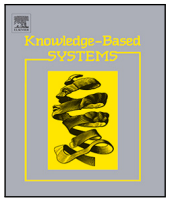
Knowledge-Based Systems 297 (2024) 111921

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



XTime: A general rule-based method for time expression recognition and normalization☆

Xiaoshi Zhong [a,b,*], Chenyu Jin [a], Mengyu An [a], Erik Cambria [c]

[a] School of Computer Science and Technology, Beijing Institute of Technology, China
[b] China North Industries Computer Application Technology Institute, NORINCO Group Co. Ltd., China
[c] School of Computer Science and Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Keywords:
Time expression (timex)
Time expression recognition and normalization (TERN)
Token types
Token triples
Mapping relations
Priority relationship

ABSTRACT

Time expression (a.k.a., timex) recognition and normalization (TERN) is a crucial task for downstream research. However, previous studies have overlooked the critical characteristics of timexes that significantly impact the task. To gain deeper insights, we conduct an analysis across four diverse English datasets to examine the key attributes of timex constituents. Our analysis reveals several noteworthy observations, such as: timexes tend to very short; the majority of timexes contain time tokens; there exist strong mapping relationships between time tokens and timex types; there exists a priority relationship among timex types; and timex values exhibit only some standard formats. Based on these insights, we propose a novel general rule-based method termed XTime[1] to recognize timexes from free text and normalize them into standard formats. Notably, XTime's rules are designed in a general and heuristic manner, enabling its independence of diverse domains and text types. Experimental evaluations conducted on both in-domain and out-of-domain English datasets demonstrate that XTime consistently outperforms or performs comparably to representative state-of-the-art methods.

1. Introduction

Time expression (i.e., timex) recognition and normalization (TERN) is a fundamental task with broad implications for numerous downstream research areas and applications, including temporal event extraction and ordering [1–8], timeline construction [9–12], temporal information retrieval [13–16], temporal reasoning [17,18], and temporal question answering [19,20]. TERN comprises two primary sub-tasks: *timex recognition* and *timex normalization*. The task of timex recognition aims to extract timexes from unstructured text. For example, given the plain text "Zhizhen was born on September 18, 2021", the goal of timex recognition is to extract the timex "September 18, 2021". On the other hand, timex normalization aims to standardize timexes into specific formats of *type* and *value*, involving two sub-tasks: *type classification* and *value normalization*. For example, in the case of the timex "September 18, 2021", the goal of timex normalization is to classify the timex as a DATE and normalize its value to "2021-09-18". Historically, TERN has been resolved primarily through deterministic rules [21–26] and machine-learning methods [27–31]. However, deterministic rules often rely heavily on domain-specific knowledge and necessitate more rules when applied to new domains; while machine-learning methods may lack interpretability. To our knowledge, there has been no systematic analysis of the general statistical characteristics of timex constituents and the factors that significantly influence timex types and values.

In this paper, we undertake a systematic analysis of timexes from training sets of four diverse datasets: TimeBank [32], TE3Silver [3], WikiWars [33], and Tweets [34]. Our analysis delves into the characteristics of constituents, types, and values of timexes, yielding seven key observations. Firstly, most timexes are very short, with over 80% containing no more than three tokens (see Observation 1 for details). Secondly, more than 91.8% of timexes contain at least one time token (see Observation 2). Thirdly, the lexicon employed to convey time information is small (see Observation 3). Fourthly, words within timexes exhibit similar syntactic behavior (see Observation 4). Fifthly, strong mapping relations exist between time tokens and timex types, with specific type of time tokens predominantly associated with particular timex type (see Observation 5). Sixthly, a discernible priority relationship exists among the four timex types: DATE < TIME < DURATION < SET. Here, A < B signifies a lower priority of timex type A compared

☆ This paper is an extension of the conference paper: Xiaoshi Zhong, Aixin Sun, and Erik Cambria. 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*, pp. 420–429, 2017.

* Corresponding author at: School of Computer Science and Technology, Beijing Institute of Technology, China.
E-mail addresses: xszhong@bit.edu.cn (X. Zhong), cyjin@bit.edu.cn (C. Jin), anmy@bit.edu.cn (M. An), cambria@ntu.edu.sg (E. Cambria).

[1] Source codes and datasets are available at https://github.com/xszhong/XTime.

https://doi.org/10.1016/j.knosys.2024.111921
Received 13 December 2023; Received in revised form 9 April 2024; Accepted 6 May 2024
Available online 10 May 2024

to B (see Observation 6). Finally, standard timex values adhere to only couple of formats, primarily comprising different types of time tokens and numerals (see Observation 7).

Based on these insights, we propose a general rule-based method termed XTime to recognize timexes from English unstructured text and normalize them into standard type and value formats. XTime defines a type system featuring three kinds of token types: *time token*, *modifier*, and *numeral*, which serve to group regular expressions (regexes) associated with time-related tokens and their respective values. These token regexes, types, and values collectively constitute a set of token triples. XTime leverages three kinds of meta time information (i.e., token triples, mapping relations between time tokens and timex types, and priority relationship among timex types, as detailed in Section 4.1) to tackle the TERN task through four main steps. Firstly, XTime identifies time tokens from input text, then identifies time segments by searching the left- and right-sides of these time tokens and stores this meta information in a data structure called MetaInfo. Subsequently, XTime merges adjacent time segments along with their corresponding MetaInfo instances. Finally, XTime extracts timexes from these time segments and assigns them standardized types and values based on the information stored in MetaInfo instances (see Section 4.2). Due to its heuristic rules built atop token types and its independence from specific tokens, XTime is domain-agnostic and suitable for a wide range of domains and text types. It is worth noting that XTime is specifically tailored for English and all our experiments are conducted on English datasets.

We evaluate the efficacy of XTime across the three sub-tasks of TERN (i.e., *timex recognition*, *type classification*, and *value normalization*) on four diverse English datasets (i.e., TE-3 [3], WikiWars [33], Tweets [34], and MEANTIME [35]) against eight representative state-of-the-art methods, comprising two rule-based methods (i.e., HeidelTime [23] and SUTime [25]), five learning-based methods (i.e., ClearTK [36], UWTime [29], TOMN [37], PTime [38], ARTime [30]), and one hybrid method (i.e., XTN [39]). Given that the training sets of TE-3, WikiWars, and Tweets are used in data analysis and the design of XTime, these datasets are considered as **in-domain** datasets. By contrast, MEANTIME is regarded as an **out-of-domain** dataset. Experimental results demonstrate that XTime achieves the best results in type classification across all datasets, both in-domain and out-of-domain. Additionally, XTime achieves either the best or the second-best results in value normalization on the three in-domain datasets, surpassing those state-of-the-art baselines. This underscores the significance of our observations regarding the characteristics of timex constituents, types, and values for the TERN task. Notably, XTime demonstrate significant performance improvements over the two rule-based baselines across all the sub-tasks on all the datasets, except for value normalization on MEANTIME. In timex recognition, XTime performs comparably with those best learning-based baselines (see Section 5 for details). Our experiments also highlight the positive correlation between improved timex recognition and enhanced time normalization performance. Moreover, owing to its light-weight heuristic rules, XTime runs in real-time and can serve as a versatile yet high-quality tool for various downstream time-related linguistic tasks, such as timeline construction and temporal reasoning.

We summarize the contributions made in previous conference paper [34] and in this extension paper as follows.

- Zhong et al. [34] analyze timexes from four diverse datasets and summarize four characteristics about timex constituents (Observations 1∼ 4). This extension paper analyzes timexes from the same four datasets and summarizes three more characteristics about timex types and timex values (Observations 5∼ 7).
- Zhong et al. [34] propose a type-based method, SynTime, to recognize timexes from text leveraging token types and heuristic rules. This extension paper proposes a type-based method termed XTime (an extension of SynTime) to recognize timexes from text and normalize them into standard type and value formats using three kinds of meta time information. Similar to SynTime, XTime operates independently of specific domains and text types. Moreover, XTime runs in real-time and can serve as a user-friendly tool for various time-related tasks.
- Zhong et al. [34] conduct experiments on three in-domain datasets to showcase the efficacy of SynTime in timex recognition. This extension paper expands the experimental scope to include four diverse in-domain and out-of-domain datasets. The results demonstrate that XTime outperforms representative state-of-the-art baselines in timex normalization, particularly excelling in type classification.

The structure of this paper is organized as follows. In Section 2 we provide a comprehensive review of existing literature pertaining to methods developed for the TERN task. Subsequently, in Section 3, we present our data analysis and summarize seven key statistical observations regarding timex constituents, types, and values. Section 4 delves into the intricate details of our proposed method, XTime. Following this, Section 5 reports the results of our experimental evaluations and analyses. Finally, in Section 6, we offer our concluding remarks.

2. Related works

Research on the TERN task has been extensively documented through the TempEval competitions series [2,3,40–43], although numerous studies exist outside of these competitions. The methods developed for TERN can generally be categorized into three types, as outlined in the survey conducted by Zhong and Cambria [44]: rule-based methods, machine learning-based methods, and deep learning-based methods.

2.1. Rule-based methods for TERN

Rule-based time taggers like GUTime, HeidelTime, and SUTime primarily employ deterministic rules to recognize and normalize timexes [21–26,45]. Many of these system are designed to handle the end-to-end TERN task (e.g, HeidelTime and SUTime). Some time taggers adopt hybrid methods, combining learning methods for timex recognition with rule-based methods for timex normalization. For example, ManTime [46], CogCompTime [39,47] integrate learning methods for timex recognization while develop rules for timex normalization. SynTime defines three kinds of token types and a set of general heuristic rules for timex recognition [34]. Additionally, Bethard [48] proposes a synchronous context free grammar (SCFG) for time normalization, later extended by Escribano et al. [39] for both English and Spanish. Rule-based time taggers have demonstrated strong performance in TempEval competitions [2,3], with SynTime showing promising results on datasets such as TimeBank, WikiWars, and Tweets [34,49,50].

2.2. Machine learning-based methods for TERN

Learning-based methods primarily extract features from text and apply statistical models on these features for TERN. For instance, [27] define a compositional grammar and employ an EM-style method to learn a latent parser for TERN. UWTime [29] leverages a combinatory categorial grammar (CCG) and employ L1-regularization to learn linguistic information from context for TERN. Ning et al. [47] formulate TER as a text-chunking problem and employ a basic machine-learning method to recognize chunks corresponding to time expressions. Ding et al. [30] propose to automatically generate rules from training data, while [51] further parse timexes obtained from these rules using a distantly supervised neural semantic parser for timex normalization. In fact, most learning-based methods incorporate rule-based elements to determine the final timex values, such as TIPSem [52] and ClearTK-TimeML [36]. Additionally, Zhong and Cambria [37] propose a constituent-based tagging scheme under conditional random fields (CRFs) to model timexes for timex recognition and such constituent-based tagging scheme for named entity recognition as well [53,54].

Table 1
Statistics of the four datasets used for data analysis.

Dataset	#Docs	#Words	#Timexes	#DATE	#TIME	#SET	#DURATION
TimeBank	183	61,418	1243	1016	22	16	189
TE3Silver	2452	666,309	12,739	11,133	192	68	1346
WikiWars	22	119,468	2671	2247	118	13	258
Tweets	942	18,199	1129	761	181	36	151

2.3. Deep learning-based methods for TERN

Neural networks and deep-learning methods have been increasingly utilized for modeling and recognizing timexes [31,39,55–65]. [57] employ distributed representations and artificial neural networks to model time expressions, exploring various configurations of layers, sizes, and normalization techniques to recognize Spanish time expressions from unstructured text. In a similar vein, [59] utilize a graph convolutional network (GCN) to jointly exploit syntactic and temporal graph structures within documents, facilitating the inference of document creation dates. Chen et al. [60] investigate the modeling of time expressions using pre-trained word representations, delving into the necessity of contextualization and the resource requirements for recognizing time expressions within free text. Furthermore, [63] leverage a sequence-to-sequence encoder with contextual entity embeddings and negation constraints to resolve date-time entities in scheduling tasks. In another study, Almasian et al. [64] introduce two popular pre-trained transformer-based models to model time expressions and general temporal information. Laparra et al. [58] normalize clinical timexes by a neural-network method while [65] normalize multilingual timexes with masked language models. Lastly, [31] leverage two cutting-edge multilingual models to model time expressions across multiple languages by transferring knowledge from multiple source languages to the low-resource target language.

XTime is naturally a rule-based time tagger specifically tailored for English. Distinguished from conventional rule-based methods, XTime adopts a heuristic and flexible rule design methodology, rendering it versatile and applicable across diverse domains and text types. In contrast to both categories of learning-based methods, XTime is rooted in a systematic analysis, providing comprehensive insights and detailed explanation for experimental results. Additionally, XTime boasts a light-weight architecture, enabling its real-time processing capabilities.

3. Data analysis

3.1. Datasets

We conduct an analysis on three benchmark datasets (i.e., TimeBank [32], WikiWars [33], and Tweets [34]) and one automatically labeled dataset (i.e., TE3Silver [3]) to investigate the characteristics of timexes regarding their constituents, types, and values. **TimeBank** comprises 183 news articles and served as a benchmark dataset in the series of TempEval competitions [1–3]. **TE3Silver**, a significantly larger dataset, consists of 2452 news articles and was also utilized in TempEval-3 [3]. **WikiWars** is a domain-specific dataset sourced from Wikipedia articles detailing 22 famous wars. Since the original WikiWars dataset lacks annotations for its timex types, we manually annotate these types to facilitate analysis, as elaborated in Section 5.1. **Tweets** represents a collection of informal text data, comprising 942 tweets collected from Twitter. Table 1 presents a summary of the statistics pertaining to these four diverse datasets.

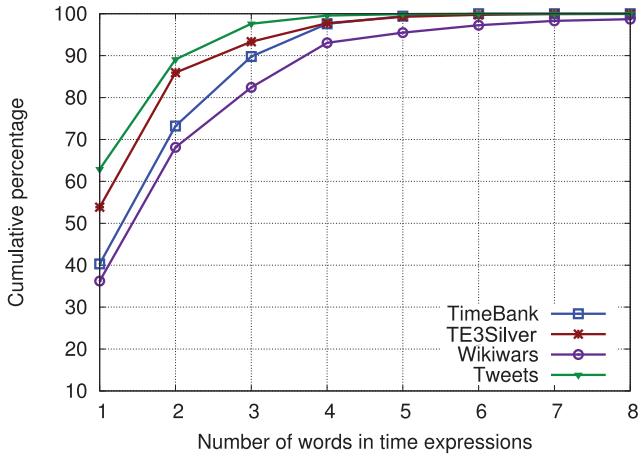


Fig. 1. Length distribution of timexes in the four datasets.

Table 2
Different statistics of timex constituents. The second column ("Avg Length") indicates the average length of timexes. The third column ("Percent") indicates the percentage of timexes that contain at least one time token. The fourth column ("#Distinct Words") indicates the number of distinct words in timexes. The final column ("#Distinct Time Tokens") indicates the number of distinct time tokens in timexes.

Dataset	Avg length	Percent	#Distinct words	#Distinct time tokens
TimeBank	2.00	94.61	130	64
TE3Silver	1.70	96.44	214	80
WikiWars	2.38	91.81	224	74
Tweets	1.51	96.01	107	64

3.2. Observations

We conduct an analysis on timexes from TimeBank, TE3Silver, and the training sets of WikiWars and Tweets.[2] Our analysis yields seven main observations. Despite the diverse nature of the four datasets, encompassing variations in corpus sizes and domains, we observe strikingly similar characteristics in the constituents, types, and values of their respective timexes.

Observation 1. *Timexes are very short. More than 80% of timexes contain no more than three words and more than 90% of timexes contain no more than four words.*

Fig. 1 depicts the length distribution of timexes across the four datasets.[3] Despite originating from diverse sources, including news articles, Wikipedia articles, and tweets, with variations in text lengths, a notable consistency emerges in the distribution of timex lengths. Specifically, the proportion of one-word timexes spans from 36.23% in

[2] Note that our analysis focuses solely on the training sets rather than the whole datasets for the purpose of investigating the characteristics of timex constituents, types, and values. TE3Silver is only utilized for analyzing timex constituents (see Observations 1∼ 4), while it is utilized for analyzing timex types and values (see Observations 5∼ 7) due to its timex labels not being considered as ground-truth.

[3] The length-frequency distribution of timexes is further analyzed together with other entities in Zhong et al. [66].

Table 3
Top 10 POS tags assigned to the words in timexes. Frequency (*Fr.*) indicates the number of times that a POS tag is assigned to words in timexes; percentage (*Pr.*) is based on its assignments to words in timexes and all words in documents.

TimeBank			TE3Silver			WikiWars			Tweets		
Tag	*Fr.*	*Pr.*	Tag	*Fr.*	*Pr.*	Tag	*Fr.*	*Pr.*	Tag	*Fr.*	*Pr.*
NN	587	6.66	NNP	6902	8.77	CD	2113	67.85	NN	572	15.27
DT	396	7.16	CD	4582	22.11	NNP	1294	8.87	CD	323	55.40
CD	351	11.60	NN	3233	3.26	NN	783	5.70	RB	189	25.40
JJ	347	8.74	DT	2080	3.15	DT	582	4.67	DT	161	18.05
NNP	336	5.09	JJ	1940	3.82	IN	363	2.48	NNP	155	6.55
NNS	156	4.17	NNS	1356	3.19	JJ	328	3.82	JJ	118	12.38
RB	162	9.44	RB	597	3.52	RB	261	8.23	NNS	116	18.10
IN	76	1.13	IN	512	0.62	NNS	234	3.82	IN	20	1.24
,	20	0.61	,	175	0.56	,	171	2.80	JJR	10	17.86
CC	9	0.60	CC	69	0.38	VBG	28	1.50	VBP	9	2.72

WikiWars to 62.91% in Tweets. This suggests a tendency, particularly prevalent in informal communication, to employ succinct expressions for conveying time information. The second column of Table 2 provides insights into the average length of timexes, revealing an average length of approximately two words per timex.

Observation 2. *More than 91% of timexes contain at least one time token.*

The third column of Table 2 reports the percentage of timexes containing at least one time token. Notably, we observe that a significant majority, at least 91.81% of timexes, contain at least one time token. It is worth noting that some timexes may not explicitly contain time tokens but rely on other timexes for temporal context; for example, in the string "2 to 8 days", the timex "2" depends on the timex "8 days". This observation underscores the importance of time tokens in composing timexes. Consequently, effective time recognition necessitates the accurate identification of associated time tokens.

Observation 3. *Only a small set of time keywords are used to express time information.*

Upon examining timexes across all four datasets, we observe that the set of keywords employed to convey time-related information is small. The fourth and final columns of Table 2 present the counts of distinct words and distinct time tokens found within timexes. Notably, these words and tokens are manually normalized before counting, with their variants disregarded. For example, both "year" and "5yrs" are counted as a singular time token "year". Numerals are excluded from the counting process. The last column of Table 2 shows that despite variations in dataset sizes, domains, and text types, the numbers of their distinct time tokens remain comparable.

Across the four datasets, we identify a total of 350 distinct words and 132 distinct time tokens. Of the 123 distinct time tokens, 45 are shared across all the four datasets, while 101 appear in at least two datasets. This observation underscores a high degree of overlap among time tokens across datasets, indicating a substantial intersection between timexes and their associated time tokens.

Observation 4. *Part-of-speech (POS) information cannot distinguish timexes from common words, but within timexes, POS tags can help distinguish their constituents.*

Table 3 showcases the top 10 POS tags that appear within timexes, along with their respective proportions over the whole text in each dataset.[4] Among the 40 POS tags observed (10 × 4), a striking 37 exhibit percentages below 20%, which the remaining 3 are CD. This suggests that POS tags alone cannot provide sufficient information to distinguish timexes from common words. Nevertheless, the most prevalent POS tags within timexes are NN*, JJ, RB, CD, and DT. Within timexes, time tokens usually have NN* and RB, modifiers have JJ and RB, and numerals have CD. This observation indicates a consistent behavior among similar constituents with timexes, akin to how linguists define POS for language [68]. Drawing inspiration from the definition of POS for language, we are prompted to define a type system for timexes, recognizing them as integral components of language.

Observation 5. *There exist strong mapping relations between time tokens and timex types: a specific type of time tokens primarily associated with a particular timex type.*

Table 4 presents the distribution of time tokens appearing in each type of timexes.[5] The format utilized in Table 4 is denoted as "Pr/Pr^1", where Pr denotes the percentage of time tokens appearing in the respective type of **the whole timexes**, as defined by Eq. (1), while Pr^1 denotes the percentage of time tokens appearing in the respective type of **the one-word timexes** (note: a one-word timex contains solely a time token, without any modifier nor numeral), as defined by Eq. (2).

$$Pr(W,T) = \frac{Count(W,T)}{Count(W)} \tag{1}$$

where T represents a specific timex type and $T \in \{\underline{\text{DATE}}, \underline{\text{TIME}}, \underline{\text{DURATION}}, \underline{\text{SET}}\}$, W denotes a specific type of time tokens (with 17 types in total; see Table 4 and Table 8), $Count(W)$ denotes the total number of time tokens of type W, while $Count(W,T)$ denotes the number of time tokens of type W that appear in all timexes of type T. For each W, we have $\sum_T Pr(W,T) = 100\%$.

$Pr(W,T)$ calculates the percentage of time tokens of type W appearing in the timexes of type T across the whole timexes. By contrast, $Pr^1(W,T)$ calculates this percentage solely based on one-word timexes:

$$Pr^1(W,T) = \frac{Count^1(W,T)}{Count^1(W)} \tag{2}$$

where $Count^1(W)$ denotes the total number of time tokens of type W exclusively within one-word timexes, while $Count^1(W,T)$ indicates the number of time tokens of type W appearing in one-word timexes of type T. $Pr^1(W,T)$ indicates, within one-word timexes, the percentage of time tokens of type W appearing in timexes of type T. Similar to $Pr(W,T)$, for each W, we have $\sum_T Pr^1(W,T) = 100\%$.

We utilize the same fifteen time tokens defined by Zhong et al. [34], with the exception of "TIMEUNIT", which is further divided into "TIMEUNIT $_S$", "TIMEUNIT $_C$", and "TIMEUNIT $_D$". TIMEUNIT $_S$ indicates TIMEUNIT in singular form (e.g., "month" and "day"), TIMEUNIT $_C$ indicates plural and continuous TIMEUNIT (e.g., "months" and "days"), while TIMEUNIT $_D$ indicates plural and discrete TIMEUNIT

[4] We employ Stanford POS Tagger [67] to obtain these POS tags. Detailed descriptions of these POS tags can be found at https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html.

[5] To distinguish between token types, timex types, and MetaInfo attributes, we underline timex types and start MetaInfo attributes with "·". For example, DATE represents a token type, while DATE denotes a timex type, and ·DATE signifies a MetaInfo attribute.

Table 4
Percentage of a specific type of time tokens appearing in a particular type of timexes. The format is "Pr/Pr^{1}", where Pr indicates the percentage calculated using **the whole timexes** while Pr^{1} indicates the percentage calculated solely based on **one-word timexes**. "DURA" indicates DURATION. Values exceeding 50% are highlighted in bold. "–" indicates the absence of such type of time tokens in the respective timex type.

Time token	TimeBank (%)				WikiWars (%)				Tweets (%)			
	DATE	TIME	SET	DURA	DATE	TIME	SET	DURA	DATE	TIME	SET	DURA
YEAR	**100/100**	–/–	–/–	–/–	**99/100**	0.8/–	–/–	0.5/–	**93/95**	5.9/5.3	–/–	0.9/–
MONTH	**98/100**	1.0/–	1.0/–	–/–	**98/100**	1.3/–	–/–	0.4/–	**97/100**	2.8/–	–/–	–/–
WEEK	**93/100**	5.4/–	2.0/–	–/–	**80/100**	20/–	–/–	–/–	**81/100**	12/–	7.7/–	–/–
DATE	–/–	–/–	–/–	–/–	–/–	–/–	–/–	–/–	**100/100**	–/–	–/–	–/–
TIME	–	**100**/–	–/–	–/–	–/–	**100/100**	–/–	–/–	–/–	**100/100**	–/–	–/–
DAYTIME	–/–	**94/100**	5.9/–	–/–	–/–	**98/100**	–/–	2.4/–	–/–	**96/100**	4.2/–	–/–
TIMELINE	**99/99**	1.2/0.7	–/–	–/–	**100/100**	–/–	–/–	–/–	**99/100**	0.3/–	–/–	0.3/–
TIMEUNIT $_S$	**89/50**	–/–	1.6/–	9.6/**50**	**69/100**	2.5/–	1.0/–	27/–	**67/94**	2.4/2.0	4.2/–	26/4.1
TIMEUNIT $_C$	16/–	–/–	–/–	**84/100**	27/–	2.5/29	–/–	**70/71**	5.1/–	–/–	1.3/–	**94/100**
TIMEUNIT $_D$	–/–	–/–	**89**/–	11/–	–/–	–/–	–/–	–/–	–/–	–/–	**100/100**	–/–
SEASON	**92/100**	–/–	8.3/–	–/–	**100/100**	–/–	–/–	–/–	**100/100**	–/–	–/–	–/–
DECADE	**80**/–	–/–	–/–	20/–	**100100**	–/–	–/–	–/–	–/–	–/–	–/–	–/–
PERIODICAL	–/–	–/–	**75/75**	25/25	–/–	–/–	**100**/–	–/–	–/–	–/–	**100/100**	–/–
DURATION	**64/65**	–/–	–/–	37/35	6.7/8.3	–/–	–/–	**93/92**	–/–	–/–	7.1/–	**93/100**
HOLIDAY	–/–	–/–	–/–	–/–	**90/100**	10/–	–/–	–/–	**88/100**	5.9/–	–/–	5.9/–
TIMEZONE	–/–	**100**/–	–/–	–/–	–/–	**100**/–	–/–	–/–	–/–	**100**/–	–/–	–/–
ERA	–/–	–/–	–/–	–/–	**100/100**	–/–	–/–	–/–	–/–	–/–	–/–	–/–

Table 5
Mapping relations between time tokens and timex types.

Time token	Timex type
YEAR, MONTH, WEEK, DATE, TIMELINE, ERA, TIMEUNIT $_S$, SEASON, DECADE, HOLIDAY	DATE
TIME, DAYTIME, TIMEZONE	TIME
PERIODICAL, TIMEUNIT $_D$	SET
TIMEUNIT $_C$, DURATION	DURATION

(e.g., "Septembers" and "Springs"). Such division is due to TIMEUNIT $_S$ being generally classified as the timex type DATE (e.g., "this month" and "this day"), TIMEUNIT $_C$ being classified as DURATION (e.g., "8 months" and "3 days"), and TIMEUNIT $_D$ being classified as SET (e.g, "Septembers" and "Springs").

Table 4 presents that a particular type of time tokens predominantly appear in a specific type of timexes across both the whole timexes and only the one-word timexes. Precisely, YEAR, MONTH, WEEK, DATE, TIMELINE, TIMEUNIT $_S$, SEASON, DECADE, HOLIDAY, and ERA are primarily associated with DATE. TIME, DAYTIME, and TIMEZONE are primarily with TIME. TIMEUNIT $_C$ and DURATION are primarily with DURATION. PERIODICAL and TIMEUNIT $_D$ are primarily with SET. These high percentages indicate robust mapping relations between time tokens and timex types. For example, 100% of TIME appearing in TIME suggests that the presence of a TIME token in a timex will invariably result in its classification as TIME. Particularly, when focusing solely one-word timexes, the majority of these percentages escalate to exceptionally high levels, with many reaching 100%. This underscores the robustness of these mapping relations, as depicted in Table 5.

It is worth noting that not all the instances of a specific type of time tokens are exclusively associated with a particular type of timexes. This is because a timex may encompass multiple time tokens or modifiers/numerals that collectively influence the timex's final type.

Observation 6. *There exists a general priority relationship among the four timex types: DATE < TIME < DURATION < SET. Modifiers and numerals may elevate a timex from a lower-priority type to a higher one.*

Table 6 displays the occurrences of time tokens from a particular timex type in the remaining three timex types. Here, "{·}" denotes the set of time tokens listed in Table 5 that map to the respective timex type. For example, {TIME} denotes the three time tokens TIME, DAYTIME, and TIMEZONE that map to the timex type TIME.

Table 6 reveals several patterns: (1) time tokens {DATE}[6] (i.e., YEAR and other nine time tokens) extensively appear in the other three timex types, (2) time tokens {TIME} (i.e., TIME, DAYTIME, and TIMEZONE) widely appear in DURATION and SET but not in DATE, (3) time tokens {DURATION} (i.e., TIMEUNIT $_C$ and DURATION) widely appear in SET without appearing in DATE or TIME, and (4) time tokens {SET} (i.e., PERIODICAL and TIMEUNIT $_D$) generally do not appear in the other three timex types.[7] This suggests a discernible priority relationship among the four timex types: DATE < TIME < DURATION < SET. Here, DATE < TIME indicates a lower priority of DATE compared to TIME.

This priority relationship implies that if a timex includes a high-priority time token, then it is unlikely to be classified into a low-priority type. For example, a timex containing a PERIODICAL will not be classified as DATE, TIME, DURATION; instead, the timex will be classified

Table 6
Number of time tokens that map to a timex type appearing in other three types of timexes. "{·}" denotes the set of time tokens in Table 5 that map to the timex type, e.g., {TIME } denotes the time tokens TIME, DAYTIME, and TIMEZONE that map to TIME.

Dataset	Time tokens	Timex type			
		DATE	TIME	DURATION	SET
TimeBank	{DATE}	–	24	2	11
	{TIME}	0	–	0	2
	{DURATION}	0	0	–	0
	{SET}	0	0	1	–
WikiWars	{DATE}	–	93	53	0
	{TIME}	0	–	1	0
	{DURATION}	0	0	–	0
	{SET}	0	0	0	–
Tweets	{DATE}	–	39	9	8
	{TIME}	0	–	1	5
	{DURATION}	0	0	–	2
	{SET}	0	0	0	–

[6] Note that {DATE} denotes the set of time tokens that map to the timex type DATE, instead of the timex type DATE.

[7] The exception of {SET} appearing in DURATION is attributed to an annotation error.

Table 7
Standard formats of timex values under each timex type and the attribution compositions for each format. "DURA" denotes "DURATION". "CC" indicates a specific century; "DDD" indicates a specific decade; "YYYY" a specific year; "MM" a specific month; "ww" the week of year; "w" the day of week; "DD" the day of month; "hh" a specific hour; "mm" a specific minute; "ss" a specific second; "dd" daytime (MO/AF/NI); "SS" a specific season (SP/SU/FA/WI); "N" a specific number; "U" indicates the time unit greater than a day; "u" indicates the time unit less than a day.

Type	Value format	Example timex	Example value	Attribute composition
DATE	PRESENT_REF, PAST_REF, FUTURE_REF	now	PRESENT_REF	·TIMELINE
	CC	20th century	19	·CENTURY
	DDD	the late 1960s	196	·DECADE
	YYYY	2013	2013	·YEAR
	YYYY-MM	April 2018	2018–04	·MONTH, ·YEAR
	YYYY-MM-XX	January day	2013-01-XX	·MONTH, ·YEAR, ·TIMEUNIT_S=D,
	YYYY-MM-DD	March 21, 2013	2013–03–21	·DATE, ·MONTH, ·YEAR, ·NUMBER
	EEYYYY	493 BC	BC0493	·ERA, ·YEAR, ·NUMBER
	EEYYYY-MM	April 480 BC	BC0480–04	·MONTH, ·YEAR, ·ERA
	YYYY-Www	the last week	2013-W11	·TIMEUNIT_S=W, ·WEEKOFYEAR
	YYYY-Www-WE	this weekend	2013-W40-WE	·TIMEUNIT_S=WE
	YYYY-SS	last summer	2012-SU	·SEASON
TIME	YYYY-MM-DDThh:mm:ss, YYYY-MM-DDThh:mm, YYYY-MM-DDThh	15:00 Saturday	2013-03-23T15:00	·DATE, ·WEEK, ·TIME
	YYYY-MM-DDTdd	Friday afternoon	2013-03-22TAF	·WEEK, ·DATE, ·DAYTIME
DURA	PNU, PTNu	24 h	PT24H	·TIMEUNIT_C, ·NUMBER
	PXU, PTXu	few minutes	PTXM	·TIMEUNIT_C
	PNU, PTNu	a month	P1M	·TIMEUNIT_S, ·NUMBER
SET	XXXX, XXXX-XX, XXXX-XX-XX, XXXX-WXX	annually	XXXX	·PERIODICAL, ·TIMEUNIT_D
	XXXX-XX-XXTdd	every morning	XXXX-XX-XXTMO	·DAYTIME
	XXXX-WXX-w	every Friday	XXXX-WXX-5	·WEEK
	XXXX-SS	every winter	XXXX-WI	·SEASON

as SET. Similarly, a timex containing a DAYTIME will not be classified as DATE, but may be classified as TIME, DURATION, or SET.

In general, a modifier or numeral can elevate a timex to a higher-priority type. For example, consider the timex "every afternoon": while the time token "afternoon" (DAYTIME) typically maps to TIME, the modifier "every" (FREQUENT) transforms the timex into SET.

Observation 7. *Standard timex values exhibit only a few formats, which are composed of different types of time tokens and numerals.*

We observe that each type of timexes has only a limited number of standard value formats, as summarized in the second column of Table 7. It shows that despite the potentially vast number of timexes in a dataset, their value formats remain relatively fixed. Moreover, we observe that only 0.56% of timexes in TimeBank, 0.08% in WikiWars, and 0.27% in Tweets have multiple time tokens of the same token type. This suggests that timex values predominantly consist of different types of time tokens and numerals. Furthermore, most modifiers have minimal impact on timex values.

4. Xtime: Timex recognition and normalization with meta time information and general heuristic rules

XTime defines a comprehensive type system, comprising a set of token types, to group timex-related tokens and store their assigned values. Built upon these token types, XTime designs general heuristic rules to recognize timexes from free text and normalize them into standard type and value formats. Fig. 2 shows the architecture of XTime, which operates across three levels: token level, type level, and rule level. At the type level, token types group timex-related tokens, while heuristic rules at the rule level operate on these token types rather than individual tokens. Such design ensures the generality of the heuristic rules, as they operate on token types such as "YEAR" and "MONTH" rather than specific tokens like "2021" and "September". Consequently, these heuristic rules are applicable across diverse domains and text types, unaffected by the specifics of individual tokens.

Fig. 3 provides the overview of XTime for the TERN task in practice. It mainly consists of two components: XTime construction and timex recognition and normalization. In XTime construction, as shown on the left-hand side, XTime is initially constructed with three kinds of meta time information: (1) token triples, (2) mapping relations between time tokens and timex types, and (3) priority relationship among timex types. After initial construction, XTime can be directly applied for the TERN task. Additionally, XTime offers flexibility through easy expansion; by incorporating timex-related token triples from training data, it adapts seamlessly to diverse domains and text types. In timex recognition and normalization, as shown on the right-hand side, XTime executes through four key steps. Initially, XTime identifies time tokens from the input POS-tagged text and assigns them token types and values to form token triples. Surrounding the identified time tokens, XTime then searches for modifiers and numerals to construct time segments. Subsequently, these time segments are merged into larger time segments. Finally, XTime transforms these time segments into timexes and assigns them pre-defined types and standard-format values.

4.1. Xtime construction

XTime is constructed by importing general heuristic rules to operationalize the functionalities of three crucial kinds of meta time information: (1) **token triples**, (2) **mapping relations** linking time tokens to timex types, and (3) **priority relationship** among timex types. It is important to note that during the construction stage, XTime does not process any text but rather focuses solely on implementing the functionalities of these three kinds of meta time information. This subsection exclusively describes meta time information, while the subsequent subsection delves into intricate details of how general heuristic rules execute the functionalities of these meta time information when processing text with XTime.

4.1.1. Token types and token triples

XTime establishes a comprehensive type system for timexes, comprising 17 token types for time tokens, 9 token types for modifiers, and 2 token types for numerals. Token types to tokens is like POS tags to words. For example, "September" is tagged with a POS tag of NNP, while being assigned a token type of MONTH.

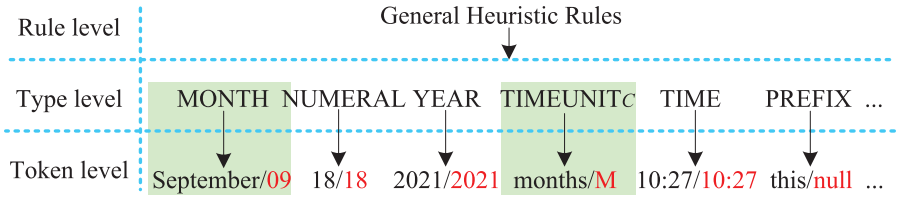


Fig. 2. Architecture of XTime. It is organized into three distinct levels: token level, type level, and rule level. At the type level, token types group timex-related tokens and assign them corresponding values. At the rule level, heuristic rules operate on token types and are independent of specific tokens, ensuring their applicability across various domains and text types. Examples of token triples include <September, MONTH, 09>, <months, TIMEUNIT$_C$, M>, where assigned token values are highlighted in red.

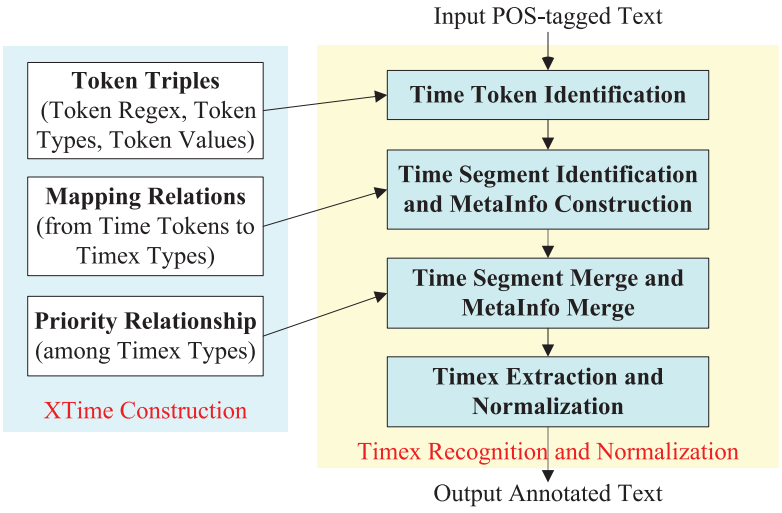


Fig. 3. Overview of XTime for TERN in practice, with two main components: XTime construction and timex recognition and normalization. The left-hand side shows the XTime construction with three kinds of meta time information: (1) token triples, (2) mapping relations between time tokens and timex types, and (3) priority relationship among timex types. The right-hand side shows the main steps of XTime recognizing timexes from free text and normalizing them into standard type and value formats.

Table 8
Token types used in XTime for time tokens, modifiers, and numerals. The final column denotes the number of distinct tokens within each type, excluding variants. "–" signifies cases where the token type involves digit variations and cannot be counted.

Type	Token type	Description	Examples	#Tokens
Time Token	YEAR	year instances	1986, 1989, 2021	–
	MONTH	month instances	February, September	12
	WEEK	day of week	Monday, Sunday	7
	DATE	date instances	2021–09–18	–
	TIME	time instances	10:27, 03:45:32	–
	DAYTIME	time within a day	morning, afternoon	27
	TIMELINE	relative to today	yesterday, tomorrow	12
	TIMEUNIT $_S$	singular time units	year, month	15
	TIMEUNIT $_C$	continuous time units	year(s)	15
	TIMEUNIT $_D$	discrete time units	year(s)	15
	SEASON	season instances	Winter, Fall, Spring	5
	DECADE	decade instances	1910s, fifties	–
	HOLIDAY	holiday instances	Christmas	20
	PERIOD	period instances	daily	9
	DURATION	duration instances	5-year	–
	TIME_ZONE	time zones	GMT, UTC	6
	ERA	era AD and BC	AD, BC	2
Modifier	COMMON	common modifiers	the, about, alternate	36
	QUANTITATIVE	quantitative modifiers	some, several, half	7
	FREQUENT	frequent modifiers	each, every	2
	OPERATE $_{Pre}$	prefix operate modifiers	next, previous, last	7
	OPERATE $_{Suf}$	suffix operate modifiers	ago, before, later	4
	DURA	duration modifier	old	1
	LINKAGE	link time tokens	and, or, to, -	4
	IN_ARTICLE	indefinite articles	a, an	2
	COMMA	comma	,	1
Numeral	NUMBER	numbers	20, 2021, 18	–
	ORDINAL	ordinals	third, fifth	–

Table 9
Some examples of token triples. "null" indicates that the token has no value.

Token triple	Token triple
<"(January| Jan\.?)('s)?" , MONTH , 01>	<"Aprils'?| Aprs\.?'?" , TIMEUNIT $_D$, 04>
<"saturday('s)?| sat\.?" , WEEK , 6>	<"fifteenth| 15th" , ORDINAL , 15>
<"years| yrs" , TIMEUNIT $_C$, Y>	<"afternoon('?s)?" , DAYTIME , AF>
<"annually" , PERIODICAL , XXXX>	<"each| every" , FREQUENT , null>

Time Token. Time tokens are those words that directly express time information, such as year, season, month, etc. Existing rule-based time taggers like HeidelTime and SUTime have manually collected a large number of time-related keywords. XTime defines 17 token types for time tokens and uses their names similar to Joda-Time classes[8]: YEAR, MONTH, WEEK, DATE, TIME, DAYTIME, TIMELINE, TIMEUNIT $_S$, TIMEUNIT $_C$, TIMEUNIT $_D$, SEASON, DECADE, PERIODICAL, DURATION, HOLIDAY, TIMEZONE, and ERA.

Modifier. Modifiers appear around time tokens and modify them. XTime defines 9 token types for modifiers according to their semantic functions and possible positions within timexes: COMMON, QUANTITATIVE, FREQUENT, OPERATE $_{Pre}$, OPERATE $_{Suf}$, DURA, LINKAGE, IN_ARTICLE, and COMMA.

Numeral. Timexes usually contain numerals, which are either time tokens, e.g., the "2021" in the timex "September 2021", or modifiers, e.g., the "10" in the timex "10 days". XTime defines 2 token types for numerals: NUMBER and ORDINAL.

Table 8 presents descriptions, examples, and numbers of token types for time tokens, modifiers, and numerals utilized in XTime.

Given that XTime is an extension of SynTime, which focuses solely on time recognition, XTime has been enhanced to tackle the end-to-end TERN task. This enhancement involves a further subdivision of token types into fine-grained ones, as described earlier and presented in Table 8. Additionally, XTime utilizes the same token regexes as SynTime and collects token values from SUTime[9] [24] for these token regexes, in alignment with the annotation standards of TimeML [69] and TimeBank [32].

These token regexes, token types, and token values collectively form a set of token triples. Each token triple is composed of a token regex, a token type, and a token value in the following format:

Token triple :=<*token regex*, *token type*, *token value*>

While all time tokens and numerals are associated with specific values, Observation 7 suggests that most modifiers do not influence timex values. Consequently, the token triples of most modifiers lack assigned values. Table 9 presents some examples of such timex-related token triples.

4.1.2. Mapping relations from time tokens to timex types

Observation 5 indicates strong mapping relations between time tokens and timex types. These mapping relations are as summarized in Table 5. They play a crucial role in XTime for determining the types of timexes.

4.1.3. Priority relationship among timex types

Observation 6 indicates a general priority relationship among the four timex types: DATE < TIME < DURATION < SET. This priority relationship also plays an important role in XTime for determining the types of timexes.

4.2. Timex recognition and normalization

In this subsection, we elaborate on how XTime utilizes the three kinds of meta time information alongside general heuristic rules to recognize timexes from unstructured text and normalize them into standard type and value formats. This process comprises four key steps: (1) time token identification, (2) time segment identification and MetaInfo construction, (3) time segment merge and MetaInfo merge, and (4) timex extraction and normalization. We provide a step-by-step illustration of this procedure using the example depicted in Fig. 4.

4.2.1. Time token identification

Identifying time tokens is a straightforward process accomplished by simply looking all words in input text at the token triples. When a word matches any of the token regexes for time tokens, XTime assigns the word a corresponding token type and token value, creating a pair of <*token type*, *token value*>. For example, in Fig. 4, XTime identifies the word "September" and assigns it the pair <MONTH, 09>, assigns "2021" the pair <YEAR, 2021>, and assigns "10:27:00" the pair <TIME, 10:27:00>.

In addition to identifying time tokens, XTime also identifies modifiers and numerals during the scanning of the input text. Any word matching the token regexes for modifiers and numerals is likewise assigned a <token type, token value > pair. In the example illustrated in Fig. 4, XTime assigns the word "18" the pair <NUMBER, 18> and assigns "," the pair <COMMA, null>.

4.2.2. Time segment identification and MetaInfo construction

In this step, XTime primarily identifies time segments from identified time tokens and stores the meta information into a MetaInfo structure. This process consists of two main sub-steps: (1) time segment identification and (2) MetaInfo construction.

Time Segment Identification. The goal of time segment identification is to search the surroundings of each previously identified time token for modifiers and numerals, then assemble the time token with any modifiers and numerals to create a time segment. This searching follows simple general heuristic rules, where the key idea is to expand the left and right boundaries of the time token.

Initially, each time token forms an individual time segment. If a time token is either a PERIOD or DURATION, then no further searching is required. Otherwise, both the left and right sides of the time token are searched for modifiers and numerals. During leftward searching, if a COMMON, FREQUENT, QUANTITATIVE, OPERATE $_{Pre}$, DURA, NUMBER, ORDINAL, or IN_ARTICLE token type is encountered, then the searching continues. Similarly, during rightward searching, if a OPERATE $_{Suf}$, NUMBER, or ORDINAL token type is encountered, then the searching continues. Both leftward and rightward searchings terminate when reaching a COMMA or LINKAGE or a non-modifier/numeral word. The leftward searching does not exceed the previous time token, and the rightward searching does not exceed the subsequent time token.[10] A time segment comprises exactly one time token, along with zero or some modifiers/numerals. For example, as depicted in Fig. 4, XTime identifies three time segments, namely S_1: "MONTH/September NUMBER/18 COMMA/,", S_2: "COMMA/, YEAR/2021", and S_3: "TIME/10:27:00".

[8] https://www.joda.org/joda-time

[9] https://github.com/stanfordnlp/CoreNLP/tree/main/src/edu/stanford/nlp/time/rules

[10] It is worth noting that such searching operates based on token types rather than specific tokens themselves.

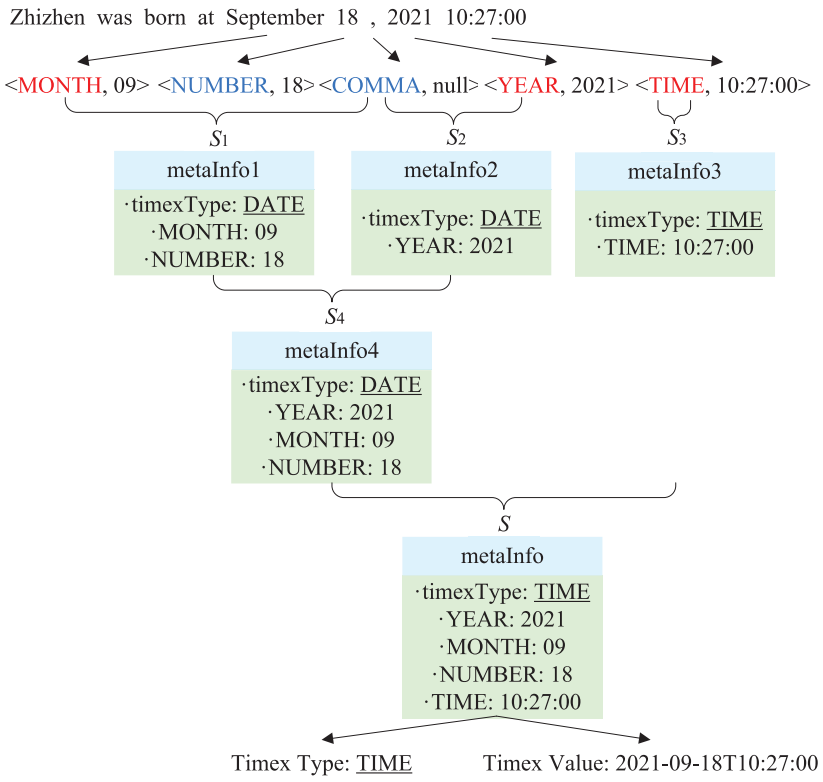


Fig. 4. An example of XTime recognizing and normalizing timexes from free text. XTime firstly assigns each time token a <*token type*, *token value*> pair, then searches the surroundings of time tokens for modifiers and numerals to form time segments and initializes a MetaInfo instance for each identified time segment. Subsequently, XTime merges some time segments and their corresponding MetaInfo instances, and finally extracts timexes from final time segments and determines their types and values.

Table 10
Attributes defined in the MetaInfo structure. MetaInfo defines 18 attributes for time tokens and numerals, 1 attribute for reference date, and 1 attribute for the timex type of the MetaInfo instance. The attribute notations start with a "·".

Attribute	Description	Example value
·refDate	reference date	2013–03–22
·timexType	timex type of the MetaInfo	DATE, TIME, SET, DURATION
·CENTURY	century instances	19, 20
·DECADE	decade instances	193
·YEAR	year instances	2012, 1956
·MONTH	month instances	01∼12
·WEEK	week instances	1∼7
·DAY	day instances	1∼31
·DATE	date instances	1905-02-20, 2021-09-18
·TIMELINE	timeline instances	PRESENT_REF, FUTURE_REF, PAST_REF
·WEEKOFYEAR	weeks of year	01∼52
·TIME	time instances	10:27, 03:07:35
·DAYTIME	daytime instances	MO, AF, NI
·ERA	era instances	BC, AD
·PERIODICAL	periodical instances	XXXX-XX-XX
·NUMBER	number instances	1, 100
·ORDINAL	ordinal instances	1, 100
·TIMEUNIT_S	TIMEUNIT $_S$ instances	Y, M, W, D
·TIMEUNIT_C	TIMEUNIT $_C$ instances	Y, M, W, D
·TIMEUNIT_D	TIMEUNIT $_D$ instances	WE, SP, FA

These is a special category of time segments that do not contain any time tokens. These time segments depend on other adjacent time segments for context. For example, in the string "8 to 20 days", the token-type sequence of "to 20 days" forms a time segment, while the token-type sequence of "8 to" forms a dependent time segment.

MetaInfo Construction. MetaInfo is a data structure designed to store pertinent information regarding identified time segments, facilitating subsequent determination of timex types and values. This structure defines 18 attributes for time tokens and numerals, 1 attribute for reference date, and 1 attribute for the timex type of a MetaInfo instance. Table 10 presents and elucidates these 20 attributes. To distinguish MetaInfo attributes from token types, we start attribute notations with a "·". For example, DATE denotes a token type while ·DATE signifies a MetaInfo attribute. It is worth noting again that DATE denotes a timex type, while {DATE } denotes a set of time tokens mapping to the timex type DATE .

For each identified time segment, XTime generates and initializes a corresponding MetaInfo instance. Taking the example of the identified

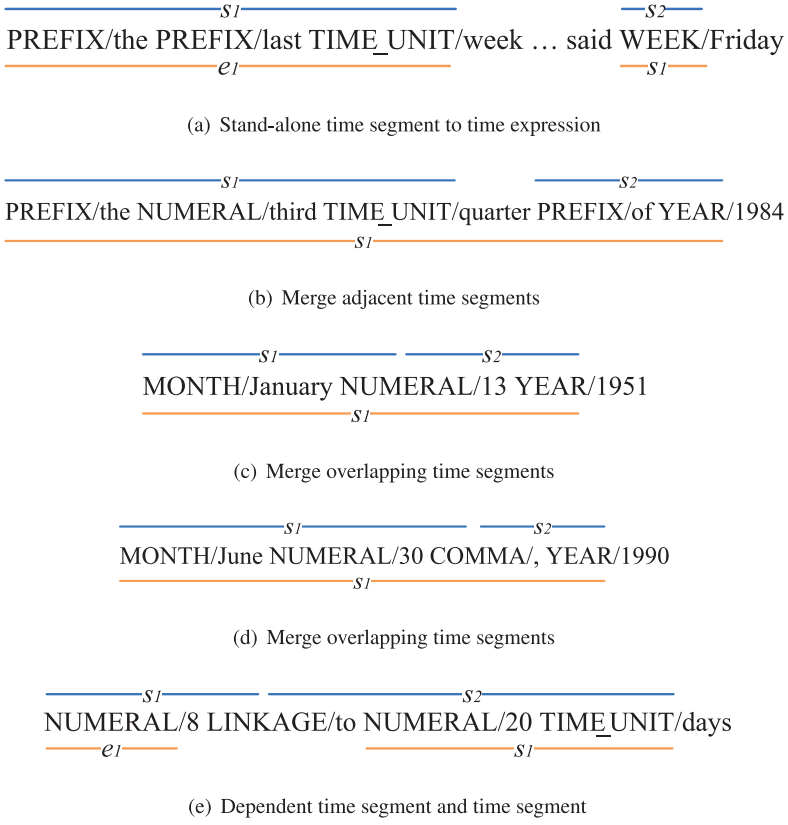


Fig. 5. Example time segments and timexes. The above labels are from time segment identification while the below labels are for timex extraction.

time segment S_1 "September 18," depicted in Fig. 4, XTime executes the following tasks in this sub-step:

- Stores the <*token type*, *token value*> pairs of the identified time segment in the respective attributes of the MetaInfo instance. For example, for the identified time segment "September 18," comprising the <MONTH, 09>, <NUMBER, 18>, and <COMMA, null> pairs, XTime sets the attribute "·MONTH" to "09" and the attribute "·NUMBER" to "18" in metaInfo1.
- Sets the attribute "·timexType" to the appropriate timex type based on the time token of the identified time segment, utilizing the mapping relations outlined in Table 4. For example, XTime sets the attribute "·timexType" of metaInfo1 to the timex type DATE since the time token "September" corresponds to MONTH, which maps to DATE as per the mapping relations in Table 5. If the time segment contains a QUANTITATIVE, then the "·timexType" is set to DURATION; if it contains a FREQUENT, then the "·timexType" is set to SET.
- Sets the attribute "·refDate".

4.2.3. Time segment merge and MetaInfo merge

As a timex may comprise multiple time tokens, it may also contain multiple time segments, of which each is represented by a distinct MetaInfo instance. To streamline this data, XTime merges adjacent time segments and some overlapping time segments, forming larger time segments and consolidating corresponding MetaInfo instances into new ones.

Time Segment Merge. XTime scans identified time segments from the beginning to the end. A standalone time segment constitutes a timex (see Fig. 5(a)). The primary focus lies in handling adjacent or overlapping time segments. If two time segments s_1 and s_2 are adjacent, then XTime merges them into a new time segment s_1 (see Fig. 5(b)). In cases where s_1 and s_2 overlap at a shared boundary, the shared boundary may be a modifier or a numeral. If the shared boundary is neither a COMMA nor a LINKAGE, then XTime merges s_1 and s_2 (see Fig. 5(c)). However, if the shared boundary is a LINKAGE, then XTime extracts s_1 as a timex and continues scanning. When the shared boundary is a COMMA, XTime merges s_1 and s_2 only if the previous and subsequent tokens of the COMMA satisfy three conditions: (1) the previous token is a time token, NUMBER, or ORDINAL; (2) the subsequent token is a time token; and (3) the token types of the previous and subsequent tokens differ (see Fig. 5(d)). It is important to note that while Fig. 5 showcases these examples with token types with tokens together, the heuristic rules operate solely on token types.

MetaInfo Merge. When merging time segments into a larger one, XTime concurrently merges their corresponding MetaInfo instances into a new MetaInfo instance. During this merging process, XTime determines the "·timexType" of the new MetaInfo instance by selecting the higher-priority timex type from the two original MetaInfo instances being merged. For example, as shown in Fig. 4, the "·timexType" of the new instance metaInfo is set by TIME since it takes precedence over the "·timexType" of metaInfo4 (DATE) and metaInfo3 (TIME). On the other hand, Observation 7 suggests that timex values typically consist of different types of time tokens and numerals, implying that different MetaInfo instances within a timex generally possess different attributes. Consequently, XTime simply amalgamates all attributes from two MetaInfo instances into a new one. For example, as shown in Fig. 4, metaInfo1 includes "·MONTH" and "·NUMBER", metaInfo2 includes "·YEAR", while metaInfo3 includes "·TIME". XTime combines the "·MONTH" and "·NUMBER" attributes from metaInfo1 with the "·YEAR" attribute from metaInfo2, then further merges these with the "·TIME" attribute from metaInfo3 into metaInfo.

Following the merging of all individual MetaInfo instances into new ones, the resulting instance serves as the final MetaInfo instance used to determine the final type and value of a timex. It is worth noting that each final time segment corresponds to a single final MetaInfo instance.

4.2.4. Timex extraction and normalization

In this step, XTime extracts a timex from each final time segment and assigns the timex pre-defined type and standard-format value based on the information stored in the corresponding final MetaInfo instance.

Timex Extraction. From a final time segment, the timex is directly exported as a sequence of tokens from the sequence of token types.

Type Classification. The final type of a timex is determined by the "·timexType" attribute of the final MetaInfo instance corresponding to the final time segment. For example, as illustrated in Fig. 4, the identified timex "September 18, 2021 10:27:00" has its final type determined by the "·timexType" attribute of the final time segment metaInfo: TIME.

Value Normalization. Determining the final value of a timex from the final MetaInfo instance involves two primary sub-steps: *determining value format* and *determining final value*. Let us begin with the first sub-step of determining value format. Observation 7 indicates that a specific timex type corresponds to a limited set of value formats. Therefore, for each final MetaInfo instance, XTime firstly selects candidate value formats from the second column of Table 7 based on its "·timexType". Subsequently, XTime determines the final value format from these candidates based on the attribute compositions of the final MetaInfo instance listed in the last column of Table 7. For example, as shown in Fig. 4, the "·timexType" attribute of metaInfo is TIME, prompting XTime to firstly determine candidate value formats: "YYYY-MM-DDThh:mm:ss", "YYYY-MM-DDThh:mm", "YYYY-MM-DDThh", and "YYYY-MM-DDTdd". Then, considering the attributes of metaInfo (i.e., "·*YEAR:2021*", "·*MONTH:09*", "·*NUMBER:18*", and "·*TIME:10:27:00*"), XTime concludes that the value format for the example timex is "YYYY-MM-DDThh:mm:ss".

In the second sub-step of determining final value, XTime utilizes the "*attribute*:*value*" information from the final MetaInfo instance to derive the timex's final value. Observation 7 suggests that each value format is composed of different types of time tokens and numerals corresponding to different types of attributes. Therefore, XTime utilizes the attribute values to populate the positions in the value format. For example, as shown in Fig. 4, the attribute value "2021" fills the "YYYY" position, "09" fills "MM", "18" fills "DD", and "10:27:00" "hh:mm:ss". Finally, XTime normalizes the example timex into the standard value "2021-09-18T10:27:00".

5. Experiments

We evaluate the quality of XTime on four diverse datasets, including both in-domain and out-of-domain datasets, for the three sub-tasks of TERN (i.e., *timex recognition*, *type classification*, and *value normalization*) against eight representative state-of-the-art methods, including both rule-based and learning-based methods.

5.1. Experimental setup

Datasets. The evaluation of XTime spans across four diverse datasets: TE-3 [3], WikiWars [33], Tweets [34], and MEANTIME [35]. **TE-3** serves as a benchmark dataset widely used in timex analysis evaluations. **WikiWars** was initially constructed under the TIMEX2 scheme without timex types. To comprehensively analyze the characteristics of timex types and values, we manually annotate types for its timexes. Specifically, we teach two annotators the knowledge about timexes according to the standards of TimeML [69] and TimeBank [32], then the two annotators manually assign timex types to all WikiWars timexes independently. The initial agreement of their annotations is 98.94%, then they discuss to reach final agreement. **Tweets** was initially constructed for timex recognition and lacked precise annotation for the types and values of its timexes. To get a high-quality dataset for evaluation, the same two annotators are trained to correct the types and values according to TimeML and TimeBank standards. **MEANTIME** is a multilingual dataset with 120 English news articles collected from Wikinews and translations in Spanish, Italian, and Dutch. Despite TE-3 and MEANTIME containing corpora in different languages, only their English portions are considered for experiments.

These datasets are categorized into in-domain and out-of-domain datasets based on their usage during our data analysis and XTime's development. TE-3, WikiWars, and Tweets, used for our data analysis and XTime's design, are considered in-domain, while MEANTIME, not used in these processes, is considered out-of-domain. In terms of splitting training and test sets, TE-3 utilizes TimeBank as its training set and TE3Platinum as its test set [3]. For WikiWars and Tweets, we follow previous research protocols to set their training and test sets. All the data of MEANTIME serves as the test set. The performance evaluation of a model is conducted on test sets.

The statistical details of TimeBank, WikiWars, and Tweets are presented in Table 1 while those of TE3Platinum and MEANTIME are reported in Table 11.

State-of-the-art Baselines. We compare XTime to eight representative state-of-the-art methods: HeidelTime [22], SUTime [24], ClearTK [36], UWTime [29], TOMN [37], PTime [38], ARTime [30], and XTN [39]. HeidelTime and SUTime are rule-based methods, ClearTK, TOMN, PTime, and ARTime are learning-based methods, while UWTime and XTN are hybrid methods. Both **HeidelTime** and **SUTime** design deterministic rules for the end-to-end TERN task. **CearTK** derives lexical, syntactic and semantic features, and applies multiple classifiers like support vector machines and logistic regression for different subtasks of TERN. **TOMN** focuses on timex recognition by leveraging a constituent-based tagging scheme to model timexes under conditional random fields. **PTime** proposes to automatically generate abstracted patterns and uses extended budgeted maximum coverage model to select appropriate patterns for timex recognition while **ARTime** learns from training data to automatically generate normalization rules for timex normalization. ARTime has two versions: ARTime and ARTime+H. Ding et al. [30] report that ARTime+H performs better than ARTime, therefore we report the performance of ARTime+H in this paper. **UWTime** uses a combinatory categorial grammar and L1-regularization to learn linguistic information from context for the TERN task. **XTN** leverages a learning method based on XLM-RoBERTa for timex recognition and type classification, and a rule method based on a synchronous context free grammar for value normalization.

Evaluation Metrics. Like previous researches, we employ the widely used toolkit of TempEval-3 [3] to report the three standard metrics *Precision* (*Pre.*), *Recall* (*Rec.*), and F_1 (as defined by Eqs. (3), (4), and (5), respectively) under both *strict match* and *relaxed match* for the performance of TERN's three sub-tasks.

$$Pre. = \frac{TP}{TP + FP} \tag{3}$$

$$Rec. = \frac{TP}{TP + FN} \tag{4}$$

$$F_1 = \frac{2 \times Pre. \times Rec.}{Pre. + Rec.} \tag{5}$$

where TP is the number of evaluated targets that are in both ground-truth and prediction, FP is the number of evaluated targets that are in prediction but not in ground-truth, while FN is the number of evaluated targets that are in ground-truth but not in prediction.

Bethard [48] utilizes a metric called *normalization accuracy* (*Acc.*), as defined by Eq. (6), to evaluate the performance of a system in normalizing gold timexes.

$$Acc. = \frac{SYS}{GOLD} \tag{6}$$

where $GOLD$ is the number of total gold timexes while SYS is the number of timexes that are correctly predicted by a system. The task of normalizing gold timexes is known as gold timex normalization in Escribano et al. [39] and pure timex normalization in this paper (see Section 5.2.3), where all timexes are assumed to be correctly recognized, with 100% *Pre.*, *Rec.*, and F_1 in timex recognition.

Table 11
Statistics of TE3Platinum and MEANTIME.

Dataset	#Docs	#Words	#Timexes	#DATE	#TIME	#SET	#DURATION
TE3Platinum	20	6375	138	96	4	4	34
MEANTIME	120	13,982	484	400	18	4	62

Table 12
Overall performance of XTime and baselines on the three in-domain datasets. The best results are highlighted in bold while the second best are underlined. Some results are reported directly from their original papers or previous researches.

Dataset	Method	Timex recognition						Normalization	
		Strict match			Relaxed match			Type	Value
		Pre.	*Rec.*	F_1	*Pre.*	*Rec.*	F_1	F_1	F_1
TE-3	HeidelTime	83.85	78.99	81.34	93.08	87.68	90.30	82.1	77.6
	SUTime	78.72	80.43	79.57	89.36	91.30	90.32	80.3	67.4
	ClearTK	85.90	79.70	82.70	93.75	86.96	90.23	–	–
	UWTime	86.10	80.40	83.10	94.60	88.40	91.40	85.4	**82.4**
	TOMN	**92.59**	90.58	91.58	**95.56**	93.48	94.51	–	–
	PTime	85.19	83.33	84.25	92.59	90.58	91.58	–	–
	ARTime+H	–	–	–	–	–	–	86.0	78.7
	XTN-D	–	–	–	93.48	93.48	93.48	89.9	76.8
	XTN-N	–	–	–	95.31	88.41	91.73	88.7	79.7
	XTime	91.43	**92.75**	**92.09**	94.29	**95.65**	**94.96**	**90.7**	82.0
WikiWars	HeidelTime	85.20	79.30	82.10	92.60	86.20	89.30	89.3	74.7
	SUTime	78.61	76.69	76.64	95.74	89.57	92.55	85.3	38.8
	ClearTK	87.69	80.28	83.82	96.80	90.54	93.56	–	–
	UWTime	**87.70**	78.80	83.00	**97.60**	87.60	92.30	90.3	**78.1**
	TOMN	84.57	80.48	82.47	96.23	92.35	94.25	–	–
	PTime	86.86	**87.57**	**87.21**	95.98	**96.76**	**96.37**	–	–
	ARTime+H	–	–	–	–	–	–	91.0	47.9
	XTime	80.00	80.22	80.11	92.16	92.41	92.29	**91.3**	75.3
Tweets	HeidelTime	89.58	72.88	80.37	95.83	77.97	85.98	76.4	71.3
	SUTime	76.03	77.97	76.99	88.43	90.68	89.54	82.5	67.4
	ClearTK	86.83	75.11	80.54	96.59	83.54	89.59	–	–
	UWTime	88.54	72.03	79.44	96.88	78.81	86.92	80.0	82.4
	TOMN	90.69	**94.51**	92.56	93.52	97.47	95.45	–	–
	PTime	**92.92**	94.09	**93.50**	**97.92**	**99.16**	**98.53**	–	–
	ARTime+H	–	–	–	–	–	–	92.9	89.1
	XTime	89.52	94.07	91.74	93.55	98.31	95.87	**96.9**	**93.5**

Under pure timex normalization, $Acc.$ is equivalent to $Rec.$ because SYS is equivalent to TP and $GOLD = TP + FN$; moreover, the two metrics $Pre.$ and $Rec.$ have the same value since the number of evaluated targets in prediction equals to the one in ground-truth, namely $TP + FP = TP + FN$.[11] Consequently, under pure timex normalization, all four metrics $Pre.$, $Rec.$, F_1, and $Acc.$ exhibit identical values.

5.2. Experimental results

We firstly report experimental results on in-domain datasets (i.e., TE3, WikiWars, and Tweets), and then report the ones on out-of-domain dataset (i.e., MEANTIME).

5.2.1. Results on in-domain datasets

Table 12 reports the overall performance of XTime and the eight baselines on the three in-domain datasets in the TERN task. Since XTime is an extension of SynTime [34], which focuses on timex recognition, we directly report the performance of timex recognition from Zhong et al. [34] for XTime. In this paper, we are mainly concerned with the performance of XTime in timex normalization.

Performance in Timex Normalization.

Table 12 shows that for timex normalization (which contains two sub-tasks: *type classification* and *value normalization*), XTime achieves five best and six second best results among the total six measures. Particularly, XTime significantly outperforms the four baselines (i.e., HeidelTime, SUTime, UWTime, and ARTime+H) in type classification. Specifically, XTime achieves the F_1 in type classification with 90.7% on TE-3, 91.3% on WikiWars, and 96.9% on Tweets. This confirms the importance of mapping relations from token times to timex types (see Observation 5) and the priority relationship among timex types (see Observation 6) in timex type classification.

XTime achieves the best results in value normalization on TE-3 and Tweets. The main reason is that XTime's heuristic rules are designed under the annotation scheme of TimeML [69] and TimeBank [32], and both TE-3 and Tweets are constructed under the same annotation scheme. On WikiWars, XTime performs slightly worse than UWTime (76.7% vs. 78.1%), mainly because many timexes in WikiWars are quite descriptive [34] and their values are somewhat deviated from the TimeML scheme. For example, under the TimeML scheme, the timex "two days after his arrival in Jerusalem" in WikiWars should be pruned to "two days". UWTime leverages combinatory categorial grammar to capture the information of linguistic structure and learns from training data to adapt to new annotation scheme. XTime lacks such ability of adapting to new annotation scheme.

XTime vs. Rule-based Baselines. We particularly compare XTime with the two rule-based baselines. Table 12 shows that XTime significantly outperforms HeidelTime and SUTime on all three datasets by large margins of 2.0~14.4 points in type classification and 0.6~22.3 points in value normalization. Especially on TE-3 and Tweets, the margins in type classification reach 8.6~14.4 points and the ones in value normalization reach 5.2~22.3 points. The main reason is that HeidelTime and SUTime design deterministic rules in a fixed way without a deep understanding of timex types and values. By contrast,

[11] Note that under pure timex normalization, a system is required to normalize all gold timexes, regardless of whether it can recognize or accurately normalize them; namely, the system has to assign a type and a value to each gold timex. Therefore, under pure timex normalization, the number of total predicted timexes equals to the one of total gold timexes

Table 13
Overall performance of XTime and baselines on the out-of-domain dataset, MEANTIME. The best results are highlighted in bold and the second best are underlined. The results of baselines are reported directly from [39]. Note that for type classification and value normalization, the results of the two baselines are evaluated under relaxed match, whereas those of XTime are under strict match; consequently, direct comparisons between XTime and the two baselines may not be appropriate.

Method	Timex recognition						Normalization	
	Strict match			Relaxed mtch			Type	Value
	Pre.	*Rec.*	F_1	*Pre.*	*Rec.*	F_1	F_1	F_1
HeidelTime	–	–	–	94.91	84.71	89.52	85.15	**79.69**
XTN-D	–	–	–	95.48	91.74	93.57	88.72	76.29
XTN-N	–	–	–	**96.37**	87.81	91.89	82.31	78.49
XTime	**83.58**	**80.99**	**82.27**	95.74	**92.77**	**94.23**	**89.88**	74.08

XTime leverages three kinds of meta information (i.e., token triples, mapping relations, and priority relationship) based on characteristics about timex types and values (see Section 3.2) to design heuristic rules in a general and flexible way.

XTime vs. Learning-based Baselines. XTime outperforms the two learning-based baselines, namely UWTime and ARTime, on all three datasets in type classification. This indicates that the mapping relations from time tokens to timex types and the priority relationship among timex types are very strong, extremely useful for timex type classification. Learning-based methods might not be able to fully capture such strong mapping relations and priority relationship. XTime outperforms UWTime and ARTime on the two TimeML-based datasets, namely TE-3 and Tweets, in value normalization. This verifies the usefulness of Observation 7 for timex value normalization under TimeML and TimeBank annotation scheme.

Performance in Timex Recognition.

XTime significantly outperforms the two rule-based baselines (i.e., HeidelTime and SUTime) in timex recognition and performs comparably with those best results of learning-based baselines in terms of F_1 on all three in-domain datasets under both strict match and relaxed match. Compared to learning-based methods, XTime does not require training and runs in real-time. This demonstrates the effectiveness and efficiency of XTime, especially in practice.

5.2.2. Results on out-of-domain dataset

Table 13 presents the overall performance of XTime and the two multilingual baselines, namely HeidelTime and XTN (which includes two versions: XTN-D and XTN-N), on the out-of-domain dataset, MEANTIME. The results of HeidelTime and XTN are reported directly from Escribano et al. [39] and all these results are evaluated under relaxed match.[12] Table 13 shows that XTime outperforms HeidelTime and XTN by at least 0.66 points at F_1 under relaxed match in timex recognition and at least 1.16 points at F_1 in type classification. This underscores the effectiveness of XTime on out-of-domain dataset and underscores its practical utility. However, in terms of value normalization, XTime exhibits lower performance compared to HeidelTime and XTN, with an F_1 deficit ranging from 2.21 to 5.61 points compared to the two baselines. Several factors contribute to this discrepancy.

Firstly, some annotations of MEANTIME differ from those of TimeBank and TimeML. For example, in MEANTIME, the timex "12:00 UTC Monday" is annotated with the value "2009-06-01T12:00Z", which includes a "Z" at the end, while in TimeBank, no such "Z" is included. There are 14 "Z"-ending timexes (which is about 3% of timexes) in MEANTIME. Additionally, inconsistent annotations exist in MEANTIME; for example, some "now" are annotated as "PRESENT_REF" while others are assigned specific dates. Similarly, some "today" are annotated with specific dates while other are labeled as "PRESENT_REF". By contrast, TankBank consistently annotates "now" as "PRESENT_REF" and "today" with specific dates. Secondly, annotation errors in MEANTIME contribute to the disparity in performance. For example, a timex "2012" is annotated with the value "2010" (which should be "2012"), a "2007-08-07" with "2007-07-08" ("2007-08-07"); a "the next six months" with "PM6" ("P6M"). Moreover, errors such as annotating "weekly" as "P1W" (which should be "XXXX-WXX"), "decades" as "PXD" ("PXDE"), and mislabeling specific dates for days of the week (e.g., annotating "Wednesday" as "2009-11-05" (which should be "2009-11-04") and "Friday" as "2007-10-20" ("2007-10-19")) also contribute to discrepancies. Thirdly, MEANTIME includes some single words like "this", "which", "ongoing", and "later" as timexes, whereas XTime does not treat such single words as timexes.

5.2.3. Results on pure timex normalization

Let us examine XTime's performance specifically in the task of pure timex normalization, where all timexes are assumed to be correctly recognized, with 100% precision, recall, and F_1 in timex recognition. The results of XTime in pure timex normalization are reported in Table 14, denoted by "XTime in Pure Norm". The table shows that compared to its performance in end-to-end TERN task, XTime exhibits superior performance across all three datasets (including both in-domain and out-of-domain datasets), with F_1 improvements ranging from 1.4 to 3.7 points in type classification and from 0.2 to 0.9 points in value normalization. These improvements underscore the importance of improving timex recognition accuracy in enhancing the performance of timex normalization.

As described in Section 5.1, under pure timex normalization, all four metrics *Pre.*, *Rec.*, F_1, and *Acc.* exhibit identical values. Such equivalence implies that the F_1 scores of XTime in pure timex normalization can be considered as *Acc.* results. Consequently, we can compare these *Acc.* results to those reported in previous studies conducted on the same datasets. For example, [48] reported an accuracy of 81.6% for their SCFG method on TE-3 while [39] reported 78.99% *Acc.* for their XTN method on TE-3 and 76.86% *Acc.* on MEANTIME, specifically focusing on value normalization. Table 14 illustrates that XTime attains an accuracy of 82.6% in value normalization on TE-3, surpassing both SCFG and XTN. Conversely, on MEANTIME, XTime exhibits inferior performance in value normalization compared to XTN (75.0% vs. 76.86%); the main factors for this discrepancy are discussed in Section 5.2.2.

5.2.4. Factor analysis

In XTime, token triples and mapping relations are necessary, so we analyze the impact of the priority relationship among timex types. To assess its impact, we remove the priority component from XTime by setting the ·timexType of the first MetaInfo as the final timex type. The results of XTime without priority component are presented in Table 14, denoted by "XTime w/o Priority". Notably, upon removing the priority component, XTime demonstrates inferior performance in both type classification and value normalization across all three datasets.

[12] It is worth noting that [39] report these results of all three sub-tasks (i.e., timex recognition, type classification, and value normalization) under relaxed match but do not provide results under strict match. Additionally, they provide codes for processing output files for evaluation (available at https://github.com/NGEscribano/XTN-timexes), but do not provide codes of their model for timex recognition and type classification. Therefore, we do not report their results under strict match in Table 13.

Table 14
Performance (F_1) of XTime in pure timex normalization and factor analysis of priority relationship among timex types.

Method	TE-3		WikiWars		Tweets		MEANTIME	
	Type	Value	Type	Value	Type	Value	Type	Value
XTime	90.7	82.0	91.3	75.3	96.9	93.5	89.9	74.1
XTime in Pure Norm	**92.8**	**82.6**	**94.7**	**76.2**	**98.3**	**93.7**	**93.6**	**75.0**
XTime w/o Priority	85.6	78.4	91.0	72.3	89.8	87.3	84.4	71.8

Specifically, there are decreases ranging from 0.3 to 7.1 points in type classification and from 2.3 to 6.2 points in value normalization. These results underscore the critical importance of the priority relationship among timex types for both sub-tasks of timex normalization.

5.2.5. Error analysis

During the evaluation of XTime, three primary types of errors are identified. Firstly, there are annotation errors present in datasets. For example, TE-3 annotates "the next decade" with the value "P10Y" while annotates "the following decade" as "P1DE". Secondly, there is a deficiency of time-related keywords, such as descriptive terms like "tenure" and "digital", which are not included in XTime's collection of token triples. Thirdly, there are instances of incorrect reference dates, such as XTime normalizing a timex from the Tweets dataset, "last week", to "2014-W22" instead of the correct "2014-W21". Further analysis of annotation errors specific to the MEANTIME dataset is elaborated in Section 5.2.2.

5.3. Limitations

While XTime exhibits strong performance across both in-domain and out-of-domain datasets, it is important to acknowledge three primary limitations. Firstly, XTime's design is tailored to the annotation scheme of TimeML [69] and TimeBank [32]. Consequently, it may encounter challenges when applied to datasets constructed under different annotation schemes that diverge significantly from TimeML and TimeBank, such as MEANTIME, which features value formats ending with a "Z" (i.e., "2009-06-01T12:00Z"), or datasets utilizing alternative schemes like the SCATE scheme and the SCATE corpus [58]. Secondly, XTime assumes correct tokenization and POS tagging of words. However, in practice, word tokenization and tagging may not be error-free due to limitations in the tools used. For example, the Stanford POS Tagger may misclassify words, such as assigning VBD to the word "sat" in the string "friday or sat", where "sat" should be tagged as "NNP". Thirdly, certain datasets may consider some single words (e.g., "this", "which", and "ongoing" in MEANTIME) as timexes, whereas XTime adheres to the TimeBank and TimeML annotation standards and exclude such single words from consideration as timexes.

6. Conclusion and future work

Through an analysis of timexes across four diverse datasets, we identify seven important characteristics pertaining to the constituents, types, and values of these timexes. Leveraging these insights, we introduce XTime, a rule-based method designed to recognize and normalize timexes from unstructured text into standardized type and value formats. By incorporating three kinds of meta time information and employing general heuristic rules, XTime demonstrates superior performance in timex normalization across both in-domain and out-of-domain datasets compared to representative state-of-the-art models. Furthermore, XTime exhibits competitive performance in the task of timex recognition. Notably, XTime's domain-agnostic nature and light-weight architecture enable real-time execution, making it a versatile tool applicable across various domains and text types. With its potential to serve as a reliable timex tagger for diverse time-related linguistic tasks, future improvements may involve leveraging transfer learning techniques to adapt to alternative annotation schemes and harnessing large language models like ChatGPT to augment data for few-shot timex recognition and normalization.

CRediT authorship contribution statement

Xiaoshi Zhong: Writing – review & editing, Writing – original draft, Investigation, Formal analysis, Conceptualization. **Chenyu Jin:** Software, Resources, Methodology. **Mengyu An:** Formal analysis, Data curation. **Erik Cambria:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their valuable feedback and constructive suggestions, which have significantly improved the quality of our paper. The authors would also like to thank Aixin Sun for his discussions during the previous conference paper. This research was mainly supported by a startup funding (Project No. : XSQD-202007008) from Beijing Institute of Technology, China, and partially supported by the AME Programmatic Funding (Project No. : A18A2b0046) from Agency for Science, Technology and Research (A*STAR), Singapore.

References

[1] M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, J. Pustejovsky, SemEval-2007 task 15: TempEval temporal relation identification, in: Proceedings of the 4th International Workshop on Semantic Evaluation, 2007, pp. 75–80.
[2] M. Verhagen, R. Sauri, T. Caselli, J. Pustejovsky, SemEval-2010 task 13: TempEval-2, in: Proceedings of the 5th International Workshop on Semantic Evaluation, 2010, pp. 57–62.
[3] N. UzZaman, H. Llorens, L. Derczynski, M. Verhagen, J. Allen, J. Pustejovsky, SemEval-2013 task 1: TempEval-3: Evaluating time expressions, events, and temporal relations, in: Proceedings of the 7th International Workshop on Semantic Evaluation, 2013, pp. 1–9.
[4] F. Cheng, Y. Miyao, Inducing temporal relations from time anchor annotation, in: Proceedings of NAACL-HLT 2018, 2018, pp. 1833–1843.
[5] H.-J. Lee, Y. Zhang, M. Jiang, J. Xu, C. Tao, H. Xu, Identifying direct temporal relations between time and events from clinical notes, BMC Med. Inform. Decis. Mak. 18 (2) (2018) 49.
[6] A. Naik, L. Breitfeller, C. Rose, Tddiscourse: A dataset for discourse-level temporal ordering of events, in: Proceedings of the SIGDial 2019 Conference, 2019, pp. 239–249.
[7] J. Niu, V. Ng, G. Penn, E.E. Rees, Temporal histories of epidemic events (THEE): A case study in temporal annotation for public health, in: Proceedings of the 12th Conference on Language Resources and Evaluation, 2020, pp. 2223–2230.
[8] J. Liu, J. Xu, Y. Chen, Y. Zhang, Discourse-level event temporal ordering with uncertainty-guided graph completion, in: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, 2021, pp. 3871–3877.
[9] Q.X. Do, W. Lu, D. Roth, Joint inference for event timeline construction, in: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012, pp. 677–687.
[10] J. Li, C. Cardie, Timeline generation: Tracking individuals on twitter, in: Proceedings of the 23rd International Conference on World Wide Web, 2014, pp. 643–652.

[11] A.-L. Minard, M. Speranza, E. Agirre, I. Aldabe, M. van Erp, B. Magnini, G. Rigau, R. Urizar, Semeval-2015 task 4: Timeline: Cross-document event ordering, in: 9th International Workshop on Semantic Evaluation (SemEval 2015), 2015, pp. 778–786.

[12] S. Alsayyahi, R. Batista-Navarro, TIMELINE: Exhaustive annotation of temporal relations supporting the automatic ordering of events in news articles, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023.

[13] O. Alonso, J. Strotgen, R. Baeza-Yates, M. Gertz, Temporal information retrieval: Challenges and opportunities, in: Proceedings of 1st International Temporal Web Analytics Workshop, 2011, pp. 1–8.

[14] R. Campos, G. Dias, A.M. Jorge, A. Jatowt, Survey of temporal information retrieval and related applications, ACM Comput. Surv. 47 (2) (2014) 15.

[15] R. Campos, J. Duque, T. Cândido, J. Mendes, G. Dias, A. Jorge, C. Nunes, Time-matters: Temporal unfolding of texts, in: European Conference on Information Retrieval, (492–497) 2021.

[16] M.-M. Rahoman, R. Ichise, A proposal of a temporal semantics aware linked data information retrieval framework, J. Intell. Inf. Syst. 50 (3) (2018) 573–595.

[17] A. Leeuwenberg, M.-F. Moens, A survey on temporal reasoning for temporal information extraction from text, J. Artif. Intell. Res. 66 (2019) 341–380.

[18] L. Qin, A. Gupta, S. Upadhyay, L. He, Y. Choi, M. Faruqui, TIMEDIAL: Temporal commonsense reasoning in dialog, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 7066–7076.

[19] Z. Jia, A. Abujabal, R.S. Roy, J. Strotgen, G. Weikum, TempQuestions: A benchmark for temporal question answering, in: Proceedings of the 2018 World Wide Web Conference Companion, 2018, pp. 1057–1062.

[20] Z. Jia, S. Pramanik, R.S. Roy, G. Weikum, Complex temporal question answering on knowledge graphs, in: Proceedings of the 30th ACM International Conference on Information and Knowledge Management, 2021, pp. 792–802.

[21] M. Verhagen, I. Mani, R. Sauri, R. Knippen, S.B. Jang, J. Littman, A. Rumshisky, J. Phillips, I. Mani, R. Sauri, R. Knippen, S.B. Jang, J. Littman, A. Rumshisky, J. Phillips, J. Pustejovsky, Automating temporal annotation with TARQI, in: Proceedings of the ACL Interactive Poster and Demonstration Sessions., 2005, pp. 81–84.

[22] J. Strötgen, M. Gertz, HeidelTime: High quality rule-based extraction and normalization of temporal expressions, in: Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval'10), Association for Computational Linguistics, Stroudsburg, PA, USA, 2010, pp. 321–324.

[23] J. Strotgen, J. Zell, M. Gertz, HeidelTime: Tuning english and developing spanish resources, in: Proceedings of Second Joint Conference on Lexical and Computational Semantics, SEM, 2013, pp. 15–19.

[24] A.X. Chang, C.D. Manning, Sutime: A library for recognizing and normalizing time expressions, in: Proceedings of 8th International Conference on Language Resources and Evaluation, 2012, pp. 3735–3740.

[25] A.X. Chang, C.D. Manning, Sutime: Evaluation in TempEval-3, in: Proceedings of Second Joint Conference on Lexical and Computational Semantics, SEM, 2013, pp. 78–82.

[26] H. Llorens, L. Derczynski, R. Gaizauskas, E. Saquete, TIMEN: An open temporal expression normalisation resource, in: Proceedings of 8th International Conference on Language Resources and Evaluation, 2012, pp. 3044–3051.

[27] G. Angeli, C.D. Manning, D. Jurafsky, Parsing time: Learning to interpret time expressions, in: Proceedings of 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2012, pp. 446–455.

[28] G. Angeli, J. Uszkoreit, Language-independent discriminative parsing of temporal expressions, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, 2013, pp. 83–92.

[29] K. Lee, Y. Artzi, J. Dodge, L. Zettlemoyer, Context-dependent semantic parsing for time expressions, in: Proceedings of the 52th Annual Meeting of the Association for Computational Linguistics, 2014, pp. 1437–1447.

[30] W. Ding, J. Chen, J. Li, Y. Qu, Automatic rule generation for time expression normalization, in: Findings of the Association for Computational Linguistics: EMNLP 2021, 2021.

[31] Y. Cao, W. Groves, T.K. Saha, J.R. Tetreault, A. Jaimes, H. Peng, P.S. Yu, Xltime: A cross-lingual knowledge transfer framework for temporal expression extraction, in: Findings of the 2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2022.

[32] J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, B. Sundheim, D. Radev, D. Day, L. Ferro, M. Lazo, The TIMEBANK corpus, Corpus Linguist. 2003 (2003) 647–656.

[33] P. Mazur, R. Dale, WikiWars: A new corpus for research on temporal expressions, in: Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2010, pp. 913–922.

[34] X. Zhong, A. Sun, E. Cambria, 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, Vancouver, Canada, 2017, pp. 420–429.

[35] A.-L. Minard, M. Speranza, R. Urizar, B. Altuna, M.V. Erp, A. Schoen, C.V. Son, MEANTIME, the NewsReader multilingual event and time corpus, in: Proceedings of the Tenth International Conference on Language Resources and Evaluation, LREC'16, 2016, pp. 4417–4422.

[36] S. Bethard, ClearTK-TimeML: A minimalist approach to TempEval 2013, in: Proceedings of the 7th International Workshop on Semantic Evaluation, 2013, pp. 10–14.

[37] X. Zhong, E. Cambria, Time expression recognition using a constituent-based tagging scheme, in: Proceedings of the 2018 World Wide Web Conference, Lyon, France, 2018, pp. 983–992.

[38] W. Ding, G. Gao, L. Shi, Y. Qu, A pattern-based approach to recognizing time expressions, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, (01) 2019, pp. 6335–6342.

[39] N. Escribano, G. Rigau, R. Agerri, A modular approach for multilingual timex detection and normalization using deep learning and grammar-based methods, Knowl.-Based Syst. 273 (2023) 110612.

[40] S. Bethard, L. Derczynski, G. Savova, J. Pustejovsky, M. Verhagen, SemEval-2015 task 6: Clinical TempEval, in: Proceedings of the 9th International Workshop on Semantic Evaluation, 2015, pp. 806–814.

[41] S. Bethard, G. Savova, W.-T. Chen, L. Derczynski, J. Pustejovsky, M. Verhagen, SemEval-2016 task 12: Clinical TempEval, in: Proceedings of the 10th International Workshop on Semantic Evaluation, 2016, pp. 1052–1062.

[42] S. Bethard, G. Savova, M. Palmer, J. Pustejovsky, SemEval-2017 task 12: Clinical TempEval, in: Proceedings of the 11th International Workshop on Semantic Evaluation, 2017, pp. 565–572.

[43] E. Laparra, D. Xu, S. Bethard, A.S. Elsayed, M. Palmer, SemEval 2018 task 6: Parsing time normalizations, in: Proceedings of the 12th International Workshop on Semantic Evaluation, 2018, pp. 88–96.

[44] X. Zhong, E. Cambria, Time expression recognition and normalization: A survey, Artif. Intell. Rev. 56 (9) (2023) 9115–9140.

[45] N. UzZaman, J.F. Allen, TRIPS and TRIOS system for TempEval-2: Extracting temporal information from text, in: Proceedings of the 5th International Workshop on Semantic Evaluation, 2010, pp. 276–283.

[46] M. Filannino, G. Brown, G. Nenadic, ManTIME: Temporal expression identification and normalization in the TempEval-3 challenge, in: Proceedings of the 7th International Workshop on Semantic Evaluation, 2013.

[47] Q. Ning, B. Zhou, Z. Feng, H. Peng, D. Roth, CogCompTime: A tool for understanding time in natural language, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2018, pp. 72–77.

[48] S. Bethard, A synchronous context free grammar for time normalization, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2013, 2013, p. 821.

[49] X. Zhong, Time Expression and Named Entity Analysis and Recognition (Ph.D. thesis), Nanyang Technological University, Singapore, 2020.

[50] X. Zhong, E. Cambria, Time Expression and Named Entity Recognition, in: Socio-Affective Computing, Springer Nature, 2021.

[51] W. Ding, J. Chen, L. E, J. Li, Y. Qu, Time expression as update operations: Normalizing time expressions via a distantly supervised neural semantic parser, Knowl.-Based Syst. 278 (2023) 110870.

[52] H. Llorens, E. Saquete, B. Navarro, TIPSem (english and spanish): Evaluating CRFs and semantic roles in TempEval-2, in: Proceedings of the 5th International Workshop on Semantic Evaluation, 2010, pp. 284–291.

[53] X. Zhong, E. Cambria, A. Hussain, Extracting time expressions and named entities with constituent-based tagging schemes, Cogn. Comput. 12 (4) (2020) 844–862.

[54] X. Zhong, E. Cambria, A. Hussain, Does semantics aid syntax? An empirical study on named entity recognition and classification, Neural Comput. Appl. (2021).

[55] Z.M. Kim, Y.-S. Jeong, TIMEX3 and event extraction using recurrent neural networks, in: Proceedings of the 2016 IEEE International Conference on Big Data and Smart Computing, 2016, pp. 450–453.

[56] C. Lin, T. Miller, D. Dligach, S. Bethard, G. Savova, Representations of time expressions for temporal relation extraction with convolutional neural networks, in: Proceedings of the 16th Workshop on Biomedical Natural Language Processing, 2017, pp. 322–327.

[57] M. Etcheverry, D. Wonsever, Time expressions recognition with word vectors and neural networks, in: Proceedings of the 24th International Symposium on Temporal Representation and Reasoning, 2017, pp. 1–12.

[58] E. Laparra, D. Xu, S. Bethard, From characters to time intervals: New paradigms for evaluation and neural parsing of time normalizations, Trans. Assoc. Comput. Linguist. 6 (2018) 343–356.

[59] S. Vashishth, S.S. Dasgupta, S.N. Ray, P. Talukdar, Dating documents using graph convolution networks, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2018, pp. 1605–1615.

[60] S. Chen, G. Wang, B. Karlsson, Exploring word representations on time expression recognition, Technical Report, Microsoft Research Asia, 2019.

[61] A. Kim, C. Pethe, S. Skiena, What time is it? Temporal analysis of novels, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020, pp. 9076–9086.

[62] L. Lange, A. Iurshina, H. Adel, J. Strötgen, Adversarial alignment of multilingual models for extracting temporal expressions from text, in: Proceedings of the 5th Workshop on Representation Learning for NLP, 2020, pp. 103–109.

[63] B. Patra, C. Fufa, P. Bhattacharya, C.C. Lee, To schedule or not to schedule: Extracting task specific temporal entities and associated negation constraints, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, 2020, pp. 8445–8455.

[64] S. Almasian, D. Aumiller, M. Gertz, Time for some German? Pre-training a transformer-based temporal tagger for German, in: Proceedings of the Text2Story'22 Workshop, 2022, pp. 83–90.

[65] L. Lange, J. Strotgen, H. Adel, D. Klakow, Multilingual normalization of temporal expressions with masked language models, 2022, ArXiv Preprint arXiv:2205.10399.

[66] X. Zhong, X. Yu, E. Cambria, J.C. Rajapakse, Marshall–Olkin power-law distributions in length-frequency of entities, Knowl.-Based Syst. 279 (2023) 110942.

[67] K. Toutanova, D. Klein, C.D. Manning, Y. Singer, Feature-rich part-of-speech tagging with a cyclic dependency network, in: Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, 2003, pp. 252–259.

[68] C. Manning, H. Schutze, Foundations of Statistical Natural Language Processing, MIT Press, Cambride, 1999.

[69] J. Pustejovsky, J. Castano, R. Ingria, R. Sauri, R. Gaizauskas, A. Setzer, G. Katz, D. Radev, Timeml: Robust specification of event and temporal expressions in text, New Directions in Question Answering 3 (2003) 28–34.