Chrono at SemEval-2018 Task 6: A System for Normalizing Temporal Expressions

Amy L. Olex, Luke G. Maffey, Nicholas Morton, Bridget T. McInnes
Virginia Commonwealth University, Department of Computer Science
Richmond, Virginia, USA
{alolex, maffeyl, mortonn, btmcinnes}@vcu.edu

Abstract

Temporal information extraction is a challenging task. Here we describe Chrono, a hybrid rule-based and machine learning system that identifies temporal expressions in text and normalizes them into the SCATE schema. After minor parsing logic adjustments, Chrono has emerged as the top performing system for SemEval 2018 Task 6: Parsing Time Normalizations.

1 Introduction

Understanding and processing temporal information is vital for navigating life. The human mind processes subtle temporal expressions instantly and effortlessly; however, it is difficult for computers to do the same. Identifying, processing, and utilizing this information requires knowledge and understanding of syntax, semantics, and context to link temporal information to related events and order them on a time-line. SemEval 2018 Task 6 (Laparra et al., 2018) aims to normalize fine-grained temporal information and relationships into the Semantically Compositional Annotations for Temporal Expressions (SCATE) schema developed by (Bethard and Parker, 2016). This scheme aims to improve upon the current TIMEX3/TimeML (Pustejovsky et al., 2003) standard by representing a wide variety of temporal expressions, allowing for events to act as anchors, and using mathematical operations over a time-line to define the semantics of each annotation. To address this challenge, we developed Chrono[1], a hybrid rule-based and machine learning (ML) Python package that normalizes temporal expressions into the SCATE schema.

2 The Chrono System

Our approach to building this hybrid system includes four processing phases: 1) text pre-processing, 2) flagging numeric and temporal tokens, 3) temporal expression identification, and 4) SCATE normalization.

1) Text Pre-processing: Python's Natural Language Toolkit (NLTK) WhitespaceTokenizer and part-of-speech (POS) tagger (Bird and Loper, 2004) process raw text files to identify individual tokens, token spans, and POS tags. Punctuation is not handled at this phase as it is important for identifying correct spans.

2) Flagging Numeric and Temporal Tokens: All numeric tokens are flagged regardless of context. Subsequent phases utilize contextual information to determine if a numeric token is part of a temporal expression. Depending on the task, a rule may remove all or some punctuation, and/or convert tokens to lowercase prior to parsing. In the following, RP and LC denote **R**emoving all **P**unctuation and converting to **L**ower**C**ase, respectively.

Numeric Flagging: Tokens are flagged as numeric if either 1) the token has a POS tag of "CD" (Cardinal Number), or 2) the text can be converted to a numeric expression. Textual representations of numeric expressions are converted to numerics with the Word2Number[2] Python module. A custom method recognizes ordinals from "first" to "thirty-first" and converts them into the associated numerics 1 to 31, respectively. LC normalization is done prior to parsing textual numerics.

Temporal Flagging: Temporal tokens are flagged through rule-based parsing using lists of key words and regular expressions. This phase is more liberal in its identification of a temporal token than the SCATE normalization phase, so

[1] https://github.com/AmyOlex/Chrono

[2] https://github.com/akshaynagpal/w2n

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 97–101
New Orleans, Louisiana, June 5–6, 2018.

it identifies a broader range of potential temporal tokens that are refined in future steps. Tokens may be numeric and temporal simultaneously. Numeric tokens with the characters '$', '#', or '%' are NOT marked as temporal. The following types of tokens are flagged as temporal:

- Formatted date patterns using '/' or '-': mm/dd/yyyy, mm/dd/yy, yyyy/mm/dd, or yy/mm/dd
- Formatted time patterns matching hh:mm:ss
- Sequence of 4 to 8 consecutive digits matching range criteria for 24-hour times or for a year, month, and/or day (e.g. 1998 or 08241998).
- Spelled out month or abbreviation, e.g. "Mar." or "March", are flagged after RP except periods as they are required to retrieve correct spans.
- Days of the week, e.g. "Sat." or "Saturday", are parsed similar to months.
- Temporal words indicating periods of time, e.g. "yesterday" or "decade", are flagged after RP and LC.
- Mentions of AM and PM in any format are flagged after RP except periods.
- The parts of a week, e.g. "weekend" and "weekends", are flagged after RP and LC.
- Seasons of the year are flagged after RP and LC.
- Various parts of a day, e.g. "noon" or "morning", are flagged after RP and LC.
- Time zones are flagged after RP.
- Other temporal words, e.g. "this", "now", "nearly", and others, are flagged after RP and LC.

3) Temporal Expression Identification: A temporal expression is represented by a *temporal phrase*, which we define as two or more consecutive temporal/numeric tokens on the same line, or an isolated temporal token, with some exceptions. If a numeric token contains a '$', '#', or '%', or the text 'million', 'billion', or 'trillion' it is not included in a temporal phrase as these generally refer to non-temporal values. Additionally, isolated numeric tokens are not considered a temporal phrase.

4) SCATE Normalization: Chrono parses each temporal phrase into zero or more SCATE entities, links sub-intervals, and disambiguates the SCATE entities "Period" and "Calendar-Interval" via a machine learning module. Chrono implements 32 types of entities with 5 parent types that have been described by (Bethard and Parker, 2016). Parsing strategies differ depending on the composition of a temporal phrase being parsed. Each temporal phrase is interrogated by all of the following parsing strategies.

Formatted Dates and Times: Formatted dates/times are parsed using regular expressions. To identify which format the date/time is in, Chrono looks for a 2-digit or 4-digit year first, then uses that position for orientation to identify the remaining elements. If a formatted date/time is identified, then the appropriate sub-intervals are linked during element parsing. 4-digit years take precedence over 2-digit years.

Numeric Dates and Times: Header and metadata for Newswire articles frequently have numeric dates listed with no punctuation (e.g. "19980218" codes for "Feb, 18 1998"), and isolated 4-digit year mentions are frequent. After formatted dates and times are parsed, any phrase containing a numeric token is interrogated for a potential date or year mention. If a numeric token is 4-digits it is tested for a year between 1500 and 2050, 6-digit tokens are parsed for 2-digit year/month/day, and 8-digit strings are parsed for a 4-digit year and 2-digit month/day. All elements must be in the proper range, otherwise the token is skipped. Appropriate sub-intervals are linked during element parsing.

24-hour Time: 24-hour times are identified by either the format $hhmmzzz$, where zzz is the time zone, or a 4-digit number that has not been classified as a year. Hour digits must be less than 24 and minutes less than 60. Sub-intervals are linked at this time if existing. Time zones are handled separately and are linked back to the hour entity during the final sub-interval linking step.

Temporal Token Search: The majority of textual temporal entities are identified by looking for specific tokens. Token categories include days of the week, months, parts of a day/week, time zones, and other temporal operators such as "early", "this", "before", etc. Prior to looking for these tokens, text is normalized by RP and LC. Exceptions to RP include searching for day/month abbreviations, such as "Sat." or "Aug.". In these cases periods are not removed because they are part of the SCATE span. Another exception to RP and LC is identifying mentions of AM or PM where periods are kept and text is not converted to lowercase

in order to capture variations like "PM" or "p.m.". Non-temporal mentions of the months or seasons of the year "may", "march", "spring", and "fall" are disambiguated using POS tags, where tokens that refer to a temporal entity generally have a POS tag of "NN" or "NP". Sub-intervals are not linked during token searches.

Text Year: Another special case of parsing temporal tokens are textual representations of years such as "nineteen ninety-seven". The Word2Number Python module was modified to recognize these phrases. Previously, it would add 19 and 97 together instead of returning 1997.

Periods and Calendar-Intervals: The same temporal token can refer to either a SCATE "Period" or "Calendar-Interval". For example, in the phrases "in a week" vs "next week" the token "week" is classified differently. Due to language intricacies it is difficult to define a rule-base system to disambiguate these entities as the classification is contingent on the topic being discussed where phrasing around the entity can be different for each instance. Thus, Period/Calendar-Interval tokens are initially identified by a token search using a defined list of terms, then the identified term and its span are passed to a ML algorithm for classification.

Machine Learning Classification: Four ML algorithms are available in Chrono to differentiate between "Period" and "Calendar-Interval" entities using contextual information. Chrono implements Naive Bayes (NB), Neural Network (NN), Decision Tree (DT), and Support Vector Machine (SVM). Binary feature vectors for all implementations have the following features:

- temporal_self: If the target is flagged as temporal, this feature is set to "1".
- temporal_context: If there is at least one temporal word within a 5-word window up- or down-stream of the target this feature is set to "1".
- numeric: If there is a numeric expression either directly before or after (a 1-word window) the target, this feature is set to "1".
- context: All words within a 5-word window are identified as features and set to "1" if that word is present. Prior to identifying these features all words are normalized with RP and LC. The 5-word window includes crossing sentence boundaries before and after the target word.

We use NLTK with default parameters to implement NB and DT, NN is a simple feed-forward network with three hidden layers implemented using Python's Keras package [3] with epochs set to 5 and batch set to 10, and SVM is implemented using SciKitLearn (Pedregosa et al., 2011) with C set to 0.05 and max iterations set to 3.

Ordinals: Ordinals such as "first" or "3rd" are classified as an "NthFromStart" entity in the SCATE schema. These mentions are identified by normalizing with RP and LC before searching for the ordinal tokens representing the numbers 1-31.

Sub-Interval Linking: After all SCATE entities are identified, all temporal phrases are re-parsed to identify sub-intervals within each phrase. For example, entities in the phrase "August 1998" are parsed by two different methods leaving the sub-interval link vacant. During sub-interval linking, the year "1998" has the "August" entity added as a sub-interval. Sub-interval linking reviews entities from the smallest to the largest, adding missing sub-intervals as needed. This method assumes each temporal phrase contains zero or one of each type of SCATE entity.

Next/Last Parsing: Determining whether an entity is referring to a date in the future, "Next", or past, "Last", depends on context and the document time (doc-time). Next/Last parsing is done after all other parsing, and checks two cases: 1) if a temporal phrase contains a year, no additional annotation is made, and 2) if specific modifier words are present (e.g. "next" or "last") immediately preceding a temporal expression, the modifier is annotated with a sub-interval referencing the following temporal entity. If neither of these cases hold, the year is set as the doc-time year, and the month and day are compared to the full doc-time to determine if it occurs before or after. Note the year assumption is not always valid and more complex, content-based parsing may be required to achieve higher precision. Finally, if a day of the week (e.g. "Saturday") is mentioned, Chrono finds the first preceding verb in the sentence, and if it is past tense the temporal entity is annotated as "Last", otherwise it is annotated as "Next".

3 Results

Training and evaluation of Chrono utilizes the Newswire corpus, consisting of 81 documents, provided by the task organizers. Average preci-

[3] https://github.com/keras-team/keras

sion, recall, and F1-measure of 5-fold cross validation for Track 1 (parsing) are reported in Table 1 (annotations for "Event" and "Modifier" are ignored). Scores for "100% Correct Entity" consider the entity location and all properties (like sub-intervals), and scores for "Correct Span" only consider the entity location.

On average, all ML algorithms perform similarly for the "100% Correct Entity". All versions also obtain a higher F1 score when only considering correct spans versus getting all entity properties correct. This indicates that Chrono correctly identifies the majority of temporal entities, but has trouble parsing some of the properties.

ChronoNN processed the final evaluation dataset, which consisted of 20 previously unseen Newswire articles, and received a F1 of .44. The evaluation dataset contained five articles from BBC that were not represented in the training dataset. Chrono's low performance indicates that it may be over-fit to the the training dataset. This is one downfall of rule-based systems, where new rules need to be developed for each new type of temporal representation. Upon further review we found the submitted version of Chrono had three minor parsing flaws that resulted in unintentional false positives.

1) Formatted dates such as "2013-02-22" were being parsed twice. The first parse specifically looked for a 4-digit year and identified all correct entities, then the second parse looked for a formatted date with a 2-digit year, but didn't check to see if a year had already been found, so returned a 2-digit year with the value "22". This was easily fixed by having the 2-digit year parser check for a 4-digit year flag before proceeding (month and day flags were already implemented).

2) 24-hour time priority was incorrectly placed above 4-digit year. This resulted in any isolated 4-digit year being parsed as a 24-hour time expression rather than a year as originally intended. A simple flip of parsing order resolved this issue.

3) Numeric temporal expressions, such as an isolated 4-digit year, were being parsed as a whole phrase rather than breaking out each token within the phrase. For example, the year in the phrase "Last 1953" was not being identified because it was not in a phrase all by itself. To fix this the parsing function was edited to loop through each token in a phrase (a method that was already implemented in most other parsers and was just overlooked here).

100% Correct Entity			
	P	**R**	**F1**
Chrono NB	.686	.630	.657
Chrono NN	.684	.629	.656
Chrono DT	.687	.632	.658
Chrono SVM	.689	.630	.660
Correct Span			
Chrono NB	.823	.752	.786
Chrono NN	.820	.749	.783
Chrono DT	.822	.751	.785
Chrono SVM	.827	.755	.789
Evaluation Results			
Chrono NN	.46	.42	.44
Post-Evaluation Results			
Chrono NN	.61	.50	.55

Table 1: Chrono results on Newswire corpus for Track 1. All standard errors are $<= 0.03$, and no method performed statistically significantly better than another.

ChronoNN received a Post-Evaluation F1 of .55 for Track 1 after implementing these fixes, which sets ChronoNN as the top performing system for SemEval 2018 Task 6, Track 1.

4 Conclusions and Future Work

Chrono is currently the top performing system for Track 1 of Task 6, but there are still many areas that can be improved. Notably, we plan to implement "Event" and "Between" parsing, as well as refine current strategies as new temporal expressions are identified. Utilizing sentence tokenization instead of relying on new lines could improve phrase identification; however, this did not appear to be a major source of error in parsing the Newswire dataset. Additionally, usability can be improved by moving all parsing rules to separate, customizable files. We also plan to expand ML use to additional disambiguation tasks, and implement an ensemble system utilizing all four ML methods. We aim to extract the temporal phrase parser into a stand-alone system and compare it's performance directly to existing programs like SUTime (Chang and Manning, 2012) and HeidelTime (Strtgen and Gertz, 2010) as it has done a decent job of identifying temporal entities in this challenge. Finally, we will evaluate Chrono's performance on the THYME dataset (Styler IV et al., 2014) using the post-evaluation submission system.

References

Steven Bethard and Jonathan Parker. 2016. A semantically compositional annotation scheme for time normalization. In *Lrec*, volume 2016, pages 3779–3786.

Steven Bird and Edward Loper. 2004. Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

Angel X. Chang and Christopher D. Manning. 2012. Sutime: A library for recognizing and normalizing time expressions. In *Lrec*, volume 2012, pages 3735–3740.

Egoitz Laparra, Dongfang Xu, Steven Bethard, Ahmed S. Elsayed, and Martha Palmer. 2018. Semeval 2018 task 6: Parsing time normalization. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, SemEval '18, New Orleans, LA, USA. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

James Pustejovsky, José M Castano, Robert Ingria, Roser Sauri, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R Radev. 2003. Timeml: Robust specification of event and temporal expressions in text. *New directions in question answering*, 3:28–34.

Jannik Strtgen and Michael Gertz. 2010. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 321–324, Stroudsburg, PA, USA. Association for Computational Linguistics.

William F. Styler IV, Steven Bethard, Sean Finan, Martha Palmer, Sameer Pradhan, Piet C. de Groen, Brad Erickson, Timothy Miller, Chen Lin, and Guergana Savova. 2014. Temporal annotation in the clinical domain. *Transactions of the Association for Computational Linguistics*, 2:143.