Learning to Identify Historical Figures for Timeline Creation from Wikipedia Articles

Sandro Bauer[1], Stephen Clark[1], Thore Graepel[2]

[1] University of Cambridge, Cambridge, United Kingdom
`firstname.lastname@cl.cam.ac.uk`
[2] Microsoft Research Cambridge, Cambridge, United Kingdom
`thore.graepel@microsoft.com`

Abstract. This paper addresses a central sub-task of timeline creation from historical Wikipedia articles: learning from text which of the person names in a textual article should appear in a timeline on the same topic. We first process hundreds of timelines written by human experts and related Wikipedia articles to construct a corpus that can be used to evaluate systems that create history timelines from text documents. We then use a set of features to train a classifier that predicts the most important person names, resulting in a clear improvement over a competitive baseline.

1 Introduction

Creating a timeline that contains the most important events in a country's history from a history article can be seen as an instance of text summarisation; while the most prominent events in the country's history will figure in the timeline, less important details are likely to be ommitted. Timelines of this kind would be useful tools in education, for instance, in particular with the emergence of web applications that let users learn about the history of the world in new, interactive ways. Our goal is to use historical information available in Wikipedia to construct these timelines automatically.

This paper examines an important sub-problem of this task. We try to learn which historical figures from a textual article (on a country, a field of science, or similar) should be included in the corresponding history timeline. Our work differs from previous approaches in that we use a corpus of 279 gold-standard timelines and related Wikipedia articles. The material used for this purpose includes timelines created by human experts (on concepts such as *History of New York*), as well as textual Wikipedia articles about the same concepts. This data is then used to train a classifier that predicts, for a set of textual articles, which person names should appear in the corresponding timeline. We use a diverse set of features and obtain a clear improvement over a competitive frequency-based baseline. This study will provide a basis for more complex systems that identify events relevant to the history of a concept.

2 Timeline corpus construction

We have processed about 600 timelines written by human experts and an equal number of sets of related textual history articles. This corpus can be used beyond the task described in this paper; it will make it possible to evaluate future automatic timeline construction systems that construct entire timelines from a set of history articles.

The history articles are Wikipedia pages whose titles start with “History of”, for example “History of Iran” or “History of biology”. When there was no such article available, we instead used the concept’s main article. When the title of a timeline mentions two concepts, for instance “Timeline of biology and organic chemistry”, but there is no single textual article on the same two topics, we use more than one textual article (in this case, “History of biology” and “Organic chemistry”). A timeline is a set of sentences, each of which is associated with a date, e.g. “2008” or “05/03/2009”. The timelines come from Wikipedia and other history websites: *HistoryWorld*[3], www.timelines.ws, *WorldAtlas*[4], *HistoryMole*[5], the BBC and datesandevents.org. Where there are multiple timelines available for a concept, we take the union of all timeline entries, since it is assumed that all entries present in *any* of the timelines on a topic are relevant to that topic.

For addressing the sub-problem of learning important person names, we first extract all `PERSON` named entities from the timelines and the corresponding textual articles in order to construct a gold-standard data set of names. There are two problems that complicate the creation of this gold standard: First, the same people are often referred to in different ways in a timeline and a textual article. For example, the names in the timeline might not contain umlauts (“Schroder”), but those in the Wikipedia article do (“Schröder”). Similarly, a person might be mentioned in the same article using more than one name (e.g. “Friedrich I.” and “Fritz”). Second, it is not obvious when a history article “corresponds” well enough to a timeline to be used as training data. The following subsections describe how we address these problems.

2.1 Creating a gold standard of person names

We start by extracting all `PERSON` NEs using the Stanford NE classifier [5]. Mentions that have only one entity candidate in the YAGO knowledge base [11] are filtered out if that candidate is not a person; this corrects for obvious cases where a named entity has been misclassified as a person.

In what follows, `TIML` designates a timeline and `TXTA` the corresponding textual article(s). For each `TIML` and `TXTA`, we infer a collection of *name sets*. A name set is a set of strings that represent the same entity, such as {*Friedrich I*, *Frederick I.*, *Barbarossa*}. To construct this set, we start with the multi-word names (such as *Pete Sampras*). Each of these names is expanded into a name set

[3] www.historyworld.net
[4] www.worldatlas.com
[5] www.historymole.com

by adding versions with all umlauts removed, all diacritics removed, all titles of nobility removed, and so forth. If the name string has only one candidate entity in YAGO that is a person, we add all the *alternative names* contained in YAGO for that entity too. Finally, we check whether any of the names in the current name set are contained in one of the name sets already constructed. If this is the case, the two sets are merged.

Once we have a set of such name sets for each `TIML` and `TXTA`, we need to infer which name set in `TIML` corresponds to which one in `TXTA` (if any). We use a method which ensures that whenever a name set from the timeline and one from the textual article have a name in common, the two sets are linked to each other. For each name set tl from `TIML`, we check whether any name set txt_1 from `TXTA` contains one of the names in tl. If so, we create an overlap pair. If a second name set txt_2 in `TXTA` contains one of the names in tl, we merge the two name sets txt_1 and txt_2. The same applies in the opposite direction.

Note that we cannot simply use entity linking systems since some of the persons do not have a YAGO entry, and hence different versions of the same name can only be grouped together using the rules above. In addition, our scenario does not even require names to be disambiguated to canonical entities in a knowledge base. We merely want to avoid that two names referring to the same person are not matched only because, for example, some diacritics are missing. We found our heuristic to work well on the development set.

Next, we check whether any name with more than two tokens can be matched to a two-word name in a different name set; this way two names such as *David William Cameron* and *David Cameron* end up in the same name set. Finally, we add all single-word person names (such as "Cameron") to the name sets using a heuristic. For each short name in `TIML` and `TXTA` respectively, we obtain all name sets in both `TIML` and `TXTA` in which at least one of the multi-word names ends with the same last name. If there is more than one such name set, we pick the one with the highest total number of occurrences of its member names.

2.2 Finding high-quality pairs of timelines and textual articles

Another problem is that we cannot assume that a timeline on a particular subject is an accurate reflection of the textual article in our corpus, since it is not known which textual resources the timeline authors used when putting together the timeline. Some Wikipedia timelines do contain hints in the form of links to related textual articles (e.g., the Wikipedia timeline on French history contains the following sentence: "To read about the background to these events, see *History of France.*"), but even in these cases it is not clear that the timelines represent all of the underlying textual content. A timeline author who creates a timeline from a textual article might decide that a part of the article is not relevant at all to the target concept, or a timeline might cover a different time window than the corresponding textual article. For example, there are timelines that only contain events after 1900, or textual articles that cover very recent events, while the corresponding timeline was last updated two years ago. It is also possible that either the timeline or the textual article is biased towards a particular subject area; for example, a timeline on a country might be biased towards armed conflicts. To alleviate this problem, a set of filters is used to ensure that there is sufficient overlap between a timeline and a textual article. We require a minimum number of 5 persons from the textual page that also appear on the timeline. In addition, we require the proportion of persons from the textual article that are also in the timeline to be between 0.1 and 0.9; cases where very few or almost all of the people in the textual article are mentioned in the timeline should be avoided. In the former case, we expect that the timeline sentences are not a good representation of the textual article's content. In the latter case, the timeline might not focus on important events only.

Finally, two more constraints are used to filter out low-quality pairs. Firstly, we require an absolute mininum number of 5 persons on both the timeline and the textual page. This filters out cases where one of the documents is too short. Secondly, a timeline is not allowed to contain more than 5 times the number of persons in the corresponding textual article. This removes cases where the timeline is too fine-grained and hence not a succinct description of the most important events on a topic, but rather a detailed event log (such as what happened in the last minutes before the first airplane hit the World Trade Center). Applying all these constraints resulted in a set of 279 pairs of timelines and textual articles.

3 System description

The task of the system is to decide for each of the persons in the textual article(s) whether it should be mentioned in the timeline. To give an example, consider the following sentence taken from a Wikipedia article: "(...) the Hohenstaufen empire under *Frederick I Barbarossa* reached its peak in (...) the marriage of his son *Henry* (...) to (...) *Constance* (...)." *Frederick I Barbarossa* is present in the timeline, while *Henry* and *Constance* are not. To make these decisions, all mentions of all names in a name set such as {*Frederick I Barbarossa*, *Friedrich I*} are considered by the system.

In a first setting, a separate feature vector is created for each mention of one of the names in a name set. Alternatively, we use one combined feature vector for all mentions with the same name, where the feature values of overlapping features are either averaged or summed. At test time, a name set is accepted if at least 50% of all vectors receive a score above a threshold value tuned on the development set. As scores for the individual vectors we use the pseudo-probabilities output by LibLINEAR[6] (which uses a linear kernel).

4 Feature set

We experiment with features that take into account the local and global context of the mentions, as well as external knowledge available about the persons. This

[6] www.csie.ntu.edu.tw/ cjlin/liblinear/

Table 1. Structural features

frequencyOfEachNEPerTextualPage: the frequency of each name on the textual page(s)
frequencyOfEachNEPerTextualPage_dividedByAllNEs: the frequency of each name on the textual page(s) divided by the total number of NEs
wordIdInSentence: the position of the NE in the containing sentence (word index of the first word of the NE)
numberOfNEsInSentence: the number of NEs in the containing sentence
indexOfCurrentStanfordNE: the index of the NE in question in the list of all NEs in the current sentence
indexOfCurrentSentence: the index of the sentence that contains the NE in question in the list of all sentences of the current paragraph
indexOfCurrentParagraph: the index of the paragraph that contains the NE in question in the list of all paragraphs of the current subsection
numberOfDifferentWritings: the number of different writings of the current name (i.e. the size of the corresponding name set)
occursInHeader: whether or not the current NE is in the Wikipedia document's header

includes structural features intended to capture effects such as: historical figures central to a concept will appear more often in an article; or important names are unlikely to appear in a sentence with a high number of other names. We also investigate the influence of document-level structure; for instance, some names might be so important for the article's topic that they are mentioned in the document's header; or the most important figures tend to appear early on in a paragraph.

Next, we build n-gram features (uni- and bigrams in a window of 3 words around the mention), as well as syntactic features that capture effects such as whether important historical figures are more or less likely to occur in a subject position or in a list of names. The length of co-reference chains that a name is part of (given by the Stanford NER toolkit) is also used, since important people might be mentioned multiple times in the form of pronouns.

Finally, we use semantic features to exploit the fact that many of the person names in the Wikipedia articles are linked to the person's own article. For each such article, there is a corresponding entity in YAGO that has types associated with it. Types in YAGO can be divided into WordNet types (at the top of the hierarchy) and Wikicategory types. We construct one feature per WordNet type of the target entity; we do not use Wikicategory types since these are too fine-grained.

Further semantic features include co-occurrences of the person name (e.g. *Charles de Gaulle*) and the concept name (such as *France*) in the same sentence; for this, we consider all sentences in the same article as well as in the person's Wikipedia article (that of *Charles de Gaulle* in this case). There is also a feature that indicates whether the concept name occurs in one of the section headers of the person's article.

Table 2. Linguistic features

unigrams: all unigrams (lemmatised) in a window size of +/- 3 around the mention, e.g. *abdicate*
unigrams_dist: all unigrams (lemmatised) in a window size of +/- 3 around the mention, annotated with the distance in words from the NE, .g. *abdicate_+1*
bigrams: all bigrams (lemmatised) in a window size of +/- 3 around the mention, e.g. *abdicate_after*
bigrams_dist: all bigrams (lemmatised) in a window size of +/- 3 around the mention, annotated with the distance in words from the NE, e.g. *abdicate_after_+1*
postags: all POS tags in a window size of +/- 3 around the mention, e.g. *VBD*
postags_dist: all POS tags in a window size of +/- 3 around the mention, annotated with the exact distance, e.g. *VBD_-1*
dep_degree1_in: the types of all incoming dependency edges of any word that is part of the NE (e.g. *dobj*)
dep_degree1_out: the types of all outgoing dependency edges of any word that is part of the NE (e.g. *xcomp*)
dep_degree2_in: the types of all incoming dependency chains of length 2 starting from any word that is part of the NE (e.g. *dobj_det*)
dep_degree2_out: the types of all outgoing dependency chains of length 2 starting from any word that is part of the NE (e.g. *xcomp_det*)
dep_is_local_or_global_subject: indicator features that indicate whether any of the words in the NE are the subject of any clause of the current sentence or of the main clause
dep_is_local_or_global_object: indicator features that indicate whether any of the words in the NE are the object of any clause of the current sentence or of the main clause
dep_numOfsisterDeps: the number of "sister dependencies" of the same type (i.e. dependencies of the same type starting from a "parent token" of the current token in the dependency graph). This measures how long the list of names is that a name occurs in), and categorical features that fire if there are at least one, two or three sister dependencies.
part_of_corefchain: a binary feature that fires if the current name is part of a coreference chain
part_of_corefchain_longestChain: the length of the longest co-reference chain the current name mention is part of
part_of_corefchain_atleast5sentences: a feature that fires if the longest co-reference chain is longer than 5 sentences
part_of_corefchain_atleast_x_InBetweenSentence: features that fire if there are at least x sentences between the sentences of the co-reference chain involving a person name
part_of_corefchain_lengthOfBiggestGapAtLeast_x: features that fire if there are at least x sentences between any two sentences of the co-reference chain involving a person name
part_of_corefchain_distanceFromFirstToLastSentenceAtLeast_x: the distance (in number of sentences) between the first and last sentences of any co-reference chain the current mention is part of

Table 3. Semantic features

conceptName_thisSentence: a feature that indicates whether the name of the concept (we remove "History of" from the article title, so hat concept names are "Germany" or "Biology") co-occurs with the name of the person in the same sentence **conceptName_allSentences**: a feature that indicates whether the name of the concept co-occurs with the name of the name of the person in **any** sentence in the textual document
entity_linked_to: a feature that fires if the name is linked to the person's Wikipedia article
sentenceOnTargetArticle_containsConceptName: a feature that indicates whether the name of the concept co-occurs with the name of the name of the person in **any** sentence in the person's Wikipedia article (if the name is linked to this article) **section_or_subsectionOnTargetArticle_containsConceptName**: a feature that indicates whether the name of the concept occurs in a section or subsection title in the person's Wikipedia article (if the name is linked to this article)

5 Experiments

The 279 timelines which pass the areforementioned criteria are divided into a development set of 20 timelines (with an average of 81.9 name sets per timeline), a test set of 30 timelines (with an average of 83.7 sets) and a training set comprising the remaining 229 timelines. In order to ensure that the various sets contain timelines that are comparable in terms of how well they represent the textual article, we first divide the timelines into ten buckets based on what percentage of the name sets from the timeline is also present in the textual article, and then pick timelines from each of the buckets at random until the test and development sets are full. The rest becomes the training set.

Two different sets of metrics are used to evaluate our approach. First, we calculate precision, recall and F-score values across all timelines (micro-averaged), based on the tuned threshold value of 0.6. Since there are some names in the gold standard timeline that do not appear in the textual article, we restrict our evaluation to those that are contained in the article.

Alternatively, the task can be interpreted as trying to assign higher scores to those names from a textual document that occur in a timeline on the same topic than to those that are not contained in the timeline. To obtain an indication of the quality of the ranking provided by the pseudo-probabilities, we calculate Average Precision (AP) scores for each timeline individually, and from this an aggregate MAP score across all timelines.

Evaluating our system is complicated by the errors made by the Stanford NER system. For instance, there are named entities that are misclassified as persons. Therefore, a second evaluation setting is used where erroneous `PERSON` instances are removed manually; we do not investigate which person names the tagger missed out on since we are primarily interested in the goodness of the ranking for the person names detected.

Table 4. Results

Set and system		P	R	F1	MAP
DV+fp	SVM	0.5430	0.6619	0.5966	0.6454
DV+fp	BL	0.4916	0.4905	0.4911	0.5813
DV-fp	SVM	0.5347	0.6616	0.5914	0.6658
DV-fp	BL	0.5052	0.4924	0.4987	0.6020
TE+fp	SVM	0.4551	0.6391	0.5316	0.6002
TE+fp	BL	0.3379	0.6116	0.4353	0.5245

An obvious choice for a baseline is the number of times a person occurs on the textual page. The ranking is provided by the total frequency of all names in the name set and is compared to the one output by the classifier. To resolve ties, we order all name sets with the same frequency count by the position where the first mention occurs in the article; in other words, the baseline ranking assumes that persons mentioned early on in a document are more important. For deciding whether a name set is accepted by the frequency baseline we similarly tune a threshold value on the development set (a proportion of the maximum number of times that the names of any single name set occur in the textual article; here set to 0.1).

6 Results

Results for the setting where one feature vector is constructed per mention are given in Table 4; "-fp" and "+fp" indicate whether erroneous `PERSON` instances are removed or retained; DV and TE refer to the development and test sets, respectively. Whether we use separate feature vectors for each NE or combined vectors does not influence the performance too much. Using the frequency baseline leads to competitive results; many important names occur more than once in an article, while others mostly occur only once. In all cases, the classifier achieves an improvement of more than 5% over the baseline. We also experimented with non-linear kernels, but this did not improve the results.

7 Related work

We are not aware of any existing work that has addressed the problem of constructing historical timelines from free-form text using a large corpus of timelines and textual documents on the same topic in the way we are proposing.

Some existing work on timeline generation is based on temporal information extraction techniques. Events and timestamps are first extracted from unstructured text in the form of TimeML `EVENT` [9] and `TIMEX` instances; the task of extracting the timestamps is usually referred to as *time normalisation* [2]. Relations among these events, such as `BEFORE`, `AFTER` and `CONTAINS` are then learned using a variety of approaches. This has been the focus of the TempEval shared tasks (see [12] for the most recent one) which are all based on the TimeML language. Kolomiyets et al. [7] are the first to learn a global timeline for a document based on such information extraction techniques. They use dependency parsing algorithms to arrange all the events in a document along a timeline. All these approaches, however, do not discuss the selection of important events for a particular downstream application such as the creation of a history timeline.

A second strand of research deals with timeline generation from news streams based on a user query. Chieu and Lee [4] apply extractive summarisation techniques to a collection C of documents to generate a timeline containing the top n most relevant events on a user query. They ask annotators to do their own research to produce a timeline on a given query, such as Bush (they are free to use whatever resource they like). Yan et al. [13] use gold-standard timelines created by professional editors and define four desirable criteria for timelines (relevance to a query, coverage, relevance, coherence). As many other similar works, they have access to a large number of documents on a single topic and the creation date for each document. Nguyen et al. [8] use a corpus of newswire texts along with timelines authored by journalists to extract salient events from news streams; they too make use of temporal processing techniques based on the TimeML framework. Their system can exploit co-occurrences of terms in multiple documents to identify salient events. Our task is more difficult in that the algorithm has to identify important entities from the local context alone.

A number of approaches focus on placing search engine results to a given user query (e.g. *Barack Obama*) along a timeline. They assume that important events appear frequently. Some approaches have used temporal information extraction techniques to process temporal information available in retrieved text documents [1]. The most closely related work is that of Chasin et al. [3]; they address the problem of creating history timelines from Wikipedia articles by classifying TimeML events into more and less important ones. The authors obtain human relevance judgments for events contained in a small number of textual documents, which is costly; they also point out that reaching inter-annotator agreement is difficult. Our study, on the other hand, is based on a novel, automatically constructed corpus of gold-standard timelines created by humans.

Finally, a great amount of work has gone into visualising information that is readily available in the form of time-labelled events [10, 6].

8 Conclusion

In this paper, we have addressed a sub-task central to the goal of creating informative history timelines from Wikipedia articles automatically: inferring the most important person entities that should be contained in the timeline. For this, we have created a corpus of timelines and textual history articles which can also be used to evaluate systems that construct complete timelines. For the person detection task, it has been demonstrated that using a variety of features results in a clear improvement over a competitive baseline. We are going to use this study as a building block in an algorithm that constructs full timelines.

Acknowledgments. SB is funded by a Microsoft Research PhD Scholarship.

References

1. Alonso, O., Gertz, M., Baeza-Yates, R.: Clustering and Exploring Search Results Using Timeline Constructions. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management. pp. 97–106. CIKM '09, Hong Kong, China (2009)
2. Bethard, S.: A Synchronous Context Free Grammar for Time Normalization. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, (EMNLP 2013). pp. 821–826 (2013)
3. Chasin, R., Woodward, D., Witmer, J., Kalita, J.: Extracting and Displaying Temporal and Geospatial Entities from Articles on Historical Events. The Computer Journal pp. 403–426 (2013)
4. Chieu, H.L., Lee, Y.K.: Query Based Event Extraction Along a Timeline. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 425–432. SIGIR '04, Sheffield, United Kingdom (2004)
5. Finkel, J.R., Grenager, T., Manning, C.: Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. pp. 363–370. ACL '05, Ann Arbor, Michigan, USA (2005)
6. Hienert, D., Luciano, F.: Extraction of Historical Events from Wikipedia. CoRR abs/1205.4138 (2012), http://dblp.uni-trier.de/db/journals/corr/corr1205.html#abs-1205-4138
7. Kolomiyets, O., Bethard, S., Moens, M.: Extracting Narrative Timelines as Temporal Dependency Structures. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics. pp. 88–97 (2012)
8. Nguyen, K.H., Tannier, X., Moriceau, V.: Ranking Multidocument Event Descriptions for Building Thematic Timelines. In: Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers. pp. 1208–1217. Dublin, Ireland (2014)
9. Saurí, R., Knippen, R., Verhagen, M., Pustejovsky, J.: Evita: A Robust Event Recognizer for QA Systems. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing. pp. 700–707. HLT '05, Vancouver, British Columbia, Canada (2005)
10. Sipoš, R., Bhole, A., Fortuna, B., Grobelnik, M., Mladenić, D.: Demo: HistoryViz — Visualizing Events and Relations Extracted from Wikipedia. In: Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications. pp. 903–907. ESWC 2009, Heraklion, Crete, Greece (2009)
11. Suchanek, F.M., Kasneci, G., Weikum, G.: YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia. In: Proceedings of the 16th International Conference on World Wide Web. pp. 697–706. WWW '07, Banff, Canada (2007)
12. UzZaman, N., Llorens, H., Derczynski, L., Allen, J., Verhagen, M., Pustejovsky, J.: SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In: Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013). pp. 1–9. Atlanta, Georgia, USA (2013)
13. Yan, R., Wan, X., Otterbacher, J., Kong, L., Li, X., Zhang, Y.: Evolutionary Timeline Summarization: A Balanced Optimization Framework via Iterative Substitution. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 745–754. SIGIR '11, Beijing, China (2011)