MPI-INF at the NTCIR-11 Temporal Query Classification Task

Robin Burghartz
Max Planck Institute for Informatics,
Saarbrücken, Germany
rburghar@mpi-inf.mpg.de

Klaus Berberich
Max Planck Institute for Informatics,
Saarbrücken, Germany
kberberi@mpi-inf.mpg.de

ABSTRACT

MPI-INF participated in the Temporal Query Intent Classification Task (TQIC) of the Temporalia track at NTCIR-11. This paper describes our approach to address this specific task. Our overall strategy has been to rely on established off-the-shelf components (e.g., standard classifiers from Weka and natural language processing methods from Stanford CoreNLP) and focus on feature engineering. Devised features include *surface* (e.g., n-grams), *linguistic* (e.g., capturing whether the query is a question), and *temporal* (e.g., statistics about publication dates and temporal expressions). We provide details on their precise definition and report on their effectiveness.

Keywords

Temporal Query Classification, Temporal Information Retrieval

Team Name

MPII

Subtasks

TQIC

1. INTRODUCTION

Document collections are increasingly longitudinal and contain documents published over extended periods of time. This is a direct result of affordable storage, an increased awareness that born-digital contents are worth keeping, and efforts to digitize old contents published long ago. Examples of longitudinal document collections include newspaper archives, social media, but also today's Web, which now contains web pages published decades ago. Time matters when searching such document collections and research in temporal information retrieval [6, 9] has looked into how temporal aspects of queries and documents can be leveraged to improve retrieval effectiveness.

Temporal query intent classification, as one task in temporal information retrieval pioneered by Jones and Diaz [13], seeks to determine whether the user's information need has a *temporal dimension*. The TQIC task of NTCIR-11 Temporalia [12] takes this idea further and also aims at determining the *temporal orientation* of the user's information need, i.e., whether she is interested in information from/about the past, present, or future. This is useful when searching longitudinal document collections, for instance, to automatically select a temporal retrieval model [7, 10, 16, 18] or inform it about the user's time period of interest.

The TQIC task is challenging mainly because there is little input from users: they cast their information needs into keyword queries, so that we are typically left with only 3-5 query keywords and the time when the query was issued. With this brevity comes ambiguity – we could sometimes only guess the information need behind a query and our interpretations differed. Also, it limits the applicability of certain methods – we found, for instance, dependency parsing not to work reliably on short query strings.

When setting out to work on the TQIC task, we deliberately decided to focus on feature engineering and rely on established off-the-shelf components, including Weka [4] for machine learning and Stanford CoreNLP [1] for natural language processing. To counter the information paucity mentioned above, we eagerly pull in information from other data sources, which include the LivingKnowledge corpus and the Open Directory Project (DMOZ). The features that we devised cover a broad spectrum and include:

- *surface features* such as lemmatized and non-lemmatized n-grams that are present in the query;
- *lexical features* derived from two lexical resources signal temporal trigger words (e.g., upcoming or recent);
- *temporal features* derived from publication dates of pseudo-relevant documents as well as temporal expressions from their contents;
- *collection features* capture the selectivity of query keywords and how pseudo-relevant documents differ from the general document collection;
- *linguistic features* indicating, for instance, whether the query constitutes a natural language question or contains a personal pronoun;
- *topical features* encoding the query's broad topic in the DMOZ topic directory.

This research has mostly been carried out as part of the first author's B.Sc. thesis project. More details than in this paper can thus be found in Burghartz [8].

Outline. The rest of this paper is organized as follows. Section 2 gives a concise overview of existing prior work. Our overall approach is outlined in Section 3, before detailing on devised features in Section 4. Experiments and insights from them are the subject of Section 5. Finally, we conclude in Section 6.

2. RELATED WORK

We now briefly discuss the related work that we build upon. Our focus is on query classification and retrieval models; a more complete recent survey of temporal information retrieval is given by Campos et al. [9].

Query Classification. The most influential attempt at temporal classification of queries comes from Jones and Diaz [13] who classify queries into the three classes *atemporal*, *temporally unambiguous* and *temporally ambiguous*. For this task they consider the distribution of retrieved documents over time and create meaningful features based on this distribution. Their classification is then done on this feature set. As an example, the time distribution of retrieved documents for the query "iraq war" has distinct peaks at the years 1991 and 2003 and is therefore classified as *temporally ambiguous*, while a query like "poaching" is classified as *atemporal* since its time distribution does not exhibit any peaks. A weakness of their approach is that only *publication times* of documents are considered. Often this publication time differs from the actual *content time*. Other ideas for implicitly temporal queries were developed by Metzler et al. [17]. By analyzing query logs they investigate the automatic detection of *implicitly year qualified* queries, i.e., queries that refer to an event in a specific year without containing the year in the query string.

Retrieval Models. For *explicit* temporal intents, Berberich et al. [7] develop an approach to find temporally relevant documents if a query containing an explicit temporal expression (as for example "2001") was issued by the user. An innovation of this paper is that, in contrast to former work, it addresses *uncertainty* in the temporal expression. In this context, uncertainty refers to the imprecision that arises when there are two temporal expressions that both refer to the same time but are slightly different, as in the case where one is more vague than the other. Consider as an example the aforementioned expression "2001". This expression can refer to different time intervals: to the complete year, to some interval of the year (e.g., April to November) or to some particular day (e.g., 9/11) - it depends on the surrounding context. Now if we issue the query "new york terrorist attacks 2001", we are very likely especially interested in documents containing the date 09/11/2001, but this date is different from the temporal expression we used in our search. Even though that work does not deal with query classification, it is still related to the topic of our work because it is one of few works which determine the *content time*. Kanhabua et al. [14] follow up on this idea and combine *publication time* and *content time* by introducing a machine learning approach that chooses one of the two retrieval models based on what is probably most suitable for the given query.

3. APPROACH

Our overall approach to address the TQIC task is to rely on well-established classification algorithms, namely Naïve Bayes and J48 Decision Trees as implemented in the Weka [4] machine learning library and focus on feature engineering.

Most of our features make direct use of the query itself (e.g., its linguistic properties) or properties of pseudo-relevant documents (e.g., publication dates and temporal expressions) retrieved from the LivingKnowledge (LK) document collection using a unigram language model. As external data sources we draw in two thesauri from the Web, which inform us about words likely to indicate a temporal intent and its orientation, and the Open Directory Project (DMOZ) to determine the overall topic of the query. More details about the individual features are provided in the following Section 4.

4. FEATURE DESIGN

In this section, which forms the core of the paper, we provide details on the different features that we devised.

4.1 Surface Features

As a set of baseline features, we employ non-lemmatized and lemmatized unigrams and bigrams that occur in the training data. Lemmatization was performed using Stanford CoreNLP [1]. For unigrams, we keep all non-lemmatized and lemmatized ones and only filter out stopwords (e.g., of and and). For bigrams, we keep all non-lemmatized and lemmatized ones (including those that contain a stopword) but filter out bigrams corresponding to a temporal expression. Considering bigrams allows us to capture compounds such as exchange rate; including stopwords informs us about common phrases such as how to, which is likely to indicate an *atemporal* query. In total, we end up with 330 distinct surface features.

4.2 Lexical Features

We leverage two online thesauri [2, 3] to identify temporal trigger words, i.e., words that are strong indicators that a query falls into one of the categories *past*, *recent*, or *future*. To do so, we identify words listed as words related to past, recent, and future in the two lexical resources. Table 1 shows a subset of the temporal trigger words thus identified. We encode the presence of one of these temporal trigger words through the binary features **containsPastTrigger**, **containsRecencyTrigger**, and **containsFutureTrigger**.

past	*recent*	*future*
history	topical	prediction
ago	trendy	upcoming
past	recent	future

Table 1: Temporal trigger words

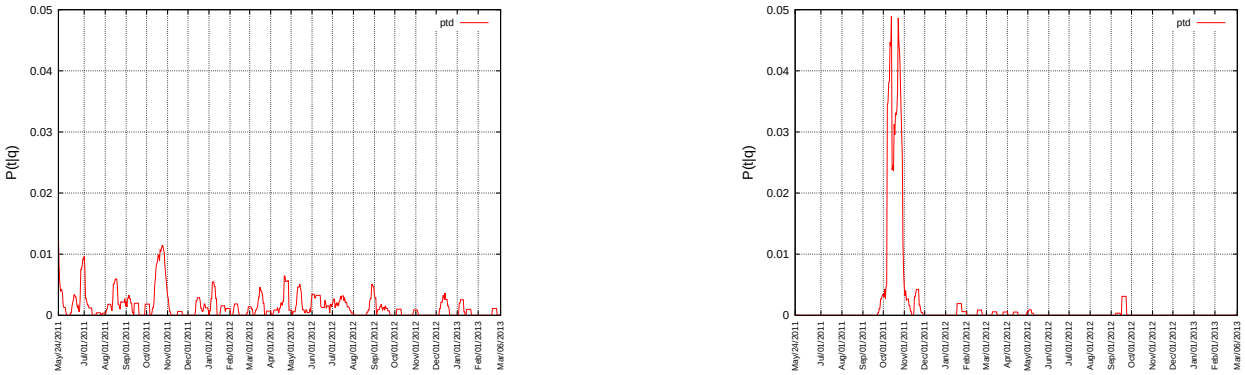
4.3 Temporal Features

Documents come with two kinds of temporal information, namely their publication dates and temporal expressions in their contents. We now describe features that leverage these two dimensions of publication time and content time.

Publication Time

We adopt the idea from Jones and Diaz [13] to construct a probability distribution from pseudo-relevant documents' publication dates. To determine the set R of pseudo-relevant documents, with $|R| = 200$ in all our experiments, we employ a unigram language model with Dirichlet smoothing [19] (with $\mu = 1000$) and let $P(q|d)$ denote the likelihood of generating the query q from document d. The publication time distribution (ptd) for a query q is then defined as

$$P_{pub}(t \mid q) = \sum_{d \in R} P(t \mid d) \cdot \frac{P(q \mid d)}{\sum_{d' \in R} P(q \mid d')} .$$



(a) weather forecast (b) occupy wall street

Figure 1: Publication time distributions

Here, t is a timestamp at day granularity and we define

$$P(t \mid d) = \begin{cases} 1, & \text{if } t \text{ is the publication date of } d \\ 0, & \text{otherwise} \end{cases} .$$

Intuitively, this definition puts high probability on days when many highly-relevant documents were published. In practice, we smooth the publication time distribution using a sliding window of seven days.

Figure 1 illustrates the publication time distributions for the queries weather forecast and occupy wall street. We can see that the distribution for the query weather forecast does not exhibit any significant peaks; smaller peaks occur due to natural fluctuations in publication volume. In contrast, the publication time distribution for the query occupy wall street exhibits peaks, which coincide with important events for the Occupy Wall Street protest movement.

We can now use statistical measures to derive features that characterize these publication time distributions. We adopt the features **temporalKL** and **temporalKurtosis** from Jones and Diaz [13]. The former is the Kullback-Leibler Divergence between the publication time distribution $P_{pub}(t|q)$ and a background publication time distribution $P_{pub}(t|D)$ for the entire document collection D. Here, $P_{pub}(t|D)$ is the probability that a randomly drawn document from the collection was published on day t. The latter is the kurtosis of the publication time distribution $P_{pub}(t|q)$. Intuitively, these two features capture how constantly relevant documents have been published and how much the publication time distribution for the query deviates from the background distribution. We expect these features to be useful in deciding whether a query is *atemporal* or not but not in distinguishing between the *past*, *recent*, and *future* categories. We do not adopt the autocorrelation feature proposed in [13], since the document collection at hand spans only two years and we thus do not expect to see periodic behavior.

Content Time

From the temporal expressions contained in pseudo-relevant documents we construct a content time distribution (*ctd*). One advantage of the content time distribution is that it is not bound to the time period covered by the document collection, but can also inform us about important events in the further past or future. Given again a set R of pseudo-relevant documents with their query likelihoods $P(q|d)$, we define the content time distribution as

$$P_{con}(t \mid q) = \frac{\sum_{d \in R} \sum_{te \in TE(d)} P(t \mid te) \cdot P(q \mid d)}{\sum_{d \in R} \sum_{te \in TE(d)} P(q \mid d)} .$$

Here, $TE(d)$ denotes the set of temporal expressions contained in document d. Each temporal expression te is mapped to a time interval $[b, e]$ at day granularity (e.g., June 2014 is mapped to $[2014/06/01, 2014/06/30]$) and we define

$$P(t|[b, e]) = \begin{cases} 1/|[b, e]|, & \text{if } t \in [b, e] \\ 0, & \text{otherwise} \end{cases}$$

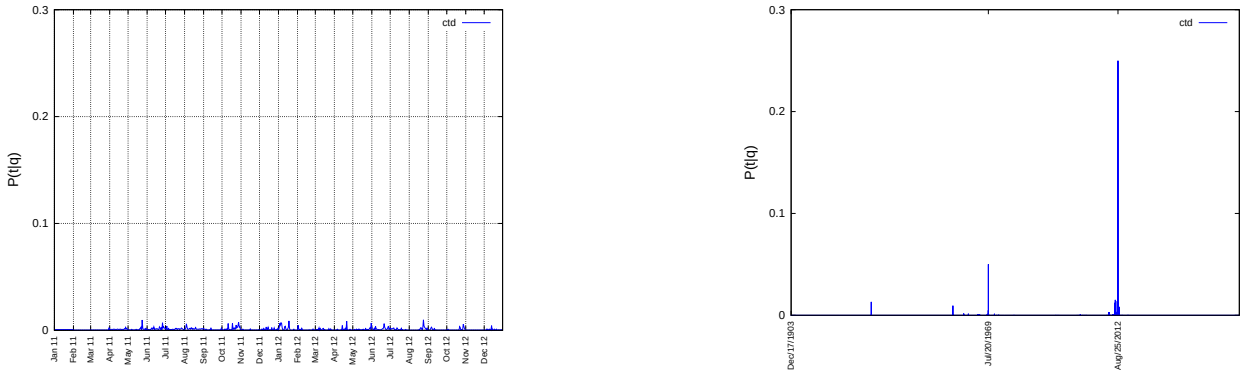
assigning uniform probability to all days in the time interval. Note that this can be seen as a simplified variant of the model proposed by Berberich et al. [7].

Figure 2 displays the content time distributions for the queries weather forecast and moon landing. As for publication time, the former query does not exhibit any peaks or irregularities. The latter query, in contrast, shows distinctive peaks at important relevant time points. These include the first moon landing on July 20 1969, thereby pre-dating the document collection by more than four decades, but also more recent relevant events such as the launch of Voyager 1 on August 25 2012.

To characterize the content time distribution relative to the query issuing time, we devise several features. As a first group of features, we determine the 10%, 25%, 50%, and 75% percentiles of the distribution and encode their distance in days (possibly a negative number) to the query issuing time. These are referred to as **ctQuantile10**, **ctQuantile25**, **ctQuantile50**, **ctQuantile75**. Complementary to that, we define three features **ctWeightOnPast**, **ctWeightOnRecent**, and **ctWeightOnFuture**, which reflect how much probability mass falls before, onto, and after the query issuing date. Finally, we also record in a feature **ctWeightOnCollectionPast** how much probability mass falls before the time period covered by the LK document collection.

Publication + Content Time

Until now, we have considered publication dates and temporal expressions only in isolation, which ignores how the



(a) weather forecast

(b) moon landing

Figure 2: Content time distributions

temporal expressions in documents relate to their publication dates. Consider the query weather forecast as a concrete example. As we have seen in Figures 1 and 2, publication dates and temporal expressions do not exhibit any interesting behavior, suggesting that the query should be classified as *atemporal*. To classify the query correctly as *future*, we propose to look into how temporal expressions in pseudo-relevant documents are positioned relative to their publication date. Intuitively, for the query weather forecast, we expect to see many temporal expressions (e.g., tomorrow or next week) that lie in the future relative to the forecasting document's publication date. As a stepping stone, we define three weights for a temporal expressions $[b,\, e]$ and a publication date t_d reflecting how much of the temporal expression lies in the past, present, or future relative to the publication date, formally

$$w_p([b,\, e],\, t_d) = \lambda \cdot \frac{|[b, t_d]| \cdot \mathbb{1}(t_d \in [b,\, e])}{|[b,\, e]|} + (1-\lambda) \cdot \mathbb{1}(e < t_d)$$

$$w_r([b,\, e],\, t_d) = \lambda \cdot \frac{\mathbb{1}(t_d \in [b,\, e])}{|[b,\, e]|} + (1-\lambda) \cdot \mathbb{1}(t_d \in [b,\, e])$$

$$w_f([b,\, e],\, t_d) = \lambda \cdot \frac{[t_d,\, e] \cdot \mathbb{1}(t_d \in [b,\, e])}{|[b,\, e]|} + (1-\lambda) \cdot \mathbb{1}(b > t_d)$$

with λ as a tunable parameter, which we set as $\lambda = 0.75$ in our experiments. Aggregating again over all pseudo-relevant documents, we define three features **relPastScore**, **relRecencyScore**, **relFutureScore** as follows

$$relPastScore(q) = \frac{\sum_{d \in R} P(q|d) \sum_{[b,\, e] \in TE(d)} w_p([b,\, e], t)}{\sum_{d \in R} \sum_{[b,\, e] \in TE(d)} P(q|d)}\ ;$$

$$relRecencyScore(q) = \frac{\sum_{d \in R} P(q|d) \sum_{[b,\, e] \in TE(d)} w_r([b,\, e], t)}{\sum_{d \in R} \sum_{[b,\, e] \in TE(d)} P(q|d)}\ ;$$

$$relFutureScore(q) = \frac{\sum_{d \in R} P(q|d) \sum_{[b,\, e] \in TE(d)} w_f([b,\, e], t)}{\sum_{d \in R} \sum_{[b,\, e] \in TE(d)} P(q|d)}\ .$$

Events

One of the most difficult types of queries are those which refer to recurrent events, for instance, french open 2013 dates. In order to resolve the event to an exact date and position it relative to the query issuing time, one would need domain knowledge about when the event usually takes place.

We present an approach which focuses on recurrent event queries and tries to resolve them to a precise date, without using any domain knowledge. The general idea is to again leverage publication dates and temporal expressions of pseudo-relevant documents, but build distributions over days of the year. Building on our above definitions of $P_{pub}(t|q)$ and $P_{con}(t|q)$, we define

$$P_{pub\_days}\,(i \,|\, q) = \sum_t \text{isAtDay}\,(t, i) \cdot P_{pub}\,(t \,|\, q)$$

$$P_{con\_days}\,(i \,|\, q) = \sum_t \text{isAtDay}\,(t, i) \cdot P_{con}\,(t \,|\, q)$$

with $i \in [1, 365]$ as a day of the year and isAtDay$(t,\, i)$ indicating whether the date t was the i-th day of the year. We further combine the two distributions above as

$$P_{days}(t|q) = 0.5 \cdot P_{pub\_days}(t|q) + 0.5 \cdot P_{con\_days}(t|q)\ .$$

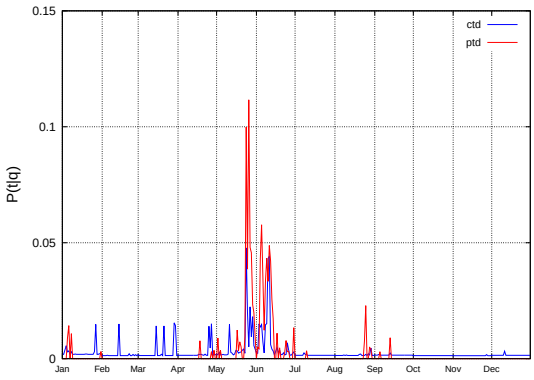


Figure 3: Distributions over days for the query french open derived from publication dates (*ptd*) and temporal expressions (*ctd*)

To illustrate this idea, Figure 3 shows the distributions obtained from publication dates and temporal expressions for the query french open. Both curves have clear peaks and we can estimate the approximate date of the event. In fact, the French Open tournament takes place from late May until early June. Note that the publication time curve shows clearer peaks; this is because publication dates always correspond to a specific day; in contrast, in the content time method we often add weights to many days at once due to the uncertainty of temporal expressions.

When given a year-qualified query (e.g., french open 2014), we use the above distribution $P_{days}(i|q)$ to determine the likely date of the event and its relative position to the query issuing time. First, we remove the year (e.g., 2014) from the query and determine the corresponding distribution $P_{days}(i|q)$. Second, we inspect the distribution and determine the day of the year i having maximal probability. If its probability exceeds 0.05, we assume the date of the event as the day i within the year given in the query. We encode whether this day lies in the past or future relative to the query issuing time using two binary features **lkPastEvent** and **lkFutureEvent**. In addition, we keep a feature **lkRecentEvent** that records the ratio $P_{days}(j|q)/P_{days}(i|q)$ where i is the determined day of the year for the event and j is the day of the query issuing time.

Temporal Dictionary

Our last set of temporal features captures for individual query keywords whether they tend to co-occur in documents together with temporal expressions referring to the past, present, or future. This idea is akin to the approach described by Jatowt et al. [11]. We construct a *temporal dictionary* from the LK document collection, which records for every word w four scores

- $dictPastScore(w)$ as the fraction of sentences containing w that also contain a temporal expression which refers to the past relative to the document's publication date;
- $dictRecencyScore(w)$ as the fraction of sentences containing w that also contain a temporal expression which refers to the document's publication date;
- $dictFutureScore(w)$ as the fraction of sentences containing w that also contain a temporal expression which refers to the future relative to the document's publication date;
- $avgTimex(w)$ as the average number of temporal expressions co-occurring with w in a sentence.

When given a query q, we look up these scores for every word $w \in q$ and keep the average and maximum as features (e.g., **avgDictPastScore** and **maxDictPastScore** for the first score).

4.4 Linguistic Features

Every query is run through Stanford CoreNLP [1] to perform part-of-speech (POS) tagging, named entity recognition (NER), and detect temporal expressions. From the resulting annotations, we derive a set of linguistic features, which we describe now in more detail.

POS Features. We define the following features based on the query's sequence of POS tags:

- **startsWithNoun** corresponds to the regular expression `(DT )?NN(S)? (.)*` and indicates whether the query starts with a noun;
- **startsWithInflectedVerb** signals whether the query starts with an inflected verb corresponding to the POS tags `VBD`, `VBP`, or `VBZ`;
- **startsWithNonInflectedVerb** indicates whether the query starts with a non-inflected verb corresponding to the POS tags `VB` or `VBG`. We hypothesize that non-inflected verbs tend to indicate *atemporal* queries (e.g., lose weight quickly);
- **containsPersonalPronoun** signals whether the query contains a personal pronoun such as I, you, or we but not it, which we observe based on the POS tag `PRP`;
- **startsWithWh** reflects whether the query starts with one of the POS tags `WP`, `WP$`, or `WRB`. Queries that start with such as wh-pronoun are often questions.

Named Entities. Based on the named entities detected we introduce three features. The first two, **containsEntity** and **isEntity**, indicate whether the query contains or corresponds in its entirety to a named entity belonging to the types `Person`, `Organization`, `Location`, or `Misc`. The third feature **containsDuration** indicates whether an entity of type `Duration` occurs. We contemplated creating more fine-grained features (e.g., **containsPerson**, **containsOrganization**, and **containsLocation**), but discarded this idea due to the limited amount of training data and the danger of overfitting.

Tense. Part-of-speech tagging also informs us about the tenses of verbs in the query. We encode those in three binary features **containsPastTense**, **containsPresentTense**, and **containsFutureTense**.

Temporal Expressions. We also make use of temporal expressions found in the query by the SUTime component of Stanford CoreNLP. More precisely, we determine a time interval from the value attributes of returned TIMEX3 tags and encode their relative position to the query issuing time in the features **containsPastDate** and **containsFutureDate**. Two additional features provide more information when the query issue time falls inside the determined time interval. The first one, **recentIntervalSize**, encodes the size of the time interval in days (e.g., 30 for June 2014) and is otherwise set to a large constant. The second one, **sectionOfInterval**, encodes the position of the query issuing time within the determined time interval. For instance, for a query issued on June 10th 2014 which contains June 2014, this feature assumes the value 0.33. Finally, we leverage a list of common holidays and introduce a feature **containsHoliday**. This feature reflects whether the query mentions a non-year-qualified holiday that does not occur within the next 100 days. The value of 100 is chosen arbitrarily, but we think it is important to somehow express the difference between imminent holidays and holidays that are further away.

4.5 Topical Features

Often, queries which look very similar can end up in different temporal classes. As a concrete example, consider the

queries number of neck muscles and number of millionaires in usa. While the former is clearly *atemporal*, the latter should be classified as *recent*. With enough training data our surface features will be able to handle such cases, for instance, by picking up neck muscles as a cue for a health-related query which is likely to be *atemporal*. As a workaround, for our limited training data, we introduce features that seek to capture the broad topic of the query. To this end, we make use of the Open Directory Project (DMOZ) [5], which is the largest human-edited directory on the Web. We issue the query against DMOZ and analyze the top-level categories (e.g., `Arts`, `Business`, or `Sports`) of the top-20 web pages returned. We take the majority top-level category, with ties broken according to lexicographic order, as the category of the query and keep it as a single feature **dmozTopic**.

4.6 Other Features

To capture the selectivity of query keywords, we introduce three additional features, namely (i) **minIDF**, (ii) **avgIDF**, and (iii) **maxIDF** corresponding to the minimum, average, and maximum inverse document frequency of any query keyword. In addition, we introduce a feature **resultRelevance**, which is defined as the sum of query likelihoods of result documents in R. Finally, as another characterization of the query result, we keep track of the average number of temporal expressions per result document in a feature **avgTimex**.

5. EXPERIMENTS

We now describe our experimental evaluation and its results. Section 5.1 describes our insights regarding features effectiveness obtained on the training data. The feature sets used for our formal run submissions are listed in Section 5.2.

5.1 Cross Validation

As a first baseline, which our methods have to outperform, we consider a *majority* classifier. Given that the training data is perfectly balanced, with all classes occurring with probability 25%, this is equivalent to a uniform random classifier, and achieves an accuracy of 25%. Our second baseline is a text-only classifier, which only relies on the n-gram features described in Section 4.1.

To assess the overall effectiveness of the different feature groups, we conducted four experiments. Accuracies in each of these experiments are estimated using 50 runs of 10-fold cross validation.

We divided our features into seven sets. The set **surface** contains the n-gram features described in Section 4.1. The temporal trigger words described in Section 4.2 go into a feature set **lexical**. A third feature set **distribution** contains the features related to distributions of publication dates and temporal expressions, described in Section 4.3, as well as the other features described in Section 4.6. The features related to events and our temporal dictionary are kept separate and in their own respective feature sets **event** and **dictionary**. The linguisitc features are contained in the set **nlp**. The DMOZ topic feature described in Section 4.5, finally, goes into its own feature set **topical**.

Experiment 1 considers the feature sets in isolation and determines their effectiveness when used alone. We ran an exhaustive feature selection on each set, considering all possible subsets of features, and report the obtained accuracy estimates in Table 2.

Experiment 2 studies how much we can gain in accuracy by using any of our seven feature sets on-top of the baseline **surface** features. Table 3 lists the resulting accuracy estimates.

	NaïveBayes	J48
surface (baseline)	56.30	42.20
majority class (baseline)	25.00	25.00
lexical	37.50	37.15
distribution	50.93	57.08
event	44.98	41.63
dictionary	58.05	53.28
nlp	53.68	57.35
topical	35.45	35.45

Table 2: Accuracy estimates of feature sets in isolation (Experiment 1)

	NaïveBayes	J48
surface (baseline)	56.30	42.20
majority class (baseline)	25.00	25.00
surface + lexical	61.83	46.33
surface + distribution	63.00	49.83
surface + event	65.08	55.68
surface + dictionary	67.88	48.75
surface + nlp	74.38	57.95
surface + topical	57.00	44.03

Table 3: Accuracy estimates of feature sets when used on-top of surface (Experiment 2)

Experiment 3 examines the accuracy of the two classifiers when using all selected features from Experiment 1, including the baseline features. It also analyzes how the performance changes when any of the feature sets is removed. Table 4 shows the results.

	NaïveBayes	J48
surface (baseline)	56.30	42.20
majority class (baseline)	25.00	25.00
all selected features	74.98	61.30
all selected - distribution	75.45	62.08
all selected - nlp	68.50	51.23
all selected - topical	74.75	57.88
all selected - lexical	73.63	57.20
all selected - dictionary	73.13	63.85
all selected - event	73.65	62.73
all selected - surface	62.88	55.15

Table 4: Accuracy estimates when removing individual feature sets (Experiment 3)

Experiment 4, instead of combining the features selected for each feature set in isolation for Experiment 1, considers *all* features described in this work. Considering all subsets of features for feature selection is clearly prohibitively expensive. We therefore take the **surface** features again as a baseline and use simulated annealing [15] to select from the additional features. In each iteration of simulated annealing, a randomly selected feature is turned on/off and we estimate the accuracy of the resulting feature set. Simulated annealing accepts the new feature set if the accuracy has improved. Otherwise, if the accuracy has dropped, it accepts it with a probability that depends on the decrease in accuracy and the iteration. In later iterations, it is thus less likely to accept decreases in accuracy. While the algorithm will often

find a *global* optimum, it may end up in a *local* optimum. Table 5 gives the accuracies of the resulting feature set (**all selected features**) and also reports how accuracy changes when we remove one of our seven feature sets.

	NaïveBayes	J48
surface (baseline)	56.30	42.20
majority class (baseline)	25.00	25.00
all selected features	85.28	72.35
all selected - distribution	77.45	64.78
all selected - nlp	70.53	47.88
all selected - topical	(85.28)	(72.35)
all selected - lexical	82.75	65.8
all selected - dictionary	71.65	(72.35)
all selected - event	(85.28)	(72.35)
all selected - surface	63.25	58.90

Table 5: Accuracy estimates obtained when removing individual feature sets (Experiment 4)

Discussion

The first thing that we notice is that our estimates are much stronger than the accuracies obtained on the formal runs. This is most pronounced in the fourth experiment, where we reach an accuracy of over 80%. We believe that we are facing overfitting problems. Even with k-fold cross-validation, one cannot avoid this, especially if the training set is rather small, which (having only 80 training queries at hand) is obviously the case. On a small training set, some bad features might seemingly correlate with the class attribute and thus have a big influence in the final classifier. Because we have designed a large number of features and tried to find appropriate subsets via feature selection, the chance that the chosen subset contains overestimated features is high. This is not really a problem of our experimental setup, though, as there is no systematic way to overcome these overfitting issues. We cannot recognize features that correlate with the class attribute by pure chance, except by human intervention, which is obviously not available to a learning algorithm.

We can also see that the n-gram features (**surface**) are capable of correctly classifying over 50% of the queries, which is not at all what one would expect. We question whether this is a representative result and doubt that, in practice, one could achieve such good accuracy by only considering unigrams and bigrams, especially when learning on such a small training set.

Nevertheless, we can detect certain trends in the results. We notice that the linguistic features are especially effective in this task. In each of our four experiments, they are among the best performers and their removal always leads to a significant decrease in accuracy.

Our event resolution approach, represented by the feature set **event**, shows a surprisingly good performance. The accuracy estimate of 44.98% (41.63%) in Experiment 1 might not appear overwhelming, especially in comparison to the other estimates. However, one has to bear in mind that the event resolution only extracts non-zero feature values for queries which contain a year expression and which are assumed to actually contain an event. Only 23 of the 80 queries contain a year expression, so the features only take a non-zero value for a small fraction of the training set. Thus the result of 44.98% (41.63%) is very satisfying and demonstrates that distributions of publication dates and temporal expressions can successfully be used to detect events.

Another surprise is the result of our temporal dictionary feature set. It achieves better accuracy than the n-gram features (with just two selected features) and without the need to learn particular n-grams. All in all, those two features are enough to correctly classify almost 60% of the queries - this is a huge finding, which confirms that many words do have inherent temporal semantics.

The features that we obtained from looking at the distributions of publication dates and temporal expressions of pseudo-relevant documents also achieve good accuracy, yet they did not perform quite as well as we had hoped. It seems like temporal expressions from retrieved documents are only partially connected to the temporal intent behind the query that was issued; one could attempt to narrow the context of the query terms down to sentences instead of entire documents, i.e., only temporal expressions that occur in the same sentence as one of the query terms would be retrieved, which might make the distributions more accurate. This is basically what we successfully did in our dictionary approach. However, in the dictionary approach, we only tried to relate terms to the past, the present, and the future, respectively. It is still necessary to link certain n-grams to *absolute* times, such that, for example, the relevant time of the expression belmont stakes can simply be looked up in a dictionary. Unfortunately, a first attempt at realizing such a dictionary of absolute times (not described in this paper) failed and no improvements could be observed. We still think extending the temporal dictionary approach is one of the most promising approaches for future work.

5.2 Formal Run

We now list the features that we chose for our three formal runs. All of the runs used the Naïve Bayes classifier as this classifier outperformed the decision tree in our experiments on the training data. The features from Run 1 and Run 3 were chosen via a simulated annealing algorithm for feature selection (because the algorithm works probabilistically and does not find the global optimum, the selected features for both runs are different). In contrast, for Run 2 we manually assembled different features that we thought might complement each other well – of course without looking at the test data. Table 6 presents the features selected this way. In addition to these features, each selected feature subset *always* contains all n-gram features.

	Run 1	Run 2	Run 3
future	0.65	0.71	0.63
atemporal	0.73	0.76	0.80
past	0.53	0.60	0.60
recent	0.57	0.49	0.44
overall	62.33	64.00	61.67

Table 7: Performance of formal runs

Table 7 provides details on the performance of our three runs, as reported by the organizers. Indeed, Run 2 with its manually selected features performs slightly better than the other two runs. We can see that our runs do a good job at deciding whether a query is *atemporal* (with precision values consistently above 70%) or *future* (with precision values still above 60%). Their performance is worst when detecting *recent* query (no precision value above 60%). Given that we used off-the-shelf classifiers (Naïve Bayes for our formal

Run 1	Run 2	Run 3
maxDictPastScore	**avgDictPastScore**	**avgDictRecencyScore**
maxDictRecencyScore	**avgDictRecencyScore**	**maxDictFutureScore**
avgDictTemporality	**avgDictTemporality**	**lkRecentEvent**
lkRecentEvent	**lkRecentEvent**	**containsPastTrigger**
lkFutureEvent	**lkFutureEvent**	**containsFutureTrigger**
containsPastTrigger	**avgTimex**	**ctQuantile10**
containsFutureTrigger	**avgIDF**	**ctQuantile50**
ctQuantile10	**containsPastTense**	**avgTimex**
ctQuantile75	**containsPresentTense**	**ctWeightOnPast**
avgIDF	**containsFutureTense**	**maxIDF**
maxIDF	**containsPastDate**	**temporalKL**
resultRelevance	**recentIntervalSize**	**containsPastDate**
containsPastTense	**containsFutureDate**	**sectionOfInterval**
containsPresentTense	**containsHoliday**	**containsEntity**
containsPastDate	**dmozTopic**	**isEntity**
recentIntervalSize		**startsWithNoun**
containsFutureDate		**startsWithNonInflectedVerb**
containsHoliday		**containsPersonalPronoun**
containsEntity		**dmozTopic**
startsWithNonInflectedVerb		
containsPersonalPronoun		

Table 6: Selected features for the three formal runs

runs) and focused on feature engineering, we regard this a positive result.

6. CONCLUSIONS

In this paper, we have laid out an approach to tackle the TQIC task of NTCIR-11 Temporalia [12]. Relying on established off-the-shelf components (Stanford CoreNLP for natural language processing and Weka for machine learning), we focused on finding effective features for the task at hand. We found that simple n-gram features go a long way and form a strong baseline. Among our devised features, we found linguistic features (e.g., capturing whether the query corresponds to a question) and event-related features, which try to guess the date of an event mentioned in the query, most effective. As part of our future research, we plan to further investigate the substantial gap in accuracy estimates obtained on the training data and the formal runs and assess the effectiveness of our features on different query workloads or other query classification tasks.

7. REFERENCES

[1] Stanford CoreNLP `http://nlp.stanford.edu/software/corenlp.shtml`.

[2] OneLook Thesaurus `http://onelook.com`.

[3] Thesaurus.com `http://thesaurus.com`.

[4] Weka Machine Learning Library `http://www.cs.waikato.ac.nz/ml/weka/`.

[5] The Open Directory Project `http://www.dmoz.org/`.

[6] O. Alonso, M. Gertz, and R. A. Baeza-Yates. On the value of temporal information in information retrieval. *SIGIR Forum*, 41(2):35–41, 2007.

[7] K. Berberich, S. Bedathur, O. Alonso, and G. Weikum. A language modeling approach for temporal information needs. *ECIR*, pages 13–25, 2010.

[8] R. Burghartz. Temporal query classification, B.Sc. Thesis, Saarland University, 2014.

[9] R. Campos, G. Dias, A. M. Jorge, and A. Jatowt. Survey of temporal information retrieval and related applications. *ACM Comput. Surv.*, 47(2):15:1–15:41, Aug. 2014.

[10] A. Jatowt, Y. Kawai, and K. Tanaka. Temporal ranking of search engine results. In *WISE*, pages 43–52, 2005.

[11] A. Jatowt, C. man Au Yeung, and K. Tanaka. Estimating document focus time. In *CIKM*, pages 2273–2278, 2013.

[12] H. Joho, A. Jatowt, R. Blanco, H. Naka, and S. Yamamoto. Overview of ntcir-11 temporal information access (temporalia task). In *Proceedings of NTCIR-11*, 2014.

[13] R. Jones and F. Diaz. Temporal profiles of queries. *ACM Trans. Inf. Syst.*, 25(3), July 2007.

[14] N. Kanhabua, K. Berberich, and K. Nørvåg. Learning to select a time-aware retrieval model. In *SIGIR*, pages 1099–1100, 2012.

[15] S. Kirkpatrick. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, 34(5-6):975–986, 1984.

[16] X. Li and W. B. Croft. Time-based language models. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 469–475, New York, NY, USA, 2003. ACM.

[17] D. Metzler, R. Jones, F. Peng, and R. Zhang. Improving search relevance for implicitly temporal queries. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 700–701, New York, NY, USA, 2009. ACM.

[18] M.-H. Peetz and M. de Rijke. Cognitive temporal document priors. In *ECIR*, pages 318–330, 2013.

[19] C. Zhai. Statistical language models for information retrieval a critical review. *Found. Trends Inf. Retr.*, 2:137–213, March 2008.