# Synthesis of LTL Formulas from Natural Language Texts: State of the Art and Research Directions

**Andrea Brunello**[1]
University of Udine, Italy
andrea.brunello@uniud.it

**Angelo Montanari**
University of Udine, Italy
angelo.montanari@uniud.it

**Mark Reynolds**
University of Western Australia, Australia
mark.reynolds@uwa.edu.au

## Abstract

Linear temporal logic (LTL) is commonly used in model checking tasks; moreover, it is well-suited for the formalization of technical requirements. However, the correct specification and interpretation of temporal logic formulas require a strong mathematical background and can hardly be done by domain experts, who, instead, tend to rely on a natural language description of the intended system behaviour. In such situations, a system that is able to automatically translate English sentences into LTL formulas, and vice versa, would be of great help. While the task of rendering an LTL formula into a more readable English sentence may be carried out in a relatively easy way by properly parsing the formula, the converse is still an open problem, due to the inherent difficulty of interpreting free, natural language texts. Although several partial solutions have been proposed in the past, the literature still lacks a critical assessment of the work done. We address such a shortcoming, presenting the current state of the art for what concerns the English-to-LTL translation problem, and outlining some possible research directions.

**2012 ACM Subject Classification** General and reference → Surveys and overviews; Computing methodologies → Natural language processing; Computing methodologies → Machine learning; Computing methodologies → Temporal reasoning; Theory of computation → Evolutionary algorithms

**Keywords and phrases** Evolutionary algorithms, Machine learning, Natural language processing, Semantic parsing, Temporal logic

**Digital Object Identifier** 10.4230/LIPIcs.TIME.2019.17

## 1 Introduction

Linear temporal logic (LTL) is a formalism which is widely used in model checking (see, for instance, [40, 45]); moreover, LTL formulas make it possible to unambiguously describe the relationships among occurrences of events over time, a capability which turns out to be quite important in many automated reasoning fields. For these reasons, LTL can be considered as a preferred means to formalize and then reason upon natural language texts, which is most useful in tasks such as requirement specification and, more generally, the analysis of temporally-declined semantic content in texts [90].

Despite of their usefulness, the correct specification and interpretation of temporal logic formulas typically requires a strong mathematical background, and this severely limits their applicability by untrained domain experts. In such contexts, a system capable of automatically translating between English and temporal logic would be of great help.

[1] Corresponding author


Concerning the LTL-to-English translation, it may be achieved in a relatively easy way, by simply parsing the logical formula by means of an attribute grammar and applying some heuristics to make the English translation sound as natural as possible [81]. As for the opposite direction, several issues have to be dealt with, such as the inherent ambiguity of natural language, the possible lack of an explicit context in the sentences, or reference to background knowledge which may not be included in the utterance. For all these reasons, the task of translating a free, unconstrained English text into a free, unbounded LTL formula is still an open problem, although several partial solutions have been proposed over the years (see Section 3).

In this paper, we give an overall assessment of the state of the art for what concerns English-to-LTL translation. In addition, we make a critical evaluation of some of the currently available tools which may be adapted and combined for such a task, and we outline some possible future research directions.

The paper is organized as follows. In Section 2, we introduce the problem of English-to-LTL translation. Then, in Section 3, we provide a short state of the art. Next, in Section 4, we outline possible future research directions. Finally, in Section 5, the outcomes of an experimental evaluation of some of the most significant tools proposed in the literature that may be used to develop and English-to-LTL translation architecture are reported. Conclusions summarize the main contributions of the work done.

## 2 Problem definition

The problem of extracting temporal logic formulas from a natural language utterance may be considered as an instance of the *Semantic Parsing* task, that is, the process of mapping a natural-language sentence into a formal, typically machine-understandable representation of its meaning [95].

It must be observed that such a problem differs from the one of inferring logical formulas from examples. In the latter, a dataset of logs, or traces, resulting from the execution of a given system is considered, with the goal of determining which formulas characterize and distinguish between examples that describe a good behaviour of the system from those that do not (see, for instance, [70]).

In the present work, we focus on *Linear-Time Temporal Logic* (LTL for short) [79]. An LTL formula is built from a finite set of proposition letters by making use of Boolean connectives and the temporal modalities **X** (next) and **U** (until). Formally, LTL formulas can be defined as follows:

- if $p \in \mathcal{AP}$, then $p$ is an LTL formula;
- if $\psi$ and $\phi$ are LTL formulas, then $\neg\psi$, $\phi \vee \psi$, $\mathbf{X}\psi$, and $\phi\mathbf{U}\psi$ are LTL formulas.

On the basis of these temporal operators, additional modalities can be defined, most commonly the Boolean connectives $\wedge$ (and), $\rightarrow$ (imply), **true**, and **false**, and the temporal modalities **G** (globally) and **F** (eventually).

As for the semantics, let $w = a_0, a_1, a_2, \ldots$ $(a_i \in 2^{\mathcal{AP}})$ be an infinite sequence of truth evaluations for proposition letters in $\mathcal{AP}$, and let $w(i)$ denote the truth evaluation at position $i$. The satisfaction relation $\models$ between $w$ and an LTL formula can be defined as follows:

- $w \models p$ if $p \in w(0)$;
- $w \models \neg\psi$ if $w \not\models \psi$;
- $w \models \phi \vee \psi$ if $w \models \phi$ or $w \models \psi$;
- $w \models \mathbf{X}\psi$ if $w(1) \models \psi$;
- $w \models \phi\mathbf{U}\psi$ if $\exists\, i \geq 0$ s.t. $w(i) \models \psi$ and $\forall\, 0 \leq k < i$ $w(k) \models \phi$.

The semantics of the derived modalities is then defined as follows:

- $\mathbf{true} \equiv p \vee \neg p$;
- $\mathbf{false} \equiv \neg\mathbf{true}$;
- $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$;
- $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$;
- $\mathbf{F}\psi \equiv \mathbf{true}\mathbf{U}\psi$;
- $\mathbf{G}\psi \equiv \neg\mathbf{F}\neg\psi$.

As an example, consider the following statement:

> *"After the button is pressed, the light will turn red until the elevator arrives at the floor and the doors open."*

Such a situation can be captured by the following LTL formula:

$$p \rightarrow \mathbf{X}\ (\ q\ \mathbf{U}\ (\ s \wedge v\ )\ )$$

where $p, q, s$, and $v$ are proposition letters corresponding to the button being pressed, the light turning red, the elevator arriving, and the doors opening, respectively.

## 3 A short state of the art

Over the years, several authors have devised methodologies to translate English sentences into LTL formulas. However, they typically do not deal with the most general case of translating natural, unbounded, English sentences into general, unbounded LTL formulas: assumptions are made that either reduce the generality of the input text or the output formula.

In [29], the authors present a pattern-based approach to the encoding of property specifications for finite-state system verification. Among other contributions, they provide a set of 55 LTL-based formulas that should capture typical patterns that come into play in the design of concurrent and reactive systems, such as the existence of a specific condition or the conditional response to a stimulus. A detailed description of such patterns, together with a repository of examples, are available on the SAnToS laboratory website [5]. Such a set of patterns has been later extended to deal with more advanced scenarios, such as, for instance, timed property specifications [42] and real-time systems [49].

In [72], the authors start from the LTL repository available in [5], identifying the 8 patterns that, according to their analysis, are the most frequently used in temporal requirement specification, covering over 80% of the cases they encountered, that all lie in the aerospace domain. Then, they develop a set of shallow classifiers (such as Random Forests [22] and Support Vector Machines [25]) that are capable of detecting the presence of such patterns in textual technical specifications. As for the predictor attributes, they make use of a bag-of-words approach, enriched with information derived from Part of Speech (PoS) tagging. It should be observed that no solution is given to the problem of instantiating a detected pattern on the specific textual data. Thus, such an approach is not able of deriving a complete translation from English to the restricted set of LTL formulas they consider.

In [100], an algorithm is presented to translate a property, which is specified by making use of predefined subset of English (referred to as *controlled English)*, to LTL. The approach relies on syntactical properties only, being based on text processing techniques like grammatical dependency parsing.

In [59], LTL is used as a formalization technique within an integrated system for generating, managing, and executing controllers for autonomous robots. The idea is that a user should be capable of instructing a robot by means of natural language, so no assumption is made on the

form of English text that is given as input to the system. A textual instruction is processed making use of tools for speech tagging, dependency parsing, and null element restoration (see, for instance, [36]). Then, all verbs are identified, together with their arguments. Finally, each verb is mapped into a set of *senses* in VerbNet [84]. Although in principle no restriction is made on the kind of input phrases that a user may submit to the system, in practice the translation is based on a mapping between *senses* and LTL formulas, that is in turn based on different, manually specified, combinations of so-called *macros*.

In [96], a framework for requirement consistency management is presented. In order to formalize requirements, the system automatically translates natural language descriptions of functionalities into a logic representation. However, a restricted English grammar is considered, to avoid problems such as semantic ambiguity and to make the overall translation process easier.

In [41], the ARSENAL framework for translating natural language requirements into analyzable formal representations is discussed. Given an input sentence, the first step exploits both domain-specific and domain-independent knowledge to identify entity n-grams, such as, for instance, "Lower Desired Temperature", which are then converted into single terms like, e.g., `Lower_Desired_Temperature`. In addition, common expressions, such as "is greater than or equal to", are converted into simple terms, like `dominates`, in an attempt to regularize the input text and reduce its complexity. Next, a dependency parsing step is performed to extract grammatical relationships between phrase elements. These pieces of information, together with PoS tags, are then fed to a so-called *semantic processor*, that guides the translation from English to an internal, intermediate representation. It should be observed that such a module relies on a set of hand-made rules, that inevitably are domain-specific and can only work for restricted scenarios. Finally, the intermediate representation can be converted into several formalisms, including LTL.

Last but not least, a *controlled natural language* is defined in [83], which can be used to specify restrictions on how a system model interacts with its environment. Sentences formulated in such a constrained language are then automatically translated to LTL by means of hard-coded rules.

Broadening the attention scope to other kinds of (temporal) logical formalisms, the following contributions are worth mentioning.

In [32], the authors present a tool for the automatic translation of natural language sentences into formulas of the action-based temporal logic ACTL, in the context of the formalization of reactive systems requirements. A corpus of sentences is analyzed and, starting from it, a context-free grammar is manually built that allows one to parse the considered instances. Such a grammar is augmented with attributes, that allow one to generate the target ACTL formula during the parsing process.

In [71], a translation method is discussed that converts constrained English utterances into a representation level based on Kamp's *Discourse Representation Theory* [46]. Such an intermediate representation is then translated into an ACTL temporal logic formula.

In [30], the authors present a methodology to translate natural language instructions into a formal logical description of goals, that can then be issued to and followed by robots. The target formalisms are CTL$^*$ and first-order dynamic logic (FDL). The translation process is carried out by means of a hand-made combinatory categorial grammar [87].

In [43], hand-made attribute grammars are employed to generate Computation Tree Logic (CTL) formulas from Hardware Description Language (HDL) code comments written in English.

Finally, in [89], a two-phase approach to parsing natural language into formal logic is presented. The first phase aims at extracting the generic structure of the logical expression associated with a given natural language utterance. The general idea is that of building a dependency parse of the input utterance, and then attaching to its nodes $\lambda$-expressions chosen from a pre-specified finite set, by means of a specifically trained model. The node assignment to $\lambda$-expressions is then evaluated to a generic logical form structure. The second phase instantiates the discovered pattern on the specific utterance data. Some considerations are made about the possibility of adopting LTL as a target formalism in future work.

Overall, the typical approach followed by these studies can be summarized as follows: given an input English utterance, preprocess it to extract syntactical information, which may include part of speech tagging, dependency parsing, semantic role labelling, and so on. Then, enrich the input with these pieces of information. Finally, run an attribute grammar-based parser, or rely on some hand-made rules, to derive a translation into a target logical format. A notable exception is the work of [89], where a fully-supervised learning setting is considered.

# 4 Possible future research directions

In the following, on the basis of the analysis of related work done in the previous section, we outline some possible approaches to the problem of translating open English utterances into general LTL formulas. For each of them, we identify existing tools and solutions from the literature that can possibly be exploited.

## 4.1 LTL synthesis via classical semantic parsing

Classical approaches to semantic parsing involve the definition of a suitable grammar, which it is possible to rely on to parse a given phrase, and to contextually generate a desired output. Typically, attribute grammars or combinatory categorial grammars have been considered in the literature for this task (see Section 3), although specifically developed grammars have also been proposed [69].

As already pointed out, the problem of generating LTL formulas from English texts can be viewed as an instance of semantic parsing. Some frameworks that allow one to develop semantic parsers are available in the literature, most notably KRISP [47], SEMPRE [19], Cornell Semantic Parsing Framework [18], SippyCup [11], and WASP [15].

However, the main difficulty remains that of defining a suitable grammar, a task that for real-world applications cannot be performed by hand, especially when, instead of restricting to a specific domain, we refer to a general translation scenario. In an attempt to facilitate this task, several approaches to grammar induction have been proposed over the years. Some of them are specifically oriented to natural language processing, like, for instance, those presented in [28, 44, 86, 101]; others are more general, e.g., [53, 77]. Nevertheless, they all require a quite large amount of training data, a problem that is shared also by the (more promising) strategy presented in Section 4.2.

## 4.2 LTL synthesis as a translation problem

Machine translation can be viewed as the use of software to translate text or speech from one language to another. Several approaches to translation have been proposed over the years.

The first proposed one was the so-called *rule-based paradigm*, in which explicit rules are given that guide the translation. The translation can be done in two different ways: either directly mapping the source language into the target one, or relying on some sort

of interlingual representation. Of course, the richness and complexity of natural language imply that such handcrafted approaches are only suitable for very constrained domains. Nevertheless, some platforms for rule-based translation are available, such as, for instance, Apertium [35].

Subsequently, *statistical machine methods* have gained in popularity. In such a case, translations are not generated by ad-hoc developed rules anymore, but are, instead, generated on the basis of statistical models, whose parameters are trained on bilingual text corpora.

At the moment, the most popular approach is the one based on *neural machine translation*, where a large artificial neural network is used to predict, given an utterance in the source language, the likelihood of a sequence of words in the target language. Several frameworks have been proposed in the literature to develop general-purpose neural networks, such as, for instance, PyTorch [9], Tensorflow [13], and Keras [6]. Recently, OpenNMT [48] has been presented as an open source toolkit specifically oriented to neural machine translation. The system prioritizes efficiency, modularity, and extensibility, and it allows one to develop state-of-the-art solutions, based on Convolutional Neural Networks, LSTM, attention mechanisms, and word- as well as character-based embeddings, through a simple general-purpose interface, that in principle requires only source and target files to be provided. Besides language translation, the framework also supports other sequence generation tasks, such as, for example, summarization, image-to-text and speech recognition[2].

It is worth pointing out that most of the translation models from the literature are based on a sequence-to-sequence architecture [91]. Nevertheless, since a logic formula can be represented by a tree-like structure, it might be better to opt for a sequence-to-tree approach [64], or for other *constrained decoding* techniques (see, for instance, [10]).

Since the English-to-LTL mapping can be viewed as a language translation problem, it makes sense to think of training a neural network model to perform such a task. The best candidate framework for doing that seems to be, to date, the previously-discussed OpenNMT. Unfortunately, a major problem has to be solved in order to actually pursue this approach, that is, the lack of a large training dataset in which English utterances are paired with the corresponding LTL formulas. The University of Kansas provides an online repository of temporal logic formulas that represent the content of technical specifications [5]. However, just about a hundred of English-to-LTL pairings are available. LTLStore[3] is an online repository of around one thousand LTL formulas, collected from previous studies. Although they lack an English counterpart, they may still be useful as *monolingual* data [85].

Various approaches can be followed to address the problem of the scarcity of training data. We would like to outline three of them.

The first, most trivial one consists of randomly generating a large set of LTL formulas by means of a suitable grammar. Then, rules may be derived to translate each formula into an English utterance. Also, some post-processing techniques can be applied to make the resulting text more realistic, such as replacing proposition letters by a set of words that represent them (e.g., `button_pressed` might be mapped to *"the button is pressed"*). Of course, the generated sentences would still make use of a very constrained kind of English. Nevertheless, such an artificial training dataset may be used as a starting point to determine which are the most promising translation techniques.

The second approach consists of finding a kind of "bridging dataset", i.e., a tranining dataset that maps each English utterance in a formal representation that, in turn, can be more or less easily converted into LTL by means of hardcoded rules. As an example, it may

[2] Additional details can be found on OpenNMT website: `http://opennmt.net/`.
[3] LTLStore project website: `https://gitlab.lrz.de/i7/ltlstore`.

be worth investigating the data used in semantic parsing competitions, such as, for instance, SemEval[4], that over the years included some tracks concerned with the evaluation of temporal content in natural language phrases (this is the case with TempEval). Other possibly useful resources are the TimeML [80] annotated corpora TimeBank [82] and AQUAINT [2], and the ACE (Automatic Content Extraction) [1] and WikiWars [66] corpora, that all provide texts manually labeled with events, temporal expressions, and relationships.

A third recent research line has managed to train both neural machine and statistical machine translation systems using monolingual corpora only [16, 17, 51, 52]. Although accuracy results are still far from those guaranteed by state of the art, bilingual data-based models, such approaches are worth some further investigation.

## 4.3 LTL synthesis through evolutionary computation

Both LTL synthesis via semantic parsing and LTL synthesis via machine translation follow consolidated paths in the field. Now, we work out an alternative path that makes use of evolutionary computation.

*Evolutionary Algorithms* (EAs) are adaptive meta-heuristic search algorithms, inspired by the process of natural selection, biology, and genetics. Unlike blind random search, they are capable of exploiting historical information to direct the search into the most promising regions of the search space and, in order to achieve that, their basic characteristics are designed to mimic the processes that, in natural systems, lead to adaptive evolution [31].

In nature, a population of individuals tends to evolve, in order to adapt to the environment; in EAs, each individual represents a possible solution for the optimization problem, and its degree of "adaptation" to the problem is evaluated by means of a *fitness* function, which can be single- or multi-objective. In the first case, at the end of the optimization process, the single, best solution is returned. In the second case, the algorithm searches for multiple optimal solutions in parallel, that are returned in the form of a so-called *Pareto-front*. Multi-objective approaches are particularly suitable for multi-objective optimization. In both cases, the elements of the population tend to iteratively evolve toward better solutions, going through a series of generations in which the *crossover* (hybridization of two solutions to generate a solution offspring) and *mutation* (random changes performed to a solution) operators are applied to selected individuals. At each generation, the individuals that are considered best by the fitness function are given a higher probability of being transmitted to the following one.

The English-to-LTL mapping problem may be thought of as an optimization one (that in turn can be solved via evolutionary computation): given an input utterance, the task is that of finding the temporal formula that best matches the content of the text.

In the following, the main issues concerning the exploitation of an evolutionary algorithm for the English-to-LTL mapping problem are discussed. They can be summarized as follows: (*i*) how the single solutions are represented; (*ii*) how the population is initialized; (*iii*) which evolutionary operators (crossover, mutation) are employed; (*iv*) which fitness function is used.

### 4.3.1 Solution representation and population

Each instance in the population represents an LTL formula, which may be conveniently coded by means of a tree-based data structure. As for the initialization of the population, one may proceed as follows. In the first step, all proposition letters are extracted from the

[4] SemEval 2019 website: `http://alt.qcri.org/semeval2019/`.

given utterance. This can be done by means of a suitable predicate extraction process, that can rely on available tools, such as, for instance, PredPatt [7, 97], or on custom-tailored solutions based on *Semantic Role Labeling* or *Open Information Extraction*, making use of natural language processing frameworks such as Stanford CoreNLP [65], the SpaCy-based [12] AllenNLP [39], and TextPro [78]. Once all proposition letters have been extracted, candidate LTL formulas may be randomly generated, ensuring both to use all of the propositional letters (since the goal is that of fully encoding what is happening in the utterance), and to respect LTL syntactical correctness constraints.

### 4.3.2 Evolutionary operators

The representation of LTL formulas by means of tree-like structures makes it quite straightforward to implement the crossover and mutation operators. Both of them can, indeed, modify or generate a solution by acting on subtrees, e.g., by adding, removing, raising, or swapping them, always making sure that each proposition letter is appearing at least one time in the resulting formula.

### 4.3.3 Fitness function

In order to establish how good a given formula is, it is necessary to determine how well it captures the pieces of information contained in the utterance. This is the most difficult step, since a properly designed fitness function should be capable of capturing a semantic parallelism between the logical formula and the natural language text, with an accuracy well beyond the one that can be provided, for instance, by commonly used techniques such as Latent Dirichlet Allocation (LDA) [21].

A possible solution can be that of extracting a suitable model from the text, capable of capturing all pieces of information that are relevant for the evaluation of the LTL formula. In order to do that, an initial step, as already discussed in the case of population initialization, can be the extraction of all proposition letters from the utterance. Then, these letters can be arranged in a Kripke structure [50], that is, a model that keeps track of how and when they hold. To generate such a model, the best approach seems to be the one pursued in [94], where a deep learning solution is developed that is capable of assigning time intervals to the verbs in a given text[5] (a similar work is discussed in [58]). Once such time intervals have been determined, the structure can be derived in a fairly straightforward way and, based on it, the (still non trivial) question now becomes "how much and how well the given LTL formula is describing what is happening in the Kripke model?". Finally, a second objective must be considered, that is, generating easy to read, and thus understandable, formulas. Such a condition can be enforced by using fairly natural metrics such as the nesting degree of the formula, the number of Boolean and/or temporal operators employed, and so on.

## 4.4 LTL synthesis through event identification and ordering

Finally, the English-to-LTL translation problem can be addressed with an approach which turns out to be somehow more hardwired than the ones discussed before, but, nevertheless, makes use of some advanced machine learning methodologies.

The idea is that of first determining the syntactic structure of the utterance and the events it refers to, then identifying the main agent and action of each of them, and finally establishing an ordering over them. Based on such pieces of information, it should be easier

[5] A working implementation can be found at: `https://hub.docker.com/r/sidvash/temporal`.

to derive a set of rules for the generation of the corresponding LTL formula, at least up to a fixed, maximum degree of nesting. Of course, the translated utterances could be fed in as training data to any other solution developed according to the approaches described in Section 4.1 and Section 4.2. In the following, we give a more detailed account of these phases.

As for the syntactic analysis, several frameworks can be found in the literature, most notably SpaCy [12], AllenNLP [39], which is based on SpaCy, Stanford CoreNLP [65], NLTK [63], and TextPro [78]. All of them allow one to perform the following syntactic analysis tasks, that are necessary to carry out the translation:

- *part of speech tagging*, by which components such nouns, verbs, and adjectives are identified;
- *construction of the dependency tree*, that establishes the relationships between "head" words and words which modify such heads;
- *coreference resolution*, that is, the identification of all the expressions that refer to the same entity in the text;
- *recognition of noun/prepositional/verb phrases*, that may help in the later identification of proposition letters (for instance, *"the big blue ocean"* should be considered as a single entity).

Then, proposition letters are to be extracted, relying on tools like, e.g., PredPatt [7, 97], or on custom solutions, based on the previously-introduced frameworks, making use of either:

- *semantic role labeling*, that is, the process that assigns labels to words or phrases in a sentence that indicate their semantic role, such as, for instance, agent or action, or
- *open information extraction*, that generates a structured, machine-readable representation of the relevant data in the text, typically in the form of n-ary propositions.

The next step is that of correlating proposition letters by making use of Boolean connectives and temporal modalities. To this end, other than relying on the dependency tree and on specific keyword extraction, it is possible to consider:

- *temporal expression recognition and normalization*, that is, the task of identifying sequences of tokens in a given text that denote a point in time, a duration, or a frequency, and of interpreting them. A lot of work has been done in this respect (see, e.g., [27, 54, 61, 98]), including a number tools that can be used for the task: PTime [8], SUTIME [23], UWTIME [56], ClearTK's TimeML module [4], HeidelTime [88], cogCompTime [76], SynTime [99], and TextPro's TimePro module [78]. In addition, AllenNLP and Stanford coreNLP contain some modules that can be used as well. The typical performance is higher than 80% precision and recall on benchmark data such as the Platinum dataset from the TempEval3 workshop [93].
- *temporal relation identification*, i.e., the task of determining a partial or total ordering of events (e.g., identified by verbs) happening in an utterance. While this can still be considered an open problem [26], a lot of work has already been done in the literature [20, 37, 60, 67, 73, 74, 75], also driven by important domain requirements, such as those in the medical field [34, 55, 92]. Some tools are available that either (*i*) encode temporal relationships by means of graphs, as it happens with CATENA [68], ClearTK's TimeML module [4], TextPro's TempRelPro module [78], cogCompTime [76], CAEVO [3], and TIPsem [14, 62], or, perhaps more naturally, (*ii*) assign a time interval to each event, like in [94] (we already refer to it in Section 4.3) and [58].

Other resources that can be successfully exploited are WordNet [33], a well-known lexical database for the English language that groups words into sets of synonyms and records a number of relations among them, VerbNet [84], which is the largest on-line verb lexicon currently available for English, PPDB [38], a database containing hundreds of million

paraphrase pairs, which capture many meaning-preserving syntactic transformations, and VerbOcean, a repository that encompasses the semantic relationships that in English language typically hold between verbs, including a *happens-before* relation [24].

## 5 An experimental evaluation of existing tools

This section is devoted to a critical experimental evaluation of some of the tools that have been previously discussed. For such a purpose, two datasets have been considered.

The first one is the *US Government's Accident Injuries* collection[6], that contains information on all accidents, injuries, and illnesses reported by mine operators and contractors beginning with 1/1/2000. Each instance corresponds to an accident and is characterized, among all other pieces of information, by a free-text narrative describing the event.

The second dataset has been provided by *Main Roads Western Australia*[7], and it includes information on fatal, serious, and medical car crashes recorded between 2013 and 2017 in Western Australia. As it happens with the previous collection, each instance is characterized by a narrative describing how the incident occurred.

The two datasets are representative of two different kinds of narratives: in the mining case, the textual description is typically short, and involves a single participant. On the contrary, the car crash narratives are longer and more articulated, may involve several participants, and tend to thoroughly describe the event.

For reference, the following representative utterance from the mining accidents dataset is taken into account:

> *Employee was cleaning up at the Primary Crusher with the Dingo skid steer. The employee slipped and fell while operating the skid steer and the machine pinned him against the cement retaining wall.*

As for the car crashes dataset, the following representative instance is considered:

> *B drove east on Gibbins Road, Coolup. It appears B has stopped at the intersection of Maryfield Road and allowed a vehicle to pass before proceeding into the intersection. B2 was driving north on Maryfield Road, has observed B's vehicle at the intersection noticing that B appeared to be looking straight ahead. As B has proceeded into the intersection, B2 has applied the brakes on his vehicle, however he was unable to avoid B's vehicle and has 't-boned' it, colliding with the drivers side of B's vehicle. B has died at the scene from injuries sustained. A preliminary test conducted on B2 returned a negative reading.*

The remainder of this section considers two main tasks that, according to Section 4.4, have to be carried out in order to synthesize an LTL formula that correctly formalizes an utterance, that is, the extraction of the proposition letter candidates, and the identification of the temporal relationships among them.

Concerning the extraction of proposition letter, we may identify two distinct steps: first, *coreference resolution*, by which all references to the same entity are identified and normalized[8]; second, the *extraction of predicates*, i.e., tuples composed of an agent, the action that is being carried out, and possible other arguments.

---

[6] `https://catalog.data.gov/dataset/accident-injuries`

[7] `https://www.mainroads.wa.gov.au/Pages/default.aspx`

[8] Observe that coreference resolution implies entity identification which, nevertheless, is a much easier task, that in general can be reliably solved by investigating the dependency tree and the PoS tags of the given utterance.

**Table 1** Predicates extracted by PredPatt on the mining accidents dataset instance. Results have been reworked for the ease of reading.

| Agent | Action |
|---|---|
| Employee | was cleaning up |
| Employee | slipped |
| Employee | fell |
| the Dingo skid | steer |
| the Dingo skid | pinned Employee |

As for coreference resolution, AllenNLP [39] relies on the state-of-the-art algorithm presented in [57]. Tested on the mining accidents dataset instance, it identifies two entities with coreferences, highlighted in red and blue colours:

> ***Employee*** *was cleaning up at the Primary Crusher with* ***the Dingo skid steer****.* ***The employee*** *slipped and fell while operating* ***the skid steer*** *and the machine pinned* ***him*** *against the cement retaining wall.*

Although the result is fairly accurate, the algorithm fails to detect the *the machine* reference to the blue entity. Turning to a more challenging example, this is the result of its execution on the second text[9]:

> ***B*** *drove east on Gibbins Road,* ***Coolup****. It appears* ***B*** *has stopped at the intersection of Maryfield Road and allowed a vehicle to pass before proceeding into the intersection.* ***B2*** *was driving north on Maryfield Road, has observed* ***B's*** *vehicle at the intersection noticing that* ***B*** *appeared to be looking straight ahead. As* ***B*** *has proceeded into the intersection,* ***B2*** *has applied the brakes on* ***his*** *vehicle, however* ***he*** *was unable to avoid* ***B's*** *vehicle and has 't-boned' it, colliding with the drivers side of* ***B's*** *vehicle.* ***B*** *has died at the scene from injuries sustained. A preliminary test conducted on* ***B2*** *returned a negative reading.*

Focusing on just two entities, it is immediately clear that the algorithm has erroneously associated *Coolup* with *B2*. Even worse, judging from the result it seems that *B2* has been applying the brakes on *B*'s vehicle, and that *B* was unable to avoid himself. These two errors are enough to totally misinterpret the semantic content of the utterance.

Once the entities have been found and normalized, it is possible to look for the candidate proposition letter. In this respect, we investigated the use of PredPatt [7, 97], since it is still a major component in many recent, state-of-the-art NLP tools (see, for instance, [94]). To do that, we relied on two versions of the chosen instances in which coreference resolution has been already correctly solved by hand.

The predicates extracted by PredPatt on the mining data utterance are reported in Table 1. As it can be seen, the algorithm outputs 5 candidates. Again, with the exception of the fourth candidate, the result is quite natural and intuitive, except perhaps for the fact that the fourth and fifth candidates refer to the *the Dingo skid* entity instead of *the Dingo skid steer*, considering *steer* as a verb.

[9] Note that the coreference resolution process extracted more than two entities. However, for the sake of clarity, we concentrate here on the two main ones, that are, *B* and *B2*.

**Table 2** Predicates extracted by PredPatt on the car crashes dataset instance. Results have been reworked for the ease of reading.

| Agent | Action |
|---|---|
| B | drove east |
| It | appears |
| A vehicle | allowed to pass |
| [UNK] | was driving north |
| B | appeared to be looking straight ahead |
| B | has proceeded into the intersection |
| B2 | has applied the brakes on B2's vehicle |
| [UNK] | avoid B's vehicle |
| [UNK] | 't-boned' it |
| [UNK] | has died at the scene from injuries |
| A preliminary test | returned negative reading |

Table 2 reports the predicates extracted by PredPatt algorithm on the car crash dataset instance. Here, there are some spurious candidates as, for example, the one with *It* as the main agent. Moreover, for several results, the agent is not clear (*UNK*), or simply wrong (*A vehicle allowed to pass*). Finally, some predicates are entirely missed, like in the case of *B has stopped at the intersection*.

Finally, let us consider the task of extracting the temporal relationships that hold between each pair of predicates. Our original intention was that of evaluating cogCompTime [76] as the tool for generating an ordering graph over the set of events, and the approach presented in [94] as the tool for assigning time intervals to events, since they are two recently proposed solutions. Unfortunately, since various weeks cogCompTime is experiencing some server problems (`http://groupspaceuiuc.com/temporal/#`), that do not allow us to experiment with it. Thus, in the following, just the time interval approach presented in [94] is evaluated with respect to the two selected utterances.

Figure 1 shows its results when applied on the mining dataset instance. As it can be seen, the two occurrences of *steer* are still erroneously recognized as actions. Moreover, while the interval assigned to *operating* seems to be an appropriate choice, the tool fails in identifying its overlap with the interval related to *cleaning*. Furthermore, the events *fell* and *slipped* appear to be swapped.

Considering the car crashes instance, as depicted in Figure 2, the outcomes are unfortunately quite poor, and do not seem to follow any reasonable pattern.

At this point, by looking at the underwhelming results obtained by the tools we tested, we decided not to proceed with the LTL formula synthesis task any further.

## 6 Conclusions

Linear-time temporal logic (LTL) is a logical formalism which is commonly adopted in formal verification, in particular in consistency and model checking, and it can be a very useful tool for capturing the temporal content of natural language utterances. Such capabilities can be exploited in many important applications, including disambiguation of technical requirements, log analysis, and automated reasoning over temporal data.

Unfortunately, as it emerges from our comprehensive analysis of the literature and of existing tools, a general enough solution, that is capable of translating free, natural English texts into unbounded, general LTL formulas is still missing.

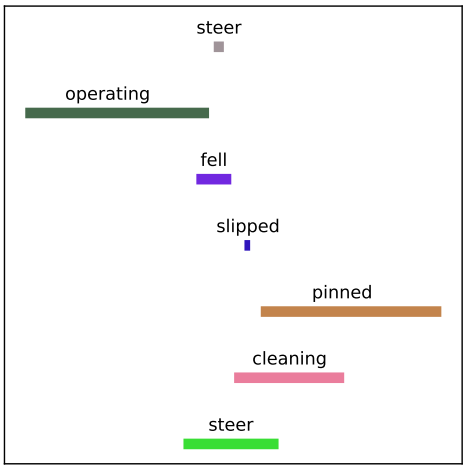


**Figure 1** Events timeline in the mining dataset instance.

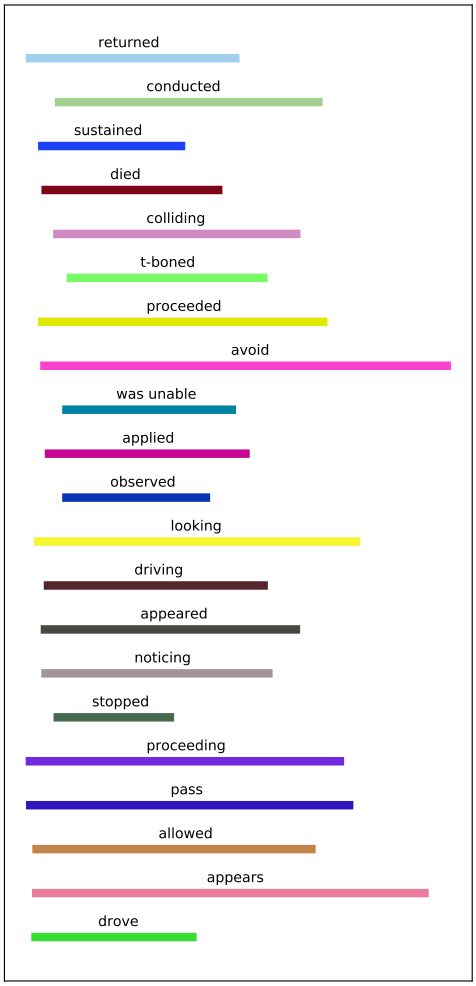


**Figure 2** Events timeline in the car crashes dataset instance.

In this paper, we provided a comprehensive review of the state of the art, as defined by the relevant literature; moreover, an empirical evaluation has been conducted on some of the currently available NLP tools, based on two real world instances. Results show that further research work is needed on methods and tools for various crucial tasks, such as coreference resolution, predicate extraction, and temporal relation identification, in order to achieve a performance which is good enough to allow for the synthesis of LTL formulas from unconstrained, natural language texts. We highlighted some possible research directions that can be followed to tackle such a complex problem.

## References


1 ACE project website. Accessed: April 2019. URL: https://www.ldc.upenn.edu/collaborations/past-projects/ace.

2 AQUAINT corpus website. Accessed: April 2019. URL: https://catalog.ldc.upenn.edu/LDC2002T31.

3 CAEVO corpus website. Accessed: April 2019. URL: https://www.usna.edu/Users/cs/nchamber/caevo/.

4 ClearTK GitHub page. Accessed: April 2019. URL: https://cleartk.github.io/cleartk/.

5 Kansas State University CIS Department, Laboratory for Specification, Analysis, and Transformation of Software (SAnToS Laboratory), Property Pattern Mappings for LTL. Accessed: April 2019. URL: http://patterns.projects.cs.ksu.edu/.

6 Keras framework website. Accessed: April 2019. URL: https://keras.io.

7 PredPatt GitHub page. Accessed: April 2019. URL: https://github.com/hltcoe/PredPatt.

8 PTime website. Accessed: April 2019. URL: http://ws.nju.edu.cn/ptime/.

9 PyTorch GitHub page. Accessed: April 2019. URL: https://github.com/pytorch.

10 Semantic parsing with AllenNLP. Accessed: April 2019. URL: https://github.com/allenai/allennlp/blob/master/tutorials/getting_started/semantic_parsing.md.

11 Sippycup GitHub page. Accessed: April 2019. URL: https://github.com/wcmac/sippycup.

12 SpaCy website. Accessed: April 2019. URL: https://spacy.io/.

13 TensorFlow framework website. Accessed: April 2019. URL: https://www.tensorflow.org/.

14 TIPSem GitHub page. Accessed: April 2019. URL: https://github.com/hllorens/otip.

15 WASP semantic parser website. Accessed: April 2019. URL: http://www.cs.utexas.edu/users/ml/wasp/.

16 Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Unsupervised Statistical Machine Translation. *CoRR*, abs/1809.01272, 2018. arXiv:1809.01272.

17 Mikel Artetxe, Gorka Labaka, and Eneko Agirre. An Effective Approach to Unsupervised Machine Translation. *CoRR*, abs/1902.01313, 2019. arXiv:1902.01313.

18 Yoav Artzi. Cornell SPF: Cornell semantic parsing framework. *arXiv preprint*, 2013. arXiv:1311.3011.

19 Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1533–1544, 2013. URL: http://aclweb.org/anthology/D/D13/D13-1160.pdf.

20 Steven Bethard, Timothy A. Miller, Dmitriy Dligach, Chen Lin, and Guergana Savova. Neural Temporal Relation Extraction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 746–751, 2017. URL: https://aclanthology.info/papers/E17-2118/e17-2118.

21 David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. URL: http://jmlr.org/papers/v3/blei03a.html.

22 Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001. doi:10.1023/A:1010933404324.

**23** Angel X. Chang and Christopher D. Manning. SUTime: A library for recognizing and normalizing time expressions. In *Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC*, pages 3735–3740, 2012. URL: `http://www.lrec-conf.org/proceedings/lrec2012/summaries/284.html`.

**24** Timothy Chklovski and Patrick Pantel. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP*, pages 33–40, 2004. URL: `http://www.aclweb.org/anthology/W04-3205`.

**25** Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995. `doi:10.1007/BF00994018`.

**26** Leon Derczynski. *Automatically Ordering Events and Times in Text*, volume 677 of *Studies in Computational Intelligence*. Springer, 2017. `doi:10.1007/978-3-319-47241-6`.

**27** Wentao Ding, Guanji Gao, Linfeng Shi, and Yuzhong Qu. A Pattern-based Approach to Recognizing Time Expressions. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

**28** Arianna D'Ulizia, Fernando Ferri, and Patrizia Grifoni. A survey of grammatical inference methods for natural language learning. *Artificial Intelligence Review*, 36(1):1–27, 2011. `doi:10.1007/s10462-010-9199-1`.

**29** Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. Patterns in Property Specifications for Finite-State Verification. In *Proceedings of the 1999 International Conference on Software Engineering, ICSE*, pages 411–420, 1999. `doi:10.1145/302405.302672`.

**30** Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul W. Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *2009 IEEE International Conference on Robotics and Automation, ICRA*, pages 4163–4168, 2009. `doi:10.1109/ROBOT.2009.5152776`.

**31** A. E. Eiben and James E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, 2015. `doi:10.1007/978-3-662-44874-8`.

**32** Alessandro Fantechi, Stefania Gnesi, Gioia Ristori, Michele Carenini, Massimo Vanocchi, and Paolo Moreschini. Assisting Requirement Formalization by Means of Natural Language Translation. *Formal Methods in System Design*, 4(3):243–263, 1994. `doi:10.1007/BF01384048`.

**33** Christiane Fellbaum. WordNet. *The Encyclopedia of Applied Linguistics*, 2012.

**34** Olivier Ferret, Xavier Tannier, Aurélie Névéol, and Julien Tourille. Temporal information extraction from clinical text. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL*, pages 739–745, 2017. URL: `https://aclanthology.info/papers/E17-2117/e17-2117`.

**35** Mikel L. Forcada, Mireia Ginestí-Rosell, Jacob Nordfalk, Jim O'Regan, Sergio Ortiz-Rojas, Juan Antonio Pérez-Ortiz, Felipe Sánchez-Martínez, Gema Ramírez-Sánchez, and Francis M. Tyers. Apertium: a free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144, 2011. `doi:10.1007/s10590-011-9090-0`.

**36** Ryan Gabbard, Seth Kulick, and Mitchell P. Marcus. Fully Parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 2006. URL: `http://aclweb.org/anthology/N/N06/N06-1024.pdf`.

**37** Diana Galvan, Naoaki Okazaki, Koji Matsuda, and Kentaro Inui. Investigating the Challenges of Temporal Relation Extraction from Clinical Text. In *Proceedings of the 9th International Workshop on Health Text Mining and Information Analysis 31, 2018*, pages 55–64, 2018. URL: `https://aclanthology.info/papers/W18-5607/w18-5607`.

**38** Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: the paraphrase database. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 758–764, 2013. URL: `http://aclweb.org/anthology/N/N13/N13-1092.pdf`.

**39** Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640, 2018. `arXiv:1803.07640`.

40 Rob Gerth, Doron A. Peled, Moshe Y. Vardi, and Pierre Wolper. Simple on-the-fly automatic verification of linear temporal logic. In *Proceedings of the 15th International Symposium on Protocol Specification*, pages 3–18, 1995.

41 Shalini Ghosh, Daniel Elenius, Wenchao Li, Patrick Lincoln, Natarajan Shankar, and Wilfried Steiner. ARSENAL: automatic requirements specification extraction from natural language. In *Proceedings of the 8th NASA Formal Methods International Symposium*, pages 41–46, 2016. doi:10.1007/978-3-319-40648-0_4.

42 Volker Gruhn and Ralf Laue. Patterns for Timed Property Specifications. *Electronic Notes in Theoretical Computer Science*, 153(2):117–133, 2006. doi:10.1016/j.entcs.2005.10.035.

43 Christopher B. Harris and Ian G. Harris. Generating formal hardware verification properties from Natural Language documentation. In *Proceedings of the 9th IEEE International Conference on Semantic Computing, ICSC*, pages 49–56, 2015. doi:10.1109/ICOSC.2015.7050777.

44 Christopher B. Harris and Ian G. Harris. GLAsT: Learning formal grammars to translate natural language specifications into hardware assertions. In *Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition, DATE*, pages 966–971, 2016. URL: http://ieeexplore.ieee.org/document/7459447/.

45 Claude Jard and Thierry Jéron. On-Line Model Checking for Finite Linear Temporal Logic Specifications. In *Proceedings of the International Conference on Computer Aided Verification*, pages 189–196, 1989. doi:10.1007/3-540-52148-8_16.

46 Hans Kamp. A theory of truth and semantic representation. *Formal semantics-the essential readings*, pages 189–222, 1981.

47 Rohit J. Kate and Raymond J. Mooney. Using String-Kernels for Learning Semantic Parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics ACL*, 2006. URL: http://aclweb.org/anthology/P06-1115.

48 Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 67–72, 2017. doi:10.18653/v1/P17-4012.

49 Sascha Konrad and Betty H. C. Cheng. Real-time specification patterns. In *Proceedings of the 27th International Conference on Software Engineering, ICSE*, pages 372–381, 2005. doi:10.1145/1062455.1062526.

50 Saul A. Kripke. Semantical Considerations on Modal Logic. *Acta Philosophica Fennica*, 16(1963):83–94, 1963.

51 Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised Machine Translation Using Monolingual Corpora Only. In *Proceedings of the 6th International Conference on Learning Representations, ICLR*, 2018. URL: https://openreview.net/forum?id=rkYTTf-AZ.

52 Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-Based & Neural Unsupervised Machine Translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, 2018. URL: https://aclanthology.info/papers/D18-1549/d18-1549.

53 Pat Langley and Sean Stromsten. Learning Context-Free Grammars with a Simplicity Bias. In *Proceedings of the 11th European Conference on Machine Learning, ECML*, pages 220–228, 2000. doi:10.1007/3-540-45164-1_23.

54 Egoitz Laparra, Dongfang Xu, and Steven Bethard. From Characters to Time Intervals: New Paradigms for Evaluation and Neural Parsing of Time Normalizations. *Transactions of the Association of Computational Linguistics*, 6:343–356, 2018. URL: https://transacl.org/ojs/index.php/tacl/article/view/1318.

55 Hee-Jin Lee, Yaoyun Zhang, Min Jiang, Jun Xu, Cui Tao, and Hua Xu. Identifying direct temporal relations between time and events from clinical notes. *BMC Medical Informatics and Decision Making*, 18(S-2):23–34, 2018. doi:10.1186/s12911-018-0627-5.

**56** Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. Context-dependent Semantic Parsing for Time Expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*, pages 1437–1447, 2014. URL: `http://aclweb.org/anthology/P/P14/P14-1135.pdf`.

**57** Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end Neural Coreference Resolution. *CoRR*, abs/1707.07045, 2017. `arXiv:1707.07045`.

**58** Artuur Leeuwenberg and Marie-Francine Moens. Temporal Information Extraction by Predicting Relative Time-lines. *CoRR*, abs/1808.09401, 2018. `arXiv:1808.09401`.

**59** Constantine Lignos, Vasumathi Raman, Cameron Finucane, Mitchell P. Marcus, and Hadas Kress-Gazit. Provably correct reactive control from natural language. *Autonomous Robots*, 38(1):89–105, 2015. `doi:10.1007/s10514-014-9418-8`.

**60** Zhengzhong Liu, Teruko Mitamura, and Eduard H. Hovy. Graph-Based Decoding for Event Sequencing and Coreference Resolution. *CoRR*, abs/1806.05099, 2018. `arXiv:1806.05099`.

**61** Hector Llorens, Leon Derczynski, Robert J. Gaizauskas, and Estela Saquete. TIMEN: an open temporal expression normalisation resource. In *Proceedings of the 8th International Conference on Language Resources and Evaluation, LREC*, pages 3044–3051, 2012. URL: `http://www.lrec-conf.org/proceedings/lrec2012/summaries/128.html`.

**62** Hector Llorens, Estela Saquete, and Borja 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. URL: `http://aclweb.org/anthology/S/S10/S10-1063.pdf`.

**63** Edward Loper and Steven Bird. NLTK: the natural language toolkit. *CoRR*, cs.CL/0205028, 2002. `arXiv:cs.CL/0205028`.

**64** Weicheng Ma, Zhaoheng Ni, Kai Cao, Xiang Li, and Sang Chin. Seq2tree: A tree-structured extension of LSTM network, 2017. URL: `https://openreview.net/forum?id=HJ0WtefAW`.

**65** Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*, pages 55–60, 2014. URL: `http://aclweb.org/anthology/P/P14/P14-5010.pdf`.

**66** Pawel P. Mazur and Robert Dale. WikiWars: A new corpus for research on temporal expressions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 913–922, 2010. URL: `http://www.aclweb.org/anthology/D10-1089`.

**67** Paramita Mirza. Extracting Temporal and Causal Relations between Events. *CoRR*, abs/1604.08120, 2016. `arXiv:1604.08120`.

**68** Paramita Mirza and Sara Tonelli. CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In *Proceedings of the 26th International Conference on Computational Linguistics, COLING*, pages 64–75, 2016. URL: `http://aclweb.org/anthology/C/C16/C16-1007.pdf`.

**69** Smaranda Muresan, Tudor Muresan, and Judith L Klavans. Lexicalized Well-Founded Grammars: Learnability and Merging. *Technical Report, Columbia University*, 2005.

**70** Daniel Neider and Ivan Gavran. Learning Linear Temporal Properties. In *Proceedings of the 2018 Formal Methods in Computer Aided Design Conference, FMCAD*, pages 1–10, 2018.

**71** Rani Nelken and Nissim Francez. Automatic Translation of Natural Language System Specifications into Temporal Logic. In *Proceedings of the 8th International Conference on Computer Aided Verification, CAV*, pages 360–371, 1996. `doi:10.1007/3-540-61474-5_83`.

**72** Allen P. Nikora and Galen Balcom. Automated Identification of LTL Patterns in Natural Language Requirements. In *Proceedings of the 20th International Symposium on Software Reliability Engineering, ISSRE*, pages 185–194, 2009. `doi:10.1109/ISSRE.2009.15`.

**73** Qiang Ning, Zhili Feng, and Dan Roth. A Structured Learning Approach to Temporal Relation Extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1027–1037, 2017. URL: `https://aclanthology.info/papers/D17-1108/d17-1108`.

**74** Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. Improving Temporal Relation Extraction with a Globally Acquired Statistical Resource. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 841–851, 2018. URL: `https://aclanthology.info/papers/N18-1077/n18-1077`.

**75** Qiang Ning, Zhongzhi Yu, Chuchu Fan, and Dan Roth. Exploiting Partially Annotated Data for Temporal Relation Extraction. *CoRR*, abs/1804.08420, 2018. `arXiv:1804.08420`.

**76** Qiang Ning, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. CogCompTime: A tool for understanding time in natural language. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 72–77, 2018. URL: `https://aclanthology.info/papers/D18-2013/d18-2013`.

**77** Hari Mohan Pandey, Ankit Chaudhary, and Deepti Mehrotra. Grammar induction using bit masking oriented genetic algorithm and comparative analysis. *Applied Soft Computing*, 38:453–468, 2016. `doi:10.1016/j.asoc.2015.09.044`.

**78** Emanuele Pianta, Christian Girardi, and Roberto Zanoli. The TextPro Tool Suite. In *Proceedings of the 6th International Conference on Language Resources and Evaluation, LREC*, 2008. URL: `http://www.lrec-conf.org/proceedings/lrec2008/summaries/645.html`.

**79** Amir Pnueli. The Temporal Logic of Programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57, 1977. `doi:10.1109/SFCS.1977.32`.

**80** James Pustejovsky, José M. Castaño, Robert Ingria, Roser Saurí, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. TimeML: Robust specification of event and temporal expressions in text. *New Directions in Question Answering*, pages 28–34, 2003.

**81** Aarne Ranta. Translating Between Language and Logic: What is Easy and What is Difficult. In *Proceedings of the International Conference on Automated Deduction*, pages 5–25. Springer, 2011.

**82** Nils Reimers, Nazanin Dehghani, and Iryna Gurevych. Temporal Anchoring of Events for the TimeBank Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*, 2016. URL: `http://aclweb.org/anthology/P/P16/P16-1207.pdf`.

**83** Tainã Santos, Gustavo Carvalho, and Augusto Sampaio. Formal Modelling of Environment Restrictions from Natural-Language Requirements. In *Proceedings of the 21st Brazilian Symposium on Formal Methods: Foundations and Applications, SBMF*, pages 252–270, 2018. `doi:10.1007/978-3-030-03044-5_16`.

**84** Karin Kipper Schuler, Anna Korhonen, and Susan Windisch Brown. VerbNet overview, extensions, mappings and applications. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics on Human Language Technologies*, pages 13–14, 2009. URL: `http://www.aclweb.org/anthology/N09-4007`.

**85** Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving Neural Machine Translation Models with Monolingual Data. *CoRR*, abs/1511.06709, 2015. `arXiv:1511.06709`.

**86** Bradford Starkie. Inferring Attribute Grammars with Structured Data for Natural Language Processing. In *Proceedings of the 6th International Colloquium on Grammatical Inference: Algorithms and Applications, ICGI*, pages 237–248, 2002. `doi:10.1007/3-540-45790-9_19`.

**87** Mark Steedman. *The syntactic process*, volume 24. MIT press Cambridge, MA, 2000.

**88** Jannik Strötgen and Michael Gertz. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324, 2010. URL: `http://aclweb.org/anthology/S/S10/S10-1071.pdf`.

**89** Giancarlo Sturla. *A two-phased approach for natural language parsing into formal logic*. PhD thesis, Massachusetts Institute of Technology, 2017.

**90** Weiyi Sun, Anna Rumshisky, and Özlem Uzuner. Temporal reasoning over clinical text: the state of the art. *Journal of the American Medical Informatics Association*, 20(5):814–819, 2013. `doi:10.1136/amiajnl-2013-001760`.

**91** Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *CoRR*, abs/1409.3215, 2014. `arXiv:1409.3215`.

**92** Julien Tourille. *Extracting Clinical Event Timelines: Temporal Information Extraction and Coreference Resolution in Electronic Health Records. (Création de Chronologies d'Événements Médicaux: Extraction d'Informations Temporelles et Résolution de la Coréférence dans les Dossiers Patients Électroniques)*. PhD thesis, University of Paris-Saclay, France, 2018. URL: `https://tel.archives-ouvertes.fr/tel-01997223`.

**93** Naushad UzZaman, Hector Llorens, James F. Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. TempEval-3: Evaluating events, time expressions, and temporal relations. *CoRR*, abs/1206.5333, 2012. `arXiv:1206.5333`.

**94** SIddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. Fine-Grained Temporal Relation Extraction. *CoRR*, abs/1902.01390, 2019. `arXiv:1902.01390`.

**95** Yorick Wilks and Dann Fass. The preference semantics family. *Computers & Mathematics with Applications*, 23(2-5):205–221, 1992.

**96** Rongjie Yan, Chih-Hong Cheng, and Yesheng Chai. Formal consistency checking over specifications in natural languages. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE*, pages 1677–1682, 2015. URL: `http://dl.acm.org/citation.cfm?id=2757200`.

**97** Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. An Evaluation of PredPatt and Open IE via Stage 1 Semantic Role Labeling. In *Proceedings of the 12th International Conference on Computational Semantics, IWCS*, 2017. URL: `https://aclanthology.info/papers/W17-6944/w17-6944`.

**98** Xiaoshi Zhong and Erik Cambria. Time Expression Recognition Using a Constituent-based Tagging Scheme. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW*, pages 983–992, 2018. `doi:10.1145/3178876.3185997`.

**99** Xiaoshi Zhong, Aixin Sun, and Erik Cambria. Time Expression Analysis and Recognition Using Syntactic Token Types and General Heuristic Rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 420–429, 2017. `doi:10.18653/v1/P17-1039`.

**100** Lukáš Žilka. Temporal logic for man. Master's thesis, Brno University of Technology, 2010.

**101** Szilvia Zvada and Tibor Gyimóthy. Using Decision Trees to Infer Semantic Functions of Attribute Grammars. *Acta Cybernetica*, 15(2):279–304, 2001. URL: `http://www.inf.u-szeged.hu/actacybernetica/edb/vol15n2/Zvada_2001_ActaCybernetica.xml`.