Towards Time Inference for Timeline Generation of Historical Wikipedia Articles

Michael Mara
Computer Science Department
Williams College
Williamstown, MA 01267
Email: mtm1@williams.edu

***Abstract*—In this paper, we combine several temporal information extraction techniques to generate timelines from historical Wikipedia articles. We extend HeidelTime to significantly increase precision and recall of temporal expression extraction in our domain, while modestly increasing performance in other domains. We then use a Markov Logic Network acting on time points, and a simple heuristic to assign explicit time intervals to every event in an article, improving performance significantly from prior work.**

I. Introduction

This paper involves temporal information extraction and inference. Information extraction (IE) is the problem of extracting easily processed relational data from unstructured plain text. Research in this area has increased to keep pace with the ever-growing body of information-rich natural language content available on the internet. Temporal information extraction and inference from plain text is a critical and daunting task for natural language processing researchers, but as Ling and Weld [1] note, most information extraction research ignores the temporal information inherently tied to any non-static facts. They go on to acknowledge the literature on temporal expressions in natural language is vast , but mostly focus on specific subproblems, rather than tackling temporal IE as a whole. This paper will attempt to tackle full temporal IE. The motivating application is improving the accuracy of the automated timelines generated by EVOGATE, developed by Chasin and Woodward, which is currently used to create a map and timeline of events from historical wikipedia articles [2]. This involves very accurate temporal expression extraction, and accurately inferring the time intervals of events in plaintext.

II. Related Research

A focal point for temporal information extraction research was the 2007 TempEval challenge [3], which asked participant to focus on three tasks:

Task A	For a restricted set of event terms, identify temporal relations between events and all time expressions appearing in the same sentence. (NOTE: The restricted set of event terms is to be specified by providing a list of root forms. Time expressions will be annotated in the source, in accordance with TIMEX3.)
Task B	For a restricted set of event terms, identify temporal relations between events and the Document Creation Time (DCT). (NOTE: The restricted set of events will be the same as for Task A. DCTs will be explicitly annotated in the source.)
Task C	Identify the temporal relations between contiguous pairs of matrix verbs. (NOTE: matrix verbs, i.e. the main verb of the matrix clause in each sentence, will be explicitly annotated in the source.)

Many approaches have yielded fair success on these tasks or similar tasks, such as Conditional Random Fields, Markov Logic Networks, and rule-based approaches [4]. Yoshikawa et al. [5] succeeded in getting the highest F-score on all three tasks by jointly solving them with Markov Logic Networks. However, solving these tasks is not sufficient for generating a useful timeline from plaintext.

One drastically limiting factor is the relations specified by the TempEval guidelines are a small subset of the Allen interval-algebra relations, which are qualitative relations between two sets of time intervals, such as BEFORE, or OVERLAPS. [6] This is obviously insufficient for timelines, which require exact points in time or time intervals, specified by either a single time or a beginning and end time, for every event displayed.

According to the official Wikipedia style guide, temporal expressions relating to the time of authorship should be avoided except for current events articles [1]. Thus Task B is not highly relevant for our purposes, and our approach this summer disregards document creation time all together. Tasks A and C, however, will certainly be a part of any complete temporal extraction scheme.

[1] http://en.wikipedia.org/

Other promising approaches to temporal information extraction not focused specifically on the TempEval tasks include the hybrid Markov Logic Network and deep semantic parsing approach of Ling and Weld [1] for developing the tightest set of temporal bounds for every event in an article directly implied by the text. One major emphasis of Ling and Weld's work is that they assigned times to events only if the times were directly implied by the text. For many applications, a sufficient criterion is to be reasonably sure of time bounds. Wikipedia in particular, by its nature of being editable by anyone, is subject to factual errors (though not significantly more than standard encyclopedias [7]). Thus direct implication may be a stricter criterion than we would want for developing timelines based on information in Wikipedia.

Another unique approach is aggregating massive amounts of search data from the web combined with shallow parsing to create fuzzy sets representing time intervals for events. [8] This approach is notable for explicitly encoding the uncertainty of time bounds of events.

III. Problem Description

Our problem is to create a program to take a Wikipedia article as input, and output the events in the text along with an explicit temporal interval for each one. This output can then be displayed by the EVOGATE system on a timeline. In order to accomplish this, we will need to extract the sparse temporal information in the text and extrapolate the start and end points for each event.

IV. Temporal Expression Extraction

A. HeidelTime

We use HeidelTime, developed by Strotgen and Gertz[9], to extract all temporal expressions from the wikipedia articles. HeidelTime is an easily extendable rule-based temporal expression extraction system based on regular expressions. Each extracted expression is annotated with its type, value, and the rule it was found with. In the 2010 TempEval-2 challenge[2], HeidelTime was used to tackle task A: Determine the extent of the time expressions in a text as dened by the TimeML timex3 tag. In addition, determine value of the features type and val. The possible values of type are time, date, duration, and set; the value of val is a normalized value as dened by the timex2 and timex3 standard.

Two versions of HeidelTime were tested, one optimized for precision, and one optimized for recall. In the extraction portion, both versions of HeidelTime outperformed their competitors, with an F-score of 86%. In addition, the precision-optimized version had the highest accuracy in assigning the value attribute (85%), as well as a respectable accuracy of (96%) for the type attribute. The publicly available HeidelTime version is a slightly improved version of the precision optimized version submitted for TempEval-2. The results for all three versions can be seen in Table **??**.

[2]http://semeval2.fbk.eu/

HeidelTime uses hand-crafted rules for temporal expression extraction and normalization. The extraction rules are based on regular expressions, but can also take into account part-of-speech (POS) constraints on the tokens extracted. For this reason, HeidelTime is used in a pipeline, with a POS tagger directly before it. Every extraction rule is coupled with a normalization rule, which takes the extracted expression and produces a normalized value as defined by the timex2 and timex3 standard. Extraction rules make use of expression resources, which are simply reusable regular expressions that are categorized and named. For example, the expression resource reSeason (invoked in the extraction rules as "%reSeason") is

$$reSeason = ([Ss]pring|[Ss]ummer|[Ff]all|[Aa]utumn|[Ww]inter).$$

Note that '[]' denotes that exactly one of the expressions inside occurs, and '|' is a boolean OR. Thus "%reSeason" matches the appropriate strings denoting seasons, such as "spring" and "Autumn."

Normalization rules take the extraction rules and create an expression representing the value of the extracted expression. They make use of normalization resources. Normalization resources are just a mapping between extracted expressions and their normalized value. For example, a portion of normSeason is:

$$"Summer","SU"$$

$$"fall","FA"$$

which denote a mapping from "Summer" to "SU" and "fall" to "FA". Assuming "%reYear4Digit" stands for four consecutive digits, a full rule could then be

$$EXTRACTION = "\%reSeason\%reYear4Digit",$$

$$NORMALIZATION = group(2) - normSeason(group(1))$$

$group(x)$ denotes the expression from the xth regular expression resource in the extraction rule. An example of an extraction and normalization of this rule would be "Summer 2011" is extracted and normalized to "2011-SU" in line with the timex2 and timex3 guidelines.

HeidelTime also supports "negative" rules. These rules are identical to others, except the normalization value is simply "REMOVE." This signals to HeidelTime to surpress any positive matches which overlap with an expression captured by the negative rule. An example of how such a rule would be used is to stop "2000 people" from being extracted as the year 2000.

HeidelTime supports underspecified values during the extraction step. An example of an underspecified value is "UNDEF-July." In this case, in a post processing step, HeidelTime uses the last specified time or the document creation time to fill in the missing values. The choice to use the document creation time is specified when running the program. For news articles, using the DCT greatly improves results, but for historical articles, DCT are usually irrelevant.

Chancellorsville		
	Default HeidelTime	Augmented HeidelTime
Precision	0.892857143	0.980392157
Recall	0.980392157	0.980392157
F-score	0.934579439	0.980392157

TABLE I
OUR RULE ADDITIONS IMPROVED PRECISION BY 0.09 WHILE LEAVING RECALL UNCHANGED, RESULTING IN A SIGNIFICANT F-SCORE INCREASE OF 0.05.

Fredericksburg		
	Default HeidelTime	Augmented HeidelTime
Precision	0.933333333	0.948717949
Recall	0.810810811	1.000000000
F-score	0.867768595	0.973684211

TABLE II
IN THIS ARTICLE OUR RULE ADDITIONS RESULTED IN PERFECT RECALL AND A SLIGHT IMPROVEMENT TO PRECISION, RESULTING IN A SIGNIFICANT F-SCORE INCREASE OF 0.11.

B. Improving HeidelTime

We tested HeidelTime in its publicly-available form on our article pool. In our initial tests, we checked the HeidelTime results of two articles, the battle of Chancellorsville and the Battle of Fredericksburg. Our trial runs demonstrated a few shortcomings of the system as-is. It missed some reasonably common time formats (such as "830 a.m."), and sometimes failed to extract durations. More significantly, especially for historical articles detailing troop movements, much like Chasin's approach, it mistakenly extracts "march" when it does not refer to the month, which throws off the accuracy of the values for subsequent underspecified expressions. There were various other minor errors as well.

HeidelTime keeps its code separate from the rules and resources it uses, so additions to the system are easy and involve no compilation. We began by adding a few new rules to the system, to cover for the most obvious shortcomings during our trial run. We then ran a preliminary comparison of the original system and our augmented version on two of our articles. The results (displayed in Tables I and II) show that our augmented version drastically improves both precision and recall.

These results were exciting, however, we wished to make sure that our new rules weren't overly specific to our domain, and we wanted a more rigorous testing environment. Thankfully, the Ruprecht-Karl University Heidelberg Database Systems Research Group's website, maintained by Gertz, contains testing and evaluation scripts for HeidelTime based on a number of corpora, including the TempEval-2 dataset, Timebank 1.2, and the WikiWars corpus.

The TempEval-2 English dataset is actually based on the Timebank 1.2 dataset, and is an order of magnitude smaller. Thus Timebank 1.2 results can provide a more robust evaluation of our changes.

The WikiWars corpus[10] is a collection of 22 war articles from Wikipedia, with temporal expressions annotated with timex2.[3] Both Timebank and the TempEval-2 dataset are comprised of newswire articles. Historical articles are significantly different in style and presentation, and improvements on the results run on a corpus comprised solely of such articles is more likely to translate into improvements in extraction for our Wikipedia articles than improvements specific to newswire articles.

The evaluation scripts[4] provide precision, recall, and F-score results for five categories: Extraction (lenient), Extraction (strict), Normalization (value attribute), Extraction and Normalization (lenient and value attribute), and Extraction and Normalization (strict and value attribute). Strict extraction means for an expression extraction to be a true positive the expression must exactly match that of the gold standard. Lenient extraction relaxes this constraint to be that the extracted expression must overlap with the gold standard expression. The Normalization category requires the value of the extracted expression to exactly match that of the gold standard, and the combined categories require the extracted expression to meet the criteria of both of the component categories. The scripts also provide an easy to analyze document detailing each expression, and its value for both the gold standard and the results of the HeidelTime extraction. This allows us to easily analyze what changes improve the system the most.

Thus we began the iterative process of adding, removing, and modifying rules, expression resources, and normalization resources, and comparing the results on the WikiWars dataset. We continued this process for over 50 iterations. Our results can be seen in Table **??**. The value attribute recall and precision remained largely the same, with massive gains in all other areas. Recall improved the most, with an 8.1% improvement in lenient extent, and a 9.1% improvement in strict extent. However, all other values improved by at least 3%.

These were quite substantial results, but we wanted to make sure our gains were not because of specializing HeidelTime specifically for historical wikipedia articles, at the cost of precision and recall for other datasets. In order to guard against this, we tested our adjusted HeidelTime on the TimeBank 1.2 dataset. Our results, though not showing as dramatic of gains as on the WikiWars dataset, nevertheless show no decrease in accuracy or precision in any category, and, indeed show modest recall gains of over a full percent in both the lenient and strict extents. This gives us some confidence that our modifications are not over specialized.

Finally, we tested our improved HeidelTime on our pool of Wikipedia articles, taking a random sample of 200 sentences. The results will appear in a table in the final version of this paper.

C. Further Improvements

There are plenty of patterns we could still add to HeidelTime to further increase precision and accuracy, and continuing our iterative improvement process would likely yield

[3]Available for download free of charge from http://www.timexportal.info/
[4]Available on http://dbs.ifi.uni-heidelberg.de/

WikiWars				
		Default HeidelTime	Augmented HeidelTime	Improvement
Lenient Extent	Precision	0.940 = 94.0%	0.974 = 97.4%	+0.034 = +3.4%
	Recall	0.821 = 82.1%	0.902 = 90.2%	+0.081 = +8.1%
	F-score	0.877 = 87.7%	0.937 = 93.7%	+0.060 = +6.0%
Strict Extent	Precision	0.851 = 85.1%	0.901 = 90.1%	+0.050 = +5.0%
	Recall	0.743 = 74.3%	0.834 = 83.4%	+0.091 = +9.1%
	F-score	0.793 = 79.3%	0.866 = 86.6%	+0.073 = +7.3%
Value	Precision	0.896 = 89.6%	0.896 = 89.6%	+0.000 = +0.0%
	Recall	0.900 = 90.0%	0.902 = 90.2%	+0.002 = +0.2%
	F-score	0.898 = 89.8%	0.899 = 89.9%	+0.001 = +0.1%
Lenient Extent + Value	Precision	0.842 = 84.2%	0.872 = 87.2%	+0.030 = +3.0%
	Recall	0.736 = 73.6%	0.808 = 80.8%	+0.072 = +7.2%
	F-score	0.785 = 78.5%	0.839 = 83.9%	+0.054 = +5.4%
Strict Extent + Value	Precision	0.787 = 78.7%	0.826 = 82.6%	+0.039 = +3.9%
	Recall	0.688 = 68.8%	0.764 = 76.4%	+0.076 = +7.6%
	F-score	0.734 = 73.4%	0.794 = 79.4%	+0.060 = +6.0%

TABLE III
A COMPARISON OF THE DEFAULT HEIDELTIME AND OUR MODIFIED HEIDELTIME ON THE WIKIWARS DATASET. OUR CHANGES LEFT THE PRECISION AND RECALL OF THE VALUE ATTRIBUTE NEARLY UNTOUCHED, WHILE DRASTICALLY IMPROVING PRECISION AND RECALL IN ALL OTHER AREAS.

TimeBank 1.2				
		Default HeidelTime	Augmented HeidelTime	Improvement
Lenient Extent	Precision	0.905 = 90.5%	0.907 = 90.7%	+0.002 = +0.2%
	Recall	0.914 = 91.4%	0.928 = 92.8%	+0.014 = +1.4%
	F-score	0.909 = 90.9%	0.917 = 91.7%	+0.008 = +0.8%
Strict Extent	Precision	0.835 = 83.5%	0.835 = 83.5%	+0.000 = +0.0%
	Recall	0.843 = 84.3%	0.854 = 85.4%	+0.011 = +1.1%
	F-score	0.839 = 83.9%	0.844 = 84.4%	+0.005 = +0.5%
Value	Precision	0.862 = 86.2%	0.862 = 86.2%	+0.000 = +0.0%
	Recall	0.862 = 86.2%	0.862 = 86.2%	+0.000 = +0.0%
	F-score	0.862 = 86.2%	0.862 = 86.2%	+0.000 = +0.0%
Lenient Extent + Value	Precision	0.780 = 78.0%	0.782 = 78.2%	+0.002 = +0.2%
	Recall	0.788 = 78.8%	0.800 = 80.0%	+0.012 = +1.2%
	F-score	0.784 = 78.4%	0.791 = 79.1%	+0.07 = +0.7%
Strict Extent + Value	Precision	0.732 = 73.2%	0.732 = 73.2%	+0.000 = +0.0%
	Recall	0.740 = 74.0%	0.747 = 74.7%	+0.007 = +0.7%
	F-score	0.736 = 73.6%	0.739 = 73.9%	+0.003 = +0.3%

TABLE IV
A COMPARISON OF THE DEFAULT HEIDELTIME AND OUR MODIFIED HEIDELTIME ON THE TIMEBANK 1.2 DATASET. ONCE AGAIN OUR CHANGES LEFT THE PRECISION AND RECALL OF THE VALUE ATTRIBUTE NEARLY UNTOUCHED, THIS TIME HAVING MODEST PRECISION GAINS AND DECENT RECALL IMPROVEMENTS IN ALL OTHER AREAS.

increasingly accurate results. One change that would immediately increase value precision in the WikiWars dataset is full support for years in BC. We've add the extraction patterns, however, years BC are currently ignored for the purposes of post-processing underspecified values.

However, there are two large areas currently keeping precision and recall. First is inconsistent gold standard labeling. For example, in the WikiWars dataset, "the same time" is labelled in every occurrence as a temporal expression. In the TimeBank 1.2 dataset, on the other hand, although "the same time" occurs many times in similar contexts to the appearances in the WikiWars articles, it is never annotated as a temporal expression. Currently we simply disregard this expression, but adding it in would instantly increase extraction recall on the WikiWars dataset a non-negligible amount, while simultaneously decreasing extraction precision in the TimeBank 1.2 dataset.

The other, larger area, can be improved without modifying datasets. Currently the underspecified value post-processing step fills in the underspecified value with data from the last fully specified temporal expression. This causes errors, such as normalizing an expression of "January" following "December 2008" to "2008-01", when it should be "2009-01". Such errors are then propagated until the next temporal expression that specifies the exact year. Inaccurate post-processing is what leads to the greatest number of errors in the value attribute. A slightly more sophisticated heuristic may be able to drastically improve value recall and precision, and a significantly more sophisticated system, perhaps deriving temporal relations, or simply using lexicographical clues could perhaps increase value precision and recall to near-perfect levels.

V. EVENT EXTRACTION

We extract events from the Wikipedia articles by using Evita, developed by Pustejovsky et al. as an event recognizer for question/answer systems. The events extracted by

Evita are mostly action verbs, but alse include some nouns and adjectives that indicate events occurred. Evita, part of the TARSQI toolkit[5], annotates events with polarity, aspect, modality, tense, non-finite morphology, class, which we can use to filter events that are unlikely to be temporally located. We used Evita off-the-shelf; for more information, read the original paper introducing Evita[11].

VI. Temporal Relation Extraction

In order to order events relative to each other, we use a Markov Logic Network to generate the most likely set of temporal relations. Markov Logic Networks, introduced by Domingos et al. [12] are an extension of first order logic with probabilities other than one (true) or zero (false). Formulas hold probabilistically, so its simple to encode soft rules and probabilistic inference. The example given in [5] of a soft formula is

$$futureTense(e) \implies !beforeDCT(e),$$

that an event in the future tense is likely, but not certain, to be temporally located after the document creation time.

The de facto standard for temporally annotated corpora is the TimeBank 1.2 corpus (also used in our temporal expression extraction tests). The TimeBank corpus consists of about 300 newswire articles human-annotated with temporal events and expression and relations between them, given in Allen interval algebra notation. Allen notation consists of 13 qualitative relations between two intervals, which are depicted in Figure 1. As mentioned earlier, the TempEval tasks used a very small subset of these relations, and most temporal extraction research has avoided using the full set of relations. Ideally, our Markov Logic Network should only need to generate one type of relation. Thankfully, as noted by [1], if we treat every temporal event as an interval with a start and end point, then all thirteen relations can be represented by inequalities between the start and end points of the two intervals. This means we can set out to only find $after(a, b)$, which we take to mean a is after or at the same time as b. We can encode equality between two time points a and b as

$$after(a, b) \text{ AND } after(b, a).$$

For an example of an Allen relation encoded with these inequalities, consider the two intervals $A = [a_0, a_1]$ and $B = [b_0, b_1]$, where A finishes B. This is equivalent to:

$$after(a_0, b_0) \text{ AND } after(b_1, a_1) \text{ AND } after(a_1, b_1)$$

Similar conversions can be created for all 13 Allen relations. Using these conversions, we can use TimeBank for our training data.

The atoms for our Markov Logic Network are the events from Evita and the time expressions from HeidelTime. There are two main categories of features of our Markov Logic Network. The first is an $apriori\_after$ relation among the atoms from HeidelTime. Since HeidelTime exactly temporally locates time expressions, we can generate after relations such as January 1952 is after December 1951 before running the MLN. We code the hard rule

$$apriori\_after(x, y) \implies after(x, y),$$

to ensure that the world outputted by our MLN still contains these relations.

The other features are generated by the Stanford Parser [13]. The relevant soft formula we encode are of the form

$$dep(x, y) \implies after(point(x), point(y)),$$

following [1], where this is a second-order formula, where $point()$ is either the start or endpoint and *dep* stands for any one of the 50+ Stanford dependency. Stanford dependencies are relations between pairs of words in a sentence and are generated by the Stanford parser [14]. Examples of Stanford dependencies are dobj (direct object) and prep_in (related by the preposition "in").

We run the Stanford parser on the TimeBank training set, in order to learn the weights on these formulas. The higher the learned weight, the more likely it is that $after(point(x), point(y))$ is a relation in the final world outputted by an MLN. A negative weight corresponds to a formula that is actually likely to not hold. We only consider the dependencies between events, or between temporal expressions, or between one of each.

We also incorporate a global weighted formula softly enforcing transitivity:

$$after(a, b) \text{ AND } after(b, c) \implies after(a, c).$$

This constraint is soft, because there are inconsistencies in our training data that would break this constraint. This is an unfortunate limitation, but one to be expected of a large human-annotated corpus. Like Weld [1], we use a positive prior for this transitivity rule, mostly because the gold dataset

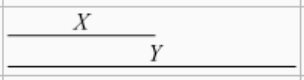
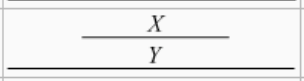
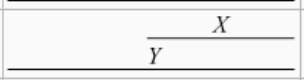
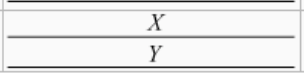
Relation	Illustration	Interpretation
$X<Y$ $Y>X$		X takes place before Y
$X\mathrm{m}Y$ $Y\mathrm{mi}X$		X meets Y (*i* stands for ***inverse***)
$X\mathrm{o}Y$ $Y\mathrm{oi}X$		X overlaps with Y
$X\mathrm{s}Y$ $Y\mathrm{si}X$		X starts Y
$X\mathrm{d}Y$ $Y\mathrm{di}X$		X during Y
$X\mathrm{f}Y$ $Y\mathrm{fi}X$		X finishes Y
$X=Y$		X is equal to Y

Fig. 1. An illustration of the 13 Allen interval relations. All of the relations depicted except equality have inverses.

[5]Download details available at http://www.timeml.org/site/tarsqi/toolkit/download.html

often lack transitively closed relations that should be included for complete information.

Unlike [1], we are not using a statistical relational learner to improve results; as we rely on our a priori after relations to pick up the slack. However, an intended extension for future work is a feature relating main verbal events of adjacent sentences. This will allow for more relations between sentences, leading to higher recall and ultimately a more accurate timeline.

Our results in using the MLN to generate temporal expressions will be available in the final draft of this paper.

VII. Synthesis

At this point we have two separate systems, a state-of-the-art temporal expression extractor improving upon HeidelTime, and a temporal relation extractor, following the basic implementation of the TIE system. We still need to integrate the results of these two mostly separate systems, to generate accurate timelines for Wikipedia articles. We show the results of three different approaches.

A. Naive Approach

We first create a baseline by assigning events in a sentence the tightest time-bounds of any date or time expression extracted by HeidelTime in the same sentence. Any events with no bounds after this process acquire bounds through a simple heuristic as follows: determined lower bounds (start points) are propagated forwards, upper bounds (end points) are propagated backwards. If there are still undefined bounds, they are filled in with the closest bound of their type in either direction. Events with identical start bounds are assumed to occur in the order in which they appear in the article. This fully specifies all bounds in an article, so long as at least one upper bound and one lower is specified by a temporal expression. Since each temporal expression has a start and endpoint, this means all bounds are specified so long as there is at least one temporal expression in the entire article. This approach is near identical to the one taken by Chasin.[2]

B. Basic Improvements

Our second approach uses some more data and basic heuristics to improve accuracy. Events are only given the time bounds of temporal expressions if there is a Stanford dependency between the two. If there is a dependency between an event and a duration expression, that event is determined to have taken as long as the expression in question. Otherwise temporal bounds are propagated as in the naive approach.

C. Incorporating the MLN

Our final and most sophisticated approach is largely similar to the last approach. However, instead of propagating bounds based on proximity in the original, bounds are propagated using the after relations generated by the Markov Logic Network.

D. Results

Our results can be seen in a table in the next draft of this paper.

VIII. Future Work

There are several directions to take this work in the future for whoever decides to extend this work. Feature selection for the Markov Logic Network could lead to massive improvements in temporal relation extraction. A simple filter of events unlikely to be temporally relevant could decrease the problem size and difficulty. One could attempt to use fuzzy sets or incorporate explicit uncertainty scores in order to get more accurate timelines. Fuzzy sets, or uncertainty is easily represented visually on a timeline by use of a gradient.

The most useful extension would be the creation of a gold standard annotated corpus based on historical articles. The training data for the Markov Logic Network consisted solely of newswire articles, which are of a decidedly different nature than a historical article, and a more domain-specific corpus would be incredibly useful. WikiWars, though quite useful for improving temporal expression extraction, lacks temporal relation annotation, and thus cannot currently be used to train our MLN. Also, any corpus creation where the articles are annotated with explicit time intervals would open large avenues of research. One such avenue would be trying to learn how long certain events are likely to take using a machine learning approach, and then using this information to estimate the time interval durations in the timeline more accurately.

As mentioned in the temporal expression section of this paper, improving HeidelTime is an area with several low-hanging fruit. More accurate temporal expression extraction will directly lead to more accurate timeline generation. This project's focus was mostly on improving HeidelTime, with some effort going into creating a working Markov Logic Network. Thus the synthesis section represents early first efforts at creating good timelines from temporal expressions and relations, and it is likely there are ways of improving the timelines independently of improving expression or relation extraction.

IX. Conclusion

We have presented significant improvements to HeidelTime, creating what is currently the most accurate system for the extraction of temporal expressions from historical articles. We also reimplemented an MLN, most following in the footsteps of TIE, for temporal relation extraction, and present decent results. Finally, we presented early attempts to combine these two systems in order to generate temporal bounds for all events in a document, and gave suggestions for future improvements.

References

[1] X. Ling and D. S. Weld, "Temporal information extraction," *AAAI*, 2010.

[2] R. Chasin, D. Woodward, and J. Kalita, "Extracting and displaying temporal entities from historical articles," *NCMT*, 2011.

[3] M. Verhagen, R. J. Gaizauskas, F. Schilder, M. Hepple, J. Moszkowicz, and J. Pustejovsky, "The tempeval challenge: identifying temporal relations in text." *Language Resources and Evaluation*, pp. 161–179, 2009.

[4] I. Mani, "Recent developments in temporal information extraction," *RANLP 2003*, 2003.

[5] K. Yoshikawa, S. Riedel, M. Asahara, and Y. Matsumoto, "Jointly identifying temporal relations with markov logic," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ser. ACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 405–413. [Online]. Available: http://portal.acm.org/citation.cfm?id=1687878.1687936

[6] J. Allen, "Maintaining knowledge about temporal intervals," *C. ACM*, 1983.

[7] J. Giles, "Internet encyclopedias go head to head," *Nature*, 2005.

[8] S. Schockaert, M. De Cock, and E. Kerre, "Reasoning about fuzzy temporal information from the web: towards retrieval of historical events," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 14, pp. 869–886, 2010, 10.1007/s00500-009-0471-8. [Online]. Available: http://dx.doi.org/10.1007/s00500-009-0471-8

[9] J. Strötgen and M. Gertz, "Heideltime: High quality rule-based extraction and normalization of temporal expressions," in *Proceedings of the 5th International Workshop on Semantic Evaluation*, ser. SemEval '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 321–324. [Online]. Available: http://portal.acm.org/citation.cfm?id=1859664.1859735

[10] P. Mazur and R. Dale, "Wikiwars: a new corpus for research on temporal expressions," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 913–922. [Online]. Available: http://portal.acm.org/citation.cfm?id=1870658.1870747

[11] R. Saurí, R. Knippen, M. Verhagen, and J. Pustejovsky, "Evita: A robust event recognizer for qa systems," *Proceedings of HLT/EMNLP 2005*, pp. 700–707, 2005.

[12] M. Richardson and P. Domingos, "Markov logic networks," *Machine Learning*, vol. 62, pp. 107–136, 2006, 10.1007/s10994-006-5833-1. [Online]. Available: http://dx.doi.org/10.1007/s10994-006-5833-1

[13] D. Klein and C. D. Manning, "Fast exact inference with a factored model for natural language parsing," *Advances in Neural Information Processing Systems*, vol. 15, pp. 3–10, 2003.

[14] M.-C. de Marneffe, B. MacCartney, and C. D. Manning, "Generating typed dependency parses from phrase structure parses," *LREC 2006*, 2006.