International Journal of Information and Communication Sciences
2024, Vol. 9, No. 2, pp. 24-32
https://doi.org/10.11648/j.ijics.20240902.11

Research Article

Practical Results for General Models of Temporal, Spatial and Semantic Text Processing

Boghiu Șerban Paul*

Faculty of Computer Science, "Alexandru Ioan Cuza" the University of Iași, Iași, Romania

Abstract

Large-scale language models (LLMs) dominate tasks such as natural language processing and computer vision, but using their spatial and temporal predictive power still needs to be improved. The main focus of the task described in this paper is to offer a technical solution that implements the Time Yards Model (TYM), which means putting in evidence time events and their evolution in a novel book or large text in an unsupervised manner. This article has a ground foundation and uses a manually annotated novel to train multiple state-of-the-art machine learning techniques to divide time segments. The aim is to identify specific text chunks based on several features (part-of-speech and part-of-sentence) to branch off the temporal tracks. This manuscript uses TimeML specification language to mark relevant words in time and space to achieve tempo-spatial investigation. Based on the observations, the proposed algorithm extensively analyses various novels presented for comparison, underlining their main strengths and weaknesses. Comprehensive experiments on diversified datasets show that our proposed solution successfully unlocks the potential for spatial and temporal forecasting with reasonable accuracy. Remarkably, our approach has achieved competitive performance compared to relevant papers from this field. Various text-processing methods are described, and different natural language processing frameworks and libraries are compared.

Keywords

Formalism, Time, Temporality, Named Entity Recognition, Geoparsing, Recurrent Neural Network, Natural Text Processing

1. Introduction

The main focus of the research described in this paper is to offer a technical solution that implements the Time Yards Model, which means putting in evidence time tracks in a book text. This paper has as a ground foundation an already manually annotated novel used to train a Convolutional Neural Network (CNN) to identify time tracks. Several text-processing techniques are described, and different text-processing frameworks are compared. A few approaches regarding time segmentations are proposed and evaluated.

2. Definitions

The following definitions clarify terms frequently used in TYM, which will be referred to repeatedly in this report.

2.1. Events

The notion of event is close to that described in TimeML [1], mainly attributed to verbs and deverbatives. Syntactically, an event is realised at the level of a clause of a sentence, i.e. a word sequence around a main active verb or a verb compound. Semantically, an event mainly exposes an action in

*Corresponding author: serban.boghiu@info.uaic.ro (Boghiu Șerban Paul)

Received: 4 March 2024; **Accepted:** 15 April 2024; **Published:** 25 December 2024

which a verb is surrounded by roles, such as agent, recipient, etc., whose syntactic realisations in the clause are subject, direct object, obliques, etc.

Although not always explicitly expressed, events are positioned on the story time axis, either reported to time moments or intervals (year, month, week, day, hour, Christmas, Thanksgiving, etc.) or other events and then, they can be simultaneous, partially intersecting, placed before or after [2]. However, in many cases, there is no concrete positioning of an event concerning a temporal expression denoting a date, a moment or another event. As such, events that comprise a book's semantic content can be partially ordered on the story time axis.

Further, entities are usually involved in events. These can be book characters, objects, abstract ideas, etc. In TYM, persons (book characters) are of particular interest. They fulfil different verb roles, some being more important than others. In our analysis, the main concern is the semantic roles of agents and recipients (i.e. fulfilling syntactic positions of subjects, direct objects and indirect objects relating to a main verb.

2.2. Time Segments (TSs)

At the basis of our model conceptualising a practical approach towards TYM and of central focus in this paper is the notion of Time Segment (TS), which represents a continuous span of the book text or a sequence of clauses observing three constraints: the unity of time, of characters and perspective [3-5].

The *unity of time* refers to the fact that the period on the story time axis of the events that make up the TS are either synchronous or contiguous in their time sequentiality.

The *unity of characters* refers to the fact that the characters fulfilling the prominent roles (agent, recipient) remain constant throughout the TS.

The *unity of perspective* refers to the fact that the events constituting the TS are told by the same narrator (the author or one of the book characters).

2.3. Time Tracks (TTs)

By slightly reinterpreting the original definitions [3, 4], a TT can be described as a maximal sequence of time segments partially ordered concerning the story time axis, in which a stable set of characters is involved and whose stories are told from a unique perspective. The restriction of being maximal refers to the story time axis: a time track should include the "destiny" of a set of characters extended to the whole span of the book text. The restriction referring to a stable set of characters is selectional and not one that touches the semantic content of the text because it is known that, in a text, characters meet and separate, interfere and split continuously during the story's development. As such, a multitude of time tracks could be interrogated about a text for different sets of characters involved, and, in particular, a time track can be searched for for only one book hero. The notion of a time track has much in common with a narrative thread. By following just one character, one can perceive this hero's involvement in other characters' destinies, as depicted by the author.

Let's also notice that the sequence of TSs constituting a TT may leave uncovered periods on the story time axis, therefore unknown developments from the destinies of the corresponding characters, segments of time in their told lives about which the narrator does not say anything.

As said, each TT is attached to a specific set of characters and a particular narrator. Along one TT attached to a declared set of characters, specific points can be put in evidence:

1) the *start point*: its leftmost point on the story time axis. This is where the characters in the declared set are mentioned together for the first time in the novel,
2) the *end point*: its rightmost point on the story time axis. This is where the characters in the declared set are mentioned together for the last time in the novel:
3) a *split point* of a TT: an interior point within the time of the novel story in which the current TT is interrupted because its specific set of characters split, at least one character from the set exiting the set. A particular type of split point called an *interruption point*, refers to situations where the author temporally interrupts the narration of a specific set of characters. As such, a split point of a TT attached to one character only is always an interruption point because it represents a point on the story time axis where the author interrupts the narration about that particular character;
4) a *merge point*: an interior point within the period of the novel story in which the current TT meets another TT specific to another set of characters. This represents a point in the story where the set of characters specific to the current TT meets with at least another character. A particular type of merge point called a *restore point*, refers to situations where an interrupted narration on a specific set of characters is restored.

3. A Synopsis of the Processing Pursuit

The logical pursuit in implementing this model is as follows:

1). identify events, which is to find clause boundaries;
2). put in evidence temporal relations, either of an event concerning a fixed temporal borne or between events;
3). put in evidence the set of characters (person-type entities) participating in an event;
4). put in evidence the perspective from which a particular event is told;
5). identify time segment boundaries as a continuous sequence (in the length of the text) of events in which the time flows sequentially and/or is stationary;
6). detect the TT of a given set of characters (as dictated by

the user) in two steps: select among the detected set of TSs for those which include the given set and reorder this chosen set of TSs concerning their periods.

The following will describe each of these steps from an implementation point of view.

3.1. Event Segmentation

Events are the main brick in composing time segments. At the level of the text, an event is offered by a syntactic construct built around a verb, in which proper nouns, personal pronouns, etc., participate in the position of the subject, direct objects and other syntactic roles. To detect basic structures such as events, the processes of tokenisation and POS-tagging are realised for each sentence using the NLTK framework[1], which is immediately followed by chunking. The tokens are characterised by their part of speeches, while chunking offers the possibility to define patterns that constitute the event's structure. Such patterns are described in more detail in Chapter 4.

Event detection comes after performing the preprocessing phase, where all the needed information to build an essential event is extracted (character, temporal expression, location, action). All events must have a set of characters assigned, which can be specifically mentioned in the sentence, deduced by a coreference resolution module, or inferred based on heuristics. When no character of a type person is mentioned in an event, considering the same set of characters as of the previous event in sequence can be a successful heuristic. The same logic is applied to temporal expressions. They must be fully anchored (by specific temporal expressions of type date, moment, etc.) or take over the last identified temporal anchor from previous events (the current event will be immediately after the previous one on a time axis).

Events are detected in an initial phase using patterns for chunking, but they go through a second test where keywords from the specific list of conjunctions are identified in the sentence. A third test is made when no conjunction is found, but many verbs are present in the sentence. In this case, delimiter punctuation signs are identified, such as:,., -, ...,;. If no such sign is identified, the events are split in the proximity of the second verb. The described steps only apply to sentences with multiple verbs. In cases where a sentence contains only one verb, a single event equal to the whole sentence is extracted.

The following pseudocode can describe this algorithm for detecting events in multiple-verb sentences:

```
1). chunks = GetChunks(sentence)
2). if(chunks.length > 0)
3). SplitEventsByConjInChunks(chunks);
4). else if (ConjFromListFound(sentence))
5). SplitEventsByConjInList(sentence);
6). else if (PunctSignFound(sentence))
7). SplitEventsByPunctSign(sentence);
8). else SplitEventsByVerbs(sentence);
```

3.2. Identifying Temporal Relations

Once the events are individualised, they should be placed in the temporal order. The subordinating conjunctions provide essential clues in establishing relations of time precedence or succession between events. They provide a transition between two ideas in the same sentence, always indicating a temporal, spatial, or cause-to-effect relationship.

Temporal expressions are constructive in identifying the temporal anchoring of events. There are more cases to consider in time-anchoring the current event[2]:

Suppose the current event contains a temporal expression of type interval. In that case, this event should be attributed inside this temporal interval, as in the following example: 1. *John played tennis with Bill last week.* Unit 1 describes an event temporally anchored by the interval-type temporal expression the *previous week.*

Suppose no temporal expression is contained in the current event. In that case, the event's time is considered contingent or immediately following one of the previous events, as in the following example: 1. *Last week, John played tennis with Bill.* 2. *He won by 3 to 2*. Here event in Unit 2 follows the one in Unit 1.

On the other hand, coordinating and correlative conjunctions connect events of the same importance, which are usually simultaneous.

Conjunctive adverbs may not necessarily express space or time relationships between events but are clues indicating their belonging to the same TS.

The considered levels of annotation refer to:

1. phrase annotation (in an incipient phase);
2. lexical and morphological annotation (as revealed by a tokeniser, a lemmatiser and a POS-tagger);
3. Time annotation, as revealed by packages such as Duckling, SUTime, and others. The conjunctions mentioned connect temporal events and offer more significance to isolated temporal expressions found by libraries.

A complete list of used conjunctions can be found in Table 1.

[1] https://www.nltk.org/

[2] This list should be extended.

Table 1. *Conjunctions are used to link events.*

Subordinating conjunctions	*after, although, as, as soon as, because, before, by the time, even if, even though, every time, if, in case, now that, once, since, so that, then, the first time, unless, until, when, whenever, whether or not, while, why*
Coordinating conjunction	*for, and, nor, but, or, yet, so*
Correlative conjunction	*as/as, both/and, either/or, hardly/when, if/then, just as/so, neither/nor, not only/but also, no sooner/than, not/but, rather/than, scarcely/ when, what with/and, whether/or*
Conjunctive adverbs	*additionally, again, almost, anyway, as a result, in addition, besides, indeed, comparatively, consequently, contrarily, comparatively, consequently, conversely, elsewhere, equally, eventually, finally, further,* furthermore, elsewhere, hence, henceforth, however

3.3. Identifying Characters Mentioned in Events

Identifying characters involved in events is directly connected with the TS constraint called *unity of characters*. This task can be realised as a cooperation of more components, among which the recognition of name entities and anaphora resolution is of primary importance.

An event should have a character or a set of characters associated with it, which can be identified through a series of Named Entity Recognition frameworks, described in detail in Chapter 4. If no such entity is found, the coreference resolution module should be called to resolve other nominal expressions that could mention the characters involved in the event.

To accomplish the Named Entity Recognition (NER) task, a few libraries are used, with the focus on the entity of type person. Here, ANNIE from GATE [6], SpaCy [3] and OpenNLP[4]. Unlike ANNIE GATE, which only specialises in English, SpaCy offers support for at least 64 languages, including Romanian. OpenNLP also supports processing texts in a few languages, but Romanian isn't one of them, with the main focus being English. They all offer the same functions: tokenisation, POS tagging, sentence segmentation, lemmatisation, named entity recognition, etc. The only difference is in the volume of training data and the response time (NER may be more time-consuming depending on the amount of data available to search).

An anaphora resolution resolver must also be used to recognise coreference chains of persons. ANNIE GATE was used for this task, which provides the Pronominal Coreference module based on a JAPE grammar formalism. The text must be passed through a preprocessing phase where it is split into sentences, and for each one, a list of persons, organisations and locations is extracted using the NER tool they also provide. Information about the gender of each named entity is considered, as well as if it is a pleonastic "it" coreference, to ensure the best match.

Once the entity mentions the type of persons identified, linking them to clauses (therefore identifying the involvements of those characters in events) is a simple assignment task. Once this is done, the unity of character constraint can be solved.

3.4. Perspective Identification

The unity of perspective aims to identify a specific event's narrator and impose a boundary to the time segment where the perspective changes. The perspective is usually attributed to the author or one of the characters participating in the story.

For simplification reasons, this study will ignore the constraint that mentions the unity of perspective; therefore, only texts with only one narrator will be considered.

Detecting TS boundaries

Segmentation of the text for time segments is described in detail in Section IV of this paper.

Detecting Time Tracks

Time tracks and their corresponding unities are easy to identify and represent when given a text. For this, some notation conventions will be used:

Let $C = \{c_0, c_1, \ldots c_n\}$ be a set of characters;

Let $T = \{t_0, t_1, \ldots t_n\}$ be a set of temporal anchors;

Let $P = \{p_0, p_1, \ldots p_n\}$ be a set of perspectives (narrators) - for this article, $P = \{p_0\} = \{$"author"$\}$ will be considered;

Let $EV = \{ev_0, ev_1, \ldots ev_n\}$ be a set of events, where each event can be linked to a set of characters, a temporal anchor, a location, an action (verb with a specific tense) and an order number: $ev_i = (c_i, t_,, l_i, a_,, o_i)$. The order number helps order events with the same tense and no temporal expressions as-

[3] https://spacy.io/

[4] https://opennlp.apache.org/docs/1.9.2/manual/opennlp.html

sociated with them but are linked through one of the above-mentioned conjunctions.

Suppose events are linked through a subordinating conjunction. In that case, the order number will be lower for the first one if the conjunction is a precedence one (*before*) or more significant if the conjunction is a succession one (*after*). For the other conjunction types, the order number will be the same for all events linked by them. The order number is increased incrementally depending on the temporal expressions or conjunctions identified within. The order number will be recalculated each time a temporal expression is found, and the event needs to be repositioned on the time axis.

Let TST = {Past, Present, Future} be a set of TS types used to position time tracks - they will be extracted using the tenses of verbs identified in events.

Let TS = {ts_0, ts_1, … ts_n} be a set of time segments, where each time segment contains a list of events ts_i = {[ev_i,.., ev_n]}

Let TT = {tt_0, tt_1, … tt_n} be a set of time tracks, where each time track contains an array of tuples consisting of a time segment and the time segment type (ts_i, tst_i)

Given an element from a time track (ts_i, tst_i) for all time events {ev_{i0}, … ev_{in}} in ts_i the verbs must have the same tense category (Past, Present, Future) as tst_i.

All events {ev_{i0}, … ev_{in}} must have successive order numbers and temporal expressions (if any).

Our goal in this paper is to present a proposal for identifying time tracks so that the flow of the text can be represented on a time axis depending on multiple criteria, such as characters or locations.

For example, let's consider a fragment from the annotated novel *Map of the Invisible World* [7] that is used to train a neural network for finding the boundaries of time segments. Suppose it applies to the following span of text from the novel: *This is Adam. He came to live in this house when he was five years old. Now, he is sixteen, and he has no memory of his life before he came here.*

For this small fragment, the following sets can be defined:

C = {"Adam"};

T = {"five years old", "now"};

P = {"author"};

EV = {ev_1, ev_2, ev_3, ev_4, ev_5, ev_6}, where:

ev_1 = "This is Adam"

= ("Adam", "", "", "is", 1)

ev_2 = "He came to live in this house"

= ("Adam", "", "", "came", 2)

ev_3 = "when he was five years old"

= ("Adam","five years old","","was", 3)

ev_4 = "Now he is sixteen"

= ("Adam", "now", "", "is", 4)

ev_5 = "and he has no memory of his life"

= ("Adam", "", "", "has", 5)

ev_6 = "before he came here"

= ("Adam", "", "", "came", 6)

TS = {ts_1, ts_2, ts_3, ts_4}, where:

ts_1 = (ev_1)

ts_2 = (ev_2, ev_3)

ts_3 = (ev_4, ev_5)

ts_4 = (ev_6)

TT = {tt_1, tt_2}, where:

tt_1 = [(ev_1, Present), (ev_4, Present), (ev_5, Present)]

tt_2 = [(ev_2, Past), (ev_3, Past), (ev_6, Past)]

4. Data Extraction

A corpus already annotated to TYM was used to automatically detect time segments and build time tracks [4]. Apart from this, a preprocessing phase is necessary to provide the elementary features needed for the learning phase of a neural network approach. Not only a few libraries are used, but many, and the results are merged for two reasons: in many cases, some libraries may have better results than others, and features must be combined to provide more details. The preprocessing phase extracts from the raw text the details of information necessary to perform event detection and segmentation into TSs.

The preprocessing phase

The first step in data preprocessing is to split the text into token sentences, notate parts of speeches, and identify chunks and tasks realised (for English) with help from the NLTK framework. This is very helpful in developing the rule-based and dictionary-based modules for extracting temporal expressions and events and building the corresponding time tracks. NLTK was chosen because it identifies not only the part of speeches of tokens but also their relevant morpho-syntactic information. For example, for verbs, it identifies the person and the tense, while for conjunctions, it identifies if it's a coordinating or a subordinating one. This framework comes with the possibility of creating chunking rules that prove very helpful in detecting and linking events.

For example, a sentence may contain several clauses linked or not by conjunctions with the same tenses. To identify which events should be linked, patterns can be utilised, such as the ones described below using the part of speeches from Annex 1:

Sarah sings and dances at the same time.

{<NN.?>*<VBZ>*<CC>*<NN.?>*<VBZ>}

Sarah plays the guitar, and she dances as well.

{<NN.?>*<VBZ>*<CC>*<PRP.?>*<VBZ>}

She is going home before night comes.

{<PRP.?>*<VBZ><VBG>*<IN>*<PRP.?><VBP>}

In addition, future tenses can not be detected by NLTK, so patterns must be used to identify them, such as:

John is going to arrive soon.

{<NN.?><VBZ><VBG><TO>}

John will arrive tomorrow.

{<NN.?><MD><VB>}

Jane will be watching TV when I come home.

{<NN.?><MD><VB><VBG>}

I shall have completed the task by tomorrow.

{<PRP.?><MD><VB><VBD>}

The second step is to extract temporal expressions. SUTime, Duckling, and SpaCy were used to process the text from a temporal perspective.

For the example mentioned above, SUTime and SpaCy annotated *five years old* and *now* as temporal expressions of type Duration and Date:

<TIMEX3 tid="t1" type="DURATION" value="P5Y">five years old</TIMEX3>

<TIMEX3 tid="t2" type="DATE" value="PRESENT_REF">Now</TIMEX3>

On the other hand, Duckling only found the temporal expression *five years* of type Duration with basic settings, but the advantage is that new rules can be configured:

<TIMEX3 tid="t1" type="DURATION" value="P5Y">five years old</TIMEX3>

Table 2 shows some statistics regarding the temporal expressions extracted for the exact text used as the testing dataset.

Table 2. *Statistics of extracted temporal expressions.*

Framework	Type	Number of temporal expressions
SUTime	Date & Time	1012
	Duration	280
Duckling	Date & Time	1011
	Duration	279
SpaCy	Date & Time	1012
	Duration	278

Although these frameworks can extract temporal expressions and durations, placing events on a time axis is often insufficient. This is where the rule-based approach comes in.

Identifying named entities of type Person and Organization is equally vital for creating characters' timelines. Because they are used for the same purposes, statistics from the ANNIE GATE, SpaCy, and OpenNLP frameworks can be found in Table 3.

Table 3. *Statistics of extracted locations and named entities.*

Framework	Type	Number of expressions
ANNIE GATE	Persons	49
	Organisations	5
SpaCy	Persons	51
	Organisations	2
OpenNLP	Persons	47
	Organisations	3

Coreference resolution is the next step in identifying characters, although they are not mentioned explicitly in the text. Since this paper only considered texts in English, ANNIE GATE was used for this task due to its simplicity in use and provided data structures. For other languages, SpaCy can be considered.

5. Data Training

The training set consists of the annotated novel *Map of the Invisible World* [7] in both English and Romanian in XML format. An example of a small fragment with two-time segments and their corresponding describing entities is:

<TA ID="A0" NAME="Narrator" />

<TA ID="A1" NAME="Adam" />

<TA ID="A2" NAME="soldiers" />

<TL ID="L1" NAME="unknown" PARENTID="" />

<TL ID="L2" NAME="the house" PARENTID="" />

<TS ID="TS1" SPANS="0~167" ACTORS="A1" LOCATION="L1" NAME="Adam found himself alone" TYPE="Narration" PER="A0" TEXT="1 When it finally happened, there was no violence, hardly any drama. It was over quickly, and then Adam found himself alone again. Hiding in the deep shade of the bushes, he saw this."/>

<TS ID="TS2" SPANS="167~1379" ACTORS="A1, A2, A3" LOCATION="L2" NAME="Soldiers had Karl with them and Adam remained shrouded by the dense thorny foliage" TYPE="Narration" PER="A0, A1" TEXT="The soldiers jumped from the truck on to the sandy soil. They dusted themselves off, straightening their hitched-up trouser legs and tucking their shirts into their waistbands. Their long sleeves were rolled up thickly above their elbows and made their arms look skinny and frail, and the belts they wore were so wide they seemed to stretch their waists to their chests. They laughed and joked and aimed pretend kicks at one another. Their boots were too big and they looked like clowns when they ran. They were just kids, Adam thought, just like me, only with guns.

They hesitated as they approached the veranda, talking among themselves. They were too far away; he couldn't hear what they were saying. Then, two of them went up to the house, and Karl was with them when they emerged. He was not handcuffed; he followed them slowly, walking to the truck with his uneven gait before climbing up and disappearing under the tarpaulin canopy. From a distance, he looked small, just like them, just like a child too, only with fair hair and pink skin.

Stop. Adam wanted to call out, to scream for Karl to come back. Don't leave, he wanted to shout. But he remained silent

and unmoving, shrouded by the dense thorny foliage. He could do this now: he held his breath and counted slowly from one to ten." />

<TREL ID="TI1" FROM="TS2" TO="TS1" REL="IMMEDIATELY_BEFORE" TRIGGER="this is what he saw" />

Time segments (TS) are annotated using the following tags:

ID - unique identifier;

SPANS - start and ending offset of the time segment in text;

ACTORS - list of characters;

LOCATION - id of a previously identified location (proper noun or simple noun)[5];

NAME - small description of the time segment;

TYPE - can have the following values: Narration, Remembers, Supposition, General Knowledge and Fiction[6];

PER - from whose perspective the time segment is being narrated;

TEXT - the actual text sequence.

Relations between time segments (TREL) are also present, providing the following information:

ID - unique identifier;

FROM - the starting time segment id;

TO - the ending time segment id;

REL - can have the following values: BEFORE, IMMEDIATELY_BEFORE, SIMULTANEOUS, IMMEDIATELY_AFTER, AFTER.

Time actors (TA) and time locations (TL) have the same tags:

ID - unique identifier;

NAME - proper nouns or simple nouns;

Text vectorisation must be used to train a neural network [9]. Since sentences have different sizes and characteristics, they must be converted into vectors with the same number of relevant features. The expected input for most neural networks consists of a two-dimensional array where the lines correspond to each instance (time segment in our case), and the columns correspond to the most meaningful features chosen to represent the time segment.

A first attempt at transforming text into numerical representations was made using the frequency vectoriser. The main idea is to create a dictionary with all the distinct words in the text and compute the number of occurrences for each. This can be done at the whole text level or just at the sentence level. The problem with this implementation is the size of the dictionary when it comes to processing significant texts and the useless number of zeros stored for words that may appear very rarely, negatively impacting performance. A solution to this problem was to include only relevant words, such as nouns, verbs and conjunctions, that provide valuable information. Although the computational time and dictionary size decreased significantly, the model did not represent the data in a manner that can be reliable for future testing.

The solution to this problem came using the Term frequency-inverse document frequency (TF-IDF) [10]. The main idea is that some words may be found often but only sometimes offer significant information. So, the problem is solved by normalising the term frequency, considering the whole text. In other words, if a word is found very often in a sentence (or time segment) but rarely in others, then the term is much more significant. For example, a character may be found mentioned multiple times in a sentence (explicitly or identified by coreference resolution), but more rarely in others, then it gains more importance.

After computing the corresponding table for all significant terms (nouns, verbs, conjunctions), only the most significant ones were kept (the chosen TF-IDF value had to be > 0.7).

The TF-IDF method was used to find the most significant terms that must be embedded later in the word2vec format [11]. This format is widely used as a model for neural networks, and this article will focus on the classification task using a Convolutional Neural Network (CNN) [8].

Other features have been added as well, being mostly composed of counts:

Tense of verbs in a time segment - quantified as 1 = Past, 2 = Present, 3 = Future;

Number of verbs;

Number of conjunctions for each type - quantified as 1 = coordinating conjunction, 2 = subordinating conjunction;

Number of characters;

Counts of associated temporal expressions - quantified as numbers of: 1 = precedence, 2 = simultaneity, 3 = succession;

Difference between associated temporal expressions from different TS where the relation from TS1 and TS2 equals to BEFORE or IMMEDIATELY_BEFORE quantified as: 1 = precedence, 2 = simultaneity, 3 = succession - the expected result should be 1 - same logic is applied for other relation types.

The training set consisted of 376-time segments and 382 time segment relations for the English language. Because the size of the annotated TSs in the corpus is insufficient for training a neural network that requires at least a few thousand data points, the extracted events were considered from each TS as training data points since the text from each segment was of a decent length. As a result, over 10,000 events were extracted. The training set consisted of 80% events, and the test set consisted of 20%.

6. Results

Time segments were identified using the classifications retrieved from CNN for every event. Every extracted event from the testing set has the same features computed as the events in the training data set. The goal is to classify each using the Past, Present, or Future categories.

Time segments are then obtained by concatenating events of the same category and in ascending order of their starting and ending position (offset) in text until the category changes. This point indicates another time axis starts, so another time

[5] Not used in this implementation.

[6] Idem

segment should be delimited.

Time tracks can easily be obtained by comparing the temporal expressions associated with the time segments: compare the first identified temporal expression from TS1 with the last identified temporal expressions from TS2 to establish one of the relations: Before, Simultaneous, and After.

The model was evaluated using a confusion matrix in Table 4 for a testing set of 2,000 extracted events. The number of extracted time segments was equal to 93.

Table 4. Resulted from the confusion matrix.

n = 2000 events	Predicted Past	Predicted Present	Predicted Future
Actual Past	476	287	98
Actual Present	143	439	113
Actual Future	67	114	263

Table 5. POS detected by NLTK.

Part of Speech	Symbol
Coordinating conjunction	CC
Preposition/subordinating conjunction	IN
Modal (could, will)	MD
Noun, singular (cat, tree)	NN
Noun plural (desks)	NNS
Proper noun, singular (Sarah)	NNP
Proper noun, plural	NNPS
Personal pronoun	PRP
Infinite marker (to)	TO
Verb (ask)	VB
Verb gerund (judging)	VBG
Verb past tense (pleaded)	VBD
Verb past participle (reunified)	VBN
Verb, present tense, not 3rd person singular (wrap)	VBP
Verb, present tense with 3rd person singular (bases)	VBZ

7. Conclusions

Most NLP frameworks performed well, identifying almost the same number of temporal expressions, named entities and organisations. Although ANNIE GATE offers excellent support in English and their data structures are easier to use, SpaCy is recommended for extending the language portfolio.

Although the training set was fragile, this obstacle was overcome by using events for classification instead of time segments. A training set consisting of a few thousand-time segments is required for better precision in future studies. The selected features also highly influence the classification method, so a feature selection step was proposed to identify relevant ones.

Abbreviations

LLMs	Large-scale Language Models
TYM	Time Yards Model
CNN	Convolutional Neural Network
TS	Time Segment
TT	Time Track
POS	Part of Speech
TREL	Relations Between Time Segments
TA	Time Actors
TL	Time Locations
NER	Named Entity Recognition
TF-IDF	Term Frequency-inverse Document Frequency

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Pustejovsky, James & Ingria, Robert & Saurí, Roser & Castaño, José & Moszkowicz, Jessica & Katz, Graham. (2005): The Specification Language TimeML, TimeML Project.

[2] James F. Allen, George Ferguson (1994): Actions and Events in Interval Temporal Logic, Journal of Logic and Computation, Volume 4, Issue 5, October 1994, pp. 531–579, https://doi.org/10.1093/logcom/4.5.531

[3] Cristea, Dan and Macovei, Andreea (2017): Why Time Resembles Rail Yards? A Way to Look at Time in Textbooks, in Romanian Journal of Information Science and Technology, Vol. 19, 1-2, 2016, pp. 31-43.

[4] Macovei, Andreea (2021): Deciphering temporal and semantic relations in texts, Ph.D. thesis, "Alexandru Ioan Cuza" University of Iași, Faculty of Computer Science.

[5] Boghiu, Șerban (2020): Spatio-Temporal and Semantic Information Extraction, internal Ph.D. report, "Alexandru Ioan Cuza" University of Iași, Faculty of Computer Science.

[6] Cunningham, Hamish & Maynard, Diana & Bontcheva, Kalina & Tablan, Valentin (2002): GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02).

[7] Tash Aw (2009): Map of the Invisible World, https://booksvooks.com/map-of-the-invisible-world-pdf-tash-aw.html

[8] Amit, Kumar Sharma & Sandeep, Chaurasia & Srivastava, Devesh Kumar (2020): Sentimental Short Sentences Classification by Using CNN Deep Learning Model with Fine Tuned Word2Vec, Procedia Computer Science, Volume 167, 2020, Pages 1139-1147, ISSN 1877-0509. https://doi.org/10.1016/j.procs.2020.03.416.

[9] Kozhevnikov, Vadim & Pankratova, Evgeniya (2020): Research of the Text Data Vectorization and Classification Algorithms of Machine Learning. Theoretical & Applied Science. https://doi.org/10.15863/TAS.2020.05.85.106

[10] Mingyong Liu, Jiangang Yang (2012): International Conference on Computer Technology and Science (ICCTS 2012). https://doi.org/10.7763/IPCSIT.2012.V47.9

[11] Ying Xie, Linh Le, Yiyun Zhou, Vijay V. Raghavan (2018): Deep Learning for Natural Language Processing, Handbook of Statistics, Elsevier, Volume 38, 2018, Pages 317-328, ISSN 0169-7161, ISBN 9780444640420. https://doi.org/10.1016/bs.host.2018.05.001