gathered from involved parties. Knowledge graphs are better suited to factual information, therefore we decided to only include the first part of the news message as our baseline complex text. Additionally to this complex news message text, we created a simplified version of the text, containing only the key points of the news message in simple sentences. We use this simplified text with the hypothesis that relation extraction should be easier than on the news message.


Simple Text Dataset

1) The Nord Stream Pipeline has a leak.
2) The Nord Stream Pipeline is in the Baltic sea.
3) The Baltic sea is in the Danish area.
4) The Nord Stream Pipeline is important to the energy war.
5) There is tension between Europe and Moscow.
6) Europe and Moscow are both involved in an energy war.

Complex Text Dataset

1) Danish authorities on Monday asked ships to steer clear of a five nautical mile radius off the island of Bornholm after a gas leak overnight from the defunct Russian-owned Nord Stream 2 pipeline drained into the Baltic Sea.

• • •

12) The pipeline, which was intended to double the volume of gas flowing from St. Petersburg under the Baltic Sea to Germany, had just been completed and filled with 300 million cubic metres of gas when Germany cancelled it days before the invasion.

Figure 1: Overview of the dataset generation process. From the news message, a simple and complex text dataset is built, from which a simple and complex triples dataset is extracted for evaluation purposes

We annotated both texts with full triples, consisting of two nodes and the relation between them, as shown in Figure 1. We used the following conditions: 1) Each triple must be fully stated in the text, 2) each triple must indicate factual information, and 3) each triple must adhere to the (subject, relation, object) format. The first condition ensures that no common-sense

[2]https://www.reuters.com/business/energy/pressure-defunct-nord-stream-2-pipeline-plunged-overnight-operator-2022-09-26

world knowledge is included, or other information that is known to the annotator but cannot be found in the text. The second condition excludes opinions or non-factual statements such as 'Denmark's energy agency gave a statement'. The final condition excludes statements about statements, so no additional time or location information, which means the triple (operator, disclosed, pressures drop) is extracted instead of (operator, (disclosed, pressure drop, on Monday)). The constructed baseline knowledge graphs can be seen in Figures 2 and 3. The full news message, the complex and simple texts, the complex and simple triples and the graph visualisations can all be found in our repository[1].

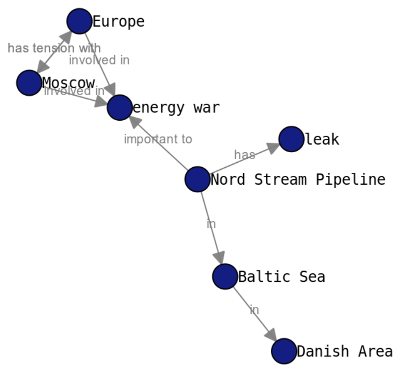


Figure 2: Simple text knowledge graph

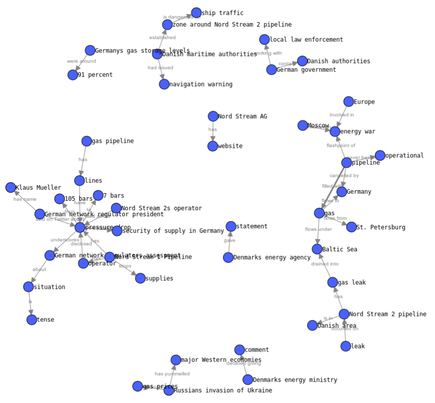


Figure 3: Complex text knowledge graph

3.2. Methods

In this section, we describe the different methods we apply on the news message and its simplified version described above. We test all methods on sentence level and on document level. These methods each have their strengths and weaknesses, which can be vary depending on the applied use case. An overview of the methods and their properties is given in Table 1. Each method is compared based on the type of information it can extract (Extracts), whether it can generate additional type information (Type info), whether weights are included in its output (Weights), whether its output follows a formal standard (Standard output), whether external knowledge outside of document input is provided in the output or just information strictly from the given text (External info), and on which model it is based on (Base model). Each method and its parameters is described below.

KeyBERT In creating a knowledge graph, selecting informative and representative nodes is crucial. Therefore, we implemented KeyBERT [23] which is based on BERT [17]. We set parameters to a top-n of 15 and a diversity of 0.5, along with Maximal Marginal Relevance (MMR) to ensure a balance between similarity and diversity in the extracted keywords. MMR minimises redundancy and maximises diversity by selecting keywords similar to the document iteratively, enhancing the quality of keywords for graph construction.

Table 1
A comparison between relation extraction methods

	Extracts	Type info	Weights	Standard output	External info	Base model
COOC	Linked entities	✗	✓	✓	text only	-
KeyBERT	Entities	✗	✓	✓	text only	BERT
REBEL	Triples	~	✗	✓	external	BART
KnowGL	Triples	✓	✗	✓	external	BART
GPT-3.5	Triples	~	✗	✗	external	GPT
GPT-4	Triples	~	✗	✗	external	GPT

Co-occurrences (COOC) While useful in a pipeline because of its high quality concepts, keyword extraction does not provide us with triples; there are no relations between the nodes. For extracting triples, we implemented several approaches. Firstly, we implemented a co-occurrence algorithm as a baseline, similarly to previous work by De Boer and Verhoosel [24] and Bakker et al. [34]. The algorithm works by analysing the frequency with which words appear together within a sentence or document. It is suitable as a baseline method because of its simplicity and interpretability. We set the maximum distance for words that occur together to 5. Further, a threshold of 0.9 is used to define a minimum amount of times a word pair must occur.

REBEL and KnowGL We also implemented the REBEL model [2]. The REBEL model is specifically designed for extracting relation triplets from raw text and is built upon the BART Transformer model [35]. We implemented the REBEL model using the number of beams and the number of return sequences settings of 5 on sentence level, and both on 30 on document level. KnowGL [30] is an extension to REBEL with entity types. We set the KnowGL parameters to the same values as REBEL.

GPT-3.5 Turbo and GPT-4 Alternatively, we implemented relation extraction using two generative LLMs. Previous approaches using these models for relation extraction showed promise [31, 32], but no extensive analysis has been performed yet. Two models from the GPT family were tested: GPT-3.5 Turbo [18] and GPT-4 [36]. We selected these models due to their superior performance on most tasks and their availability. We queried both models with the following prompt: "*Take the following document: [text], Extract all relations in this text to a graph. The graph format must be in JSON, with nodes and edges. Make sure to include the three parts of the 'subject', 'object' and 'relation' triple for each relation you find. Think carefully before you answer.*" The temperature was kept at the low setting of 0.3 to prevent hallucinations of relations that can not be found in the text.

3.3. Evaluation

To evaluate the performance of above methods on extracting nodes and triples from our dataset, we use precision, recall, and the F1 score; commonly used metrics for information retrieval tasks. Correctness of nodes and triples are compared manually to the dataset, and nodes/triples that are semantically identical to the baseline are counted as correct (e.g., 'the pipeline' instead of 'pipeline' is approved, yet 'Danish' instead of 'Danish Area' is disapproved). The triple is only evaluated as correct when both the nodes and the relation is correct.

Additionally, we measure the average Clustering Coefficient CC_{avg} [37] of each of the graphs. The CC_{avg} is based on the density of the neighbourhood surrounding each node of the graph, and is calculated by counting for each node the amount of triples it is either a subject or object of, divided by the total amount of possible triples ($n \times n - 1$, with n=total amount of nodes). With this metric, we measure the completeness of the graph. As discussed by Guéret et al. [37], the aim of knowledge graphs should not be to be fully complete ($CC_{avg} = 1$), as most links would be meaningless, but a high clustering coefficient indicates a well-cohesive graph. Finally, we offer a brief qualitative analysis highlighting distinctive qualities of various methods from the perspective of a graph developer.

4. Results

In this section, we state the results from the methods described in Section 3.2 on the dataset as discussed in Section 3.1, based on the metrics described in Section 3.3. First, we describe the performance on node level, where only the extracted entities are evaluated (Section 4.1). Second, we present the results evaluated on the triple level, considering the full extracted triples (Section 4.2). Third, we compare the extracted graphs on their density (Section 4.3). Finally, we provide a short qualitative evaluation on observations we made during evaluation (Section 4.4).

4.1. Nodes

The node extraction performance of the methods is shown in Tables 2 and 3. As shown in Table 2, performance on the simple text is overall high, with GPT-3.5 Turbo on sentence level scoring an F1 of 1, matching perfectly to the ground truth. While REBEL on document level scores lowest, on sentence level REBEL scores a perfect recall and high precision. As seen in Table 3, on the complex text, the highest precision is scored by KeyBERT on document level. Highest recall and F1-measure are by GPT-3.5 Turbo on sentence level. On the complex text, F1 and recall scores are lower on a document level in all cases except for co-occurrences. No such pattern can be observed for precision.

4.2. Triples

The triple extraction performance of the methods is shown in Tables 4 and 5. Note that due to KeyBERT only providing entities and COOC only providing linked entities (see Table 1), both methods score 0 on all metrics and are left out. Table 4 shows that both GPT-3.5 Turbo on sentence level and GPT-4 on document level score a perfect precision on the simple text, with GPT-3.5 scoring a perfect recall and F1-measure as well. KnowGL scores the lowest F1-measure on the simple text. On the complex text, as seen in Table 5, GPT-3.5 Turbo scores the highest F1-measure on sentence level, yet GPT-4 scores the highest precision on document level and recall on sentence level. KnowGL on document level notably extracted no correct triples. Recall on document level was again lower for all methods than on sentence level.

Table 2
Scores on Node level for Simple text

Method	Level	Prec.	Rec.	F1
KeyBERT	doc	0.875	1	0.933
KeyBERT	sen	0.875	1	0.933
COOC	doc	0.438	1	0.609
COOC	sen	0.438	1	0.609
REBEL	doc	0.429	0.429	0.429
REBEL	sen	0.875	1	0.933
KnowGL	doc	0.667	0.571	0.615
KnowGL	sen	0.636	1	0.778
GPT-3.5	doc	0.875	1	0.933
GPT-3.5	sen	**1**	**1**	**1**
GPT-4	doc	0.875	1	0.933
GPT-4	sen	0.875	1	0.933

Table 3
Scores on Node level for Complex text

Method	Level	Prec.	Rec.	F1
KeyBERT	doc	**0.8**	0.261	0.393
KeyBERT	sen	0.537	0.783	0.637
COOC	doc	0.232	0.696	0.348
COOC	sen	0.232	0.696	0.348
REBEL	doc	0.667	0.174	0.276
REBEL	sen	0.628	0.587	0.607
KnowGL	doc	0.571	0.087	0.151
KnowGL	sen	0.559	0.413	0.475
GPT-3.5	doc	0.604	0.63	0.617
GPT-3.5	sen	0.633	**0.826**	**0.717**
GPT-4	doc	0.657	0.5	0.568
GPT-4	sen	0.529	0.804	0.638

Table 4
Results for Simple text on Triple level

Method	Level	Prec.	Rec.	F1
REBEL	doc	0.429	0.375	0.4
REBEL	sen	0.227	0.625	0.333
KnowGL	doc	0.222	0.25	0.235
KnowGL	sen	0.167	0.375	0.231
GPT-3.5	doc	0.875	0.875	0.875
GPT-3.5	sen	**1**	**1**	**1**
GPT-4	doc	**1**	0.875	0.933
GPT-4	sen	0.75	0.75	0.75

Table 5
Results for Complex text on Triple level

Method	Level	Prec.	Rec.	F1
REBEL	doc	0.143	0.073	0.097
REBEL	sen	0.079	0.122	0.096
KnowGL	doc	0	0	-
KnowGL	sen	0.122	0.122	0.122
GPT-3.5	doc	0.333	0.195	0.246
GPT-3.5	sen	0.434	0.561	**0.489**
GPT-4	doc	**0.625**	0.244	0.351
GPT-4	sen	0.364	**0.585**	0.449

4.3. Graph Density

The density of the graphs extracted by the methods is shown in Figures 4 and 5, based on the average clustering coefficient CC_{avg} metric (see Section 3.3). KeyBERT does not produce relations and therefore the density cannot be calculated. The density of the baseline knowledge graphs is shown as a line, which is CC_{avg}=0.054 for Simple text and CC_{avg}=0.00085 for Complex text. As seen in Figure 4, on the Simple text, KnowGL on document level and REBEL both score highest and COOC on sentences has the lowest density score. The density of GPT-3.5 Turbo's graph is closest to the baseline's density. Results of the complex text can be seen in Figure 5. Because on document level REBEL and KnowGL produce high outlier density scores (CC_{avg}=0.061 for KnowGL and CC_{avg}=0.027 for REBEL), both are left out of the Figure. On complex text, REBEL and KnowGL score higher than the baseline, while GPT-4's density on document level is closest to the baseline density.

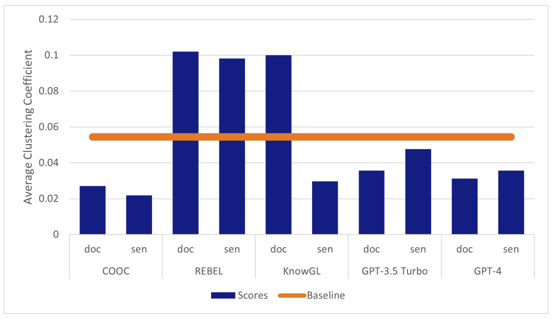


Figure 4: CC_{avg} density scores for Simple text

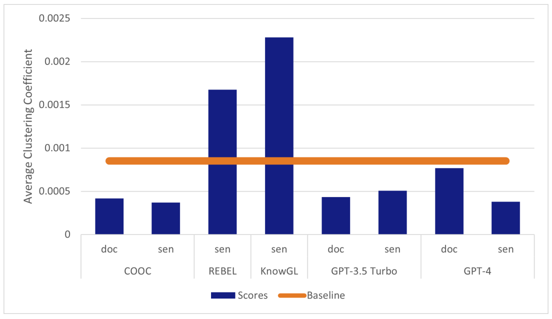


Figure 5: CC_{avg} density scores for Complex text

4.4. Qualitative Analysis

Co-occurrences (COOC) COOC generates odd results from a knowledge graph developer's standpoint. It only outputs unigrams, so many full entities are missed. For example, 'Danish', 'maritime' and 'authorities' are extracted instead of 'Danish maritime authorities'. Furthermore, many verbs are incorrectly found as entities (e.g., 'declined', 'ran'). Also, due to only providing linked relations, no triples are incorrect as no semantic value is given when a connection between subject and object does exist. For example, 'st, cooc, petersburg' while St. Petersburg exists as a city, or 'pressure, cooc, pipeline' while pipeline and pressure are connected because it is the pipeline's pressure.

KeyBERT Because KeyBERT provides no relations altogether, sometimes entities are picked up what ideally would be considered the relation of a triple. For example, the simple text baseline includes 'Europe, tension, Moscow' as (subject, relation, object)-triple, yet KeyBERT provides 'tension' as entity.

REBEL and KnowGL Where the results of the COOC method entail too little semantic value, REBEL and KnowGL often provide too much additional information. For example, KnowGL includes world knowledge, providing triples such as 'Germany, shares border with, Denmark' or 'St. Petersburg, located in or next to body of water, Baltic Sea'. While including this world knowledge can be useful to create well-connected knowledge graphs, it is rather a knowledge graph extension step with additional information outside of the text. This is less effective when generated triples are slightly incorrect or non-informative, such as 'Danish, located in or next to body of water, Danish area'.

However, opposed to other methods, KnowGL does include symmetric relations (e.g., providing both 'Nord Stream AG, owner of, website' and 'website, owned by, Nord Stream AG'), which from the perspective of knowledge graph development is an informative feature, although a proper schema might be able to infer them. Due to the REBEL classifying the text to relations it is trained on, often the relation part of the triples are semantically slightly incorrect, such as 'Nord Stream 2, product or material produced, gas' instead of 'Nord Stream 2 pipeline, is filled with, gas'.

GPT The GPT models both often provided nodes made out of multiple entities, sometimes even providing entire subclauses as nodes, such as 'currently not known what had caused the pressure drop' or 'leak today occurred on the nord stream 2 pipeline in the danish area'. Keeping the goal of knowledge graph development in mind, smaller connected entities are preferable. However, this is also a strength of this approach, as the baseline knowledge graph include occasional long entities (e.g., 'German network regulator president'), which are only picked up by GPT.

5. Discussion

In the process of annotating the news message and its simplified version, we tried to adhere to the text as much as possible. However, this approach has its limitations, as it precludes the inclusion of details that a graph developer might typically incorporate into the model. For example, by adhering to our restrictions (see Section 3.1) we did not include world knowledge facts and meta-triples which abstract over what is in the text. During annotation, we made some abstractions but chose to leave them out of the dataset, due to the difficulty in objectively determining which additional information should be incorporated. This is always a limitation of manually creating knowledge graphs, as graph developers are influenced by their world and domain knowledge on deciding what is relevant to include. For extracted knowledge graphs, flattened results without abstractions are often acceptable, but for an ontology, abstractions and external world knowledge are essential. Some methods do have such additional knowledge. During evaluation, we noticed that KnowGL, and to lesser extent REBEL, include additional triples such as 'St. Petersburg is located in Russia'. According to our annotated data, this is not counted as a correct triple. However, such additional triples containing external world knowledge might be valuable and desirable, depending on the use case.

In the results, all methods on the complex text demonstrated higher recall scores on sentence-level extraction. This makes sense, as more triples are extracted when methods are ran on sentence level. Triple counts are included in our repository[1]. The complex text yielded much lower performance scores that the simple text. While to be expected, this is an indication that the effectiveness of relation extraction is influenced by the complexity of text. For node extraction, a similar pattern could be observed. Scores were also influenced by the fact that most methods are suitable for relation extraction, not necessarily node extraction, whereas KeyBERT scores high on the simple text, and has a high precision on the complex text. However, we are not necessarily interested in separate entities, they have to be relevant for forming a knowledge graph. This is reflected in the qualitative results, where KeyBERT identifies 'tension' as an entity, where ideally this is a relation. The GPT models outputs the nodes separately from the relations, resulting in some nodes that are not part of triples. Similarly to KeyBERT, from a graph developer's perspective this is not desirable. An advantage of KeyBERT and all relation extraction methods except GPT, is that depending on the method, parameters can be set — such as the amount of results to be extracted — giving more control over the results and making such methods more suitable for pipeline usage. Overall, the results from the GPT models were of decent quality, as can be seen from their high scores. However, the output content and its format differed per run and required manual post-processing to evaluate the results. This makes this method less suitable for use in a pipeline. The output consistency can be influenced to a certain extent by utilising few-shot learning and/or function calling. Further research into these consistency methods would make a valuable addition to our work.

The Graph Density results indicate that REBEL and KnowGL are able to generate high density knowledge graphs, which might be desirable depending on the use case. However, GPT-3.5 Turbo performs closest to the baseline density on simpler texts, while GPT-4 performs closest on complex texts. Note that with the complex text, the graphs tend to have very low CC_{avg} scores overall, because the larger and more complicated the text, the more possible nodes, and thus the denominator of the formula growing exponentially. While having denser graphs as results might not always be better, it is an interesting metric as there might be use cases where you would want a highly connected knowledge graph, or have your graphs meet a threshold of connectivity, for which REBEL and KnowGL might be more useful than GPT.

6. Conclusion and Future Work

In this paper, we created a small annotated dataset containing different versions of a news message annotated with triples, and provided a comprehensive comparison of relation extraction methods within the context of this dataset. Our primary objective was to assess relation extraction methods on a real-life use case scenario, where the resulting graph should reflect a manually created graph. Our results indicate that the generative Large Language Models (LLMs) GPT-3.5 Turbo and GPT-4 outperform the other relation extraction methods we tested. However, in the qualitative analysis performed additionally to the evaluation metrics, we noted that alternative approaches like REBEL and KnowGL exhibit strengths in leveraging external world knowledge to enrich the graph beyond the textual content alone.

For future work, a comparison of a broader range of generative LLMs on this task would be a promising direction, since our results show their superior performance for knowledge graph extraction. Another future venue lies in exploring alternative methods for evaluating the quality and performance of extracted knowledge graphs, since standard metrics do not convey all qualitative aspects of an extracted graph. Additionally, our results show that on complex texts, performance of all methods is lacking. We treated this as an end-to-end task, but improvements might be made by including the methods in a pipeline with pre- and post-processing steps. Finally, extending this work with a larger and more diverse dataset of knowledge graphs based on real-world text would offer a valuable opportunity to evaluate and fine-tune language models on this task. Looking forward, such avenues offer significant potential to further advance the field of knowledge graph extraction.

Acknowledgments

We extend our gratitude to the NATO Science & Technology Organization Centre for Maritime Research & Experimentation (CMRE) for providing us with a practical use case, and we extend special thanks to Dr. Pawel Kowalski for his guidance in the maritime domain and previous work. Additionally, we would like to thank the *DEFRAUDify*, *TrustLLM*, and *RVO Causal Relations in Behavioural Modelling* projects for their support.

References

[1] M. Y. Jaradeh, K. Singh, M. Stocker, A. Both, S. Auer, Information extraction pipelines for knowledge graphs, Knowledge and Information Systems 65 (2023) 1989–2016.

[2] P.-L. Huguet Cabot, R. Navigli, REBEL: Relation extraction by end-to-end language generation, in: Findings of the Association for Computational Linguistics: EMNLP 2021, Association for Computational Linguistics, 2021, pp. 2370–2381.

[3] E. A. Feigenbaum, The art of artificial intelligence: themes and case studies of knowledge engineering, in: Proceedings of the 5th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'77, Morgan Kaufmann Publishers Inc., 1977, p. 1014–1029.

[4] R. R. Bakker, Knowledge Graphs: representation and structuring of scientific knowledge, Ph.D. thesis, University of Twente, 1987.

[5] A. Singhal, Introducing the knowledge graph: things, not strings, 2012. URL: https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[6] A. C. Khadir, H. Aliane, A. Guessoum, Ontology learning: Grand tour and challenges, Computer Science Review 39 (2021) 100339.

[7] X. Zou, A survey on application of knowledge graph, in: Journal of Physics: Conference Series, volume 1487, IOP Publishing, 2020, p. 012016.

[8] F. N. AL-Aswadi, H. Y. Chan, K. H. Gan, From ontology to knowledge graph trend: Ontology as foundation layer for knowledge graph, in: Iberoamerican Knowledge Graphs and Semantic Web Conference, Springer, 2022, pp. 330–340.

[9] R. Studer, V. R. Benjamins, D. Fensel, Knowledge engineering: Principles and methods, Data & knowledge engineering 25 (1998) 161–197.

[10] P. Buitelaar, P. Cimiano, B. Magnini, Ontology learning from text: methods, evaluation and applications, volume 123, IOS press, 2005.

[11] A. C. Khadir, H. Aliane, A. Guessoum, Ontology learning: Grand tour and challenges, Computer Science Review 39 (2021) 100339.

[12] M. N. Asim, M. Wasim, M. U. G. Khan, W. Mahmood, H. M. Abbasi, A survey of ontology learning techniques and applications, Database, The Journal of Biological Databases and Curation 2018 (2018) 1–24. doi:10.1093/database/bay101.

[13] A. Hari, P. Kumar, WSD based ontology learning from unstructured text using transformer, Procedia Computer Science 218 (2023) 367–374.

[14] C. Niklaus, M. Cetto, A. Freitas, S. Handschuh, A survey on open information extraction, arXiv preprint arXiv:1806.05599 (2018).

[15] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, Advances in neural information processing systems 26 (2013).

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 5998–6008.

[17] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[18] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child,

A. Ramesh, D. Ziegler, J. Wu, C. Winter, D. Amodei, Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.

[19] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, W. Fedus, Emergent abilities of large language models, 2022. arXiv:2206.07682.

[20] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, P. Fung, Survey of hallucination in natural language generation, ACM Computing Surveys 55 (2023). doi:10.1145/3571730.

[21] T. Al-Moslmi, M. G. Ocaña, A. L. Opdahl, C. Veres, Named entity extraction for knowledge graphs: A literature overview, IEEE Access 8 (2020) 32862–32881.

[22] G. Salton, C.-S. Yang, C. T. Yu, Contribution to the theory of indexing, Technical Report, Cornell University, 1973.

[23] M. Grootendorst, Keyword extraction with BERT, 2020. URL: https://towardsdatascience.com/keyword-extraction-with-bert-724efca412ea, accessed on: 07-03-2024.

[24] M. De Boer, J. Verhoosel, Creating and evaluating data-driven ontologies, International Journal on Advances in Software 12 (2019).

[25] N. Peng, H. Poon, C. Quirk, K. Toutanova, W.-t. Yih, Cross-sentence n-ary relation extraction with graph lstms, Transactions of the Association for Computational Linguistics 5 (2017) 101–115.

[26] C. Wang, X. Liu, Z. Chen, H. Hong, J. Tang, D. Song, Deepstruct: Pretraining of language models for structure prediction, arXiv preprint arXiv:2205.10475 (2022).

[27] Y. Ma, A. Wang, N. Okazaki, Dreeam: Guiding attention with evidence for improving document-level relation extraction, arXiv preprint arXiv:2302.08675 (2023).

[28] J. Wang, W. Lu, Two are better than one: Joint entity and relation extraction with table-sequence encoders, arXiv preprint arXiv:2010.03851 (2020).

[29] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest, E. Simperl, T-rex: A large scale alignment of natural language with knowledge base triples, in: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), 2018.

[30] G. Rossiello, M. F. M. Chowdhury, N. Mihindukulasooriya, O. Cornec, A. M. Gliozzo, Knowgl: Knowledge generation and linking from text, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 37, 2023, pp. 16476–16478.

[31] R. M. Bakker, G. J. Kalkman, I. Tolios, D. Blok, G. A. Veldhuis, S. Raaijmakers, M. H. de Boer, Exploring knowledge extraction techniques for system dynamics modelling: Comparative analysis and considerations, in: Proceedings of the Benelux Conference on Artificial Intelligence (BNAIC), 2023.

[32] Z. Wan, F. Cheng, Z. Mao, Q. Liu, H. Song, J. Li, S. Kurohashi, GPT-RE: In-context learning for relation extraction using large language models, arXiv preprint arXiv:2305.02105 (2023).

[33] B. P. Allen, L. Stork, P. Groth, Knowledge engineering using large language models (2023). arXiv:2310.00637.

[34] R. M. Bakker, M. H. de Boer, A. P. Meyer-Vitali, B. J. Bakker, S. A. Raaijmakers, A hybrid approach for creating knowledge graphs: Recognizing emerging technologies in dutch companies, HHAI2022: Augmenting Human Intellect (2022) 307–309.

[35] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, arXiv preprint arXiv:1910.13461 (2019).

[36] OpenAI, GPT-4 technical report, 2023. `arXiv:2303.08774`.

[37] C. Guéret, P. Groth, C. Stadler, J. Lehmann, Assessing linked data mappings using network measures, in: The Semantic Web: Research and Applications: 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31, 2012. Proceedings 9, Springer, 2012, pp. 87–102.