TimeDict: Time Expression Recognition Using a Simple and Expansible Dictionary

Authors
School of Computer Science and Engineering
Nanyang Technological University, Singapore
emails

ABSTRACT

Extracting time expressions from free text is a critical and fundamental task for many applications. We analyze time expressions from four datasets and observe that only a small group of words are used to directly express temporal information, and the words demonstrate similar syntactic behaviour. Based on our observations, we propose a type-based approach, named TimeDict, to recognize time expressions. Specifically, we built a dictionary for time-related keywords in three main groups: *Time Token*, *Modifier*, and *Numeral*. Example time tokens include day of week (*e.g.,* Monday) and time duration (*e.g.,* 2-hour). A modifier is a word that may appear before or after a time token, such as, 'next' and 'ago'. Numeral are numbers and ordinals. To recognize time expressions, TimeDict first identifies the time tokens from raw text, then searches for modifiers and numerals surrounding the time tokens to form time segments. Finally, the time segments are merged into time expressions. Thanks to the comprehensive handcrafted rules available in exiting tools, TimeDict can be easily initialized by importing existing rules for recognizing time tokens, modifiers, and numerals. More importantly, as a dictionary, TimeDict can be easily expanded to include more keywords for its effectiveness on different types of text. For example, in tweets, 'tomorrow' may be spelt as 'tmr'. Experimental results show that TimeDict outperforms state-of-the-art methods on benchmark datasets and tweets data.

1. INTRODUCTION

Time expression or temporal expression plays an important role in information retrieval and many applications in natural language processing [1]. Recognizing time expressions in free text has attracted significant attention from researchers since last decade [26, 27, 29].

There are two main approaches for time expression recognition, rule-based approach [4, 21, 22] and machine learning based approach [3, 11]. Rule-based time expression taggers recognize time expressions based on carefully designed rules, and are able to recognize most time expressions written in regular forms. However, it is difficult for rule-based taggers to recognize time expressions written in informal manner, *e.g.,* time expressions in tweets. Machine learning based approach, on the other hand, requires training data for good performance, and may suffer from recognizing less frequent time expressions.

In our study, we analyze the time expressions in four datasets: TimeBank [18], Gigaword [16], WikiWars [15], and Tweets. While the first three datasets have been used in early studies, the tweets data is created through manual annotation in our research. From the analysis, we make four observations. First, most time expressions are very short, with 80% of time expressions containing no more than three tokens and more than 90% containing no more than four tokens, on all datasets. Second, the vocabulary used to express time information is very small, with a small group of keywords. Example keywords include the words for seasons, months, and days of the week. On all datasets, at least 93% of time expressions each contain at least one time token. Third, surrounding the time tokens are the modifiers like 'next' and 'ago' and numerals. The last observation is that the Part-of-Speech (POS) tags of tokens in time expression demonstrate similar syntactic behaviors.

Based on the observations, we propose a type-based approach, named TimeDict, to recognize time expressions. Specifically, we construct a dictionary for time-related keywords[1] in three main groups: *Time Token*, *Modifier*, and *Numeral*. Time tokens are the small group of keywords that directly express time information, including month of the year (*e.g.,* 'Feburary'), time unit (*e.g.,* year), time duration (*e.g.,* 2-hour), and so forth. Modifiers are the words that modify time tokens; they may appear before or after time tokens. For example, modifier 'next' appears before time token 'year' in time expression 'next year' and 'ago' appears after 'years' in time expression 'years ago'. Numeral are numbers and ordinals. Two example time expressions containing numerals are 'the third week' and 'January 20'. Because more than 93% of time expressions contain time tokens, we first identify time tokens from free text, and then recognize the modifiers and numerals. If a number belongs to a time token, *e.g.,* '2008', it will not be identified as a numeral.

Thanks to the comprehensive rules available in existing tools, we can initialize TimeDict by importing the patterns for recognizing time tokens, modifiers, and numeral. In our implementation, TimeDict is derived from the rules in SUTime, a state-of-the-art rule-based tagger that achieved the highest recall in TempEval-3 [4, 5]. Because TimeDict is a dictionary, it can be expanded by simply adding new keywords under the different types.

Naturally, TimeDict is a rule-based tagger. The key difference between TimeDict and other rule-based taggers lies in the way defining ***types***. The type definition in TimeDict is inspired by Part-of-Speech in which "linguists group some words of language into classes (sets) which show similar syntactic behaviour, and often a typical semantic

Submission to WWW '17 April 3–7, 2017, Perth, WA, Australia

ISBN ISBN. . . $15.00

DOI: DOI

[1] In this paper, we use 'keyword' and 'token' interchangeably for words acceptable to predefined patterns in regular expression, *e.g.,* patterns for year instances and date instances.

type" [14]. TimeDict defines types for tokens according to their syntactic behaviors. Existing rule-based taggers define types for tokens based on their semantic meaning. For example, SUTime defines 5 modifier types: operators, relative modifiers, frequency modifiers, approximate modifiers, and early and late modifiers.[2] TimeDict also defines 5 modifier types: modifiers that appear before time tokens; modifiers that appear after time tokens; and modifiers that link two time expressions; and two types of special modifiers (comma and indefinite article). Table 5 lists the details of the types in TimeDict. Accordingly, SUTime designs rules for each type based on their meanings, and deals with each type separately. TimeDict defines rules based on the token type and their relative positions in time expression. That is why we call TimeDict is a type-based approach. The number of rules in TimeDict is much smaller than that in SUTime. Becuase of the small number of rules, TimeDict is much more flexible and easily expansible.

We evaluate TimeDict against three state-of-the-art baselines, namely, HeidelTime, SUTime, and UWTime, on three datasets (TimeBank, WikiWars, and Tweets). TimeBank and WikiWars are benchmark datasets for time expression extraction.[3] Tweets dataset is manually annotated in this study. Experimental results show that our TimeDict significantly outperforms the three state-of-the-art methods on TimeBank and Tweets datasets. On WikiWars dataset, TimeDict achieves comparable results with baseline methods. More importantly, TimeDict achieves the best recalls on all three datasets in both of strict match measures and relaxed match measures. To sum up, we make the following contributions in this study.

- We analyze time expressions from four datasets, make four observations. Our TimeDict is designed based on these observations.
- We propose a type-based time expression tagger, TimeDict, that includes time-related keywords and simple rules for effective time expression recognition from free text.
- We conduct experiments on three datasets to demonstrate the effectiveness of TimeDict against state-of-the-art baselines.

2. RELATED WORK

Many studies on time expression extraction are reported through TempEval exercises which aim at automatically identifying time expressions, events, and time relations in text [26, 27, 29]. The task of time expression identification is divided into two subtasks: extraction and normalization. In this study, we focus on time expression extraction only.

Rule-based Time Expression Extraction. Rule-based temporal taggers like GUTime, HeidelTime, and SUTime, predefine time-related words and regular expression patterns [4, 21, 28]. HeidelTime [21] hand-crafts extraction rules with time resources like weekdays, seasons, or months, and leverages language clues like part-of-speech to identify the time expression; the extracted time expressions are then normalized to standard form in a pipeline framework UIMA (Unstructured Information Management Architecture).[4] SUTime [4] is based on a cascade strategy of finite automata [10] on regular expression patterns over tokens [6]. It first matches from simple regular expressions over tokens, and then expands regular expressions over chunks. After that, it filters those ambiguous expressions. Rule-based temporal taggers achieve very good results in TempEval exercises. In particular, HeidelTime reaches the highest F_1 of 86% in TempEval-2 [29] and SUTime achieves the best F_1 of 91.3% on relaxed matching evaluation in TempEval-3 [26].

As aforementioned, TimeDict is also rule-based tagger and the key differences between TimeDict and existing rule-based taggers are the definition of type. More specifically, TimeDict defines types for tokens according to their syntactic behaviors, not on their semantic meaning. Simple rules are then defined on the types.

Machine Learning based Approach. Machine learning based methods have also been reported in TempEval exercises. These methods extract features from the text and apply statistical models on these features for time expression extraction. Example features include character features (*e.g.,* first 3 characters, last 3 characters), word features (*e.g.,* current token, previous token, lemma), syntactic features (*e.g.,* part-of-speech, chunks), gazetteer features (*e.g.,* match in a dictionary), and semantic features (*e.g.,* semantic role, lexical semantics) [3, 9, 13]. The statistical models include Markov Logic Network, Logistic Regression, Support Vector Machines, Maximum Entropy, and Conditional Random Fields [3, 9, 13, 25]. Some models obtained good performance, and even achieved the highest F_1 of 82.71% on strict matching measure in TempEval-3 [3].

Outside the TempEval exercises, Angeli *et al.* leveraged compositional grammar and employed a EM-style approach to learn a latent parser for interpreting time expressions as concrete entity [2]. In the method named UWTime, Lee *et al.* handcrafted a Combinatory Categorial Grammar (CCG) [20] to define a set of lexicon with rules and used L1-regularization to learn linguistic context, so as to identify time expressions [11]. In these two methods, linguistic information is explicitly used in the learning. Particulary in [11], CCG could capture rich structure information of language, similar to rule-based method. Tabassum *et al.* focused on resolving only the dates in tweets, and used distant supervision to extract time expressions [24]. They used five temporal types and assigned one of them to each word, which is similar to our way of defining time-related types over tokens. However, they focus on only the date type, while our TimeDict can extract all time expressions. Moreover, TimeDict does not involve learning.

Time Expression Normalization Methods in TempEval exercises mostly define rule for time expression normalization [3, 9, 13, 21, 25, 28]. Due to the high similarity of the rule systems, Llorens *et al.* suggested to construct a large knowledge base as a public resource for this task [12]. On the other hand, some researchers treat the normalization process as a learning task and use machine learning methods [11, 24]. Lee *et al.* [11] and Tabassum *et al.* [24] used AdaGrad algorithm [7] and a log-linear algorithm, respectively. In this research, we only focus on time expression extraction, and normalization could be achieved using existing methods.

3. TIME EXPRESSION ANALYSIS

3.1 Dataset

Our data analysis is conducted on four datasets: TimeBank, Gigaword, WikiWars, and Tweets. TimeBank corpus [18], consisting of 183 news articles, is a benchmark dataset used in TempEval series [26, 27, 29]. Gigaword corpus [16] is a large scale Silver evaluation dataset in TempEval-3. It contains 2,452 news articles and is *automatically* labeled by time expression taggers TIPSem, TIPSem-B [13], and TRIOS [25]. WikiWars corpus is derived from Wikipedia articles about wars [15]. Tweets is our manually annotated dataset. It contains 942 tweets each contain at least one time expression. Table 1 summarizes the datasets and more details about

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

[3] Gigaword dataset was not used in our experiments because the labels in the dataset was automatically generated by other taggers which may not be the ground truth labels.

[4] http://uima.apache.org

Table 1: Statistics of the datasets (a tweet here is a document)

Dataset	#Documents	#Words	#TIMEX
TimeBank	183	61,418	1,243
Gigaword	2,452	666,309	12,739
WikiWars	22	119,468	2,671
Tweets	942	18,199	1,127

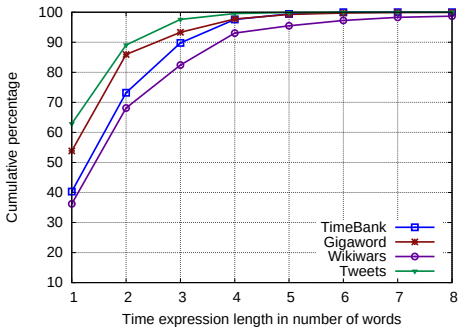

Figure 1: Length distribution of time expressions on all datasets

the datasets are in Section 5.

3.2 Observation

In this section, we report 4 observations made from the time expressions in the 4 datasets.

OBSERVATION 1. *Time expressions are very short. More than 80% of time expressions contain no more than three words and 90% of them contain no more than 4 words.*

Figure 1 plots the length distribution of the time expressions in four datasets. Although the texts are collected from different sources (*e.g.,* news articles, Wikipedia articles, tweets) and vary in sizes, the length of time expressions follow a similar distribution. This observation relates to the principle of least effort [30]. That is, people will act under the least effort so as to minimise the cost of energy at both individual level and collective level to the usage of language [30]. Time expression in free text is a part of language. In particular, the unit-length (*i.e.,* one word) time expressions range from 36.23% in WikiWars dataset to 62.91% in Tweets dataset. In informal communication, people use minimum number of words to express time information. The third column in Table 2 reports the average length of the time expressions. On average, time expressions are about two words.

OBSERVATION 2. *Only a small group of time-related keywords are used to express time information.*

From the time expressions in all four datasets, we observe that only a small group of time-related keywords are used to express time information. These keywords include the month of the year, day of the week, and time units (*e.g.,* 'month', 'day', 'year'). These keywords have been identified in most existing rule-based time expression taggers. In TimeDict, we name these time-related keywords *time tokens* (see Table 5).

Table 3 reports the number of distinct words and the number of distinct time tokens from the datasets. Note that, the words/tokens are manually normalized before counting. For example, 'month', 'months', and '5mons' are counted as one token 'month'. Numerals are ignored in the counting. Despite the different sizes of the four datasets, the number of distinct time tokens are comparable across datasets.

Table 2: The percentage of time expressions that contain at least one time token, and the average length of time expressions.

Dataset	Percentage	Average Length
TimeBank	95.15	2.00
Gigaword	97.69	1.70
WikiWars	93.18	2.38
Tweets	97.81	1.51

Table 3: Number of distinct words and the number of distinct time tokens in time expressions.

Dataset	#Distinct Words	#Distinct Time Tokens
TimeBank	130	64
Gigaword	214	80
WikiWars	224	74
Tweets	107	64

OBSERVATION 3. *More than 93% of time expressions contain at least one time token. Within time expressions, time tokens and modifiers tend to occur together.*

The second column in Table 2 reports the percentage of time expressions that contain at least one time token. Observe that at least 93.18% of time expressions contain time token(s) on across all datasets. The observation suggests that it is essential to identify time tokens for time expression recognition. We also observe that if a time expression contains both time token(s) and modifier(s), then the time token(s) and the modifier(s) tend to appear together.

OBSERVATION 4. *Part-of-speech (POS) could not distinguish time expressions from common words, but within time expressions, POS can help distinguish their constituents.*

Table 4 lists the top 10 POS tags that appear in time expressions, and their percentage in the whole text, on each dataset. Note that, the Tweets dataset includes 942 manually annotated tweets each contain at least one time expression. If including all tweets that do not contain time expressions, the percentage will be much lower. Except CD in WikiWars and Tweets datasets, most tags have smaller percentage than 50% on the four datasets. More specifically, 29 out of the 40 tags have percentage lower than 10%. This indicates that POS could not provide much information to distinguish time expressions from common words. However, we observe that the most common POS tags for tokens in time expressions are NN*, CD, DT, JJ, and RB. Within time expressions, time tokens usually have tags of NN* and RB, numerals have CD, and most modifiers have tags JJ and RB with a few NN*. Here, NN* refers to nouns in singular (NN) or plural (NNS) form, and proper nouns in singular (NNP) or plural (NNPS) form. This observation indicates that the constituents of time expressions have certain syntactic behaviors.

Inspired by the definition of Part-of-Speech in which "linguists group some words of language into classes (sets) which show similar syntactic behaviour, and often a typical semantic type" [14], we group words used to express time related information in TimeDict.

4. TIMEDICT: A TIME DICTIONARY

Figure 2 shows an overview of the proposed TimeDict. Shown in the left-hand side of the figure, TimeDict is initialized with handcrafted rules available in existing tools. In our implementation, we used rules from SUTime. After initialization, TimeDict can be directly applied on text for time expression recognition. In other words, TimeDict can be constructed without the need of training

Table 4: Top 10 more frequent POS tags assigned to the words in time expressions. Frequency is the number of times a POS tag is assigned to a word in time expression; percentage is based on its assignments to words in time expressions and all words in documents.

TimeBank			Gigaword			WikiWars			Tweets		
Tag	**Freq.**	**Percentage**	**Tag**	**Freq.**	**Percentage**	**Tag**	**Freq.**	**Percentage**	**Tag**	**Freq.**	**Percentage**
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

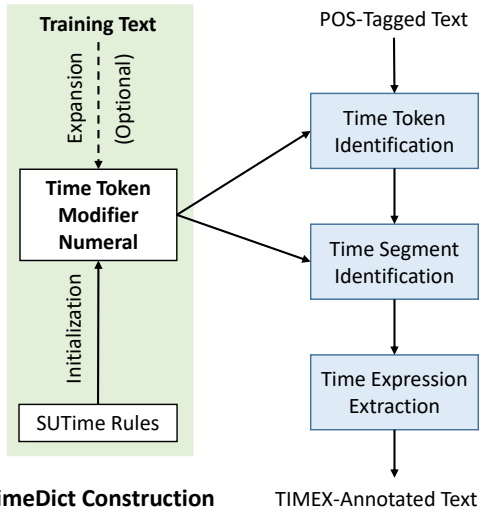

Figure 2: Overview of TimeDict. The construction of TimeDict is shown on the left-hand side, with initialization using rules from SUTime, and optional expansion using training text. The right-hand side shows the main steps in recognizing time expressions using TimeDict.

data, similar to rule-based tools. On the other hand, TimeDict can be easily expanded by incorporating new time-related keywords (or tokens) from training text. The expansion enables TimeDict to recognize time expressions from different kinds of texts where time may be expressed in different forms with different spelling (*e.g.,* 'tmr' for tomorrow) or different terms (*e.g.,* tenure).

Using TimeDict for time expression recognition consists of three main steps, shown in the right-hand side of Figure 2. TimeDict requires the text to be POS-tagged. Because POS taggers are widely available and the state-of-the-art taggers achieve very high accuracy (above 97%)[5], we do not consider this as an limitation of TimeDict. The POS tags are helpful in identifying time tokens, in the first step. In the second step, TimeDict searches for modifiers surrounding the identified time tokens to form time segments. The time segments are merged to time expressions in the last step.

4.1 TimeDict Construction

In TimeDict, we define 15 types of time tokens, 5 types of modifiers, and 1 numeral type (see Table 5).

[5] https://www.aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)

Time Token Time tokens are the words that directly specify time expressions, *e.g.,* year, season, month, date, and so on. Existing rule-based temporal taggers, *e.g.,* HeidelTime [21] and SUTime [4], have manually collected a large number of time-related words and patterns and grouped them into different types. These time-related words and patterns comprise a good dictionary. We define 15 time token types and use their names similar to Joda-Time classes.[6] In our implementation, the patterns for recognizing these time tokens are based on the rules defined in SUTime.

Modifier In SUTime, several types of modifiers are defined according to their semantic meaning. In TimeDict, we define 3 types of modifiers based on their possible positions relative to the modified time tokens. Modifiers such as 'next' and 'few' that appear before time tokens are prefix modifiers, *i.e.,* PREFIX; modifiers that appear after time tokens are under SUFFIX type. The third type of modifiers that link two time tokens are classified under LINKAGE type. We also define two special modifiers, COMMA (',') and IN_ARTICLE for indefinite articles 'a' and 'an'. COMMA usually appears as a part of a date instance, for example 'October 10, 2016'. IN_ARTICLE usually modifies a time unit, for example 'a year' or 'an hour'.

Both TimeML [17] and TimeBank corpus [18] do not treat the prepositions like 'in', 'at', and 'on' as a part of time expressions. Thus TimeDict does not collect these prepositions.

Numeral Numbers in text can be a part of a time expression *e.g.,* number 10 in 'October 10, 2016', a modifier *e.g.,* '10 days', or appear in text that is not related to time. In TimeDict, we define a numeral type for ordinals and numbers. Note that, in TimeDict, one token belongs exactly to one type. Therefore, if a number (*e.g.,* '2016') is identified as a year instance, then it will not be tagged as numeral type.

TimeDict Initialization Thanks to the comprehensive handcrafted rules for recognizing time tokens and modifiers, TimeDict can be easily initialized by using existing rules. In our implementation, we collected time tokens and modifiers from the rules defined in SUTime [7]. Note that, the time tokens are defined in regular expressions, *e.g.,* a year instance is defined by [1-2][0-9]{3}|'[0-9]{2} (in Java notation). The TimeDict initialized based on SUTime rules is denoted by TimeDict-S. The number of tokens in each TimeDict type is listed in the last column in Table 5. In the next section, we expand TimeDict-S to include more time tokens from training text.

4.2 TimeDict Expansion

Rule-based systems more or less suffer from the problem of cov-

[6] http://www.joda.org/joda-time/

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

Table 5: TimeDict consists of 15 types of time tokens, 5 types of modifiers, and 1 numeral type. The last column indicates the number of tokens, without counting variants in each type; "-" indicates the type involve change numbers in digits and cannot be counted.

Type	Description	Examples	#Tokens
Time Token			
DECADE	decade instances	1910s, fifties	-
YEAR	year instances	1970, 2006	-
SEASON	season instances	Winter	5
MONTH	month instances	February	12
WEEK	day of week	Monday	7
DATE	date instances	2016-09-07	-
TIME	time instances	03:45:32	-
DAY_TIME	time within a day	morning	27
TIMELINE	relative to today	yesterday	12
HOLIDAY	holiday instances	Christmas	20
PERIOD	period instances	daily	9
DURATION	duration instances	5-year	-
TIME_UNIT	time units	year(s)	15
TIME_ZONE	time zones	GMT, UTC	6
ERA	era AD and BC	AD, BC	2
Modifier			
PREFIX	before time token	the, about	48
SUFFIX	after time token	ago, old	2
LINKAGE	link time tokens	and, or, to, -	4
IN_ARTICLE	indefinite articles	a, an	2
COMMA	comma	,	1
Numeral			
NUMERAL	numbers, ordinals	20, third	-

erage. For example, SUTime cannot recognized the word 'tenure' in time expression, while the word is treated as a time expression in TimeBank corpus. It is equally true for TimeDict. Therefore, the ability of expanding dictionary for time-related keywords become a critical requirement for TimeDict for its effectiveness on different types of text data.

Given a time expression e, which consists of a sequence of m words $e = \langle w^1, \cdots, w^m \rangle$, we consider e is covered by TimeDict $\mathcal{D}$ if every word $w \in e$ is defined in $\mathcal{D}$, or $w \in \mathcal{D}$. Accordingly, e is not covered by $\mathcal{D}$ if there exist at least one word $w \in e$ but $w \notin \mathcal{D}$.

To expand TimeDict, the time expressions from a collection of training text need to be manually annotated. It is straightforward to list the time expressions which are not covered by TimeDict. Then the unique words from these expression that are not covered by TimeDict can be easily identified (Lines 2 - 4 in Algorithm 1). Given an uncovered word $w \notin \mathcal{D}$, whether to add w to $\mathcal{D}$ (accordingly its type in $\mathcal{D}$) is manually determined with reference to the list of time expressions containing it in the training data (Lines 5-6).

Again, based on our observations, not many words are used to express time related information. Therefore, the words to be added in TimeDict shall be generalizable to many time expressions and contribute to TimeDict's coverage. For example, '2day', 'todayy', and 'tday' are all popular spellings of 'today' in tweets, similarly '2morrow' and 'tmr' for 'tomorrow'. Adding these words to TimeDict is expected to increase its effectiveness in recognizing time expressions in tweets.[8] On the other hand, some time expressions are extremely descriptive with very specific words that are not generalizable to other time expressions. Such words will not be added to TimeDict. The Wikiwar dataset (see Section 5 for more details) contains a few such examples: *e.g.,* 'ten brutal months of the Battle of Verdun' and 'the time Arnold reached Quebec City in early November'. In the latter example, 'early November' can be easily recognized by most time tagger but not 'the time Arnold reached Quebec City', particularly for rule-based or keyword based taggers. Note that prepositions like 'in' and 'at' are not added to TimeDict because they are not considered as a part of time expression in TimeML and TimeBank corpus.

[8]The popular alternative spellings in tweets may also be obtained through Brown clustering. Word clusters obtained from 56 million English tweets are available at http://www.cs.cmu.edu/~ark/TweetNLP/cluster_viewer.html.

Algorithm 1 TimeDict expansion

Input: Time expressions: $E = \{e\}$, where $e = \langle w^1, \cdots, w^m \rangle$
Output: Expanded time dictionary: TimeDict-E $\mathcal{D}$

1: initialize: $\mathcal{D} \leftarrow$ TimeDict-S
2: **for** each $e \in E$ **do**
3: **if** $w \in e$ and $w \notin \mathcal{D}$ **then**
4: identify w and add e to the list of expressions containing w
5: **for** each identified w and its list of containing expressions **do**
6: decide whether to add w to $\mathcal{D}$ and assign its type accordingly
7: **return** $\mathcal{D}$

4.3 Time Expression Recognition

We now detail the three main steps in recognizing time expressions using TimeDict: (i) time token identification, (ii) time segment identification, and (iii) time expression extraction.

4.3.1 Time Token Identification

Given a piece of text, the task of time token identification is to identify the time-related keywords which are instances of the 15 time token types listed in Table 5.

After constructing TimeDict, the instances in the 15 time token types either are in a closed vocabulary (*e.g.,* words defined in MONTH, WEEK, and HOLIDAY), or can be matched by patterns (*e.g.,* instances in YEAR, DATE, and TIME). At first glance, time tokens identification from free text seems simple and straightforward, through string/pattern matching. However, ambiguity makes the task slightly more difficult. For example 'May' could be the fifth month in a year, a person name, or used as a modal verb, among many different meanings.[9] Another example is the word 'second' which could be a unit of time, or an ordinal. To filter out those words that match instances in TimeDict but are not time-related keywords, we use part-of-speech (POS) tags of the words.

POS tagging has been a classic research problem in NLP and the state-of-the-art POS taggers achieve extremely high accuracy, above 97%. In our implementation, we use Stanford POS Tagger, hence following their notations.[10] Table 6 lists the POS tags used for filtering words matching the instances in the corresponding TimeDict types. In this table, RB denotes adverb, NN* refers to nouns in singular (NN) or plural (NNS) form, and proper nouns in singular (NNP) or plural (NNPS) form. JJ denotes adjective and CD is cardinal number. Only the words that match the instances in TimeDict and are tagged with the corresponding POS tags in the table, will be identified. That is, even though a word occurrence 'May' matches a keyword in TimeDict, but if its POS tag is MD (modal verb), it will not be identified as a time token.

At the end of time token identification, we have all the occurrences of time keywords in the 15 time token types in TimeDict listed in Table 5. For each identified occurrence, we record its position in the text (or index) and its time token type. More formally, a recognized

[9]https://en.wikipedia.org/wiki/May_(disambiguation)
[10]http://nlp.stanford.edu/software/tagger.shtml

Table 6: Matching TimeDict type with POS tag

TimeDict Type	POS Tag
DAY_TIME, TIMELINE, PERIOD	RB or NN*
DURATION	CD or JJ
PREFIX	RB or JJ or NN*
MONTH, WEEK, SEASON, TIME_UNIT, TIME_ZONE, ERA	NN*
DECADE, YEAR, DATE, TIME, HOLIDAY, NUMERAL, SUFFIX, LINKAGE, IN_ARTICLE, COMMA	-

Table 7: Notations

Notation	Semantic
w^k	A word w at the k-th index position
$t = \langle w, p, i \rangle$	A time token t, with a time token word w, TimeDict type p, at index position i
$s = \langle t, l, r \rangle$	A time segment s, containing time token t, spanning from left index position l to right index r
$e = \langle w^l, w^r \rangle$	A time expression, containing a sequence of words from left index l to right index r

time token t is a 3-tuple $t = \langle w, p, i \rangle$, where w is the word, p is its time token type, and i is its position index in the text. Next, we identify time segments by searching for modifiers and numerals surrounding the time tokens.

4.3.2 Time Segment Identification

A time segment is a sequence of words which consists of exactly one time token, and zero or more modifiers/numerals. A time segment could be a time expression on its own. For example 'two weeks ago' is a time segment which consists of a time token 'weeks' (TIME_UNIT), a numeral 'two', and a modifier 'ago' (SUFFIX). This time segment on its own is a time expression. A time expression may involve multiple successive time segments, which will be detailed in next Section.

In a time segment, modifiers and numerals may appear before or after the time token, or on both sides. The modifiers that appear before a time token include the words in PREFIX and IN_ARTICLE types, and the modifiers that appear after a time token are the words in SUFFIX types. The LINKAGE words, NUMERAL, and COMMA may appear on both sides of the time token in a time segment.

We have identified the time tokens in the previous step. Now we search for modifiers/nuermals surrounding each time token to recognize time segments. Let T be the list of time tokens t's identified from the given text, sorted by their index positions in increasing order. Algorithm 2 summarizes the main steps in identifying time segments for the time tokens in T.

Recall that a time segment is a word sequence containing time token and surrounding words. We denote time segment by $s = \langle t, l, r \rangle$, where t is the time token, l and r are the left and right position indexes (or boundaries) of the word sequence.[11] The key task in time segment identification is to determine the left and right boundaries, starting with the index of each time token (*i.e.,* $t.i$). If t's time token type ($t.p$) is either DURATION or PERIOD, then there is no need to further search for modifiers (Lines 3-4). The next two for-loops are for searching modifiers to the left (Lines 6-13) and to the right (Lines 14-21) of the time token. The search stops when reaching a linkage word or a comma, or when a non-modifier word

[11] For simplicity, we do not show the types (*i.e.,* modifier types in TimeDict) of the words in time segment.

Algorithm 2 Time segment identification

Input: The list of time tokens T, sorted in increasing order by position
Output: List of time segments S.

```
1: for each t in T do /*scan from the first time token to the last*/
2:     initialize s, s.l ← t.i and s.r ← t.i
3:     if t.p ∈ {DURATION, PERIOD } then
4:         add s to S
5:     else
6:         for k from t.i − 1 to t′.i do /*scan to the left till a time token*/
7:             if w^k ∈ {PREFIX, NUMERAL, IN_ARTICLE } then
8:                 s.l ← k
9:             else if w^k ∈ {LINKAGE, COMMA } then
10:                s.l ← k
11:                break /*stop scanning at a linkage or comma*/
12:            else
13:                break /*stop scanning at non-modifier type*/
14:        for k from t.i+1 to t′′.i do /*scan to the right till a time token*/
15:            if w^k ∈ {SUFFIX, NUMERAL } then
16:                s.r ← k
17:            else if w^j ∈ {LINKAGE, COMMA } then
18:                s.r ← k
19:                break
20:            else
21:                break
22:        add s to S
23: return S
```

is met. Note that, to determine whether a word is a PREFIX, its POS tag is also considered (see Table 6). For a LINKAGE word, if it links two time tokens or links a time token and a numeral, it is treated as a modifier; otherwise a non-modifier. The two detailed rules are not shown in Algorithm 2 for simplicity.

A special kind of time segment, named *dependent time segment*, does not contain time token. However a dependent time segment, as its name suggest, exists only next to a time segment. For example, in a word sequence '8 to 20 days', 'to 20 days' is a time segment with time token 'days' (TIME_UNIT), 'to' is LINKAGE and '8' is NUMERAL. In this case, '8 to' forms a dependent time segment, illustrated in Figure 3(e). Note that, the identification of dependent time segment is not shown in Algorithm 2 for simplicity. The process occurs between Lines 9-11, and between Lines 17-19.

4.3.3 Time Expression Extraction

The task of time expression extraction is to determine the boundaries of a time expression based on the time segments extracted in the earlier step. A time segment on its own can be a time expression. For example, 'the past few weeks' is a time segment and also a time expression, where the first three words 'the', 'past', 'few', are all PREFIX modifiers and the word 'weeks' has time token type TIME_UNIT. A stand-alone time segment in a sentence, *i.e.,* not appearing right before or after another time segment, is extracted as a time expression.

Recall that, in time segment identification, we scan words to the left and to the right of a time token, and stop at a non-modifier word or a LINKAGE and COMMA. Thus the boundary of a time segment could be a time token, a NUMERAL, or a modifier (including LINKAGE and COMMA). The main focus of time expression extraction is to deal with the cases where there are two or more adjacent or overlapping time segments in a sentence. Two time segments s_1 and s_2 are adjacent if s_1 appears immediately before s_2, or $s_1.r+1 = s_2.l$. Two time segments are overlapping if they share a same boundary $s_1.r = s_2.l$.

The time expression extraction process scans the time segments from the beginning of each sentence to the right. A stand-alone time segment (*i.e.,* not adjacent to or overlap with the next segment) is

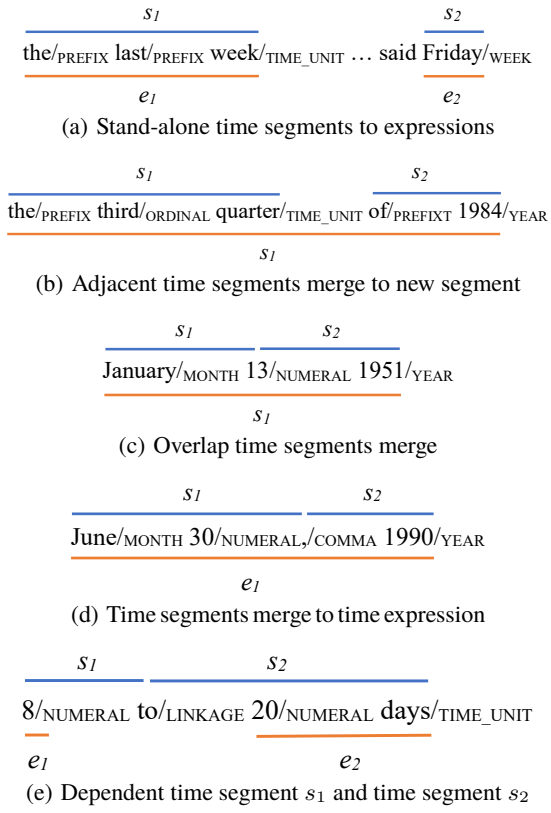

Figure 3: Example time segments and time expressions. The labels above the blue lines are for time segments obtained during time segment identification, and the labels below the orange lines are obtained during time expression extraction.

extracted as a time expression (see Figure 3(a)). If a time segment s_1 is adjacent to s_2, then s_1 and s_2 are merged to form a new time segment s_1 (see Figure 3(b)). The process continue scanning to the right. Consider s_1 and s_2 are overlapping with a shared boundary. If the word at the shared boundary is neither a COMMA nor a LINKAGE, then s_2 is merged to s_1 (see Figure 3(c)). If the word is a LINKAGE word, then s_1 is extracted as a time expression and the process continues. When the shared boundary is a COMMA, s_2 merges to s_1 only if the COMMA's previous token (*i.e.,* $w^{s_1.r-1}$) and its consecutive token (*i.e.,* $w^{s_2.l+1}$) satisfy the following three conditions: (i) $w^{s_1.r-1}$ is a time token or a NUMERAL, (ii) $w^{s_2.l+1}$ is a time token, and (iii) the types of $w^{s_1.r-1}$ and of $w^{s_2.l+1}$ are not the same (see Figure 3(d)).

As a post processing, the prepositions, linkage words, and comma, which appear at the boundary of an extracted time expression will be trimmed.

5. EXPERIMENTS

We conduct experiments on three datasets and compare TimeDict with three state-of-the-art baselines, HeidelTime, SUTime, and UWTime. For TimeDict, we report the results of two methods **TimeDict-S** and **TimeDict-E**. The former is initialized by using patterns from SUTime directly, and the latter is the expanded version by adding more keywords to the dictionary.

5.1 Experiment Setting

We use two benchmark datasets (TimeBank and WikiWars) and one manually labeled Tweets dataset in our evaluation. The Gigaword dataset [16] was labeled automatically by time taggers and has no ground truth labels, thus we do not use it in our experiments.

Datasets. TimeBank corpus contains 183 new articles, and has been used as the benchmark dataset in TempEval series for time expression identification [26,27,29]. WikiWars dataset is collection of 22 English Wikipedia articles about famous wars and annotated with TIMEX2 standard [8]. We map the annotations in TIMEX3 standard [19] in order to use the TempEval evaluation toolkit.[12]

For the Tweets dataset, we randomly sample 100 users from Twitter, and use Twitter API to collect at most 2000 tweets from each sampled user. From the collected tweets, we randomly sample 4000 tweets and use SUTime to tag them. SUTime annotates 942 tweets that contain time expressions. From the remaining 3,058 tweets, we randomly sample 500 and manually label them. Only 15 tweets contain time expressions. Thus we consider that SUTime misses about 3% time expressions in tweets. We then manually annotate the 942 tweets, according to the TimeML and TimeBank standards. Finally 1,127 time expressions are manually labeled. For the 942 tweets, we randomly sample 200 tweets as test set, and the rest 742 as training set, because a baseline method UWTime requires training.

Baseline Methods. We compare TimeDict method with three state-of-the-art methods: HeidelTime [21], SUTime [4], and UWTime [11]. HeidelTime and SUTime are both rule-based methods; they achieve the best results in TempEval-3 in terms of relaxed match [26]. UWTime uses a handcrafted Combinatory Categorial Grammar to represent time expressions and a learning method. It achieves better results than HeidelTime in TimeBank dataset and WikiWars dataset [11]. When training UWTime on Tweets dataset, we tried two setting: 1) train with only the Tweets training set; 2) train with Tweets training set and TimeBank. The second setting achieves slightly better result and we report this result.

Evaluation Metrics. We follow TempEval-3 and use their evaluation toolkit to report Precision, Recall, and F_1 in terms of *strict match* and *relaxed match*. Strict match means exact match with ground truth annotations, and relaxed match means there is overlap between the extracted time expression and the manual annotation [26].

5.2 Experiment Result

Table 8 reports the results of all methods on the three datasets, with two sets of measures: Precision, Recall, F_1 on strict match and relaxed match, respectively. Among the 18 measures, TimeDict-S and TimeDict-E achieve 12 best results and and 13 second best results. Except the strict match on WikiWars datasets, both TimeDict-S and TimeDict-E achieve F_1 scores above 91%. For the relaxed match, TimeDict achieves recalls above 92% on all the three datasets. The high recalls are consistent with our observation that at least 93.18% of time expressions contain time token(s) (see Table 2). The high recalls of TimeDict-S and TimeDict-E indicate that our method covers most of the time tokens. For precision, TimeDict-S and TimeDict-E achieves the best result on TimeBank and comparable results on the other two datasets under strict match. Under relaxed match, TimeDict performs slightly poorer than other methods but the results are above 90% on all three datasets.

5.2.1 *TimeDict-S v.s. Baseline Methods*

On TimeBank dataset, TimeDict-S achieves F_1 of 92.09% in terms of strict match and of 94.96% in relaxed match. It outper-

[12] http://www.cs.rochester.edu/~naushad/tempeval3/tools.zip

Table 8: Results of all methods on three datasets. The best measures are highlighted in bold face and the second best are underlined.

	Method	Strict Match			Relaxed Match		
		Pr	Re	F_1	Pr	Re	F_1
TimeBank	HeidelTime [23]	83.85	78.99	81.34	93.08	87.68	90.30
	SUTime [5]	78.72	80.43	79.57	89.36	91.30	90.32
	UWTime [11]	86.10	80.40	83.10	**94.60**	88.40	91.40
	TimeDict-S	91.43	92.75	92.09	94.29	**95.65**	**94.96**
	TimeDict-E	**91.49**	**93.48**	**92.47**	93.62	**95.65**	94.62
WikiWars	HeidelTime [11]	85.20	79.30	82.10	92.60	86.20	89.30
	SUTime	78.61	76.69	76.64	95.74	89.57	92.55
	UWTime [11]	**87.70**	78.80	**83.00**	**97.60**	87.60	92.30
	TimeDict-S	80.00	80.22	80.11	92.16	92.41	92.29
	TimeDict-E	79.18	**83.47**	81.27	90.49	**95.39**	**92.88**
Tweets	HeidelTime	**89.58**	72.88	80.37	95.83	77.97	85.98
	SUTime	76.03	77.97	76.99	88.43	90.68	89.54
	UWTime	88.54	72.03	79.44	**96.88**	78.81	86.92
	TimeDict-S	89.52	94.07	91.74	93.55	98.31	95.87
	TimeDict-E	89.20	**94.49**	**91.77**	93.20	**98.78**	**95.88**

Table 9: Number of time tokens and modifiers for expansion

Dataset	#Time Tokens	#Modifiers
TimeBank	4	5
WikiWars	16	19
Tweets	3	2

forms all baseline methods. On WikiWars dataset, TimeDict-S achieves competitive F_1 of 80.11% (v.s. the best one of 83.00%) in strict match and of 92.29% (v.s. the best of 92.55%) in relaxed match. One reason is that TimeDict follows the standard of TimeBank dataset which prefers short time expressions, while WikiWars dataset prefers the long expressions. For example, TimeBank will treat "2009-2010" as two time expressions; while WikiWars will treat it one. Moreover, TimeBank treats most of the prepositions like 'in', 'at', and 'before' as a time signal instead of part of time expressions, while WikiWars will treat them as part of time expressions. Therefore, TimeDict-S achieves slightly lower F_1 than UWTime and HeidelTime in strict match, but comparable F_1 with UWTime and better F_1 than HeidelTime in relaxed match. On Tweets dataset, our TimeDict-S achieves the F_1 of 91.74% in strict match and of 95.87% in relaxed match, outperforming all three baseline methods.

5.2.2 TimeDict-E v.s. TimeDict-S

Table 9 reports the number of time tokens and modifiers used to expand TimeDict-S to get TimeDict-E, on each dataset. We can see that only a few tokens are added for expansion.

On TimeBank dataset, among the added tokens, only the time token 'period' and the modifier 'fiscal' affect the result. TimeDict-E treats 'period' in phrase 'Soviet period' as a time expression, while the ground truth does not. That explains the slight drop in precision under relaxed match. Adding modifier 'fiscal' leads to the recognition of the phrase 'this fiscal year' and improves the performance in strict match. On WikiWars dataset, relatively more tokens are added, TimeDict-E performs much better than TimeDict-S, especially in recall. It improves the recall by 3.25% in strict match and 2.98% in relaxed match. One reason is that the ground truth annotations in WikiWar dataset contain a lot of long time expressions, which are very descriptive. Examples include 'the day after the US's northward crossing of the 38th-parallel border into North Korea' and 'the period of the CPA's inception on April 21, 2003'. On Tweets dataset, the added time tokens are 'tmr', 'tomo', 'tdy', and the modifiers are 'all', 'same'. The expansion of the TimeDict leads to slight improvement in performance measures.

5.3 Error Analysis

From our experimental results, we study the errors made by TimeDict and summarise our findings.

First, TimeDict can not identify most of time expressions involving proper nouns, such as, 'D-Day', 'World War I', 'Gandhian Era', 'Red All-saints' Day'. Because the keywords in TimeDict are meant to be generic and cover many time expressions, such specific words are not in the dictionary. This becomes a limitation by nature of the design. On the other hand, such time expression is unlikely to be recognized by rule-based time taggers.

Second, TimeDict follows the standard of TimeML [17] and TimeBank [18] and excludes the prepositions like 'in' and 'before'. TimeDict could not recognize the time expressions with 'in' and 'before' within the time expression, such as, 'late in 1985', 'a few months before', and 'six months before the Soviet deployment'.

Third, TimeDict is affected by incorrect POS taggs. We note that the POS tagger mistakenly assign the word 'sat' in 'friday or sat' a POS tag VBD, 'wed' in 'every mon, wed' POS tag VBN, and 'yr' in 'take yr Hot Yoga' with POS tag NN. All these wrong POS tags affect the recognition of TimeDict.

6. CONCLUSION

Time expression is one of the most important type of information in free text. The correct recognition of time expression could improve many tasks in Information Retrieval and Natural Language Processing. In this paper, we conduct an analysis of the time expressions over four datsets and observe that time expressions are in general very short, and are expressed by using a small vocabulary. More importantly, we observe that the tokens in time expression demonstrate syntactic behavior. Inspired by Part-of-Speech tagging, based on the observations, we propose a type-based time expression tagger named TimeDict. It group the time-related keywords in types and define simple rules on top of the types. As a rule-based system, TimeDict can be easily initialized by importing patterns for recognizing the tokens in these types from existing rule-based time taggers. Through experiments on three dataset, evaluated on benchmark toolkit, we show that TimeDict outperform state-of-the-art baselines, including both rule-based time taggers and machine learning based time tagger.

7. REFERENCES

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

[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*, pages 446–455, 2012.

[3] S. Bethard. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Proceedings of the 7th International Workshop on Semantic Evaluation*, pages 10–14, 2013.

[4] 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*, pages 3735–3740, 2012.

[5] A. X. Chang and C. D. Manning. Sutime: Evaluation in tempeval-3. In *Proceedings of second Joint Conference on Lexical and Computational Semantics (SEM)*, pages 78–82, 2013.

[6] A. X. Chang and C. D. Manning. Tokensregex: Defining cascaded regular expressions over tokens. Technical report, Department of Computer Science, Stanford University, 2014.

[7] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[8] L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson. Tides 2005 standard for the annotation of temporal expressions. Technical report, MITRE, 2005.

[9] 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.

[10] J. R. Hobbs, D. E. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel, and M. Tyson. Fastus: A cascaded finite-state transducer for extracting information from natrual-language text. In *Finite State Devices for Natural Language Processing*, pages 383–406, 1997.

[11] 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*, pages 1437–1447, 2014.

[12] 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*, pages 3044–3051, 2012.

[13] 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*, pages 284–291, 2010.

[14] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. Cambride: MIT Press, 1999.

[15] 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*, pages 913–922. Association for Computational Linguistics, 2010.

[16] R. Parker, D. Graff, J. Kong, K. Chen, and K. Maeda. Engilish gigaword fifth edition. 2011.

[17] 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*, 3:28–34, 2003.

[18] 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*, 2003:647–656, 2003.

[19] J. Pustejovsky, K. Lee, H. Bunt, and L. Romary. Iso-timeml: An international standard for semantic annotation. In K. Choukri, editor, *Proceedings of the Seventh conference on International Language Resouces and Evaluation (LREC'10)*, pages 394–397, 2010.

[20] M. Steedman. *Surface Structure and Interpretation*. The MIT Press, 1996.

[21] 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)*, pages 321–324, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[22] J. Strotgen and M. Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–198, 2013.

[23] 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)*, pages 15–19, 2013.

[24] J. Tabassum, A. Ritter, and W. Xu. A minimally supervised method for recognizing and normalizing time expressions in twitter. In *Conference on Empirical Methods in Natural Language Processing (arXiv)*, 2016.

[25] 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*, pages 276–283, 2010.

[26] 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*, pages 1–9, 2013.

[27] 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*, pages 75–80, 2007.

[28] 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.*, pages 81–84, 2005.

[29] 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*, pages 57–62, 2010.

[30] G. Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley Press, Inc., 1949.