Timeline Extraction from Decision Letters Using ChatGPT

Femke Bakker
University of Amsterdam
The Netherlands
femke.bakker2@student.uva.nl

Ruben van Heusden
University of Amsterdam
The Netherlands
r.j.vanheusden@uva.nl

Maarten Marx
University of Amsterdam
The Netherlands
maartenmarx@uva.nl

Abstract

Freedom of Information Act (FOIA) legislation grants citizens the right to request information from various levels of the government, and aims to promote the transparency of governmental agencies. However, the processing of these requests is often met with delays, due to the inherent complexity of gathering the required documents. To obtain accurate estimates of the processing times of requests, and to identify bottlenecks in the process, this research proposes a pipeline to automatically extract these timelines from decision letters of Dutch FOIA requests. These decision letters are responses to requests, and contain an overview of the process, including when the request was received, and possible communication between the requester and the relevant agency. The proposed pipeline can extract dates with an accuracy of .94, extract event phrases with a mean ROUGE-L F1 score of .80 and can classify events with a macro F1 score of .79.

Out of the 50 decision letters used for testing (each letter containing one timeline), the model correctly classified 10 of the timelines completely correct, with an average of 3.1 mistakes per decision letter.

1 Introduction

Timeline extraction is the process of extracting dated events and ordering them along a timeline (Cornegruta and Vlachos, 2016). The task can be seen as a variant of event extraction, where the date is the operand of the event, and a type is associated with each date-event pair.

Our goal is to retrieve all triples of the form (date, event, event class) from a given document using a pipeline consisting of SpaCy and ChatGPT. After the triples have been extracted, we place them along a timeline to create an overview of the decision process, an example of which can be seen in Figure 1. Note that each event is grounded in the document, and can be hyperlinked to the exact position in the document, allowing for quick verification. These constructed timelines can have several purposes, such as the graphical summarization of content (Hoeve et al., 2022), as well as being a part of process mining, where the event classification helps to gain insights in the different parts and their durations in a process. Furthermore, timeline extraction over a (dynamic) corpus is a valuable tool in automatic process monitoring. Our timelines are machine interpretable overviews of processes, making it easier to control and check them in real-time. Thus, timeline extraction also creates valuable metadata about temporal relations and intervals of events and event sequencing (Allen, 1983).

This study focuses on extracting timelines from decision letters produced by the Dutch government in response to a request made under the Dutch FOIA legislation. We used SpaCy to detect and extract dates from sentences, and ChatGPT to extract the event phrases and their classes. Out of the 524 triples in the test set, roughly 76% of them were classified correctly, and out of the 50 decision letters, the timelines of 10 of them were extracted perfectly.

2 Related Work

The field of timeline extraction has seen quite some interest in recent years, and it was featured as part of the SemEval 2010 TempEval and SemEval 2015 TimeLine challenges (Pustejovsky and Verhagen, 2009; Minard et al., 2015), where several aspects, such as the grounding of dates with events as well as the creation of cross-document timelines for entities were addressed. Traditionally, the systems used for timeline extraction have consisted of pipeline approaches, with a system containing a Part-of-Speech tagger, Named Entity Recognition (NER) and coreference resolution modules

Proceedings of the 7th Workshop on Challenges and Applications of Automated Extraction of Socio-political Events from Text (CASE 2024), pages 24–31

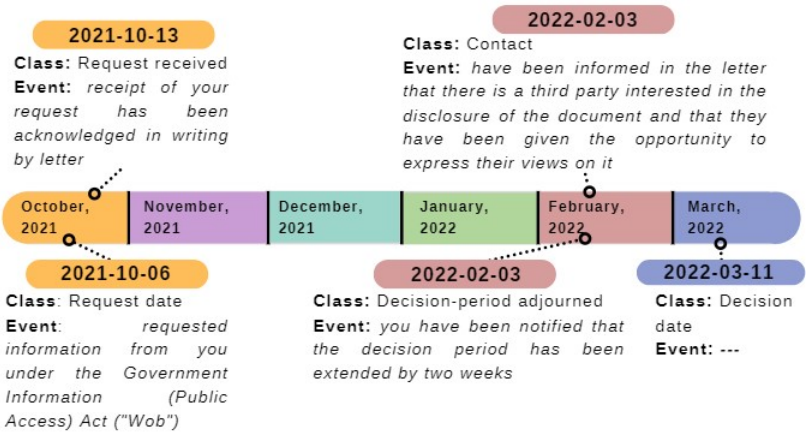

Figure 1: A timeline with five dated and classified events

in succession to extract event phrases (Aone and Ramos-Santacruz, 2000; Ahn, 2006; Minard et al., 2015), and systems such as HeidelTime (Strötgen and Gertz, 2010) to extract temporal phrases. As the annotation of these timelines can be quite expensive, some works focused on automatically constructing additional data, such as work done by Cornegruta and Vlachos (2016), who use distant supervision to create timelines for entities and use this additional data to train their pipeline model. A major downside of these pipeline approaches, as discussed by Du and Cardie (2020), is the propagation of errors from individual components in these systems, harming overall performance. The authors propose a method that does not use a pipeline, but instead uses a BERT model to extract events by posing the problem as a Question Answering task and querying the model, which is in some regards similar to our approach using ChatGPT. Another approach that replaces part of the pipeline with a neural component is work from Leeuwenberg and Moens (2018), which relies on entity annotations being present, and uses an LSTM network to predict the temporal durations of these entities, relative to each other.

With the advent of pre-trained Large Language Models (LLMs) such as ChatGPT and Llama (Touvron et al., 2023), new event extraction methods have been developed using these models (Xu et al., 2023). These methods are similar to the ones using BERT, but instead prompt these large language models to extract (actor, event, event type) triples directly from the input text. Although some of the models can be fine-tuned, a pre-trained LLM can usually perform quite well on new tasks, especially when using few-show prompting or in-context-learning. This involves providing several examples of the task that has to be performed to the model in the same prompt, helping the model in performing the task. Several techniques and best-practices for in-context learning exist, as surveyed by Dong et al. (2022). In our paper we experiment with the selection of the in-context examples, and use BM25 to select the top-k most similar datapoints from the trainingset, an approach similar to that used by Liu et al. (2021).

3 Method

3.1 Creation of the dataset

The dataset used in this research consists of 100 decision letters, written in Dutch, originating from Dutch ministries, all published in 2022. These decision letters were released as part of the WOOGLE project [1], and the documents are available as part of a curated dataset on the Dutch Scientific Data Repository (DANS)[2].

SpaCy[3] was used to extract sentences containing dates for annotation, which were subsequently filtered using regular expressions to remove false positives, resulting in a total of 812 sentences for annotation. The annotation process also included converting dates to ISO-format, such as *first of June 2021* to *01-06-2021*. The annotation was done by two annotators using an encoding scheme introduced by Schumann and QasemiZadeh (2015) for the annotation of terms and phrases in specialized domains. Events can be classified into eight possible classes, which were created through manual inspection of the decision letters, and by consulting

[1] https://woogle.wooverheid.nl/
[2] https://doi.org/10.17026/dans-zau-e3rk
[3] https://spacy.io

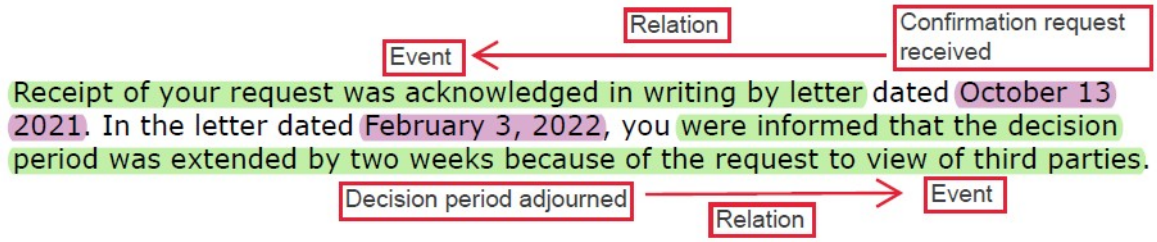

Figure 2: Example of an annotated segment of a decision letter (translated to English) with two dates each linked to one event

Task	κ	N
Date	1.0	26
Event Phrase	0.68	26
Event Class	0.91	26
Relation	0.62	26

Table 1: Inter-annotator agreement for a subset of the sentences calculated using Cohen's Kappa (N=26)

experts familiar with the Dutch FOIA legislation process.

To verify the agreement between the annotators, Cohen's Kappa was calculated for the dates, the event phrases, the event classes and the relations between dates and events (whether or not an event was linked to the correct date). The scores for these four different tasks are shown in Table 1. For the inter-annotator agreement of the event phrases, exact matching was used for the comparison, resulting in a relatively low score. However, the ROUGE-L F1 score, which measures the longest common subsequence between phrases, yields a score of 0.86, indicating a close alignment between the phrases extracted by both annotators. The Relation class also shows a relatively low score, something that is partially caused by the fact that event phrases consisting of multiple parts are rare, therefore having a large influence on the final agreement score. All event phrases consisting of a single phrase were correctly linked for both annotators. An example of an annotated text is shown in Figure 2, where two dates are linked to one event each.

The dataset was splitted equally into a training and a test set, where the two sets were split so that they each contained complete documents. The main statistics of both sets are presented in Table 2, and the distribution of the number of sentences in each document and the number of dates per sentence is shown in Figure 3. Of the 812 sentences in the dataset, 14 percent of the sentences contained more than one date.

3.2 Model

We used the *gpt-3.5-turbo-1106* checkpoint of ChatGPT, the latest iteration at the time of writing (December 2023), with the prompts written in Dutch, but translated to English for presentation in the paper. To facilitate the reproducibility of the results, the ChatGPT model was run with a temperature setting of 0.0, limiting the randomness in the output of the model. The code and dataset are publicly available on GitHub. [4]

3.3 Our timeline extraction approach

Our timeline extraction pipeline operates directly on the text extracted from a decision letter, obtained using either text extraction tools for PDF, or through optical character recognition software. Below is a brief outline of the approach, also illustrated in Figure 4.

- **Sentence Splitting** The text extracted from a PDF file is split into individual sentences with a sentence tokenizer for Dutch from NLTK.
- **Date Extraction** Sentences containing dates are extracted using SpaCy, and several rules are applied to filter out non-dates.
- **Event Phrase Extraction** Given a sentence and a list of dates, ChatGPT is prompted to return the event phrase associated with each date.
- **Event Phrase Classification** Given a list of event phrases, ChatGPT is prompted to classify the event phrase into seven possible classes.
- **Decision Date Classification** Extract the *Decision date* using regular expressions, as these dates are usually not linked to an event in text.

[4] https://github.com/irlabamsterdam/TimeLineExtractionDecisionLettersCASE

Portion	Number of Documents	Number of Sentences	Number of dates
Train	50	376	414
Test	50	445	524
Total	100	812	938

Table 2: Overview of the number of documents, number of sentences and the number of dates for both the train and test partitions

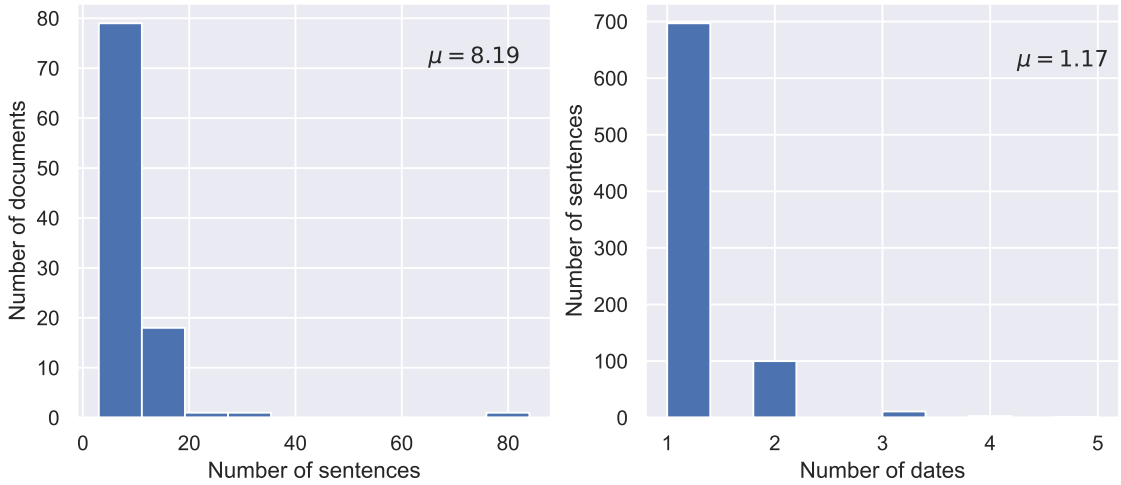

Figure 3: Number of sentences in each document for the complete dataset (N=100) and the number of dates in each sentence for all sentences in the dataset (N=812)

The individual steps of the algorithm are explained in more detail below.

Step 1: Sentence Splitting The first step in the pipeline is the splitting of the text of a document into separate sentences, which is done by using a sentence tokenizer for Dutch from NLTK. By splitting the text into sentences, sentences without dates can easily be discarded in the next step.

Step 2: Extract sentences containing dates with SpaCy In the second step, SpaCy is run to identify sentences that contain dates. This produces quite a lot of false positives, which are filtered by discarding dates that do not contain a month, as most false positives had that form.

Step 3: Extract event phrases and classes using ChatGPT The extraction and classification of the events associated with the dates from Step 2 is done by prompting the model two times. In the first step, a list of dates and a sentence containing these dates is fed to ChatGPT, and the model has to return the event phrase associated with each date, or return 'no event' if no event was detected.

Prompt: *You are given a list of dates and a sentence containing these dates. It is your task to extract the descriptions of the events happening on these dates, or to return 'No event' if no event took place on that date.*

Return your output as a list of tuples with each tuple consisting of a date and the event associated with it.

Example input: Concerning the decision on your WOO-request, October 1st, 2022

Example output: [('2020-10-01', 'Decision on your WOO-request')]

After these events were extracted, the model was prompted a second time, now with the list of event phrases, and was asked to classify them into the seven possible classes. If no event was detected in the first step then the event was automatically labelled with 'no event'.

Prompt: *You are given a list of event descriptions and it is your task to classify each of these descriptions into one of the following classes.*
1. Decision period adjourned: The decision on the WOO request has been adjourned
2. Contact: Communication took place between

the person filing the request and the relevant organization
3. WOO legislation in effect: The woo legislation came into effect, on the first of May 2020
4. Confirmation request received: The confirmation of receiving the WOO request
5. Request date: On this date a WOO request has been filed, requesting information trough the WOO legislation
6. Requested received: The WOO requested has been received by the relevant organization
7. Other: Any description that does not fall under any of the previous classes

Example input: ['you have been informed of the latest status update at the departments of ILT and RWS']

Example output: ['contact']

For both steps, the examples provided to ChatGPT were selected using an approach mentioned by Liu et al. (2021). BM25 is used to select the top examples for both the event extraction and event classification prompts. In the case of event phrase extraction, the examples were selected from the training set by retrieving the 5 most similar sentences together with their ground truth event phrases. For the classification of the event, 2 examples of similar event phrases were retrieved from the training set for each event phrase in the sentence.

Step 3: Classifying decision dates As the decision dates are usually not linked to an event in the text, but often appear at the top of a letter in a set format, these were not extracted using ChatGPT, but by using regular expressions to capture patterns such as *Datum: 2023-11-01*.

3.4 Evaluation

The evaluation of the date extraction is done by using accuracy, comparing the predicted an ground truth dates. For the evaluation of the event phrase extraction the ROUGE-L metric (Lin, 2004) is used, which computes the Longest Common Subsequence (LCS) between the tokenized representations of the ground truth and predicted texts. This metric is well-suited for the evaluation of the event extraction component, as the extracted events should be literal extracts from the letter.

To determine whether or not an extracted event phrase is correct, we follow work done by Kuhn et al. (2023) and classify an extracted event as correct only if it has a ROUGE-L F1 score of 0.5 or higher.

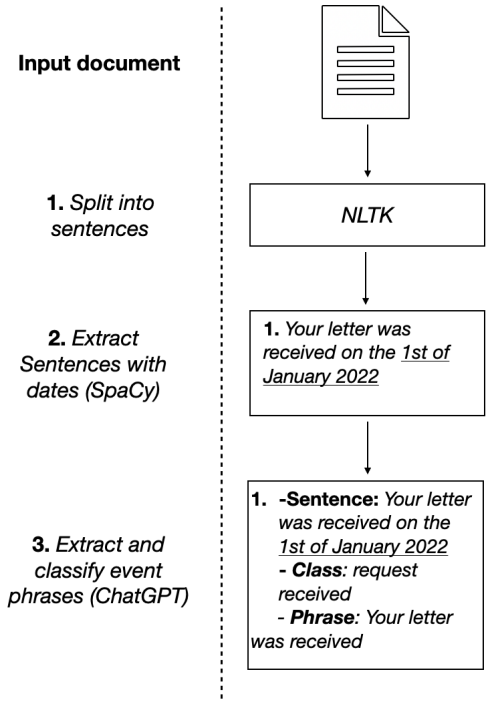

Figure 4: High level overview of the timeline extraction pipeline for an input document.

We evaluate the total accuracy of the model both in the percentage of triples that are classified correctly, as well as how many documents the pipeline classifies completely correct. For the evaluation of the event extraction and classification parts, only the triples of dates that were returned by ChatGPT were considered, as in several instances extra triples that were not in the ground truth were returned by ChatGPT. For the event classification task, the inputs to the model were the ground truth event phrases, to judge its classification performance without being influenced by the previous steps.

4 Results

4.1 Date Extraction

The date extraction part of the pipeline achieves an accuracy of .94 on all the dates in the test set. When the model is incorrect, it was usually because of ambiguity in the date, such as *In the month of June*, where the model might pick a random date belonging to that month.

4.2 Event Extraction

For the event phrase extraction, the model achieves an average ROUGE-L F1 score of .80 on the event

Table 3: Evaluation scores for event classification using ChatGPT (N=218)

	Precision	Recall	F1-score	Support
Decision period adjourned	0.93	0.93	0.93	29
Contact	0.89	0.79	0.84	42
WOO legislation in effect	1.00	1.00	1.00	16
Confirmation request received	1.00	0.98	0.99	44
Other	0.73	0.67	0.70	24
Request date	0.98	0.98	0.98	48
Request received	0.75	1.00	0.86	15

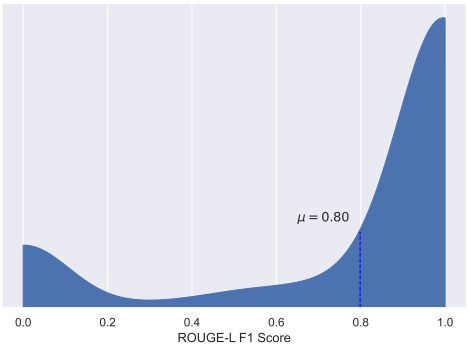

Figure 5: Distribution of the ROUGE-L F1 scores for the event phrases extracted by ChatGPT.

phrases, with a precision of .83 and a recall of .78. When thresholded at 0.5, 82% of the extracted event phrases were correct. The distribution of the ROUGE-L scores is shown in Figure 5. Although most scores are quite close to one, there is a significant number of event phrases that received a score of zero. Upon further examination it was found that these were exclusively cases where the date was not associated with an event in the ground truth, but the model still retrieved an event phrase.

4.3 Event Classification

Table 3 shows the results of the event classification with ChatGPT, with an overall macro F1 score of .79. The 'WOO legislation in effect' class achieves an almost perfect score, which is explained by the fact that this event is almost always described using the exact same phrase, simply specifying the date on which the law became effective. The model performs worst on the *other* class, which is unsurprising given the fact that this class contains all the events that could not be classified into the other classes and thus there is no clear description for what fits in this class. In these cases, the provided examples will most likely not help much either. One of the reasons that the model performs very well on the event classification task is that most of the event phrases follow similar patterns and use similar vocabulary across different documents, and thus supplying the model with similar sentences in the prompt helps in classifying the event correctly.

4.4 Decision Date Classification

The decision dates that were extracted using regular expressions achieved an accuracy of .96, where two mistakes were made out of the total of 48 triples that contained a decision date.

4.5 Timeline Construction

Out of the 524 the triples in the test set, roughly 76% of them were completely correct, where a majority of the mistakes can be attributed to ChatGPT failing to return a prediction for the date (for examples four dates being given as input put only three event phrases being returned).

Finally, we look at the correctness of the constructed timelines, where each decision letter contains exactly one timeline. In 20% of the letters, the complete timeline was extracted correctly, with the mode of the number of mistakes in a timeline being 1 and the average being 3.2. This relatively low amount of completely correct documents can be explained by the fact that documents contain on average roughly 8 dates, and thus classifying all of them correctly is quite a strict way of evaluating the performance. Although the amount of completely correct timeline is relatively low, the fact that a majority of the triples is correct and the mode of the number of mistakes is quite low, means that the graphical summarization can still be considered useful in getting a rough idea on the timelines, and mistakes can be easily spotted. Moreover, as there is a clear chronological order in the events (a request has to be received before a confirmation can be sent for example), this logic can be used to filter

out obvious mistakes in event classification, and will most likely result in even less errors.

5 Discussion

Although the proposed pipeline achieves good performance on the task of event extraction and classification for decision letters, the fact that it relies on ChatGPT, a commercial and closed-source product has certain downsides. Although we have tried to mitigate the inherent randomness in the ChatGPT model, it is possible that there are minor inconsistencies in performance between runs. A possible direction for future work is the usage of open-source LLMs such as Llama-2 to facilitate the usage of this work in practice, and to alleviate some of the aforementioned problems. The goal of this work was to evaluate a pipeline consisting of SpaCy and ChatGPT, with as little components as possible, to prevent the propagation of errors. Although several components could have been implemented by using different models, such as a parsing-based approach for the event extraction, or by using another neural model such as BERT, the fact that ChatGPT is pre-trained meant that there was very little need for training data, and a small dataset could be used for evaluating the proposed approach.

6 Conclusion

We have shown that a quite accurate timeline extractor for a specific domain can be constructed using a promptable LLM like ChatGPT, with a very limited number of training examples, in a relatively low-resource language such as Dutch, using few-shot prompting and selecting similar examples using BM25. For future work, we could look into fine-tuning an open-source LLM such as Llama for this specific task, or maybe consider generating training samples for the task using an LLM and using these to train another system.

Acknowledgements

This research was supported in part by the Netherlands Organization for Scientific Research (NWO) through the ACCESS project grant CISC.CC.016.

References

David Ahn. 2006. The Stages of Event Extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

James F Allen. 1983. Maintaining Knowledge About Temporal Intervals. *Communications of the ACM*, 26(11):832–843.

Chinatsu Aone and Mila Ramos-Santacruz. 2000. REES: A Large-Scale Relation and Event Extraction System. In *Sixth Applied Natural Language Processing Conference (ANLP)*, pages 76–83, Seattle, Washington, USA. Association for Computational Linguistics.

Savelie Cornegruta and Andreas Vlachos. 2016. Timeline Extraction using Distant Supervision and Joint Inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1936–1942.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A Survey for In-Context Learning. *arXiv preprint arXiv:2301.00234*.

Xinya Du and Claire Cardie. 2020. Event Extraction by Answering (Almost) Natural Questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683. Association for Computational Linguistics.

Maartje ter Hoeve, Julia Kiseleva, and Maarten de Rijke. 2022. Summarization with Graphical Elements. *arXiv preprint arXiv:2302.13971*.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.

Artuur Leeuwenberg and Marie-Francine Moens. 2018. Temporal Information Extraction by Predicting Relative Time-lines. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1237–1246, Brussels, Belgium. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What Makes Good In-Context Examples for GPT-3? *arXiv preprint arXiv:2101.06804*.

Anne-Lyse Myriam Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke Van Erp, Bernardo Magnini, German Rigau, and Ruben Urizar. 2015. Semeval-2015 Task 4: Timeline: Cross-Document Event Ordering. In *proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 778–786.

James Pustejovsky and Marc Verhagen. 2009. SemEval-2010 Task 13: Evaluating Events, Time Expressions, and Temporal Relations (TempEval-2). In

Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009), pages 112–116.

Anne-Kathrin Schumann and Behrang QasemiZadeh. 2015. The ACL RD-TEC Annotation Guideline: A Reference Dataset for the Evaluation of Automatic Term Recognition and Classification.

Jannik Strötgen and Michael Gertz. 2010. HeidelTime: High Quality Rule-Based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval)*, pages 321–324.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

Xin Xu, Yuqi Zhu, Xiaohan Wang, and Ningyu Zhang. 2023. How to Unleash the Power of Large Language Models for Few-shot Relation Extraction? In *Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP)*, pages 190–200, Toronto, Canada (Hybrid). Association for Computational Linguistics.