A Burstiness-aware Approach for Document Dating

Dimitrios Kotsakos
University of Athens
dimkots@di.uoa.gr

Theodoros Lappas
Stevens Institute of Technology
tlappas@stevens.edu

Dimitrios Kotzias
University of Athens
dkotzias@di.uoa.gr

Dimitrios Gunopulos
University of Athens
dg@di.uoa.gr

Nattiya Kanhabua
L3S Research Center
Leibniz Universität Hannover
kanhabua@l3s.de

Kjetil Nørvåg
Norwegian University of Science and Technology
Kjetil.Norvag@idi.ntnu.no

ABSTRACT

A large number of mainstream applications, like temporal search, event detection, and trend identification, assume knowledge of the timestamp of every document in a given textual collection. In many cases, however, the required timestamps are either unavailable or ambiguous. A characteristic instance of this problem emerges in the context of large repositories of old digitized documents. For such documents, the timestamp may be corrupted during the digitization process, or may simply be unavailable. In this paper, we study the task of approximating the timestamp of a document, so-called document dating. We propose a content-based method and use recent advances in the domain of term burstiness, which allow it to overcome the drawbacks of previous document dating methods, e.g. the fix time partition strategy. We use an extensive experimental evaluation on different datasets to validate the efficacy and advantages of our methodology, showing that our method outperforms the state of the art methods on document dating.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

burstiness; language models; temporal similarity

1. INTRODUCTION

Temporal text mining is at the core of a large number of mainstream applications. The input to such applications consists of a collection of timestamped documents. The temporal dimension can be used for, among others, event detection [1], document search [6], and text summarization [8].

The assumption made by all such applications is that the timestamp of each document is both available and accurate. In practice, however, this assumption may be false. A characteristic instance of this problem emerges in the context of large repositories of old digitized documents. Such repositories are becoming increasingly large and abundant, due to initiatives such as The National Digital Newspaper Program[1]. For such documents, the timestamp may be corrupted during the digitization process, or may simply be unavailable.

In this paper, we propose a purely statistical method based on document content and burstiness for approximating the true timestamp of a given document in the context of very large timestamped document collections. We address the problem by considering two main factors, (i) lexical similarity and (ii) burstiness. The first factor captures the intuition that similar documents are more likely to discuss the same topics and events, and are thus more likely to be associated with adjacent timestamps. The second factor builds upon the intuition that bursts in term frequency capture the trends in vocabulary usage during the corresponding timeframe and can thus prove useful in document dating.

When an event takes place (e.g. an earthquake, sports finals), the event's characteristic terms (e.g. "earthquake", "shooting", "overtime") appear more frequently in the media. In the context of document dating, our intuition is that a timeframe when many terms of the query document are bursty is more likely to overlap with the document's timestamp. Our method is the first to utilize temporal information through a burstiness-aware approach, without depending on specific language rules, datasets, or meta-information.

Our contributions can be summarized as follows:

- We propose a purely statistical algorithm for estimating the timestamp of a document based on its content and burstiness.
- Our approach reports non-fixed periods of time, in contrast to previous approaches, that report one timeframe among the pre-segmented timespan of the reference corpus.
- We provide an extensive experimental evaluation, by using three different datasets spanning different time periods, showing that our method outperforms the precision of the state of the art methods.

2. RELATED WORK

The problem of document dating has proven to be difficult for the information retrieval community. The best published

[1] `http://www.loc.gov/ndnp`

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.

http://dx.doi.org/10.1145/2600428.2609495.

results do not achieve more than 50% precision, when estimating 1-year long intervals in corpora that span 10 years, whilst using purely statistical methods [2]. The underlying reason for this, is that not all documents contain temporal information, which makes a percentage of the corpus useless for testing and training purposes.

Previous approaches addressed the problem of document dating by identifying linguistic constructs with a clear temporal interpretation (e.g. the mention of the date or time). However, such tokens can be very sparse as well as ambiguous, referring to irrelevant timeframes. Another line of work overcomes this drawback, by considering the entire vocabulary of a document [3]. While this is a clear improvement, these methods are limited by their static consideration of the candidate timeframes, since they pre-segment the timeline into intervals of the same fixed length. A language model is then used to select the interval that is most likely to be the temporal origin of the query document.

Kanhabua and Nørvåg in [5] propose a document-dating method that extends the one proposed by de Jong et al. [3]. Specifically, the authors propose the application of semantic-based preprocessing of the reference collection and apply a term-weighting scheme. Chambers proposed a discriminative model, using a Maximum Entropy classifier, as well as defining rules for processing temporal linguistic features, as year mentions in documents [2]. While this model outperformed the methods proposed by de Jong and Kanhabua, it has the limitation that it only works well for *year* predictions, because temporal linguistic features that refer to months or days are ambiguous. Moreover, in this study there were no running time experiments, an issue that we address in the current paper. In contrast with all of the above approaches, our approach has not such requirements and can handle intervals of arbitrary length.

3. PROBLEM DEFINITION

Let $\mathcal{D}$ be a collection of documents spanning a timeline of $T = t_1, t_2, ..., t_n$ of n discrete timestamps (e.g. days). Each document $d \in \mathcal{D}$ is associated with exactly one timestamp $t(d) \in T$. Given a query document $q \notin \mathcal{D}$, for which the timestamp $t(q)$ is unknown, our goal is to find the most likely interval I of size ℓ, within T, so that $t(q)$ falls within I. Among other things, our approach considers the *burstiness* of the terms in the query document q. Given a term $x \in q$, by $\mathcal{B}(x)$ we represent the set of non-overlapping bursty intervals for x.

4. OUR APPROACH

Our approach considers (i) the lexical similarity of the query document with the documents in the reference collection $\mathcal{D}$ (ii) the burstiness of the significant terms of the query document q, e.g., top-k terms ranked by *tf-idf*. The use of lexical similarity captures the intuition that similar documents are more likely to discuss similar topics and events, and are thus more likely to originate in the same timeframe. In practice, however, similar documents may appear on different timestamps across the timeline. We address this, by introducing term burstiness. When an event or topic is recorded in a textual corpus, its characteristic terms exhibit atypically high frequencies. We refer to these timeframes as *bursty intervals*. Our algorithm is orthogonal to the actual mechanism used for computing non-overlapping bursty intervals. By identifying the bursty intervals of different terms, we can identify the timeframe of relevant events, as well as relevant documents that discuss them. A conceptual view of our approach is given by the example in Figure 1. In this case, the three documents d_2, d_3, d_4 will be selected by our algorithm, since they are both close to each other and overlap with multiple bursty intervals of the considered terms.

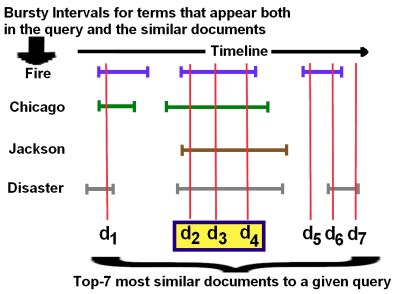

Figure 1: How `BurstySimDater` identifies the appropriate timestamp for a given query document.

We refer to our algorithm as `BurstySimDater`. The pseudocode is given in Algorithm 1.

Algorithm 1 `BurstySimDater`

Input: reference corpus $\mathcal{D}$, bursty intervals $\mathcal{B}$, query document q, max timeframe length ℓ
Output: timeframe of q

1: $\mathcal{S} \leftarrow$ top-k most similar documents to q from $\mathcal{D}$
2: $W_{\mathcal{S}} \leftarrow \emptyset$
3: **for** $d \in \mathcal{S}$ **do**
4: $\quad w_d \leftarrow 0$
5: $\quad \mathcal{Y} \leftarrow d \cap q$
6: $\quad$ **for** $x \in \mathcal{Y}$ **do**
7: $\quad\quad w_d \leftarrow w_d + |\{I \in \mathcal{B}(x) : t(d) \in I\}|$
8: $\quad w_d \leftarrow w_d / |\mathcal{Y}|$
9: $\quad W_{\mathcal{S}} \leftarrow W_{\mathcal{S}} \cup \{w_d\}$
10: $A_{\mathcal{S}} \leftarrow (d \in \mathcal{S}, W_{\mathcal{S}})$
11: $\mathcal{I} \leftarrow \texttt{GetMax}(A_S, \ell)$
12: **Return** $\mathcal{I}$

The input to the algorithm consists of the query document q, the reference corpus $\mathcal{D}$, the set of precomputed bursty intervals $\mathcal{B}$ and the upper bound on the reported timeframe ℓ. The output is an interval of length at most ℓ, within the timeline T spanned by $\mathcal{D}$.

First, the algorithm retrieves the top-k most similar documents to q from $\mathcal{D}$. In our own evaluation, we experimented, among others, with the *tf-idf* measure and the Jaccard similarity. We use the latter in the experimental section of this paper, since it led to the best results. We refer to the retrieved set of the k most similar documents as $\mathcal{S}$.

In steps 2-9, we assign a weight w_d to each document in $d \in \mathcal{S}$, based on its overlap with the burstiness patterns of its terms. Initially, w_d is set to zero. Let $\mathcal{Y}$ be the overlap of d's vocabulary with the vocabulary of the query document q. For each term $x \in \mathcal{Y}$, let $\mathcal{B}(x)$ be the precomputed set of bursty intervals for x. We then increment w_d by the number of the intervals from $\mathcal{B}(x)$ that actually contain $t(d)$. After the iteration over all terms in $\mathcal{Y}$ is complete, we normalize w_d by dividing it by $|\mathcal{Y}|$. Conceptually, the weight w_d of a document d is the average number of bursty intervals that

Table 1: Description of the datasets

#	Dataset	Start Date	End Date	# docs
1	**NYT10**	01/01/1987	12/31/1996	665,741
2	**NYT1987**	01/01/1987	12/31/1987	73,279
3	**SF-Call1**	01/01/1903	12/31/1904	144,289
4	**SF-Call2**	01/01/1908	12/31/1909	153,412
5	**TopixAll**	01/01/2008	12/31/2008	65,540
6	**TopixCanada**	01/01/2008	12/31/2008	3,326
7	**TopixSAfrica**	01/01/2008	12/31/2008	2,389

it overlaps with, computed over all the terms that it has in common with the query q. The computed weights are kept in the set $\mathcal{W}_{\mathcal{S}}$. We want to identify the interval when the most terms from the top-k similar documents are *simultaneously* bursty. This period is the intersection of intervals with the maximum sum of weights. To do this, in steps 10-11, we create an array $A_{\mathcal{S}}$ of size T, where cell i equals to the sum of weights w_d for all documents $d \in \mathcal{S}$ that were written at t_i. Next, we find the interval $\mathcal{I}$ of length ℓ with the maximum sum. By tuning ℓ, we tune the level of desired accuracy. In order to compute the sets of bursty intervals we use the `GetMax` algorithm [6]. Given a discrete time series of frequency measurements, `GetMax` returns a set of non-overlapping bursty intervals with respect to the frequencies.

5. EXPERIMENTS

For our experimental evaluation we used three real-world news datasets, each of them being a chronologically ordered sequence of documents. Table 1 describes each dataset in detail. Datasets $1, 2$ are parts of the New York Times[2] dataset, datasets $3, 4$ are articles from *The San Francisco Call* newspaper and datasets $5, 6, 7$ are articles from the website `Topix.com`, which host news articles from 181 countries. After POS tagging and Word Filtering for all competing methods, we kept only nouns, verbs and adjectives. After carefully examining the relevant literature on document dating, we chose to compare with the following methods:

`MaxEnt`: The algorithm proposed by Chambers in [2] trains a discriminative version of a Maximum Entropy classifier. We used the `MaxEnt` classifiers from the freely available Stanford toolkit, as was done in the original paper.

`NLLR`: The algorithm proposed by de Jong et al. [3] and extended by Kanhabua and Nørvåg [5] initially splits the timeline to segments of fixed (and equal) length (e.g. weeks). It then uses temporal language modeling to compare the vocabulary between each query document and the segments, in order to choose the most likely segment.

In `BurstySimDater` experiments we used $k = 10$ most similar documents. Changing this parameter did not result in a big difference in precision. In `MaxEnt` and `NLLR` we used all unigrams features. As proposed in the respective papers [2, 5] and was validated in our experiments, performance of `MaxEnt` and `NLLR` is best when all features are used. We evaluate all approaches on each of the available datasets via a 10-fold cross validation, omitting the known timestamp of a query document.

5.1 Scalability Experiments

In this experiment we applied the three methods on various random samples of increasing size from **NYT10**, which is the largest document collection. We experimented with various timeframe lengths $\ell = 1, 6, 12$ months. Figure 4 depicts the total running time for each method as a function of the sample size. X-axis illustrates the total size of the dataset, 90% of which serves as training- and 10% as testing-set. The total running time for `NLLR` includes partitioning of the dataset, indexing of the documents and building the language models for each partition. The total running time for `MaxEnt` includes the training of the Maximum Entropy Classifier. The total running time for `BurstySimDater` includes indexing of the documents and computation of the bursty intervals for all terms in the corpus. Moreover, all running time values include the computation of the reported intervals for all testing documents.

As depicted in Figure 3 `BurstySimDater` achieves the best precision for all dataset sample sizes and all timeframe lengths ℓ. More importantly, in terms of total running time `BurstySimDater` scales much better than `MaxEnt` and is directly comparable to `NLLR`. Due to the computational complexity of `MaxEnt` some of the experiments did not terminate in a reasonable amount of time.

Figure 2 illustrates the difference in total running times of the three methods for a multiple query experiment. More specifically, for each document three intervals of respective lengths $\ell = 1, 6, 12$ months were desired. The reason for the depicted running time difference is that `BurstySimDater` algorithm does not need to repeat the indexing of documents and the computation of the bursty intervals in order to evaluate the three different queries, whereas `NLLR` and `MaxEnt` need to re-partition the timeline into segments of length $\ell = 1, 6$ and 12 months. This is another benefit for not pre-partitioning the timeline into fixed-length segments.

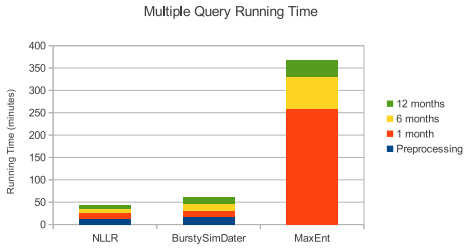

Figure 2: A multiple query experiment on a 60% sample of the NYT10 dataset yields the depicted total running time for the three approaches.

5.2 Precision Experiments

In this experiment we evaluate and compare the precision values achieved by `BurstySimDater`, `MaxEnt` and `NLLR` in all datasets and settings. In order to measure the precision values as a function of (the length of the target timeframe) ℓ, we experimented on **NYT10**, which is our largest dataset. Both `MaxEnt` and `NLLR` algorithms require a pre-segmented timeline in intervals of length ℓ. Our `BurstySimDater` algorithm has no such requirement. Instead, ℓ is provided as an *upper bound* of the reported timeframe. We tune ℓ so that the results of the competing approaches are directly comparable. We evaluate the approaches for $\ell \in [4, 12, 24, 48]$ weeks.

Table 2 contains all achieved precision values for all methods. As described above, the experiments for `MaxEnt` algorithm did not terminate in reasonable time for target time-

[2] `http://catalog.ldc.upenn.edu/LDC2008T19`

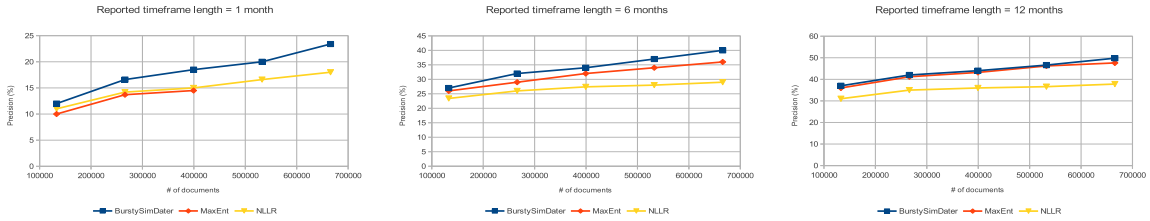

Figure 3: Comparison of precision values for the three methods vs. sample size for the NYT10 dataset. Timeframe length: *1-month*, *6-months* and *12-months*.

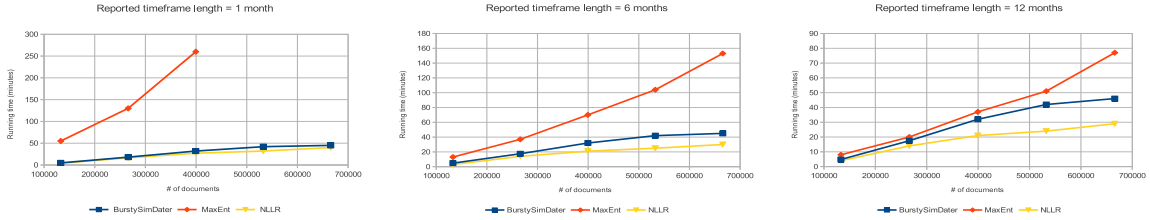

Figure 4: Comparison of total running time for the three methods vs. sample size for the NYT10 dataset. Timeframe length: *1-month*, *6-months* and *12-months*.

Table 2: Precision (%) for NYT10 dataset

Timeframe Length	NLLR	MaxEnt	BurstySimDater
1 month	18	-	**23.4**
3 months	24	-	**32**
6 months	25	36	**40**
1 year	38.4	48.6	**49.8**

Table 3: Precision for 1 month in 1 or 2 year(s)

Dataset	NLLR	MaxEnt	BurstySimDater
NYT1987	29	24	**32**
SF-Call1	35	**38.5**	**38.6**
SF-Call2	29	**36**	34
TopixCanada	44.5	61.8	**63**
TopixSAfrica	75	81	**84**

frames of length $\ell = 4$ and 12 weeks. `BurstySimDater` not only outperforms the state of the art methods in all timeframe lengths, but also this difference in precision increases as the number of candidate time intervals becomes larger (Figure 3, e.g. for target timeframe length $\ell = 1$ month).

Table 3 depicts all precision values for $\ell = 1$ month for all 1 or 2 year datasets. This experiment also demonstrates the challenging nature of the problem: while some of the documents discuss specific events, others simply discuss topics that are not relevant to current events and can thus be associated with any timestamp. `BurstySimDater` algorithm outperforms `NLLR` in all datasets for all values of ℓ, while achieving similar values to `MaxEnt`, which in turn has the scalability problems analyzed in Section 5.1. Another interesting observation comes from the results on the **TopixCanada** and **TopixSAfrica** datasets. For this corpora, the achieved precision values were significantly higher for all methods, reaching up to 63% and 84% respectively. This verifies our intuition that spatial information can be utilized to improve the results of our document-dating algorithm.

6. CONCLUSION AND FUTURE WORK

In this paper we introduced a new approach for document dating that does not depend on temporal linguistic constructs and can report timeframes of arbitrary length. In addition it clearly outperforms the state of the art in terms of running time complexity and precision, over a variety of real datasets. As future work, we will apply our approach to a document retrieval problem in the context of temporal search and leverage the determined timestamp of documents in building the temporal profile [4, 7] of a query in order to improve the overall retrieval effectiveness.

7. ACKNOWLEDGMENTS

This work has been co-financed by EU and Greek National funds of the NSRF - Research Funding Programs: "Heraclitus II fellowship, ARISTEIA - MMD", the EU funded project INSIGHT and the ERC Advanced Grant ALEXANDRIA.

8. REFERENCES

[1] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proceedings of SIGIR'1998*.

[2] N. Chambers. Labeling documents with timestamps: Learning from their time expressions. In *Proceedings of ACL'2012*.

[3] F. de Jong, H. Rode, and D. Hiemstra. Temporal language models for the disclosure of historical text. In *Proceedings of AHC'2005*.

[4] R. Jones, and F. Diaz. Temporal Profiles of Queries. In *ACM Trans. Inf. Syst.*, 2007.

[5] N. Kanhabua and K. Nørvåg. Improving Temporal Language Models For Determining Time of Non-Timestamped Documents. In *Proceedings of ECDL'2008*.

[6] T. Lappas, B. Arai, M. Platakis, D. Kotsakos, and D. Gunopulos. On burstiness-aware search for document sequences. In *Proceedings of SIGKDD'2009*.

[7] M.-H. Peetz, E. Meij, and M. de Rijke. Using temporal bursts for query modeling. In *Inf. Retr.*, 2014.

[8] X. Wan. TimedTextRank: adding the temporal dimension to multi-document summarization. In *Proceedings of SIGIR'2007*.