Online DATEing: A Web Interface for Temporal Annotations

Dennis Aumiller*
Institute of Computer Science, Heidelberg University
Heidelberg, Germany
aumiller@informatik.uni-heidelberg.de

Satya Almasian*
Institute of Computer Science, Heidelberg University
Heidelberg, Germany
almasian@informatik.uni-heidelberg.de

David Pohl
Institute of Computer Science, Heidelberg University
Heidelberg, Germany
pohl6@stud.uni-heidelberg.de

Michael Gertz
Institute of Computer Science, Heidelberg University
Heidelberg, Germany
gertz@informatik.uni-heidelberg.de

ABSTRACT

Despite more than two decades of research on temporal tagging and temporal relation extraction, usable tools for annotating text remain very basic and hard to set up from an average end-user perspective, limiting the applicability of developments to a selected group of invested researchers. In this work, we aim to increase the accessibility of temporal tagging systems by presenting an intuitive web interface, called "Online DATEing", which simplifies the interaction with existing temporal annotation frameworks. Our system integrates several approaches in a single interface and streamlines the process of importing (and tagging) groups of documents, as well as making it accessible through a programmatic API. It further enables users to interactively investigate and visualize tagged texts, and is designed with an extensible API for the inclusion of new models or data formats. A web demonstration of our tool is available at https://onlinedating.ifi.uni-heidelberg.de and public code accessible at https://github.com/satya77/Temporal_Tagger_Service.

CCS CONCEPTS

• **Information systems** → **Information extraction**; *Extensible Markup Language (XML)*; • **Human-centered computing** → *Information visualization.*

KEYWORDS

Temporal Tagging, Information Extraction, System Demonstration

ACM Reference Format:
Dennis Aumiller, Satya Almasian, David Pohl, and Michael Gertz. 2022. Online DATEing: A Web Interface for Temporal Annotations. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3477495.3531670

1 INTRODUCTION

Temporal information plays an important role in the field of Information Retrieval [3], by offering an additional relevance dimension. More recently, domain-specific applications have been investigated in narrative-centric task setups to assist with the creation of temporally informed representations of event timelines. Examples are the Text2Story workshop series [6] and the Financial Narrative Shared Task series [10].

In general, since the first developments in the area of temporal information extraction [22–24], the field has come a long way: several tried-and-tested datasets exist for the development of temporal taggers in multiple languages [19, 27, 28], and tagging performance on English data is getting close to the limit [9]. Yet, despite these developments, obtaining temporal annotations as an end-user remains a hassle: the few methods that do ship working implementations require manual installation, including specific setups of the runtime environment. Even then, specifications for parameters are not always intuitive to a user, or even properly documented. More often than not, one is only able to find implementations with outdated software from several years ago, or no code at all. Very few of the evaluated system are accompanied by any form of intuitive interface, of which none allows tagging more than a single document at a time. Furthermore, comparison of annotations between different methods remains cumbersome, and any form of inspection of tagging results is obscured due to the generally assumed output format in XML-like documents.

To address usability aspects for temporal tagging, in this work, we contribute:

(1) a unification of several tagging tools in a single user-friendly web service including batch processing,
(2) a visualization web interface for the comparison of different models on the fly,
(3) a streamlined tagging API to enable users to alternatively access methods programmatically to obtain data,
(4) the public code release[1], including detailed installation instructions for running several tools in the same environment, and instructions for an easy addition of new models.

We initially cover a range of models from rule-based systems to neural architectures for multiple languages, which demonstrate the integration of various input requirements and architecture-specific parameters into a single easy-to-use and intuitive API.

The remainder of this work is structured as follows: We briefly discuss the landscape of tools for temporal tagging, followed by a description of the available tagging models used in this work. Afterwards, we go over the system architecture. Lastly, we demonstrate

*These authors contributed equally to this work

[1]https://github.com/satya77/Temporal_Tagger_Service

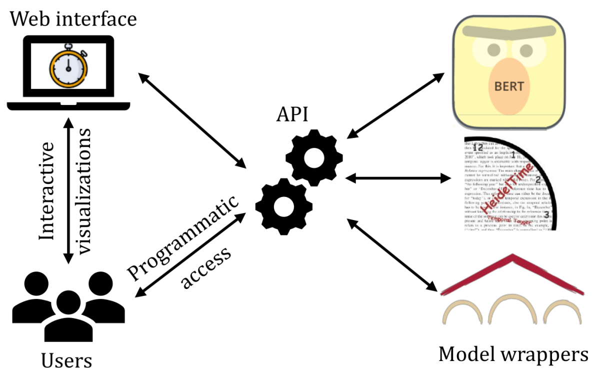


Figure 1: System architecture. Users can either interact with our system through the provided interface, or access it directly via a programmatic API. The backend binds to a diverse set of models.

intended workflows and demonstration scenarios, and conclude with a brief summary and an outlook on future extensions.

2 RELATED WORK

We primarily limit the scope of related works to tools focusing on the extraction (and normalization) of temporal expressions. For a more comprehensive overview of the field and its challenges, we refer the reader to the more extensive works by Campos et al. [5] and Nagahuna et al. [14].

Most prominently in the space of temporal tagging are HeidelTime [26] and SUTime [8]; while their performance is not necessarily state-of-the-art anymore, they remain the most frequent choices as baselines due to their relatively mature Java packages and comparatively easy integration into existing pipelines. Both also have available (and relatively recent) Python bindings.[2,3]
Further packages with publicly available code exist, but with some shortcoming: Syntime [31] and TOMN [30] provide uncompiled Java packages, UWTime [16] lacks continued support and does not run with Java 7 or newer. Python packages generally lack continued development and thus compatibility with Python3. Examples are ManTIME [11], TERNIP[4] and TEXer [13]. SpaCy[5] is probably one of the most prominent packages used nowadays, and their English models are able to tag temporal expressions in English, although without any further distinction between expression types. One extension to a more compatible TIMEX-based tagging is Timexy[6], although there is no known performance evaluation of this package on existing datasets.
More recently, Transformer-based tagging models have been introduced [1, 2, 9, 15, 25]; publicly available models (e.g., from [1, 2]) can be used through the Huggingface transformers library, but do not directly output TimeML-compatible annotations [21].

[2]https://github.com/PhilipEHausner/python_heideltime, last accessed: 2022-02-15
[3]https://github.com/FraBle/python-sutime, last accessed: 2022-02-15
[4]https://github.com/cnorthwood/ternip, last accessed: 2022-02-17
[5]https://spacy.io/, last accessed: 2022-02-17
[6]https://github.com/paulrinckens/timexy, last accessed: 2022-04-22

Rudimentary web interfaces for extracting temporal annotations exist to our knowledge only for HeidelTime and SUTime.[7,8] While SUTime allows for slightly more flexible tagging options, HeidelTime's demo is the only one to directly export annotated XML. Even then, HeidelTime's demo has no option to extract text from batches of documents. Further, none of their APIs allow for a simple visual comparison of tagging results. This is crucial, because it would give users the ability to quickly judge the appropriateness of competing annotation methods, and compare their respective results without having to manually align the hard-to-interpret XML outputs or run through several different interfaces.

3 SYSTEM IMPLEMENTATION

We briefly introduce the general setting for temporal tagging and outline the various models. For the description of our system implementation, we elaborate on the backend API server and its separate visual web interface. A conceptual view of the architecture and the interactions can be seen in Figure 1.

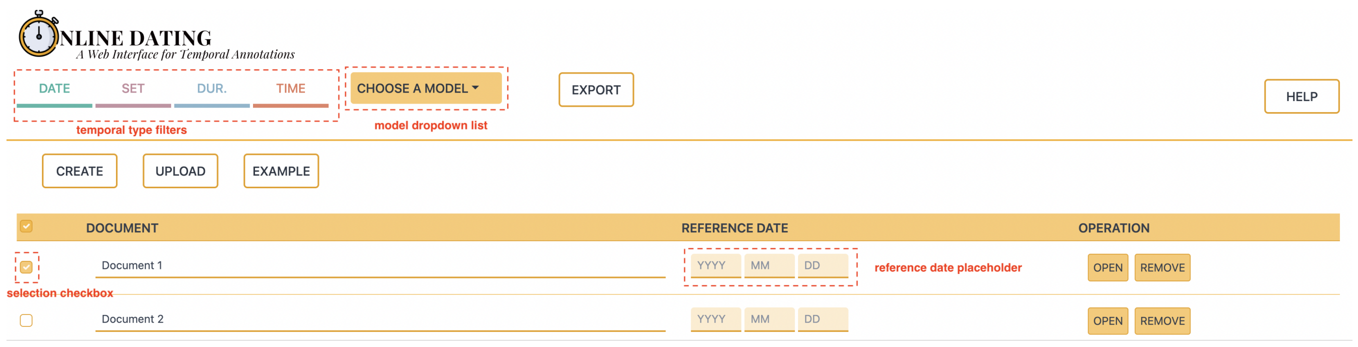
3.1 Models

Temporal tagging is the task of identification, classification, and optionally normalization of temporal information in a text. Multiple schemas for temporal tagging have been proposed, but the most prominent one is TIMEX3, which follows the XML schema defined by the TimeML standard [21]. As an example, the sentence "*Yesterday I went to the cinema*" could be tagged as "<TIMEX3 type='DATE'>*Yesterday*</TIMEX3> *I went to the cinema*". TIMEX3 tags can optionally contain a normalized representation of the date (e.g., "2022-02-20" instead of "*yesterday*"), where a reference date of the text is required to find an appropriate normalization. The type attribute represents the four available TIMEX3 classes. These are `DATE` (e.g., "28th March, 1982"), `SET` (recurring times, e.g., "every Monday"), `DURATION` (time intervals, e.g., "for two hours"), `TIME` (a more specific point in time, e.g., "tomorrow at 2pm"). Despite its widespread use, an XML file with TIMEX3 annotations is not intuitive to read and interpret for a non-expert user. Therefore, in the demo, we convert the tags to a span-based highlighting for visual inspections and use TIMEX3 only when exporting annotations.
For this demo, we include the prominent tagging tools with working Python implementations, namely HeidelTime [26] and SUTime [8] as well as Transformer-based tagging models from previous work [1, 2]. In the following, we briefly describe each model.

3.1.1 Tranformer Models. Identification of temporal expressions and assigning TIMEX3 types can be formalized as a token classification task, where the class label of each token indicates the TIMEX3 type of the temporal expression. In our case, Transformer models are fine-tuned language models for token classification of temporal tags. However, due to the data availability, these models can only be made available for a subset of languages. For English, we include three different taggers from Almasian et al. [1]:

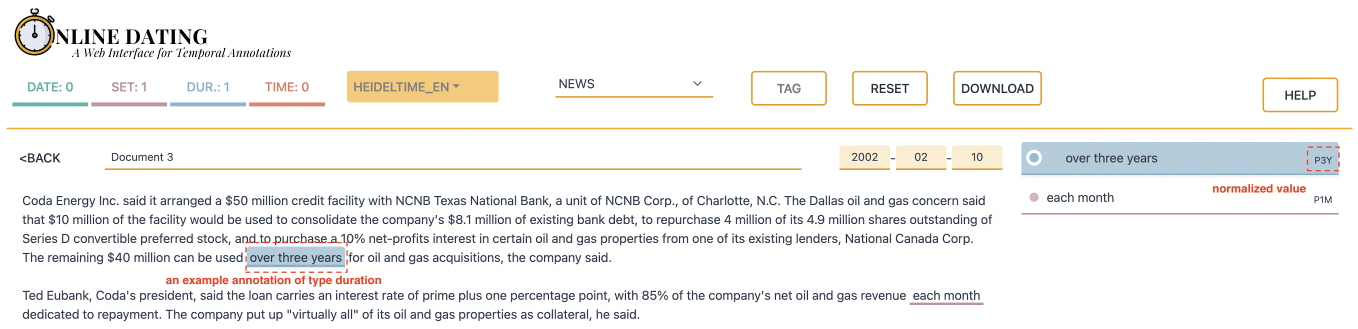
(1) **Classifier:** A BERT-based token classifier, which predicts the probability distribution over the four TIMEX3 classes (see above) for each token.

[7]https://heideltime.ifi.uni-heidelberg.de/heideltime/, last accessed: 2022-02-15
[8]http://nlp.stanford.edu:8080/sutime/process, last accessed: 2022-02-15



	DOCUMENT	REFERENCE DATE	OPERATION
✓	Document 1	YYYY MM DD	OPEN REMOVE
	Document 2	YYYY MM DD	OPEN REMOVE

(a) Document view



Coda Energy Inc. said it arranged a $50 million credit facility with NCNB Texas National Bank, a unit of NCNB Corp., of Charlotte, N.C. The Dallas oil and gas concern said that $10 million of the facility would be used to consolidate the company's $8.1 million of existing bank debt, to repurchase 4 million of its 4.9 million shares outstanding of Series D convertible preferred stock, and to purchase a 10% net-profits interest in certain oil and gas properties from one of its existing lenders, National Canada Corp. The remaining $40 million can be used over three years for oil and gas acquisitions, the company said.

Ted Eubank, Coda's president, said the loan carries an interest rate of prime plus one percentage point, with 85% of the company's net oil and gas revenue each month dedicated to repayment. The company put up "virtually all" of its oil and gas properties as collateral, he said.

(b) Editor view

Figure 2: Document overview page (top) and *editor view* (bottom). The *document view* serves as the landing page of the web interface. It allows for (batch) upload of documents and processing with specified parameters. In the *editor view*, tagging results of specific documents can be inspected, including alignment and highlighting of extracted temporal expressions. This includes color-coded distinction of different classes as well as the additional display of normalized values.

(2) **Classifier_CRF:** Similar to Classifier, but with an additional CRF layer instead of the linear layer, to account for the impact of neighboring tagging decisions on the current class label.
(3) **Classifier_DATE:** A custom BERT tagging model with an additional reference date embedding layer.

For German, we include a single model with the same architecture as the previously mentioned **Classifier** variant, based on the model presented by Almasian et al. [2]; this model has been additionally pre-trained on weakly labeled data and fine-tuned from a pre-trained checkpoint of GELECTRA-large [7]. The classification head again outputs class probabilities for each token type class, similar to the English **Classifier**.
As a caveat, token classification models only perform identification and classification of tags and are incapable of value normalization. To our knowledge, no further neural models supporting different languages exist with publicly available checkpoints, although multilingual approaches have been evaluated in the literature before [15, 25].

3.1.2 HeidelTime. A rule-based temporal tagger that uses temporal expression patterns, knowledge resources, and linguistic clues. HeidelTime both classifies and normalizes tags and is available in multiple languages. In this work, we include all languages with an explicit rule set, but do not consider the automatically translated rules. Users can further choose between different domain-specific rule sets ("news", "scientific", "narrative" or "colloquial").

3.1.3 SUTime. SUTime is another rule-based temporal tagger built on regular expression patterns, which integrates into the CoreNLP pipeline. Compared to HeidelTime, SUTime only covers American and British English, as well as Spanish.

3.1.4 Timexy. As a final component, we utilize this spaCy-based extension, which currently supports three different languages (English, German and French). Timexy also uses a rule-based pattern detection algorithm, similar to HeidelTime and SUTime. Due to its direct integration with Python, and lightweight ruleset, this method is comparably faster than other currently supported approaches.

3.2 Backend

The backbone of our API service consists of a simple routing server implemented with Flask-RESTful[9] in Python.
Since HeidelTime and SUTime are both implemented in Java, we utilize their respective Python wrappers to process documents. This adds process-related overhead to each document call, however, the respective rule-based approaches are otherwise fairly quick in their annotation. In contrast, Transformer models are called

[9] version 0.3.9

directly through Huggingface's `transformers` library [29]. Due to the size of neural architectures, annotating texts with such models is therefore mainly limited by the available CPU (or GPU) compute power.
New models can easily be added by providing a wrapper around a function call to the new library, which correctly passes the input text and optional parameters down to any new service, as long as they can be called from within Python. Direct POST requests to the API similarly need to contain a payload consisting of the respective input text, model type, and optional document reference date.
To accelerate the processing of neural models, our backend server is running on a machine with an Nvidia Titan RTX GPU. We further ensure that text segments are properly split for Transformer models to not exceed their maximum token lengths.

3.3 Web Interface

The web application is served via another Flask Python web server [10], and the interface is implemented in HTML and JavaScript. The interface accepts user input in the form of raw text files, or pasted text in the *editor view*, and passes model parameters and the request payload to the backend with AJAX. The application connects to the API with a POST request to return the tagged input. The Bootstrap dashboard from Core UI[11] and Jinja2 web template enables the interactive layout. For the web interface, we further distinguish between the following two web views:

3.3.1 Document View. The *document view*, displayed in Figure 2a, serves as the entry point for the web service. It provides a dashboard in which users can manage one or multiple files to be annotated, including an overview of already uploaded documents. Further, users may (un)select specific elements for batch processing jobs, using the selection check box on the left-hand side.
Documents can be opened in a separate view for inspection or removed from the list completely, using the respective buttons in the "OPERATION" column. We allow for multiple ways to add a new document, either by creating one from scratch or uploading from existing local files. The "CREATE" button generates a new empty document, which can be filled with content by opening it in the editor's view. Existing documents can be added with the "UPLOAD" button, which opens a separate context window where users can select one or multiple files for processing. Our interface accepts plain text files and XML documents containing a `<TEXT>` tag as input, which is the standardized representation of existing temporal tagging corpora. For demonstration, we have also included sample documents taken from the TempEval-3 dataset [28] for English and KRAUTS [27] for German samples. A random sample document can be generated by clicking on the "EXAMPLE" button.
Depending on user preferences, the interface allows for the selection of available models (cf. Section 3.1), through the model dropdown list, as well as setting the document language. When selecting a model, users are then also able to select model-specific parameters. For example, several methods also support the specification of a "reference date", which will then be used as the anchor point in the normalization step of relative temporal expressions. If needed, users can specify the document creation time of each input separately. In addition to the model specifications, users may select one or more of the four available TIMEX3 annotation classes, by enabling or disabling *temporal type filters*, which will limit the tagging process accordingly.
In the *document view*, clicking on the "EXPORT" button will trigger a batch processing job with the specified parameters of *all* selected documents at the same time, returned as a downloadable zip folder.

[10] https://flask.palletsprojects.com/en/2.0.x/, last accessed: 2022-02-19
[11] https://coreui.io/, last accessed: 2022-02-19

3.3.2 Editor View. Upon clicking the "OPEN" button on the *document view*, users may enter the *Editor view* (cf. Figure 2b). This view allows users to edit the text of a particular document. In addition, it further allows them to investigate and compare tagging results of an individual document, by previewing the extracted tags given a particular parameter choice. The model choice and type filtering is presented in a similar fashion to the processing in the *document view*. This time the type filtering section also displays the total count of each type found in the document. Each tagging result will be displayed within the text area, by underlining the temporal expression, color-coded based on the specified type colors on temporal type filters. The column on the right side of the editor contains the aggregation of all expressions. Normalized versions of the temporal expression are given if the underlying annotation model supports such a step, and subsequently requires a valid document reference date. By hovering over a particular result in either pane, the associated expression will be highlighted visually to ease the alignment of tagged texts and normalized values. To try out different models, the user can click on the "RESET" button and choose a new method from the dropdown menu. To export a single document in the *editor view*, users may click on the "EXPORT" button.

4 SYSTEM DEMONSTRATION

We describe different user workflows that can be executed with the current capabilities of our interface. In general, for all of these scenarios, one of the main advantages remains the ability to run several different methods out-of-the-box, without having to worry about individual installations.

4.1 Comparison of Tagging Results

A critical lack of functionality in previous interfaces is the inability to compare different models, due to the lack of integration of multiple annotation tools under a single interface. Since models have been generally tailored towards (or trained on) specific domains of texts, the quality of annotations can severely differ based on the input, which makes an appropriate model choice even more important.
After uploading the documents to be processed, users may inspect the annotation quality of individual samples in the *editor view*, where they can iterate through available methods and compare the annotation results and their respective quality. The comparison is possible by re-computing tagging results on a text, or opening multiple different instances of the annotation interface. This use case is also helpful to understand the particularities of different model-specific parameters at a glance and to make task-appropriate choices during initial data exploration.

4.2 Batch Processing of Document Collections

A relatively straightforward application is the processing of several texts at once, to quickly obtain annotations for a collection of documents. For this purpose, users can utilize the option to upload folders of documents via the *document view*, which is also not available in previous interfaces or would require a hand-written script for any existing software tooling.

It is both possible to provide a singular setting for all documents and process the mat once, or process select groups of documents with distinct parameter choices. This can be done by selecting the appropriate file groups in the overview menu and running with the preferred model parameters.

4.3 API Processing

In particular, due to our API design, the tool also allows for programmatic access to the annotation service. This may be desirable for several reasons:

(1) Programmatic accesses can avoid manual interaction with the web interface, eliminating user interactions, especially when dealing with a large number of documents.
(2) Users can define their own API endpoints based on custom model configurations, by extending the publicly available source. This is critical to support models with slightly different input/output specifications, e.g., due to additional parameters.
(3) Depending on the available hardware, it is possible to run the API server on a different machine, allowing multiple users to access a centrally set-up tagging interface, requiring less manual oversight over local installations.

5 CONCLUSION AND ONGOING WORK

In this work, an extensible and unified web interface for temporal tagging was presented, combining several existing annotation frameworks in a single API. We streamline the processing of groups of documents and allow users to compare tagging results of different extraction approaches.

As future extensions, the *editor view* lends itself for semi-automatic annotation tasks, where user feedback could be taken into account and models could be subsequently re-trained with feedback. For extensions of the API itself, aside from further model inclusions, we currently only support TIMEX3-compatible XML outputs; extension schemes such as SCATE [4] would be a sensible addition. Alternative options include, for example, support for temporal reasoning [17] or event relation extraction [12, 18, 20], which are more complex to annotate and visualize.

REFERENCES

[1] Satya Almasian, Dennis Aumiller, and Michael Gertz. 2022. BERT got a Date: Introducing Transformers to Temporal Tagging. (2022). forthcoming.

[2] Satya Almasian, Dennis Aumiller, and Michael Gertz. 2022. Time for some German? Pre-Training a Transformer-based Temporal Tagger for German. In *Proceedings of Text2Story - Fifth Workshop on Narrative Extraction From Texts co-located with 44nd European Conference on Information Retrieval, Text2Story@ECIR 2022, Stavanger, Norway, April 10th, 2022 (CEUR Workshop Proceedings, Vol. 3117)*, Ricardo Campos, Alípio Mário Jorge, Adam Jatowt, Sumit Bhatia, and Marina Litvak (Eds.). CEUR-WS.org, 83–90.

[3] Omar Alonso, Michael Gertz, and Ricardo A. Baeza-Yates. 2007. On the value of temporal information in information retrieval. *SIGIR Forum* 41, 2 (2007), 35–41. https://doi.org/10.1145/1328964.1328968

[4] Steven Bethard and Jonathan Parker. 2016. A Semantically Compositional Annotation Scheme for Time Normalization. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. European Language Resources Association (ELRA), Portorož, Slovenia, 3779–3786. https://www.aclweb.org/anthology/L16-1599

[5] Ricardo Campos, Gaël Dias, Alípio Mário Jorge, and Adam Jatowt. 2014. Survey of Temporal Information Retrieval and Related Applications. *ACM Comput. Surv.* 47, 2 (2014), 15:1–15:41. https://doi.org/10.1145/2619088

[6] Ricardo Campos, Alípio Jorge, Adam Jatowt, Sumit Bhatia, and Marina Litvak. 2022. The 5$^{\text{th}}$ International Workshop on Narrative Extraction from Texts: Text2Story 2022. In *Advances in Information Retrieval - 44th European Conference on IR Research, ECIR 2022, Stavanger, Norway, April 10-14, 2022, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 13186)*, Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty (Eds.). Springer, 552–556. https://doi.org/10.1007/978-3-030-99739-7_68

[7] Branden Chan, Stefan Schweter, and Timo Möller. 2020. German's Next Language Model. In *Proceedings of the 28th International Conference on Computational Linguistics*. International Committee on Computational Linguistics, Barcelona, Spain (Online), 6788–6796. https://doi.org/10.18653/v1/2020.coling-main.598

[8] Angel X. Chang and Christopher Manning. 2012. SUTime: A library for recognizing and normalizing time expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. European Language Resources Association (ELRA), Istanbul, Turkey, 3735–3740. http://www.lrec-conf.org/proceedings/lrec2012/pdf/284_Paper.pdf

[9] Sanxing Chen, Guoxin Wang, and Börje Karlsson. 2019. *Exploring Word Representations on Time Expression Recognition*. Technical Report. Microsoft Research Asia.

[10] Mahmoud El-Haj, Paul Rayson, and Andrew Moore. 2018. The First Financial Narrative Processing Workshop (FNP 2018). In *Proceedings of the LREC 2018 Workshop*.

[11] Michele Filannino, Gavin Brown, and Goran Nenadic. 2013. ManTIME: Temporal expression identification and normalization in the TempEval-3 challenge. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, 53–57. https://www.aclweb.org/anthology/S13-2009

[12] Rujun Han, Mengyue Liang, Bashar Alhafni, and Nanyun Peng. 2019. Contextualized Word Embeddings Enhanced Event Temporal Relation Extraction for Story Understanding. *CoRR* abs/1904.11942 (2019). arXiv:1904.11942 http://arxiv.org/abs/1904.11942

[13] Tianyong Hao, Alex Rusanov, and Chunhua Weng. 2013. Extracting and normalizing temporal expressions in clinical data requests from researchers. In *International Conference on Smart Health*. Springer, 41–51.

[14] Nattiya Kanhabua, Roi Blanco, and Kjetil Nørvåg. 2015. Temporal Information Retrieval. *Found. Trends Inf. Retr.* 9, 2 (2015), 91–208. https://doi.org/10.1561/1500000043

[15] Lukas Lange, Anastasiia Iurshina, Heike Adel, and Jannik Strötgen. 2020. Adversarial Alignment of Multilingual Models for Extracting Temporal Expressions from Text. In *Proceedings of the 5th Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Online, 103–109. https://doi.org/10.18653/v1/2020.repl4nlp-1.14

[16] Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent Semantic Parsing for Time Expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, 1437–1447. https://doi.org/10.3115/v1/P14-1135

[17] Artuur Leeuwenberg and Marie-Francine Moens. 2019. A Survey on Temporal Reasoning for Temporal Information Extraction from Text. *Journal of Artificial Intelligence Research* 66 (2019), 341–380. https://doi.org/10.1613/jair.1.11727

[18] Chen Lin, Timothy Miller, Dmitriy Dligach, Steven Bethard, and Guergana Savova. 2017. Representations of Time Expressions for Temporal Relation Extraction with Convolutional Neural Networks. In *BioNLP 2017*. Association for Computational Linguistics, Vancouver, Canada,, 322–327. https://doi.org/10.18653/v1/W17-2341

[19] Hector Llorens, Estela Saquete, and Borja Navarro. 2010. TIPSem (English and Spanish): Evaluating CRFs and Semantic Roles in TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Uppsala, Sweden, 284–291. https://www.aclweb.org/anthology/S10-1063

[20] Yuanliang Meng, Anna Rumshisky, and Alexey Romanov. 2017. Temporal Information Extraction for Question Answering Using Syntactic Dependencies in an LSTM-based Architecture. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 887–896. https://doi.org/10.18653/v1/D17-1092

[21] James Pustejovsky, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. ISO-TimeML: An International Standard for Semantic Annotation. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, Valletta, Malta*. European Language Resources Association.

[22] Dragomir Radev, Beth Sundheim, Lisa Ferro, Roser Saurí, Andy See, and James Pustejovsky. 2002. *Using TimeML in Question Answering.* Technical Report. Brandies University.

[23] Frank Schilder and Christopher Habel. 2001. From temporal expressions to temporal information: Semantic tagging of news messages. In *Proceedings of the ACL 2001 workshop on temporal and spatial information processing.*

[24] Andrea Setzer and Robert Gaizauskas. 2000. Annotating Events and Temporal Information in Newswire Texts. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00).* European Language Resources Association (ELRA), Athens, Greece. http://www.lrec-conf.org/proceedings/lrec2000/pdf/321.pdf

[25] Michal Starý, Zuzana Neverilová, and Jakub Valcík. 2020. Multilingual Recognition of Temporal Expressions. In *The 14th Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2020, Brno (on-line), Czech Republic, December 8-10, 2020.* Tribun EU, 67–78. http://nlp.fi.muni.cz/raslan/2020/paper2.pdf

[26] 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.* Association for Computational Linguistics, Uppsala, Sweden, 321–324. https://www.aclweb.org/anthology/S10-1071

[27] Jannik Strötgen, Anne-Lyse Minard, Lukas Lange, Manuela Speranza, and Bernardo Magnini. 2018. KRAUTS: A German Temporally Annotated News Corpus. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).* European Language Resources Association (ELRA), Miyazaki, Japan. https://www.aclweb.org/anthology/L18-1085

[28] Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013).* Association for Computational Linguistics, Atlanta, Georgia, USA, 1–9. https://www.aclweb.org/anthology/S13-2001

[29] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.* Association for Computational Linguistics, Online, 38–45. https://www.aclweb.org/anthology/2020.emnlp-demos.6

[30] Xiaoshi Zhong and Erik Cambria. 2018. Time Expression Recognition Using a Constituent-based Tagging Scheme. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, Pierre-Antoine Champin, Fabien Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM, 983–992. https://doi.org/10.1145/3178876.3185997

[31] Xiaoshi Zhong, Aixin Sun, and Erik Cambria. 2017. Time Expression Analysis and Recognition Using Syntactic Token Types and General Heuristic Rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* Association for Computational Linguistics, Vancouver, Canada, 420–429. https://doi.org/10.18653/v1/P17-1039