Time Expression Normalization with Meta Time Information

Mengyu An[1,*], Chenyu Jin[1,*], Xiaoshi Zhong[1,2,#], and Erik Cambria[3]
[1]*School of Computer Science and Technology, Beijing Institute of Technology, China*
[2]*China North Industries Computer Application Technology Institute, NORINCO Group Co. Ltd., China*
[3]*School of Computer Science and Engineering, Nanyang Technological University, Singapore*
{anmy, cyjin, xszhong}@bit.edu.cn; cambria@ntu.edu.sg

Regular Research Paper at the Research Track on Big Data and Data Science (CSCI-RTBD)

***Abstract*—Time expression (a.k.a., timex) normalization is a fundamental task for many downstream researches and applications. Previous researches mainly developed deterministic rules and machine-learning methods for the end-to-end task of timex recognition and normalization (TERN). However, deterministic rules heavily depend on specific domains while machine-learning methods are somewhat unexplainable. To better understand the task, we analyze three diverse benchmark datasets for the characteristics of timex types and values. According to these characteristics, we propose a rule-based method termed MetaTime[1] with three kinds of meta time information to normalize timexes into standard type and value formats. MetaTime is independent of specific domains and textual types. Experimental results on three diverse benchmark datasets demonstrate that MetaTime outperforms four representative state-of-the-art methods.**

***Index Terms*—time expression normalization, meta time information, token triples, mapping relations, priority relationship**

I. Introduction

Time expression (i.e., timex) recognition and normalization (TERN) is a fundamental task for numerous downstream researches and applications, such as temporal event extraction [19], [31], [32], [34], timeline construction [1], [11], [23], temporal reasoning [18], [27], and temporal question answering [13], [14]. TERN includes two sub-tasks: *timex recognition* and *timex normalization*. Timex recognition has achieved considerable progress in the pass few years [8], [10], [35]–[40]. However, timex normalization remains a challenge.

Timex normalization aims to normalize timexes to the standard formats of *type* and *value*. It contains two sub-tasks: *type classification* and *value normalization*. For example, for the timex "September 18, 2021", the goal of the task is to normalize it into the type of DATE and the value of "2021-09-18". Previously, timex normalization was mainly resolved by deterministic rules [6], [7], [20], [28], [29], [33] and learning methods [2], [3], [5], [9], [17]. However, deterministic rules heavily depend on specific domains, which lacks flexibility; while learning-based methods are somewhat unexplainable. To our knowledge, there is no research that systematically analyzes what factors affect timex types and values.

In this paper, we analyze timexes from three diverse benchmark datasets for the characteristics of timex types and values. Our analysis uses the three kinds of token types (i.e., *time token*, *modifier*, and *numeral*) defined by [40] for timex constituents. From the analysis we have three observations. Firstly, time tokens and timex types have strong mapping relations (see Observation 1 for details). Secondly, there exists a priority relationship among the four timex types (see Observation 2). Thirdly, standard timex values have only some formats and these value formats are mainly composed of different types of time tokens and numerals (see Observation 3).

According to these observations, we propose a rule-based method termed MetaTime to normalize timexes into standard type and value formats. Specifically, MetaTime uses three kinds of meta time information (i.e., token triples, mapping relations from time tokens to timex types, and priority among timex types; see Section IV-A) to capture the information of all the words of a timex and stores these meta information in a structure called MetaInfo (see Section IV-B2). Finally, MetaTime uses the MetaInfo to determine a timex's standard type and value (see Section IV-B). MetaTime designs rules on top of token types and therefore is independent of specific domains and specific textual types.

We evaluate the quality of MetaTime on three benchmark datasets (i.e., TE-3 [31], WikiWars [22], and Tweets [40]) against four representative state-of-the-art methods (i.e., HeidelTime [29], SUTime [7], UWTime [17] and ARTime [9]). Experimental results demonstrate that MetaTime achieves the best results in type classification on all the three datasets and the best results in value normalization on the TE-3 and Tweets datasets, in comparison with the four state-of-the-art methods (See Section V for details). Moreover, MetaTime runs in real-time and can be easily used as a basic tool for other time-related linguistic tasks, such as temporal reasoning [18], [27] and temporal question answering [13], [14].

To summarize, we mainly make the following contributions.

- We analyze three benchmark datasets to understand the task of timex normalization and summarize three statistical characteristics about timex types and values.
- We propose a simple rule-based method termed MetaTime to normalize timexes into standard type and value formats. MetaTime is independent of specific domains and textual types, and runs in real-time and can be easily used as a basic tool for other time-related tasks.
- Experiments on three datasets demonstrate that MetaTime outperforms four representative state-of-the-art baselines.

*Mengyu An and Chenyu Jin contributed equally. #Xiaoshi Zhong is the corresponding author. This research was mainly supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A18A2b0046).

[1]Source codes will be available at https://github.com/xszhong/MetaTime.

II. Related Works

Many researches on timex normalization are reported in the series of TempEval competitions [31], [34]. The methods for timex normalization could be categorized into two main types: rule-based methods and learning-based methods.

Rule-based Methods for Timex Normalization. Rule-based time taggers like HeidelTime and SUTime mainly design deterministic rules to normalize timexes [6], [7], [20], [28]–[30], [33]. Many of them are designed for TERN (e.g, HeidelTime and SUTime). Some time taggers are designed with mixed methods. For example, ManTime [12], ClearTK-TimeML [4], and CogCompTime [24] propose learning methods for timex recognization while develop rules for timex normalization.

Learning-based Methods for Timex Normalization. Many learning-based methods for timex normalization are also developed for the TERN task. [2] define a compositional grammar and employ an EM-style method to learn a latent parser for TERN. UWTime [17] leverages a combinatory categorial grammar (CCG) and employ L1-regularization to learn linguistic information from context for TERN. [9] propose to automatically generate rules from training data for timex normalization. In fact, most learning-based methods design rules to determine the final timex values, such as TIPSem [21] and ClearTK-TimeML [4]. [16] normalize clinical timexes by a neural-network method while [15] normalize multilingual timexes with masked language models.

MetaTime is a rule-based time normalizer and focuses on English. Compared with previous rule-based methods that design rules in a deterministic way, MetaTime designs rules in a heuristic and flexible way and is independent of specific domains and textual types. Compared with learning-based methods that are somewhat unexplainable, MetaTime is based on a systematic analysis of the characteristics about timex types and values and provides detailed explanation for experimental results. Moreover, MetaTime runs in real-time.

III. Data Analysis

A. Datasets

We analyze the following three benchmark datasets for the characteristics of timex types and values: TimeBank [26], WikiWars [22], and Tweets [40]. TimeBank contains 183 news articles and is used in TempEval competitions [31], [32], [34]. WikiWars is a domain-specific dataset collected from Wikipedia articles about 22 famous wars. Tweets is a dataset about timexes in informal text and consists of 942 tweets collected from Twitter. Table I summarizes the statistics of the three benchmark datasets.

B. Observations

While the three datasets are varied in many aspects (e.g., corpus sizes and domains), we will see that the types and values of their timexes demonstrate similar characteristics.

Observation 1: Time tokens and timex types have strong mapping relations: a specific type of time tokens mainly appear in a specific type of timexes.

TABLE I: Statistics of the three benchmark datasets

Dataset	#Words	#Timex	#DATE	#TIME	#SET	#DURATION
TimeBank	61418	1243	1016	22	16	189
WikiWars	119468	2671	2247	118	13	258
Tweets	18199	1129	761	181	36	151

Table II reports the percentage of time tokens that appear in the four types of timexes within individual dataset. The format of results in Table II is "Pr/Pr^1", where Pr denotes the percentage of time tokens that appear in the type of **the whole timexes**, as defined by Eq. (1); while Pr^1 denotes the percentage of time tokens that appear in the type of **the one-word timexes**, as defined by Eq. (2).

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

where T denotes 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. (There are 17 types of time tokens in total; see the first column of Table II.) $Count(W)$ denotes the total number of the W type of time tokens, while $Count(W,T)$ denotes the number of the W type of time tokens that appear in the T type of all timexes. For each W, we have $\sum_T Pr(W,T) = 100\%$.

$Pr(W,T)$ calculates the percentage of the W type of time tokens that appear in the T type of timexes by considering **the whole timexes**. By contrast, $Pr^1(W,T)$ calculates the percentage by considering only **the 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 the W type of time tokens that appear in only the one-word timexes, while $Count^1(W,T)$ denotes the number of the W type of time tokens that appear in the T type of the one-word timexes. $Pr^1(W,T)$ indicates that within one-word timexes, the percentage of the W type of time tokens that appear in the T type of timexes. Similarly, for each W, $\sum_T Pr^1(W,T) = 100\%$.

Table II shows that a specific type of time tokens mainly appear in a specific type of timexes in both the whole timexes and only the one-word timexes. The high percentages indicate strong mapping relations from time tokens to timex types. For example, 100% of TIME appearing in <u>TIME</u> indicates that if a timex contains a TIME, then the timex will be classified as <u>TIME</u>. When considering only the one-word timexes (where each timex contains and only contains a time token, without any modifier or numeral), almost all the percentages increase to a very high level, especially, many reach 100%. That means these mapping relations are very strong. The mapping relations are summarized in Table III.

Observation 2: There exists a general priority among the four timex types: $\underline{\text{DATE}} < \underline{\text{TIME}} < \underline{\text{DURATION}} < \underline{\text{SET}}$. Modifiers and numerals may change the type of a timex from low priority to high priority.

TABLE II: Percentage of a specific type of time tokens that appear in a specific type of timexes. The result format is "Pr/Pr^1". The percentage greater than 50% is in boldface. "-" indicates no such type of time tokens that appear in the type of timexes.

Time Token	TimeBank (%)				WikiWars (%)				Tweets (%)			
	DATE	TIME	SET	DURATION	DATE	TIME	SET	DURATION	DATE	TIME	SET	DURATION
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 III: Mapping relations from time tokens to 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

TABLE IV: Number of time tokens of a timex type appearing in other three types of timexes. "$\{\cdot\}$" denotes the set of time tokens in Table III that map to the timex type; for example, {TIME} denotes 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	-

Table IV reports the number of the time tokens of a timex type appearing in other three timex types. "$\{\cdot\}$" denotes the set of time tokens listed in Table III that map to the timex type. For example, {TIME} denotes the three time tokens TIME, DAYTIME, and TIMEZONE that map to the timex type TIME. Table IV shows that (1) time tokens {DATE}[2] widely appear in other three timex types, (2) time tokens {TIME} appear in DURATION and SET but do not appear in DATE, (3) time tokens {DURATION} appear in SET but neither appear in DATE nor in TIME, and (4) time tokens {SET} generally do not appear in other three timex types. This suggests a clear priority relationship among the four timex types: DATE < TIME < DURATION < SET.

Such priority indicates that if a timex contains a high-priority time token, then the timex is hardly classified into a low-priority type. For example, a timex that contains a PERIODICAL will neither be classified as DATE nor as TIME nor as DURATION; the timex will be classified as SET. A timex that contains a DAYTIME will not be classified as a DATE; it may be classified as TIME or DURATION or SET.

Generally, a modifier or numeral may increase a timex to a higher-priority type. For example, in the timex "every afternoon", while the time token "afternoon" (a DAYTIME) maps to TIME, the modifier "every" (a PERIOD_MOD) changes the timex to SET.

Observation 3: Standard timex values have only certain formats and these formats are composed of different types of time tokens and numerals.

We observe that each type of tiemxes have only certain standard value formats. The second column of Table V summarizes from training data the formats of timex values under each timex type. It shows that while there may be tons of timexes in a dataset, the value formats of these timexes are relatively fixed. We also find that only 0.56% of timexes in TimeBank, 0.08% in WikiWars, and 0.27% in Tweets have multiple time tokens with the same token type. This indicates that timex values are mainly composed of different types of time tokens and numerals. In addition, we find that most modifiers do not affect timex values.

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

TABLE V: Standard formats of timex values under each timex type and the attribution compositions for each format. “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	Timex Example	ValueExample	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
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
DURATION	PNU, PTNu	24 hours	PT24H	·TIMEUNIT_C, ·NUMBER
	PXU, PTXu	few minutes	PTXM	·TIMEUNIT_C
	PNU, PTNu	a month	P1M	·TIMEUNIT_S, ·NUMBER

IV. MetaTime: Meta Time Information

MetaTime uses a set of token types to group timex tokens and store the values that are assigned to these tokens. On top of the token types, MetaTime designs simple heuristic rules to normalize timexes into standard type and value formats. Figure 1 shows the overview of MetaTime for timex normalization in practice. MetaTime mainly contains two components: MetaTime construction and timex normalization.

A. MetaTime Construction

MetaTime is constructed by importing heuristic rules to implement the functions of the three kinds of *meta time information*: (1) token triples, (2) mapping relations from time tokens to timex types, and (3) priority relationship among timex types. Note that in this stage, MetaTime does not process any text, but only implements the functions of meta time information. In this subsection, we only describe the three kinds of meta time information, and in next subsection, we detail how heuristic rules implement the functions of the three kinds of meta time information when processing text.

Token Triples. We collect token regular expressions (regexes) of timexes and their token types (including *time token*, *modifier*, and *numeral*)[3] from SynTime [40]. Since SynTime focuses only on timex recognition, we further divide these token types to fine-grained ones for timex normalization. There are seventeen time tokens used in MetaTime and these time tokens are summarized in the first column of Table II. Besides token types, we also collect the values[4] from SUTime [6] for these token regexes, according to the annotation scheme of TimeML [25] and TimeBank [26].

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

$$\text{Token triple} := <\textit{token regex}, \textit{token type}, \textit{token value}>$$

All the time tokens and numerals have values. As Observation 3 indicates that most modifiers do not affect timex values, the token triples of most modifiers have no values.

[3]https://github.com/xszhong/syntime/tree/master/syntime/resources/syntimeregex
[4]https://github.com/stanfordnlp/CoreNLP/tree/main/src/edu/stanford/nlp/time/rules

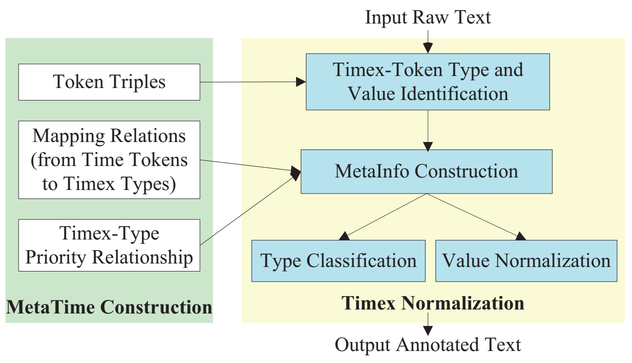

Fig. 1: Overview of MetaTime for timex normalization. The left-hand side shows the MetaTime construction, with three kinds of meta time information: (1) token triples, (2) mapping relations from time tokens to timex types, and (3) priority among timex types. The right-hand side shows the main steps of MetaTime normalizing timexes into standard formats.

Mapping Relations from Time Tokens to Timex Types. Observation 1 indicates strong mapping relations from time tokens to timex types, as summarized in Table III. These mapping relations play a crucial role in MetaTime.

Priority among Timex Types. Observation 2 indicates a general priority relationship among the four timex types: DATE < TIME < DURATION < SET. This priority relationship plays an important role in timex normalization.

B. *Timex Normalization*

In this stage, we detail how MetaTime implements the functions of the meta time information when processing text and normalizing timexes into standard type and value formats. Such process contains three steps: (1) timex-token type and value identification, (2) MetaInfo construction, (3) timex type classification and value normalization. We use the example shown in Figure 2 to detail these steps.

1) Token Type and Value Identification: The identification of token types and values for a timex is straightforward and simple, through looking all the words of the timex at token triples. If a word is matched by a token regex, then MetaTime assigns the word with the corresponding token type and value, which form a pair of <*token type*, *token value*>. As shown in Figure 2, by looking at the token triples, MetaTime assigns the word "September" with the pair <MONTH, 09>.

2) MetaInfo Construction: **MetaInfo.** MetaInfo is a structure created to store the information of token types and values identified in previous stage. MetaInfo defines 18 attributes for time tokens and numerals, 1 for reference date, and 1 for the timex type of a MetaInfo instance. The MetaInfo construction mainly contains two phases: MetaInfo initialization and MetaInfo merge.

MetaInfo Initialization. A MetaInfo instance is initialized for an identified time token. Taking the "September" shown in

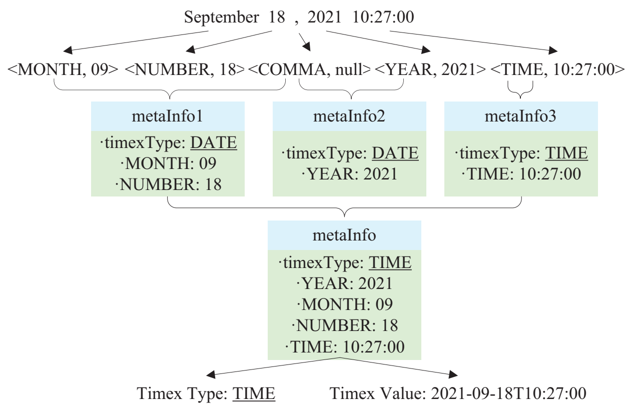

Fig. 2: An example of MetaTime normalizing timex into standard formats. MetaTime firstly assigns each word with a <*token type*, *token value*> pair, then initializes a MetaInfo instance for each identified time token, after that merges all the individual MetaInfo instances into a final MetaInfo instance, and finally determines the timex's type and value.

Figure 2 as an example, MetaTime does the following things in the initialization phase for a time token:

- Stores the pair <*token type*, *token value*> of the identified time token to the attribute in the MetaInfo instance. For example, for the "September", MetaTime stores its assigned pair <MONTH, 09> to the attribute "·MONTH" with the value of "09" in metaInfo1.
- Assigns a timex type to the attribute "·timexType" according to the token type of the identified time token and the mapping relations. For example, MetaTime sets the attribute "·timexType" of metaInfo1 by the timex type DATE, because the token type of "September" is MONTH and MONTH maps to DATE according to the mapping relations summarized in Table III.
- Sets the attribute "·refDate".
- Searches both sides of the identified time token for modifiers and numerals, without exceeding the previous and next time tokens within a timex. If encountering a numeral, then MetaTime updates the attribute "·NUMBER" or "·ORDINAL"; if encountering a QUANT_MOD, then MetaTime sets the "·timexType" by DURATION; if encountering a PERIOD_MOD, then MetaTime sets the "·timexType" by SET.

MetaInfo Merge. All the MetaInfo instances for individual time tokens are merged into a final MetaInfo instance for the timex. Observation 3 indicates that timex values are composed of different types of time tokens and numerals. That means within a timex, different MetaInfo instances generally have different attributes. For example, metaInfo1 has "·MONTH" and "·NUMBER", metaInfo2 has "·YEAR", while metaInfo3 has "·TIME". Therefore, MetaTime simply merges all the attributes into the final MetaInfo instance. The "·timexType" of the final MetaInfo instance is set by the highest-priority timex type among those individual MetaInfo instances. For example, as shown in Figure 2, the attributes "·MONTH" and "·NUMBER" of metaInfo1, "·YEAR" of metaInfo2, and "·TIME" of metaInfo3 are merged into the final instance metaInfo, and the "·timexType" of metaInfo is set by TIME because TIME is the highest-priority timex type among metaInfo1 (DATE), metaInfo2 (DATE), and metaInfo3 (TIME).

After all the individual MetaInfo instances are merged into the final MetaInfo instance, the final instance is used to determine the final type and value of the timex.

3) Timex Type Classification: The final type of a timex is set by the "·timexType" of final MetaInfo instance of the timex. As shown in Figure 2, for example, the final timex type of "September 18, 2021 10:27:00" is set by the "·timexType" of metaInfo: TIME.

4) Timex Value Normalization: Generating the timex value from a final MetaInfo instance mainly contains two steps: determining value format and determining final value.

Determining Value Format. Observation 3 indicates that a specific type of timexes have certain value formats, therefore, for each final MetaInfo instance, MetaTime firstly selects candidate value formats from the second column of Table V according to its "·timexType". From these candidate value formats, MetaTime then determines the final value format according to the final MetaInfo instance's attributes and the attribute compositions listed in the last column of Table V. As shown in Figure 2, for example, the "·timexType" of metaInfo is TIME, MetaTime firstly determines the following candidate value formats: "YYYY-MM-DDThh:mm:ss", "YYYY-MM-DDThh:mm", "YYYY-MM-DDThh", "YYYY-MM-DDTdd"; then according to metaInfo's attributes, namely "*·YEAR:2021*", "*·MONTH:09*", "*·NUMBER:18*", and "*·TIME:10:27:00*", MetaTime determines that the value format of the example timex is "YYYY-MM-DDThh:mm:ss".

Determining Final Value. MetaTime uses the *attribute*:*value* information of the final MetaInfo instance to determine the timex's final value. Observation 3 indicates that each value format is composed of different types of time tokens and numerals, which correspond to different types of attributes. Therefore, MetaTime leverages the values of these attributes to fill in the value format's positions. As shown in Figure 2, for example, the attribute value "2021" fills in the position of "YYYY", "09" fills in "MM", "18" in "DD", and "10:27:00" in "hh:mm:ss"; finally, MetaTime normalizes the example timex into the standard value "2021-09-18T10:27:00".

V. Experiments

We evaluate the quality of MetaTime on three diverse benchmark datasets (i.e., TE-3 [31], WikiWars [22], and Tweets [40]) for the task of timex normalization (including timex type classification and value normalization) against four representative state-of-the-art methods, including two rule-based methods (i.e., HeidelTime [28] and SUTime [6]) and two learning-based methods (i.e., UWTime [17] and ARTime [9]).

A. Experimental Setup

Datasets. The three datasets used to evaluate MetaTime are TE-3 [31], WikiWars [22], and Tweets [40]. The original WikiWars dataset was constructed under the TIMEX2 scheme without timex types. To fully analyze the characteristics of timex types and values, we manually annotate types for WikiWars timexes. Specifically, we teach two annotators the knowledge about timexes according to the standards of TimeML and TimeBank, then the two annotators manually assign timex types to all the WikiWars timexes independently. The initial agreement of their annotations is 98.94%, then they discuss to reach final agreement. The statistics of the three datasets are summarized in Table I.

TE-3 treats TimeBank as the training set and TE3Platinum as the test set [31]. For WikiWars and Tweets, we follow previous research to set their training and test sets. The performance of a model is reported on the test sets.

State-of-the-art Baselines. We compare MetaTime with four state-of-the-art methods: HeidelTime [28], SUTime [6], UWTime [17], and ARTime [9]. HeidelTime and SUTime are rule-based methods while UWTime and ARTime are learning-based methods. Both HeidelTime and SUTime design deterministic rules for the end-to-end TERN task. UWTime uses a combinatory categorial grammar (CCG) and L1-regularization to learn linguistic information from context for the TERN task. ARTime learns from training data to automatically generate normalization rules for timex normalization.

Since HeidelTime, SUTime, and UWTime are designed for the end-to-end TERN task, we report only their timex-normalization performance by applying their codes on the TERN task. ARTime reports the timex-normalization performance from both pure timex-normalization and end-to-end TERN tasks by using PTime [10] as its recognition model. In our experiments, we report ARTime's timex-normalization performance from both tasks. In pure timex-normalization task, timexes are assumed to be correctly recognized.

Evaluation Metric. Like previous researches, we use the toolkit of TempEval-3 [31] to report the standard metric F_1 for both timex type classification and value normalization.

B. Experimental Results

Table VI reports the overall performance of MetaTime and the four state-of-the-art baselines on the three benchmark datasets.[5]

MetaTime achieves five best and six second best results among six measures. Particularly, MetaTime significantly outperforms the four baselines in type classification. Specifically, under pure timex normalization, MetaTime achieves the F_1 in type classification with 92.8% on TE-3, 94.8% on WikiWars, and 98.4% on Tweets; under TERN, MetaTime achieves the F_1 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 1) and

[5]For the end-to-end TERN task, MetaTime uses Syntime [40] as its recognition model.

TABLE VI: Overall performance of timex normalization. "Pure" indicates the performance is reported from the pure timex-normalization task while "TERN" indicates the performance from the TERN task. For each measure, the best result is in bold-face while the second best is underlined. Some results are reported from their original papers.

Method	Task	TE-3		WikiWars		Tweets	
		Type	Value	Type	Value	Type	Value
HeidelTime [29]	TERN	82.1	77.6	89.3	74.7	76.4	71.3
SUTime [7]	TERN	80.3	67.4	85.3	38.8	82.5	67.4
UWTime [17]	TERN	85.4	81.4	90.3	**78.1**	80.0	82.4
ARTime [9]	Pure	84.8	75.4	83.2	57.3	93.7	85.9
	TERN	83.0	74.1	90.3	47.4	92.1	82.4
ARTime+H [9]	Pure	90.6	81.9	85.2	57.8	94.4	92.7
	TERN	86.0	78.7	91.0	47.9	92.9	89.1
MetaTime	Pure	**92.8**	**82.6**	**94.7**	<u>76.2</u>	**98.3**	**93.7**
	TERN	<u>90.7</u>	<u>82.0</u>	<u>91.3</u>	75.3	<u>96.9</u>	<u>93.5</u>
MetaTime w/o Priority	Pure	87.7	79.0	94.4	74.1	92.0	88.2
	TERN	85.6	78.4	91.0	72.3	89.8	87.3

the priority relationship among the four timex types (see Observation 2) in timex type classification.

MetaTime achieves the best results in value normalization on TE-3 and Tweets. That is mainly because MetaTime is designed under the annotation scheme of TimeML [25] and TimeBank [26], and both TE-3 and Tweets are constructed under the same scheme. On WikiWars, MetaTime performs worse than UWTime (76.7% vs. 78.1%), mainly because many timexes in WikiWars are quite descriptive [40] 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 CCG to capture the information of linguistic structure and learns from training data to adapt to new annotation scheme. MetaTime lacks such ability of adapting to new annotation scheme.

MetaTime vs. Rule-based Baselines. We particularly compare MetaTime with rule-based baselines, under pure timex normalization. Table VI shows that MetaTime significantly outperforms HeidelTime and SUTime on all the 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 margin in type classification reaches 8.6∼14.4 points and the one in value normalization reaches 5.2∼22.3 points. The reason is that HeidelTime and SUTime design deterministic rules in a fixed way without a deep understanding of timex types and values. By contrast, MetaTime leverages three kinds of meta time information (i.e., token triples, mapping relations, and timex-type priority) based on three characteristics about timex types and values (see Section III-B) to capture timex information in a flexible way.

Factor Analysis. In MetaTime, token triples and mapping relations are necessary, so we analyze the impact of the priority among timex types. We remove the priority component from MetaTime by setting the ·timexType of the first MetaInfo as the final timex type. The results are reported in Table VI, indicated by "MetaTime w/o Priority". After removing the priority component, MetaTime performs worse in both type classification and value normalization on all the three datasets under both pure timex-normalization and TERN tasks, with decreases of 0.3∼7.1 points in type classification and 2.6∼6.3 points in value normarlization. This indicates the importance of the priority relationship in timex normalization.

Error Analysis. There are mainly three types of errors in MetaTime's evaluation: (1) annotation errors in datasets, for example, TE-3 annotates "the next decade" with the value "P10Y" while annotates "the following decade" with "P1DE"; (2) a lack of time-related keywords, for example, MetaTime does not collect descriptive words like "tenure" and "digital" for token triples; and (3) incorrect reference date, for example, MetaTime normalizes a Tweets timex "last week" to "2014-W22" while the ground-truth is "2014-W21".

VI. Conclusion

We analyze timexes from three diverse benchmark datasets, and summarize three important statistical characteristics about the types and values of timexes. According to these characteristics, we propose a simple rule-based method termed MetaTime with three kinds of meta time information to normalize timexes into standard type and value formats. Experimental results on three diverse benchmark datasets demonstrate that MetaTime outperforms four representative state-of-the-art models in timex normalization. Moreover, MetaTime is independent of specific domains and textual types and runs in real-time and can be easily used as a basic yet high-quality timex tagger for other time-related linguistic tasks.

References

[1] S. Alsayyahi and 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.

[2] G. Angeli, C. D. Manning, and 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.

[3] G. Angeli and 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.

[4] S. Bethard, “Cleartk-timeml: A minimalist approach to tempeval 2013,” in *Proceedings of the 7th International Workshop on Semantic Evaluation*, 2013, pp. 10–14.

[5] Y. Cao, W. Groves, T. K. Saha, J. R. Tetreault, A. Jaimes, H. Peng, and 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.

[6] A. X. Chang and 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.

[7] ——, “Sutime: Evaluation in tempeval-3,” in *Proceedings of second Joint Conference on Lexical and Computational Semantics (SEM)*, 2013, pp. 78–82.

[8] S. Chen, G. Wang, and B. Karlsson, “Exploring word representations on time expression recognition,” Microsoft Research Asia, Tech. Rep., 2019.

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

[10] W. Ding, G. Gao, L. Shi, and Y. Qu, “A pattern-based approach to recognizing time expressions,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 6335–6342.

[11] Q. X. Do, W. Lu, and 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.

[12] M. Filannino, G. Brown, and G. Nenadic, “Mantime: Temporal expression identification and normalization in the tempeval-3 challenge,” in *Proceedings of the 7th International Workshop on Semantic Evaluation*, 2013.

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

[14] Z. Jia, S. Pramanik, R. S. Roy, and 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.

[15] L. Lange, J. Strotgen, H. Adel, and D. Klakow, “Multilingual normalization of temporal expressions with masked language models,” in *arXiv preprint arXiv:2205.10399*, 2022.

[16] E. Laparra, D. Xu, and S. Bethard, “From characters to time intervals: New paradigms for evaluation and neural parsing of time normalizations,” *Transactions of the Association for Computational Linguistics*, vol. 6, pp. 343–356, 2018.

[17] K. Lee, Y. Artzi, J. Dodge, and 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.

[18] A. Leeuwenberg and M.-F. Moens, “A survey on temporal reasoning for temporal information extraction from text,” *Journal of Articial Intelligence Research*, vol. 66, pp. 341–380, 2019.

[19] J. Liu, J. Xu, Y. Chen, and 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.

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

[21] H. Llorens, E. Saquete, and 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.

[22] P. Mazur and 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.

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

[24] Q. Ning, B. Zhou, Z. Feng, H. Peng, and 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.

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

[26] J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, B. Sundheim, D. Radev, D. Day, L. Ferro, and M. Lazo, “The timebank corpus,” *Corpus Linguistics*, vol. 2003, pp. 647–656, 2003.

[27] L. Qin, A. Gupta, S. Upadhyay, L. He, Y. Choi, and 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.

[28] J. Strötgen and 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)*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 321–324.

[29] J. Strotgen, J. Zell, and 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.

[30] N. UzZaman and 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.

[31] N. UzZaman, H. Llorens, L. Derczynski, M. Verhagen, J. Allen, and 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.

[32] M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, and J. Pustejovsky, “Semeval-2007 task 15: Tempeval temporal relation identification,” in *Proceedings of the 4th International Workshop on Semantic Evaluation*, 2007, pp. 75–80.

[33] 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, and J. Pustejovsky, “Automating temporal annotation with tarqi,” in *Proceedings of the ACL Interactive Poster and Demonstration Sessions.*, 2005, pp. 81–84.

[34] M. Verhagen, R. Sauri, T. Caselli, and J. Pustejovsky, “Semeval-2010 task 13: Tempeval-2.” in *Proceedings of the 5th International Workshop on Semantic Evaluation*, 2010, pp. 57–62.

[35] X. Zhong, “Time expression and named entity analysis and recognition,” Ph.D. dissertation, Nanyang Technological University, Singapore, 2020.

[36] X. Zhong and 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.

[37] ——, *Time Expression and Named Entity Recognition*. Springer Nature, 2021, vol. 10.

[38] ——, “Time expression recognition and normalization: A survey,” *Artificial Intelligence Review*, vol. 56, no. 9, pp. 9115–9140, 2023.

[39] X. Zhong, E. Cambria, and A. Hussain, “Extracting time expressions and named entities with constituent-based tagging schemes,” *Cognitive Computation*, vol. 12, no. 4, pp. 844–862, 2020.

[40] X. Zhong, A. Sun, and 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*, vol. 1, Vancouver, Canada, 2017, pp. 420–429.