Creating RSS for News Archives and Beyond

Sandip Debnath
Microsoft Corporation
1 Microsoft Way, Redmond, WA 98052
debnath@acm.org
(Work done while in Penn State University)

Abstract

RSS or Rich Site Summary is becoming an invaluable format/tool for news feeds. More and more news publishing organizations are realizing its benefits. Content publishers are joining the already heavily crowded RSS club[1]. In the era of information explosion and peer-to-peer sharing, RSS is a great format for doing content publishing, archiving, sharing and much more. However, it came late. We realize that this should have started at the same time Internet became popular and news organizations are making their on-line debut. During the last decade, an enormous amount of news articles had already been published, and (at the same time,) improperly archived due to the lack of a flexible and widely accepted format of archival. However, better late than never. As we now explore possibilities of RSS, this is the time to make the transition smooth for old unformatted news articles and make it uniform across all (new and old) news articles. To do that we realized that extracting metadata of old news articles is one of the ways to create their RSS versions. In this paper we talk about our progress in extracting news metadata with the use of support vector classifier and show that an ordering of applying the classifiers is more useful than applying them in random order. We also show preliminary results on applying TIMEX tags to extract news events, which can be very useful to go beyond RSS to create individual event lines instead of taking the whole story under a single timeline.

Introduction

RSS (Rich Site Summary, RDF Site Summary or Really Simple Syndicate), has been fast becoming the de-facto industry standard for news, advertisements, and other on-line content publishing, and notification. As we have discussed in the abstract, thousands of companies have already joined the club and the list is growing by day.

Before going deeper into the details of RSS, let us look back and see what RSS is. As its name indicates it is a way to represent rich data in a simple way. It took birth during XML revolution and the introduction of RDF schema.

RSS became such a powerful tool due to its easy-to-create feature for feed creation and simple-to-understand tagging. It has become useful for many individuals and organizations. In a self-reinforcing cycle, it is becoming a standard for on-line content publishing.

However as we discussed, every new idea comes with challenges too. Though it is easy to create new content in RSS format, the challenge comes in converting existing content in an RSS or similar XML based tagged format.

This great challenge of converting old news articles RSS/XML ready, comes with the problem of extracting metadata. Automatically extracting metadata from within the news article itself is a major problem. This ultimately boils down to using the news articles, reading them automatically, discovering editing style, and finding useful metadata such as the **DateLine, HeadLine**, or **ByLine** (reporter name/reporting agency)[2] etc. To accomplish these challenging goals, a machine learning technique was deemed the best choice.

To that extent, as support vector (**SV**) technique works great in text-based classification purposes, we chose it and tested it for several hundred news articles in this paper. We trained the classifier with a training set derived from manually news articles. We then used the classifier for the rest of the old news articles (test set). As there are multiple metadata field (**Headline, Dateline, Byline** etc.) we realized that multiple classifiers for each field rather than a single one will be more appropriate. Though it worked out well, further research revealed that certain ordering of the application of multiple classifier works better than applying them in random order.

In this paper, we first indicate our old results of extracting **Dateline** of a news article and then introduce the classification technique for **HeadLine** or **ByLine** extraction. We show that first classifying the **Dateline** and applying that knowledge in the form of extra parameter in the classification of others give improved classification accuracies. In the last part of this work, we also try to show how we can go be-

[1]Over $20,000$ content publishers are using RSS feed as found in http://www.syndic8.com including small and big names such as CNN, BBC, NASA, WIRED etc.

[2]Though **DateLine, HeadLine** etc. are not exactly RSS/News XML tag names, however this does not change the generalization of underlying methodologies or use of fundamental concepts outlined here

yond and introduce the concept of tagging individual news events with respect to temporal behaviour. We used TIMEX tagging and showed an example page which describes how important the event organization could be.

Extracting temporal information is useful, not only in RSS but also in text-summarization or question-answering. In this paper we first described temporal information extraction (finding out the **DateLine**) as that is used later for improving other metadata extraction accuracies.

DateLine is defined as the date and time (if available) of the publication of the news article, *as mentioned in the article itself*. Most news articles mention the date of publication somewhere inside the body. As mentioned above, properly calculating the exact time-lines of individual events of a news article depends on finding the **DateLine**. In this paper we focus on finding the **DateLine** of an article using Support-vector (**SV**) learning approach.

For any referenced temporal expression (e.g. "last Friday", "next month", "next weekend" etc.) to be tagged with other temporal information such as year, month, or possible date and time, it is obvious that we need to add or subtract the proper difference of time from the **DateLine** of that article. In this paper we will see that finding the **DateLine** of an article is the most important task not only to find actual time-line, but to help increase the accuracy in finding the **HeadLine** and the **ByLine**.

The problem is not trivial due to several reasons. Firstly, temporal expressions are not always represented in a standard way. It has many different linguistic and geographical format. So it needs a proper grammar. Secondly the position of **DateLine** is not unique in a news article. As we see from Figure 1 and as explained later, it is not trivial to extract **DateLine**.

We approached the problem by identifying the proper set of parameters by which we will a train a learning classifier. Agreeing with our intuition, our choice of parameter-set combined with the **SV** based classifier, produced high accuracy.

The paper is organized this way: some related works are described in the next section followed by our approach. We thereafter talk about formats, data preparation algorithms, and training and testing phases. We described the general flow of our algorithm for all cases, be it **DateLine**, **HeadLine**, or **ByLine**. We show an example of TIMEX tagging of individual as part of going beyond if a single storyline and introducing events inside the article as separate entities, and conclude thereafter.

Related Work

Generating or converting old news articles into RSS or similar XML tagged format is not studied that extensively. The closest of this research trend is tagging news articles. Specifically time-tagging news article has received attention in recent years where most of the prior work is based on Natural Language Processing (*NLP*).

Unfortunately, we have not seen much prior work to find the **HeadLine, DateLine,** or **ByLine** of an article. The reason, as explained above, is due to the popular assumption that news data can be available from the archive, properly tagged.

In temporal annotations, we have seen prior efforts by Hwang and Schubert (Hwang & Schubert 1994), Kemp and Reyle (Kemp & Reyle 1993), Lascarides and Asher (Lascarides & Asher 1993), Allen (Allen 1984; 1995), Hitzeman (Hitzeman 1993) and others. They have used knowledge sources, tense, aspect, adverbs, rhetorical relations and of course background knowledge.

Regarding temporal significance, Allen's general theory of action and time (Allen 1984) is very effective in structuring textual documents into temporally well-defined blocks. Some early approaches are very formal with finding time or time related expressions in documents but they were instrumental in setting up the ground-breaking steps. Based on that others tried to use rule-based or sometimes knowledge-based techniques. But most of the researchers related to text summarization, question answering or temporal ontology building, used or tried to use *NLP* techniques. *NLP* has its roots long back in time with Reichenbach (Reichenbach 1947) who pointed out the difference between the point of speech (time of utterance), the point (time) of the event and point of reference or the reference time.

Lascarides and Asher used "narration" relation in sentences to identify the time of events. Others have found that news stories may not be a right place to use the narrative convention. As researchers found, events in news articles are tough to order. But even before starting to order the events in a news article, the first and foremost requirement is to find out the sentences carrying any occurrences of time units such as year, month, week, day and so on.

In most cases these time units are relative, meaning they are not expressed in complete time unit formats [3]

In Time Frames (Koen & Bender 2000), Koen and Bender stated the benefits of the time augmentation of news. Their time extractor extracts time with moderate precision and recall. MIT's Questioning News System (Sack 1997) used individual documents of a set, but did not create a temporal structure as such. Other researchers such as Allen (Allen 1995), Dorr (Dorr & Olsen 1997), Mani (Mani & Wilson 2000), Lascarides (Lascarides & Asher 1993), Passonneau (Passanneau 1988), Ferro (Ferro *et al.* 2001), tried to approach it from *NLP* perspective using discourse structures, tense of the verb or the aspect. But as we have seen and explained before there is not enough evidence of classifying the temporal expressions using machine language techniques to find out the **DateLine**. It sounds obvious that without the proper **DateLine**, no technique could give the proper time-line of any events inside the article.

Regarding **HeadLine** or **ByLine** extractions of a news article, we have not seen much effort in the past. Use of regular expression is rampant and does not ensure any quality guarantee and above all not flexible. In this regard, we actually tried to compare our method with vanilla regular expression

[3] Complete time units are usually expressed in **YY::MM::DD::HH::mm::SS** following the ISO8601 guidelines or at least in a similar way, which can easily be converted to ISO8601 format using simple converter algorithms.

Figure 1: A sample Yahoo finance page with temporal expressions highlighted.

technique to show that our learning based approach not only just outperform the regular expression technique, but it also ensures a thresholded quality level.

Our Approach

We used a **SV** classifier (Hastie, Tibshirani, & Friedman 2003; Scholkopf *et al.* 2000; 2001) for all metadata extraction. For **DateLine**, we compiled our own temporal grammar and devised the **TimeFinder** algorithm (based on this grammar) to find all possible temporal expressions inside an article. We the compute the values of several parameters (these parameters are described later) for each of these expressions. We train the **SV** classifier with this data. For a new set of articles, we process them first through **TimeFinder** to generate the set of temporal expressions available. We measure the parameter values for all these expressions and use our trained classifier to find the **DateLine**s of these new articles.

The same procedure also applies for the other metadata, such as **HeadLine** or **ByLine**. Instead of temporal expressions, we use full sentences and noun phrases (using Alembic workbench) for them respectively. The general outline of this whole process is described in Figure 2.

Example

Figure 1 shows a sample HTML page from Yahoo finance [4] Web-site. Dotted rectangular boxes indicate the temporal expressions. We see several temporal expressions in the beginning of this article. However, none of them are the **DateLine**. The **DateLine** of this article is "December 06, 2002", (just above the heading **"AMERISOURCEBERGEN CORP (ABC)"**), contrary to the popular assumption of taking the first temporal expression as the **DateLine**. Clearly we can not rely on that assumption.

(A) Time format and Grammar

We needed to use a standard date and time format. According to the ISO 8601 guidelines, the standard way of

[4] http://finance.yahoo.com

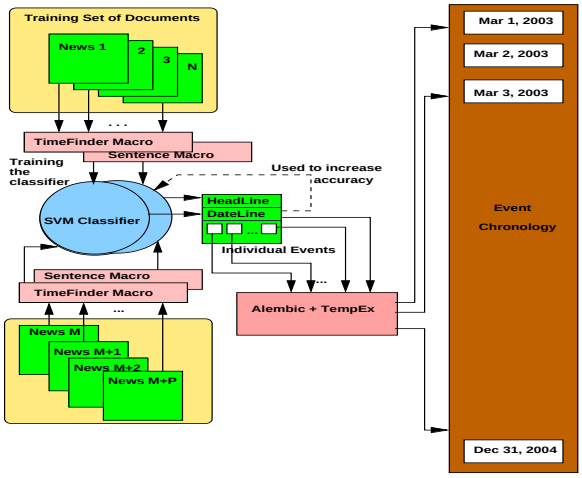

Figure 2: The steps of our learning based approach

expressing the date is **YYYY-MM-DD** and that of time is **hh:mm:ss**. There is also specifications and off-the-shelf algorithms available[5]

We devised our *own grammar* to extract temporal expressions. This include most of the time and date formats like "Jan 20, 2004", "01/20/2004", "2004-01-20", "Jan 20th, 2004", "20th Jan, 2004" etc. Moreover we also look for phrases like"2 months ago", "3 weeks after", "in 5 minutes", etc. Though every entries of "YYYY-MM-DD hh::mm::ss" could not be filled, **DateLine** classification does not depend on that. We followed the initial work by Koen and Bender (Koen & Bender 2000) to classify time expressions into these categories.

- **Interval** Such as "twenty to twenty five minutes" ((Koen & Bender 2000))
- **Age** Like "2 years old","A week after" etc.
- **Date** Such as "Jan 02, 2003", or "03/04/2004" etc.
- **Precise Time** Such as "2:00pm", or "Morning 7'O clock" etc.
- **Time Duration** "Evening", "Morning", "Dawn" etc.
- **Special Day** "Christmas", "New Year's Eve", "Thanksgiving", "Rosh Hashanah" etc.

(B) Word/Phrase

Word and phrases are extracted using our own regular expression grammar. To simplify the implementation, multiple words from the real **ByLine** has been clubbed together to represent one word. It is simple to modify it for n-gram.

[5] Arthur David Olson and others maintain a database of all current and many historic time zone changes and daylight saving time algorithms: http://www.twinsun.com/tz/tz-link.htm

(C) Sentence

Extracting sentences is the easiest, but still not a trivial tasks. Abbreviations have to be tackled properly in the grammar used to extract sentences.

Data Preparation and Training Phase

DataPreparer measured the parameter (described in next subsection) values for all temporal expressions, sentences, phrases. **ContentExtractor** is an intelligent HTML to text converter((Debnath, Mitra, & Giles 2005)), which breaks the whole page into logical blocks, identifies the redundant blocks comparing with other HTML pages from the same source and keep the informative blocks of text. During the first pass, **TimeFinder** function finds all probable temporal expressions in every article ("training article"), and converts them into ISO 8601 format. In Algorithm 1, **TimeFinder** is basically part of the **ItemFinder** algorithm.

During the second pass, **DataPreparer** asks the user to identify the item (**ByLine, DateLine** or **HeadLine** of the training article. The user identifies the correct temporal expression which can be attributed as the **DateLine** of the article. During the third pass, **DataPrepaper** pulls every expression hash key and measures the values of different parameters (described next) by using **MeasureParameterValue** function.

For **HeadLine ItemFinder** works as a sentence extractor. For **ByLine**, Alembic Natural Language Processing suite has been chosen as **ItemFinder**.

Data Parameters

The following set of parameters are used to create training data for the **SV** classifier. These parameters are measured for each temporal expression, sentence, and phrase. Most of them fall under distance measures, with some as frequency measures.

- **Paragraph Distance (PD)**: The paragraph distance consists of two parameters – how many paragraphs are there before a time expression or **PDB** and how many paragraphs are there after the time expression or **PDA**.
- **Sentence Distance (SD)**: The sentence distance consists of two parameters – how many sentences are there before a time expression or **SDB** and how many sentences are there after the time expression or **SDA**.
- **Word Distance (WD)**: The word distance also consists of two parameters – how many words are there before a time expression or **WDB** and how many words are there after the time expression or **WDA**.
- **Specific Words (SW)**: Specific words (**SW**) are also very important to properly identify the **DateLine**. Words like "By", "On", etc. has more often been seen near the **DateLine**'s temporal expression compared to other temporal expressions.
- **Specific Symbols (SS)**: In the same way we also consider the occurrences and distances between specific symbols (**SS**) and the time expressions. These symbols include special character-set like "–" or ":" which are also common near the **DateLine** expression.

Algorithm 1: DataPreparer (for Training phase): This algorithm prepares data to train **SV** classifier.

Input : HTML Page H, Parameter Set $\mathcal{P}$

Output : Training Set to train the Support-Vector Classifier

Standard: Word/Phrase Extractor,ISO 8601 standard for date and time, Sentence Extractor Algorithm, etc.

begin

$X \longleftarrow ContentExtractor(H)$

Pass 1:
$\mathcal{T} \longleftarrow$ ItemFinder(H)
Extract all words/phrases/time expressions(using our grammar)/sentences, assuming this set is $\mathcal{T}$

Pass 2: (User interface)
Ask the user to specify which word/time-line/sentence $t \in \mathcal{T}$ is the ByLine/DateLine/HeadLine.

Pass 3:
Measuring the $\mathcal{P}$ parameter values for the words/time expressions/sentences which are selected in the first pass.
for *each* $t_i \in \mathcal{T}$ **do**
  $P_{t_i} \longleftarrow MeasureParameterValue$(t,H,X);
  (P_{t_i} is a data row in the $|\mathcal{T}|X|\mathcal{P}|$ matrix.)
Prepare all parameter values in tabular format and stores them in training datafile.

end

Algorithm 2: MeasureParameterValue (used in both training and testing phase): This function calculates all the parameter values for a temporal expression in an HTML page H.

Input : Word/Time expression/Sentence t, HTML Page H, Text Page X converted from H

Output : Values of Word Data/Temporal Data/Sentence Data Parameters

begin

Function MeasureParameterValue(t, H, X**)**
We need both X, and H as some of the parameter calculations depend on HTML characters and some will be calculated from the text version of the article.
begin
  Takes the content X and breaks it into Paragraphs, Sentences, Words . . . etc.
  $\mathcal{P}$ is the Data Parameters
  for *each parameter* $p \in \mathcal{P}$ **do**
    $p_{t_i} \longleftarrow$ value of p in X for t_i;
    Push p_{t_i} in P_{t_i}.
  return P_{t_i} (A row vector)
end

end

- **Font Face Variation (FFV)**: Font face variation (**FFV**) is also another important factor which can be used to identify the location of **DateLine**. We see that usually the news publication date is placed close to the **HeadLine** of the news article and usually the **HeadLine** is written in different character size or in bold face. The regular text in normal font face follows it. Though things are not always written in the same way (that is why it is a challenging problem), yet there is a correlation between their locations and **DateLine**. We wanted to exploit this correlation and so we marked the places in the document where a change of font face occurs. Then we calculated distance $D^i \forall i \in |\mathcal{T}|$ where D^i is the shortest distance between t_i (the i^{th} temporal expression) and the marks. So if there are M places where font face changes, $D^i = \min(Distance(i,j))$, where $Distance(i,j)$ is character difference between t_i and j^{th} mark.
- **Similarity Measures (SM)**: Similarity measures involve word level similarity between sentences before and after a time expression. The reason behind choosing this parameter is the observation that usually the **HeadLine** of a news article and the first paragraph just after the **DateLine** describe the same event, sometimes even using identical words or phrases.

Some of the parameters alone may not be sufficient in distinguishing the **DateLine** from other temporal expressions but taking everything into account helped in getting high accuracy. In future, importance of specific parameters and the redundancy of others will be reported.

Testing and Evaluation Phase

Financial news articles from various (here 19) Web-sites (Table 1) have been used. Column 3 and 4 represent accuracy of **DateLine** extraction by using SV classifier as opposed to specific regular expression (RE) for each Web-site. Column 5 and 6 show **HeadLine** extraction accuracy with and without the use of **DateLine**. Extracting the **DateLine** first and applying it as a parameter for **HeadLine** extraction improves the accuracy (in most cases).

Table 1 also shows the accuracy of our algorithm. Accuracy implies the classification accuracy whereas accuracy from regular expression methods simply means the percentage of time for which the **DateLine** has been identified correctly. From this table, it is clear that our approach can be used to achieve high accuracy and can outperform a generic regular expression matching algorithm to find **DateLine**.

Interestingly when we applied the same technique for **HeadLine**, we found that first classifying the **DateLine** has an added advantage of making the **HeadLine** accuracy better. Therefore in complex cases like this, not only classification, but the proper step-by-step use of it is equally important.

Conclusion

RSS is a useful concept for metadata tagged news articles, but it is only used for last few years. We came up with metadata extraction techniques which can be used to convert

Site	**No. of articles**	**Acc of DateLine (C)**	Acc of DateLine (RE)	Acc (w/o) of HeadLine	**Acc (w) of HeadLine**
AP	37	94.59	89.18	81.08	94.59
Briefing.com	132	94.69	83.33	87.1	90.9
BusinessWeek	128	92.96	78.12	88.28	93.75
Business Wire	430	96.04	93.02	84.72	92.4
CBS	606	96.53	89.93	90.26	94.88
CCBN	92	94.56	88.04	80.43	85.86
Dow Jones	324	98.14	92.59	91.97	95.67
EDGAR	599	92.98	85.97	90.65	96.82
Forbes	451	98.66	93.34	88.91	95.34
Market Wire	158	96.2	79.11	89.87	94.93
Morningstar	14	92.85	85.71	71.42	71.42
Motley Fool	162	93.20	83.33	82.71	89.51
NewsFactor	71	91.54	83.09	91.5	95.77
PR Newswire	525	95.04	91.04	94.28	96.19
PrimeZone	64	96.87	85.93	85.93	90.62
Reuters	641	97.97	92.82	93.6	96.72
SmartMoney	63	92.06	85.71	87.3	92.06
StarMine	38	89.4	92.1	89.47	84.21
TheStreet.com	351	89.74	85.47	91.16	92.59

Table 1: The columns represent Source Web-site, Number of Articles, Accuracy of **DateLine** from SVM and using Regular Expression rules, Accuracy of **HeadLine** without the use of **DateLine** information and with the use of **DateLine** information.

Algorithm 3: FindAccuracy (testing phase): This algorithm uses classifier $\mathcal{C}$ to classify the words/temporal expressions/sentences.

Input : Classifier $\mathcal{C}$, HTML Page H
Output : Accuracy
begin
  $X \longleftarrow ContentExtractor(H)$
  Pass 1:
  $\mathcal{T} \longleftarrow$ ItemFinder(H)
  Pass 2:
  Measure the $\mathcal{P}$ parameter values for all the words/time expressions/sentences extracted in the first pass.
  for *each* $t_i \in \mathcal{T}$ **do**
    $p_{t_i} \longleftarrow MeasureParameterValue$(t,H,X);
    p_{t_i} is a data row in the $|\mathcal{T}|X|\mathcal{P}|$ matrix.
  Prepare all parameter values in tabular format and stores them in testing datafile.
  Pass 3:
  Feed the testing datafile to the classifier $\mathcal{C}$
  Find the ByLine/DateLine/HeadLine and match with the labelled dataset and find the accuracy
end

```
<doc> <S>
<HeadLine> AMERISOURCEBERGEN CORP (ABC) <HeadLine>
<ByLine> form 8-k </ByLine>
...
Other Events.</S>  <S>On <TIMEX TYPE=''DATE'' VAL=''20021205''>December 5,
2002</TIMEX>, <ENAMEX TYPE=''ORGANIZATION''>AmerisourceBergen Corporation
</ENAMEX> (the''<ENAMEX TYPE=''ORGANIZATION''>Company</ENAMEX>'') issued
a  press release providing information about an investor meeting held
<TIMEX TYPE=''DATE'' VAL=''20021206''>that day</TIMEX> by the
<ENAMEX TYPE=''ORGANIZATION''>Company</ENAMEX> in <ENAMEX TYPE=
''LOCATION''>New York City</ENAMEX>, providing disclosure of the <ENAMEX
TYPE=''ORGANIZATION''>Company</ENAMEX>'s  financial expectations for the
fiscal quarter ending <TIMEX TYPE=''DATE'' VAL=''20021231''>December 31,
2002</TIMEX> and affirming the <ENAMEX TYPE=''ORGANIZATION''>Company
</ENAMEX>'s financial expectations for the fiscal year ending <TIMEX
TYPE=''DATE'' VAL=''20030930''>September 30, 2003</TIMEX>.</S>
<S>A copy of the press release is filed as Exhibit 99.1 to this report
and incorporated herein by reference.</S>
...
```

Table 2: RSS/XML-like tagging + TIMEX Tagging of a news article.

archived news articles into RSS. To do that we approached with Support Vector based classifiers. We devised the parameter set to best extract different metadata. We also show that a step-by-step process of applying SV classifier is better than applying them randomly for different metadata such as **DateLine, HeadLine**, or **ByLine**. We also want to tag individual news events inside a news story and showed that using TIMEX it is possible to tag individual events with temporal tag. Due to space constraints we would like to elaborate this in future.

References

Allen, J. F. 1984. Towards a general theory of action and time. In *Artificial Intelligence*, 123–154.

Allen, J. F. 1995. Natural language understanding: Discourse structure, tense and aspect. In *Addison-Wesley Chapter 16:5*, 517–533.

Debnath, S.; Mitra, P.; and Giles, C. L. 2005. Automatic extraction of informative blocks from webpages. In *the upcoming proceedings of the Special Track on Web Technologies and Applications in the ACM Symposium of Applied Computing*.

Dorr, B., and Olsen, M. B. 1997. Driving verbal and coompositional lexical aspect for nlp applications. In *In the proceedings of ACL*, 151–158.

Ferro, L.; Mani, I.; Sundheim, B.; and Wilson, G. 2001. Tides temporal annotation guidelines draft - version 1.02. In *MITRE Technical report*.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2003. *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. Springer Verlog.

Hitzeman, J. 1993. *Temporal Adverbials and the Syntax-Semantics Interface*. University of Rochester, Rochester, New York.

Hwang, C., and Schubert, L. K. 1994. Interpreting tense, aspect, and time adverbials: a compositional, unified approach. In *Proceedings of the 1st International Conference on Temporal Logic*, 238–264.

Kemp, H., and Reyle, U. 1993. *From Discourse to Logic*. Kluwer Academic Publishers.

Koen, D., and Bender, W. 2000. Time frames: Temporal augmentation of the news. In *IBM Systems Journal*, volume 39, 597–616.

Lascarides, A., and Asher, N. 1993. Temporal relations, discourse structure, and commonsense entailment. In *Lingusitics and Philosophy*, 437–494.

Mani, I., and Wilson, G. 2000. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, 69–76.

Passanneau, R. J. 1988. A computational model of the semantics of tense and aspect. In *Computational Linguistics*, 44–60.

Reichenbach, H. 1947. The tenses of verb. In *Elements of Symbolic Logic*, 287–298.

Sack, W. 1997. The questioning news system. In *Technical Report presented at the MIT media Library*.

Scholkopf, B.; Smola, A.; Williamson, R.; and Bartlett, P. L. 2000. New support vector algorithms. neural computation. In *Neural Computation 12*, 1207–1245.

Scholkopf, B.; Platt, J.; Shawe-Taylor, J.; Smola, A. J.; and Williamson, R. C. 2001. Estimating the support of a high-dimensional distribution. In *Neural Computation 13*, 1443–1471.