Labeling Documents with Timestamps: Learning from their Time Expressions

Nathanael Chambers
Department of Computer Science
United States Naval Academy
nchamber@usna.edu

Abstract

Temporal reasoners for document understanding typically assume that a document's creation date is known. Algorithms to ground relative time expressions and order events often rely on this timestamp to assist the learner. Unfortunately, the timestamp is not always known, particularly on the Web. This paper addresses the task of automatic document timestamping, presenting two new models that incorporate rich linguistic features about time. The first is a discriminative classifier with new features extracted from the text's time expressions (e.g., 'since 1999'). This model alone improves on previous generative models by 77%. The second model learns probabilistic constraints between time expressions and the *unknown* document time. Imposing these learned constraints on the discriminative model further improves its accuracy. Finally, we present a new experiment design that facilitates easier comparison by future work.

1 Introduction

This paper addresses a relatively new task in the NLP community: automatic document dating. Given a document with unknown origins, what characteristics of its text indicate the year in which the document was written? This paper proposes a learning approach that builds constraints from a document's use of time expressions, and combines them with a new discriminative classifier that greatly improves previous work.

The temporal reasoning community has long depended on document timestamps to ground relative time expressions and events (Mani and Wilson, 2000; Llidó et al., 2001). For instance, consider the following passage from the TimeBank corpus (Pustejovsky et al., 2003):

> And while there was no profit *this year* from discontinued operations, *last year* they contributed 34 million, before tax.

Reconstructing the timeline of events from this document requires extensive temporal knowledge, most notably, the document's creation date to ground its relative expressions (e.g., *this year* = 2012). Not only did the latest TempEval competitions (Verhagen et al., 2007; Verhagen et al., 2009) include tasks to link events to the (known) document creation time, but state-of-the-art event-event ordering algorithms also rely on these timestamps (Chambers and Jurafsky, 2008; Yoshikawa et al., 2009). This knowledge is assumed to be available, but unfortunately this is not often the case, particularly on the Web.

Document timestamps are growing in importance to the information retrieval (IR) and management communities as well. Several IR applications depend on knowledge of when documents were posted, such as computing document relevance (Li and Croft, 2003; Dakka et al., 2008) and labeling search queries with temporal profiles (Diaz and Jones, 2004; Zhang et al., 2009). Dating documents is similarly important to processing historical and heritage collections of text. Some of the early work that motivates this paper arose from the goal of automatically grounding documents in their historical contexts (de Jong et al., 2005; Kanhabua and Norvag, 2008; Kumar et al., 2011). This paper builds on their work

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 98–106,
Jeju, Republic of Korea, 8-14 July 2012.

by incorporating more linguistic knowledge and explicit reasoning into the learner.

The first part of this paper describes a novel learning approach to document dating, presenting a discriminative model and rich linguistic features that have not been applied to document dating. Further, we introduce new features specific to absolute time expressions. Our model outperforms the generative models of previous work by 77%.

The second half of this paper describes a novel learning algorithm that orders time expressions against the unknown timestamp. For instance, the phrase *the second quarter of 1999* might be labeled as being *before* the timestamp. These labels impose constraints on the possible timestamp and narrow down its range of valid dates. We combine these constraints with our discriminative learner and see another relative improvement in accuracy by 9%.

2 Previous Work

Most work on dating documents has come from the IR and knowledge management communities interested in dating documents with unknown origins. de Jong et al. (2005) was among the first to automatically label documents with dates. They learned unigram language models (LMs) for specific time periods and scored articles with log-likelihood ratio scores. Kanhabua and Norvag (2008; 2009) extended this approach with the same model, but expanded its unigrams with POS tags, collocations, and tf-idf scores. They also integrated search engine results as features, but did not see an improvement. Both works evaluated on the news genre.

Recent work by Kumar et al. (2011) focused on dating Gutenberg short stories. As above, they learned unigram LMs, but instead measured the KL-divergence between a document and a time period's LM. Our proposed models differ from this work by applying rich linguistic features, discriminative models, and by focusing on how time expressions improve accuracy. We also study the news genre.

The only work we are aware of within the NLP community is that of Dalli and Wilks (2006). They computed probability distributions over different time periods (e.g., months and years) for each observed token. The work is similar to the above IR work in its bag of words approach to classification. They focused on finding words that show periodic spikes (defined by the word's standard deviation in its distribution over time), weighted with inverse document frequency scores. They evaluated on a subset of the Gigaword Corpus (Graff, 2002).

The experimental setup in the above work (except Kumar et al. who focus on fiction) all train on news articles from a particular time period, and test on articles in the same time period. This leads to possible overlap of training and testing data, particularly since news is often reprinted across agencies the same day. In fact, one of the systems in Kanhabua and Norvag (2008) simply searches for one training document that best matches a test document, and assigns its timestamp. We intentionally deviate from this experimental design and instead create temporally disjoint train/test sets (see Section 5).

Finally, we extend this previous work by focusing on aspects of language not yet addressed for document dating: linguistic structure and absolute time expressions. The majority of articles in our dataset contain time expressions (e.g., the year 1998), yet these have not been incorporated into the models despite their obvious connection to the article's timestamp. This paper first describes how to include time expressions as traditional features, and then describes a more sophisticated temporal reasoning component that naturally fits into our classifier.

3 Timestamp Classifiers

Labeling documents with timestamps is similar to topic classification, but instead of choosing from topics, we choose the most likely year (or other granularity) in which it was written. We thus begin with a bag-of-words approach, reproducing the generative model used by both de Jong (2005) and Kanhabua and Norvag (2008; 2009). The subsequent sections then introduce our novel classifiers and temporal reasoners to compare against this model.

3.1 Language Models

The model of de Jong et al. (2005) uses the normalized log-likelihood ratio (NLLR) to score documents. It weights tokens by the ratio of their probability in a specific year to their probability over the entire corpus. The model thus requires an LM for each year and an LM for the entire corpus:

$$NLLR(D,Y) = \sum_{w \in D} P(w|D) * log(\frac{P(w|Y)}{P(w|C)}) \quad (1)$$

where D is the target document, Y is the time span (e.g., a year), and C is the distribution of words in the corpus across all years. A document is labeled with the year that satisfies $argmax_Y NLLR(D,Y)$. They adapted this model from earlier work in the IR community (Kraaij, 2004). We apply Dirichlet-smoothing to the language models (as in de Jong et al.), although the exact choice of α did not significantly alter the results, most likely due to the large size of our training corpus. Kanhabua and Norvag added an entropy factor to the summation, but we did not see an improvement in our experiments.

The unigrams w are lowercased tokens. We will refer to this de Jong et al. model as the **Unigram NLLR**. Follow-up work by Kanhabua and Norvag (2008) applied two filtering techniques to the unigrams in the model:

1. **Word Classes**: include only nouns, verbs, and adjectives as labeled by a POS tagger
2. **IDF Filter**: include only the top-ranked terms by tf-idf score

We also tested with these filters, choosing a cutoff for the top-ranked terms that optimized performance on our development data. We also stemmed the words as Kanhabua and Norvag suggest. This model is the **Filtered NLLR**.

Kanhabua and Norvag also explored what they termed *collocation features*, but lacking details on how collocations were included (or learned), we could not reproduce this for comparison. However, we instead propose using NER labels to extract what may have counted as collocations in their data. Named entities are important to document dating due to the nature of people and places coming in and out of the news at precise moments in time. We compare the NER features against the Unigram and Filtered NLLR models in our final experiments.

3.2 Discriminative Models

In addition to reproducing the models from previous work, we also trained a new discriminative version with the same features. We used a MaxEnt model and evaluated with the same filtering methods based on POS tags and tf-idf scores. The model performed best on the development data without any filtering or stemming. The final results (Section 6) only use the lowercased unigrams. Ultimately, this MaxEnt model vastly outperforms these NLLR models.

3.3 Models with Time Expressions

The above language modeling and MaxEnt approaches are token-based classifiers that one could apply to any topic classification domain. Barring other knowledge, the learners solely rely on the observed frequencies of unigrams in order to decide which class is most likely. However, document dating is not just a simple topic classification application, but rather relates to temporal phenomena that is often explicitly described in the text itself. Language contains words and phrases that discuss the very time periods we aim to recover. These expressions should be better incorporated into the learner.

3.3.1 Motivation

Let the following snippet serve as a text example with an ambiguous creation time:

> Then there's the fund-raiser at the American Museum of Natural History, which plans to welcome about 1,500 guests paying $1,000 to $5,000. Their tickets will entitle them to a preview of...the new Hayden Planetarium.

Without extremely detailed knowledge about the American Museum of Natural History, the events discussed here are difficult to place in time, let alone when the author reported it. However, time expressions are sometimes included, and the last sentence in the original text contains a helpful relative clause:

> Their tickets will entitle them to a preview of...the new Hayden Planetarium, *which does not officially open until February 2000*.

This one clause is more valuable than the rest of the document, allowing us to infer that the document's timestamp is *before* February, 2000. An educated guess might surmise the article appeared in the year prior, 1999, which is the correct year. At the very least, this clause should eliminate all years after 2000 from consideration. Previous work on document dating does not integrate this information except to include the unigram '2000' in the model.

This paper discusses two complementary ways to learn and reason about this information. The first is to simply add richer time-based features into the model. The second is to build separate learners that can assign probabilities to entire ranges of dates, such as *all years following 2000* in the example above. We begin with the feature-based model.

3.3.2 Time Features

To our knowledge, the following time features have not been used in a document dating setting. We use the freely available Stanford Parser and NER system[1] to generate the syntactic interpretation for these features. We then train a MaxEnt classifier and compare against previous work.

Typed Dependency: The most basic time feature is including governors of year mentions and the relation between them. This covers important contexts that determine the semantics of the time frame, like prepositions. For example, consider the following context for the mention *1997*:

> Torre, who watched the Kansas City Royals beat the Yankees, 13-6, on Friday for the first time since 1997.

The resulting feature is '*since pobj 1997*'.

Typed Dependency POS: Similar to Typed Dependency, this feature uses POS tags of the dependency relation's governor. The feature from the previous example is now *'PP pobj 1997'*. This generalizes the features to capture time expressions with prepositions, as noun modifiers, or other constructs.

Verb Tense: An important syntactic feature for temporal positioning is the tense of the verb that dominates the time expression. A past tense verb situates the phrase *in 2003* differently than one in the future. We traverse the sentence's parse tree until a governor with a VB* tag is found, and determine its tense through hand constructed rules based on the structure of the parent VP. The verb tense feature takes a value of *past*, *present*, *future*, or *undetermined*.

Verb Path: The verb path feature is the dependency path from the nearest verb to the year expression. The following snippet will include the feature, *'expected prep in pobj 2002'*.

[1] http://nlp.stanford.edu/software

Finance Article from Jan. 2002

Text Snippet	Relation to 2002
...started a hedge fund before the market peaked in 2000.	before
The peak in economic activity was the 4th quarter of 1999.	before
...might have difficulty in the latter part of 2002.	simultaneous

Figure 1: Three year mentions and their relation to the document creation year. Relations can be correctly identified for training using known document timestamps.

> Supervising them is Vice President Hu Jintao, who appears to be Jiang's favored successor if he retires from leadership as expected in 2002.

Named Entities: Although not directly related to time expressions, we also include n-grams of tokens that are labeled by an NER system using Person, Organization, or Location. People and places are often discussed during specific time periods, particularly in the news genre. Collecting named entity mentions will differentiate between an article discussing a *bill* and one discussing the US President, *Bill Clinton*. We extract NER features as sequences of uninterrupted tokens labeled with the same NER tag, ignoring unigrams (since unigrams are already included in the base model). Using the Verb Path example above, the bigram feature *Hu Jintao* is included.

4 Learning Time Constraints

This section departs from the above *document* classifiers and instead classifies individual emphyear mentions. The goal is to automatically learn **temporal constraints** on the document's timestamp.

Instead of predicting a single year for a document, a temporal constraint predicts a range of years. Each time mention, such as '*not since 2009*', is a constraint representing its relation to the document's timestamp. For example, the mentioned year '2009' must occur *before* the year of document creation. This section builds a classifier to label time mentions with their relations (e.g., before, after, or simultaneous with the document's timestamp), enabling these mentions to constrain the document classifiers described above. Figure 1 gives an example of time mentions and the desired labels we wish to learn.

To better motivate the need for constraints, let

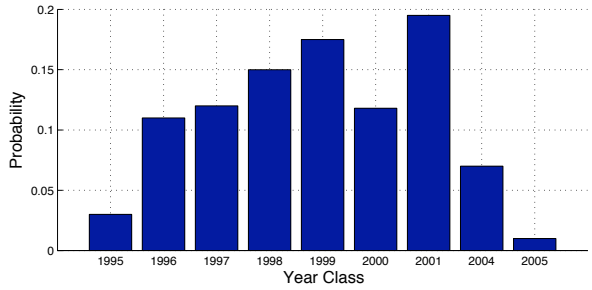

Figure 2: Distribution over years for a single document as output by a MaxEnt classifier.

Figure 2 illustrate a typical distribution output by a document classifier for a training document. Two of the years appear likely (1999 and 2001), however, the document contains a time expression that seems to impose a strict constraint that should eliminate 2001 from consideration:

> Their tickets will entitle them to a preview of...the new Hayden Planetarium, which does not officially open until February 2000.

The clause *until February 2000* in a present tense context may not definitively identify the document's timestamp (1999 is a good guess), but as discussed earlier, it should remove all future years beyond 2000 from consideration. We thus want to impose a constraint based on this phrase that says, loosely, 'this document was likely written before 2000'.

The document classifiers described in previous sections cannot capture such ordering information. Our new time features in Section 3.3.2 add richer time information (such as *until pobj 2000* and *open prep until pobj 2000*), but they compete with many other features that can mislead the final classification. An independent constraint learner may push the document classifier in the right direction.

4.1 Constraint Types

We learn several types of constraints between each year mention and the document's timestamp. Year mentions are defined as tokens with exactly four digits, numerically between 1900 and 2100. Let T be the document timestamp's year, and M the year mention. We define three **core relations**:

1. **Before** Timestamp: $M < T$
2. **After** Timestamp: $M > T$
3. **Same** as Timestamp: $M == T$

We also experiment with 7 **fine-grained relations**:

1. **One year Before** Timestamp: $M == T - 1$
2. **Two years Before** Timestamp: $M == T - 2$
3. **Three+ years Before** Timestamp: $M < T - 2$
4. **One year After** Timestamp: $M == T + 1$
5. **Two years After** Timestamp: $M == T + 2$
6. **Three+ years After** Timestamp: $M > T + 2$
7. **Same** Year and Timestamp: $M == T$

Obviously the more fine-grained a relation, the better it can inform a classifier. We experiment with these two granularities to compare performance.

The learning process is a typical training environment where year mentions are treated as labeled training examples. Labels for year mentions are automatically computed by comparing the actual timestamp of the training document (all documents in Gigaword have dates) with the integer value of the year token. For example, a document written in 1997 might contain the phrase, "in the year 2000". The year token (2000) is thus *three+ years after* the timestamp (1997). We use this relation for the year mention as a labeled training example.

Ultimately, we want to use similar syntactic constructs in training so that "in the year 2000" and "in the year 2003" mutually inform each other. We thus compute the label for each time expression, and replace the integer year with the generic *YEAR* token to generalize mentions. The text for this example becomes "in the year YEAR" (labeled as *three+ years after*). We train a MaxEnt model on each year mention, to be described next. Table 2 gives the overall counts for the core relations in our training data. The vast majority of year mentions are references to the future (e.g. after the timestamp).

4.2 Constraint Learner

The features we use to classify year mentions are given in Table 1. The same time features in the document classifier of Section 3.3.2 are included, as well as several others specific to this constraint task.

We use a MaxEnt classifier trained on the individual year mentions. Documents often contain multiple (and different) year mentions; all are included in training and testing. This classifier labels mentions with relations, but in order to influence the document classifier, we need to map the relations to individual

Time Constraint Features

Typed Dep.	Same as Section 3.3.2
Verb Tense	Same as Section 3.3.2
Verb Path	Same as Section 3.3.2
Decade	The decade of the year mention
Bag of Words	Unigrams in the year's sentence
n-gram	The 4-gram and 3-gram that end with the year
n-gram POS	The 4-gram and 3-gram of POS tags that end with the year

Table 1: Features used to classify year expressions.

Constraint	Count
After Timestamp	1,203,010
Before Timestamp	168,185
Same as Timestamp	141,201

Table 2: Training size of year mentions (and their relation to the document timestamp) in Gigaword's NYT section.

year predictions. Let T_d be the set of mentions in document d. We represent a MaxEnt classifier by $P_Y(R|t)$ for a time mention $t \in T_d$ and possible relations R. We map this distribution over relations to a distribution over years by defining $P_{year}(Y|d)$:

$$P_{year}(y|d) = \frac{1}{Z(T_d)} \sum_{t \in T_d} P_Y(rel(val(t) - y)|t) \quad (2)$$

$$rel(x) = \begin{cases} before & \text{if } x < 0 \\ after & \text{if } x > 0 \\ simultaneous & \text{otherwise} \end{cases} \quad (3)$$

where $val(t)$ is the integer year of the year mention and $Z(T_d)$ is the partition function. The $rel(val(t) - y)$ function simply determines if the year mention t (e.g., 2003) is before, after, or overlaps the year we are predicting for the document's unknown timestamp y. We use a similar function for the seven fine-grained relations. Figure 3 visually illustrates how $P_{year}(y|d)$ is constructed from three year mentions.

4.3 Joint Classifier

Finally, given the document classifiers of Section 3 and the constraint classifier just defined in Section 4, we create a joint model combining the two with the following linear interpolation:

$$P(y|d) = \lambda P_{doc}(y|d) + (1 - \lambda) P_{year}(y|d) \quad (4)$$

where y is a year, and d is the document. λ was set to 0.35 by maximizing accuracy on the dev set. See Figure 4. This optimal $\lambda = .35$ weights the constraint classifier higher than the document classifier.

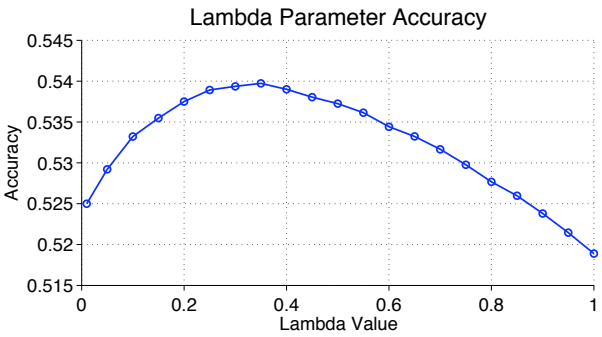

Figure 4: Development set accuracy and λ values.

5 Datasets

This paper uses the New York Times section of the Gigaword Corpus (Graff, 2002) for evaluation. Most previous work on document dating evaluates on the news genre, so we maintain the pattern for consistency. Unfortunately, we cannot compare to these previous experiments because of differing evaluation setups. Dalli and Wilks (2006) is most similar in their use of Gigaword, but they chose a random set of documents that cannot be reproduced. We instead define specific segments of the corpus for evaluation.

The main goal for this experiment setup was to establish specific training, development, and test sets. One of the potential difficulties in testing with news articles is that the same story is often reprinted with very minimal (or no) changes. Over 10% of the documents in the New York Times section of the Gigaword Corpus are exact or approximate duplicates of another document in the corpus[2]. A training set for document dating must not include duplicates from the test set.

We adopt the intuition behind the experimental setup used in other NLP domains, like parsing, where the entire test set is from a contiguous section of the corpus (as opposed to randomly selected examples across the corpus). As the parsing community trains on sections 2-21 of the Penn Treebank (Marcus et al., 1993) and tests on section 23, we create Gigaword sections by isolating specific months.

[2]*Approximate duplicate* is defined as an article whose first two sentences exactly match the first two of another article. Only the second matched document is counted as a duplicate.

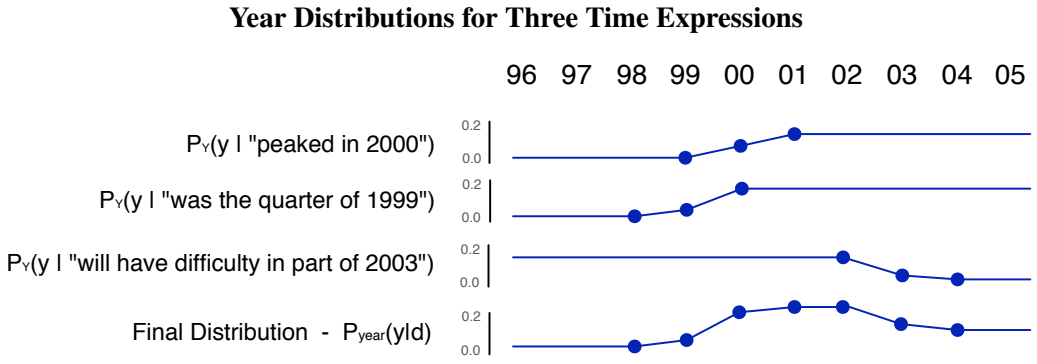

Figure 3: Three year mentions in a document and the distributions output by the learner. The document is from 2002. The dots indicate the *before*, *same*, and *after* relation probabilities. The combination of three constraints results in a final distribution that gives the years 2001 and 2002 the highest probability. This distribution can help a document classifier make a more informed final decision.

Training	Jan-May and Sep-Dec
Development	July
Testing	June and August

In other words, the development set includes documents from July 1995, July 1996, July 1997, etc. We chose the dev/test sets to be in the middle of the year so that the training set includes documents on both temporal sides of the test articles. We include years 1995-2001 and 2004-2006, but skip 2002 and 2003 due to their abnormally small size compared to the other years.

Finally, we experiment in a balanced data setting, training and testing on the same number of documents from each year. The test set includes 11,300 documents in each year (months June and August) for a total of 113,000 test documents. The development set includes 7,300 from July of each year. Training includes approximately 75,000 documents in each year with some years slightly less than 75,000 due to their smaller size in the corpus. The total number of training documents for the 10 evaluated years is 725,468. The full list of documents is online at *www.usna.edu/Users/cs/nchamber/data/timestamp*.

6 Experiments and Results

We experiment on the Gigaword corpus as described in Section 5. Documents are tokenized and parsed with the Stanford Parser. The year in the timestamp is retrieved from the document's Gigaword ID which contains the year and day the article was retrieved. Year mentions are extracted from documents by matching all tokens with exactly four digits whose integer is in the range of 1900 and 2100.

The MaxEnt classifiers are also from the Stanford toolkit, and both the document and year mention classifiers use its default settings (quadratic prior). The λ factor in the joint classifier is optimized on the development set as described in Section 4.3. We also found that dev results improved when training ignores the border months of Jan, Feb, and Dec. The features described in this paper were selected solely by studying performance on the development set. The final reported results come from running on the test set once at the end of this study.

Table 3 shows the results on the Test set for all document classifiers. We measure accuracy to compare overall performance since the test set is a balanced set (each year has the same number of test documents). **Unigram NLLR** and **Filtered NLLR** are the language model implementations of previous work as described in Section 3.1. **MaxEnt Unigram** is our new discriminative model for this task. **MaxEnt Time** is the discriminative model with rich time features (but not NER) as described in Section 3.3.2 (**Time+NER** includes NER). Finally, the **Joint** model is the combined document and year mention classifiers as described in Section 4.3. Table 4 shows the F1 scores of the Joint model by year.

Our new MaxEnt model outperforms previous work by 55% relative accuracy. Incorporating time features further improves the relative accuracy by

Model	Overall Accuracy
Random Guess	10.0%
Unigram NLLR	24.1%
Filtered NLLR	29.1%
MaxEnt Unigram	45.1%
MaxEnt Time	48.3%
MaxEnt Time+NER	51.4%
Joint	53.4%

Table 3: Performance as measured by accuracy. The predicted year must exactly match the actual year.

	95	96	97	98	99	00	01	02
P	.57	.49	.52	.48	.47	.51	.51	.59
R	.54	.56	.62	.44	.48	.48	.46	.57
F1	.55	.52	.57	.46	.48	.49	.48	.58

Table 4: Yearly results for the Joint model. 2005/06 are omitted due to space, with F1 .56 and .63, respectively.

7%, and adding NER by another 6%. Total relative improvement in accuracy is thus almost 77% from the Time+NER model over Filtered NLLR. Further, the temporal constraint model increases this best classifier by another 3.9%. All improvements are statistically significant ($p < 0.000001$, McNemar's test, 2-tailed). Table 6 shows that performance increased most on the documents that contain at least one year mention (60% of the corpus).

Finally, Table 5 shows the results of the temporal constraint classifiers on year mentions. Not surprisingly, the fine-grained performance is quite a bit lower than the core relations. The full Joint results in Table 3 use the three core relations, but the seven fine-grained relations give approximately the same results. Its lower accuracy is mitigated by the finer granularity (i.e., the majority class basline is lower).

7 Discussion

The main contribution of this paper is the discriminative model (54% improvement) and a new set of features for document dating (14% improvement). Such a large performance boost makes clear that the log likelihood and entropy approaches from previous work are not as effective as discriminative models on a large training corpus. Further, token-based features do not capture the implicit references to time in language. Our richer syntax-based features only apply to year mentions, but this small textual phenomena leads to a surprising 13% relative improvement in accuracy. Table 6 shows that a significant chunk of this improvement comes from documents containing year mentions, as expected.

	P	R	F1
Before Timestamp	.95	.98	.96
Same as Timestamp	.73	.57	.64
After Timestamp	.84	.81	.82
Overall Accuracy			**92.2%**
Fine-Grained Accuracy			**70.1**%

Table 5: Precision, recall, and F1 for the core relations. Accuracy for both core and fine-grained.

	All	With Year Mentions
MaxEnt Unigram	45.1%	46.1%
MaxEnt Time+NER	51.4%	54.3%
Joint	53.4%	57.7%

Table 6: Accuracy on all documents and documents with at least one year mention (about 60% of the corpus).

The year constraint learner also improved performance. Although most of its features are in the document classifier, by learning constraints it captures a different picture of time that a traditional document classifier does not address. Combining this picture with the document classifier leads to another 3.9% relative improvement. Although we focused on year mentions here, there are several avenues for future study, including explorations of how other types of time expressions might inform the task. These constraints might also have applications to the ordering tasks of recent TempEval competitions.

Finally, we presented a new evaluation setup for this task. Previous work depended on having training documents in the same week and day of the test documents. We argued that this may not be an appropriate assumption in some domains, and particularly problematic for the news genre. Our proposed evaluation setup instead separates training and testing data across months. The results show that log-likelihood ratio scores do not work as well in this environment. We hope our explicit train/test environment encourages future comparison and progress on document dating.

Acknowledgments

Many thanks to Stephen Guo and Dan Jurafsky for early ideas and studies on this topic.

References

Nathanael Chambers and Dan Jurafsky. 2008. Jointly combining implicit constraints improves temporal ordering. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*, Hawaii, USA.

W. Dakka, L. Gravano, and P. G. Ipeirotis. 2008. Answering general time sensitive queries. In *Proceedings of the 17th International ACM Conference on Information and Knowledge Management*, pages 1437–1438.

Angelo Dalli and Yorick Wilks. 2006. Automatic dating of documents and temporal text classification. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 17–22.

Franciska de Jong, Henning Rode, and Djoerd Hiemstra. 2005. Temporal language models for the disclosure of historical text. In *Humanities, computers and cultural heritage: Proceedings of the XVIth International Conference of the Association for History and Computing (AHC 2005)*.

Fernando Diaz and Rosie Jones. 2004. Using temporal profiles of queries for precision prediction. In *Proceedings of the 27th Annual International ACM Special Interest Group on Information Retrieval Conference*.

David Graff. 2002. English Gigaword. *Linguistic Data Consortium*.

Nattiya Kanhabua and Kjetil Norvag. 2008. Improving temporal language models for determining time of non-timestamped documents. In *Proceedings of the 12th European conference on Research and Advanced Technology for Digital Libraries*.

Nattiya Kanhabua and Kjetil Norvag. 2009. Using temporal language models for document dating. *Lecture Notes in Computer Science: machine learning and knowledge discovery in databases*, 5782.

W. Kraaij. 2004. *Variations on language modeling for information retrieval*. Ph.D. thesis, University of Twente.

Abhimanu Kumar, Matthew Lease, and Jason Baldridge. 2011. Supervised language modeling for temporal resolution of texts. In *Proceedings of CIKM*.

Xiaoyan Li and W. Bruce Croft. 2003. Time-based language models. In *Proceedings of the twelfth international conference on Information and knowledge management*.

Dolores M. Llidó, Rafael Llavori, and Mariá J. Aramburu. 2001. Extracting temporal references to assign document event-time periods. In *Proceedings of the 12th International Conference on Database and Expert Systems Applications*.

Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, David Day, Lisa Ferro, Robert Gaizauskas, Marcia Lazo, Andrea Setzer, and Beth Sundheim. 2003. The timebank corpus. *Corpus Linguistics*, pages 647–656.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Workshop on Semantic Evaluations*.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. 2009. The tempeval challenge: identifying temporal relations in text. *Special Issue: Computational Semantic Analysis of Language: SemEval-2007 and Beyond*, 43(2):161–179.

Katsumasa Yoshikawa, Sebastian Riedel, Masayuki Asahara, and Yuji Matsumoto. 2009. Jointly identifying temporal relations with markov logic. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Ruiqiang Zhang, Yi Chang, Zhaohui Zheng, Donald Metzler, and Jian yun Nie. 2009. Search result re-ranking by feedback control adjustment for time-sensitive query. In *Proceedings of the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.