Using Time-Series for Temporal Intent Disambiguation in NTCIR-12 Temporalia

Dan Li[¶], Xiaoxia Liu[†], Yunxia Zhang[†], Degen Huang[†], Jingxiang Cao[¶]
School of Foreign Languages, Dalian University of Technology[¶]
Department of Computer Science and Technology, Dalian University of Technology[†]
No.2 Linggong Road, Gaoxinyuan District, 116024 Dalian, China
linda_2013, liuxxivy, zhangyunxia@mail.dlut.edu.cn, huangdg, caojx@dlut.edu.cn

ABSTRACT
Our group DUT-NLP-EN participated in the TID subtask (English) of NTCIR-12 Temporalia and submitted three runs. The temporal intent probability distribution of four categories (past, recency, future and atemporal) for the 300 test queries are predicted through logistic regression model in all the three runs. In RUN1, four groups of features are used including trigger word, word POS, explicit time gap, temporal probability of words. Implicit time gap is added in the form of rule-based time gap in RUN2 and in the form of time-series statistics in RUN3. RUN2 performs slightly better than the rest two runs with AvgCosine of 0.732 and AvgAbsLoss of 0.210.

Team Name
DUT-NLP-EN

Subtasks
Temporal Intent Disambiguation (TID) (English)

Keywords
Temporal Intent, Query Classification, Time-Series, Bayesian Probability

1. INTRODUCTION
The temporal intent of a query is crucial in Information Retrieval. According to Nunes [1] who samples the AOL query dataset, only about 1.5% queries are explicit temporal queries, meaning that they contain explicit temporal expressions, such as "Poland 1940s", "Olympics 2008" or "top movies 2000s". Thus it is challenging to discover the underlying temporal intents.

NTCIR Temporalia task deals with four temporal intents, i.e. *atemporal*, *past*, *recency*, *future*. The Temporal Query Intent Classification (TQIC) subtask of NTCIR-11 aims to estimate the only best temporal intent of a query [2]. But one best is not an appropriate solution because many queries are ambiguous, so the Temporal Intent Disambiguation (TID) subtask of NTCIR-12, an upgraded task from TQIC, asks participants to estimate not only the temporal intent categories but also the distribution among the four temporal intent categories for a given query. TID can handle the ambiguous queries better. Further details can be referred to in the overview paper [3]. The potential applications may lie in search engines where queries are treated accordingly to the underlying distribution of the temporal categories and presented with the most temporally-related documents.

2. RELATED WORK
Comprehensive overviews of the existing literature on temporal information retrieval have been conducted [4][5]. Here we summarize several works related to TQIC in NTCIR-11.

From the results of the test collection of TQIC [2], the most difficult category to classify is *recency*, and the easiest one is *past*. It is also noticed that (1) *atemporal* queries are likely to be misclassified as either *recency* or *past* queries (16.7% and 9.6%, respectively); (2) *past* queries are likely to be misclassified as *atemporal* queries (13.1%); (3) *recency* queries are likely to be misclassified as *future* (28.2%) or *atemporal* (13.5%) queries; (4) *future* queries tend to be misclassified as *recency* queries (25.9%).

Yu et al. [6] submitted 3 runs for the TQIC subtask utilizing three types of features: time gap features, verb tense features and lemmas and named entities. In run-1 and run-2, they used Logistic Regression classifier with different Model parameters. In run-3, they used SVMlin classfiier with additional data from AOL dataset. A comparative run also was provided. Run-1 got the highest overall score and the highest *future* score. And run-3 got the highest *past* score following by Run-1 and Run-2. But the 3 runs both had low *recency* score.

Hasanuzzaman et al. [7] submitted 2 runs and both perform best in *recency* classification. They built an ensemble learning paradigm using eight different classifiers, namely Support Vector Machines, Naive Bayes, Multilayer Perceptron, Locally Weighted Learning, LogitBoost, Decision Tables, Hybrid Learning and Random Forests. It reduces bias by combining multiple classifiers instead of a single one.

Shah et al. [8] submitted 3 runs with different classifiers by using Naive Bayes Classifier in run-1, SVM classifier in run-2, and combined Naive Bayes, SVM and Decision Tree in run-3. Run-1 outperforms the other participants in terms of *atemporal*.

3. METHODOLOGY
Given a search query, the task is to estimate a distribution of the four temporal intent categories. Generally, a category distribution prediction is one the many steps of a classification model, therefore we handle the task as a classification problem. A temporal intent classification model is trained based on feature construction and supervised learning on the given query collection. Then, the same feature construction is applied to a new query, and the model is used to predict the temporal intent category distribution for a new query of interest. The architecture is showed in Figure 1.

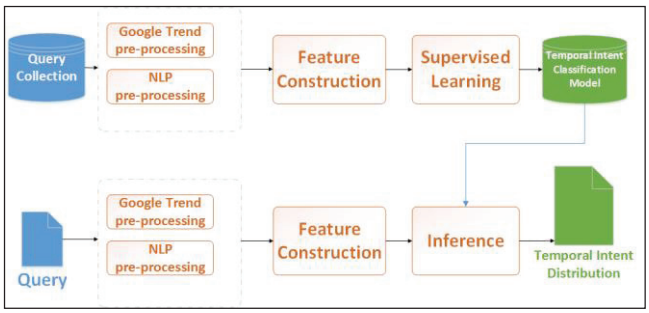

Figure 1. The architecture of modelling temporal intent category distribution

The following parts will illustrate pre-processing, feature construction, and supervised learning in detail.

3.1 Pre-processing

The pre-processing consists of NLP pre-processing and Google Trends[1] preprocessing. For NLP pre-processing, we use Stanford CoreNLP Package [9] for tokenization, lemmatizing, POS tagging, parsing, and temporal expression recognition. Temporal expressions are recognized by SUTime in CoreNLP, a rule-based temporal tagger. The temporal type and value corresponds to the TIMEX3 standard. We focus on the three temporal types (TIME, DURATION, and DATE) and list the corresponding examples as follows.

1. Bear night 1974, <Timex tid="t1" type = "TIME"> 1974TNI </Timex>
2. Election Day, <Timex tid="t3" type = "DURATION"> P1D </Timex>
3. Howard Stern Jesse Ventura 2016, <Timex tid="t18" type="DATE">2016</Timex>

We also automatically crawled search frequency data from Google Trends for all the queries. Google Trends is a public web facility that shows how often a particular search query is entered relative to the total search volume across various regions and in various languages. Each query corresponds to a csv file with time axis and frequency axis. The granularity for the time axis is one week, and the range for the frequency axis is 0 – 100. We call this original data Time Doman Data (TDD). Then we use the *periodogram* [10] function in TSA package in R language to generate the csv file of power spectrum, which describes the distribution of frequency components composing the signal that generates the corresponding TDD. The horizontal axis is frequency (in Hz), the vertical axis is the spectral density at corresponding frequencies. We call the power spectrum data Frequency Domain Data (FDD). FDD can be used to find the "hidden period" of the TDD. See examples in Figure 2.

3.2 Feature Description

As we only have a small training data, the traditional widely-used features such as bag-of-word, n-gram do not work well in the task due to data sparsity. We propose the following five groups of features.

Trigger word. By manually analyzing some samples, we find that queries containing some specific words tend to be frequently used in specific scenarios and queries containing these words are also easy to classify for human. For example, when people search "exchange rate" they want to know the current exchange rate, not the past one, therefore the queries containing "exchange rate" are often classified to *recency* without much ambiguity. We design four features in this group: *whether_past*, *whether_recency*, *whether_future*, *whether_atemporal*. The value for the four features are all boolean type. If a target query contains words in past trigger word set, then *whether_past* is set TRUE; if a target query contains words in recency trigger word set, then *whether_recency* is set TRUE; if a target query contains words in future trigger word set, then *whether_future* is set TRUE; if a target query contains words in atemporal trigger word set, then *whether_atemporal* is set TRUE. A strict set of trigger words is selected for each of the four categories, which contains 53 words. Aho–Corasick algorithm [11] is employed to match the query with the trigger word sets.

Word POS. The abundant inflectional changes in English helps to discover temporal information. In the sample query set, the intent category of many queries can be manually inferred with word POS information only. By analyzing the syntax parsing of the 93 samples in the dry run, we find that 81% have noun head and noun head queries tend to be atemporal, while 19% have verb head, the intent category of verb head can be easily classified according to verb tense. See Table 1.

Table 1. Temporal intent distribution for queries with noun head or verb head in the 93 samples

Type	P	R	F	A	Total
Noun head	16	18	16	25	75
Verb head	5	4	3	6	18

We design two features *head_word_POS* and *verb_tense*, and adapt the Stanford POS tag set to represent the feature values. For head word POS, we take the word in the ROOT dependency relation as the head word. For verb tense, we take as the target verb the root node of the maximum subtree that has a verb root node.

Word temporal probability. We assume that for the word temporal intent category distribution the training data is the same with the test data. Given a word, the probability of the four temporal intent categories can be inferred through Bayesian Probability:

$$p(c_i|w) = \frac{p(w|c_i)p(c_i)}{p(w)} = \frac{p(w|c_i)p(c_i)}{\sum_{j=1}^{4} p(w|c_j)p(c_j)}, i = 1,2,3,4. \quad (1)$$

$p(c_i) = \frac{|c_i|}{N}$, $p(w|c_i) = \frac{|wc_i|}{Nc_i}$, c_i is intent category, N is the sample number, $|c_i|$is the count of queries of category i, $|wc_i|$is the count of word w in queries of category i, $Nc_i$is the count of all the words in queries of category i.

We assume that the probability of a temporal intent category for a query is the corresponding probability sum of all the constitute words, therefore the temporal intent category distribution of a query can be presented as follows:

$$p(c_i|q) = \sum_{w \in q} p(c_i|w), \quad i = 1,2,3,4. \quad (2)$$

Explicit time gap. Only 9 queries in the 93 samples have explicit temporal expressions. The categories of all these queries conform to the dates that the temporal expressions indicate. We can see the

[1] http://www.google.com/trends

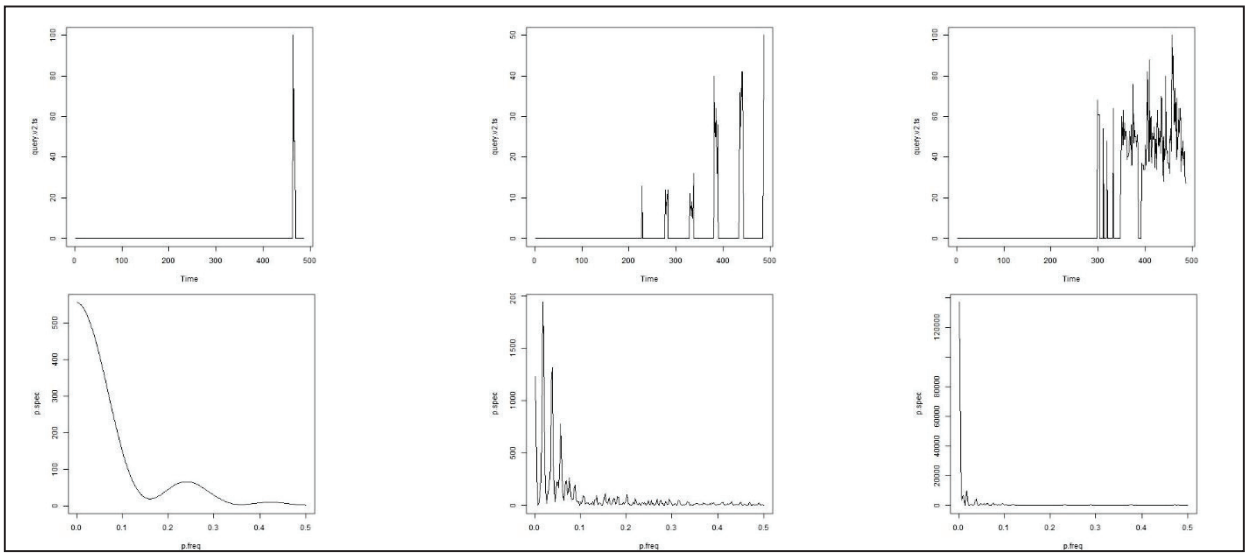

Figure 2. Examples of pairs of TDD (first line) and FDD (second line). The horizontal axis of TDD is time (in week), and the vertical axis is standardized search frequency from 0 to 100. The horizontal axis of FDD is frequency (in Hz), and the vertical axis is the spectral density at corresponding frequencies. The three query examples are "end of twinkies", "nba playoff's scores", and "when was America discovered".

explicit temporal expression is a high-quality but low-coverage indicator for temporal intent classification. We design three features, namely *explicit_time_gap_past*, *explicit_time_gap_recency*, *explicit_time_gap_future*, with boolean feature value. For each query, we extract the temporal expression (if has) from the pre-processed XML, and transfer it to the standardized date format. Then the time gap is computed by subtracting the standardized date from the issue date of the query. Finally, if time gap > 2 weeks, for example, the feature values of *explicit_time_gap_past* will be assigned TURE, *explicit_time_gap_recency* assigned FALSE, and *explicit_time_gap_future* assigned FALSE. If there is no explicit temporal expression, the three feature values are assigned FALSE.

Implicit time gap (Time-series). Implicit time gap is a supplement feature to explicit time gap. As most queries do not contain explicit temporal expressions, it is almost the most challenging work in TID task. Previous researches have extracted implicit temporal expressions from the snippets of search results, the related Wiki pages, or sometimes Wiki Infobox etc. Yet extracting temporal information from unstructured texts does not guarantee the recognition of the true date of the queries, and structured texts like Wiki Infobox suffer from low coverage of queries. We propose a novel way – to extract implicit temporal expression from search frequency data of queries. Google Trends is a good choice because of its high coverage of market and easy access.

We find that user search behaviors reflect the development trend of an event. For example in Figure 2, the event in query "end of twinkies" occurs in November 2012, and the peak of the corresponding TDD is also around November 2012; the NBA playoff in query "nba playoff's scores" occurs every May of a year, and people mostly search the related words around May; query "when was America discovered" describes a historical fact and thus is not time-sensitive, and the curve is rather flat and with no extreme value.

We design two groups of features for implicit time gap. The first one is **rule-based time gap**, which is similar to explicit time gap, also *rule_based_past*, *rule_based_recency*, *rule_based_future*, with boolean feature value. Our hypothesis is that we can discriminate periodical, occasional and time-insensitive queries through TDD and TDD of query collection. If it is periodical, the maximum point in one period near the issue date can be detected; if it is occasional, the peak point can also be detected. Finally, the time gap for periodical or occasional query is computed by subtracting the date of the detected point from the issue date. The algorithm is showed in Figure 3. The parameters are set as follows: $\text{FDD\_MEAN\_THLD} = 200$, $\text{TDD\_MR\_THLD} = 0.02$, $\text{FDD\_FREQUENCY\_THLD} = \frac{1}{2(\text{year})*52(\text{week})} \approx 0.0096$, $\text{FDD\_SPEC\_THLD} = 10000$, $\text{PMQ\_DOWN} = 2(\text{week})$, $\text{PMQ\_UP} = 4(\text{week})$.

Due to the difficult tuning process of prior parameters, we also propose another group of features, i.e. **time-series statistics**, including *max*, *mean*, *variation*, *sr*, *mr*, *stridency* for both TDD and FDD. Their feature values are real number type. *max*, *mean*, and *variation* are commonly-used features. *sr*, *mr* ,and *stridency* are adapted from [12] as follows:

$$\text{sr} = \frac{f_{max} - \max(\{f_1,\ f_2, \dots, f_T\} - \{f_{max-d}, \dots, f_{max-1}, f_{max},\ f_{max+1}, \dots, f_{max+d}\})}{\sum_{t=1}^{T} f_t} \quad (3)$$

$$\text{mr} = \frac{f_{max}}{\sum_{t=1}^{T} f_t} \quad (4)$$

$$\text{stridency} = \frac{f_{max}}{\sum_{f_t > \delta f_{max}} f_t} \quad (5)$$

where T is the length of a time-series, d is an adjustment parameter and $2d$ means the duration of a spike, δ is an adjustment parameter to control the cutoff level ($\delta = 0.2$ here).

```
RuleBasedTimeGap(tdd_list, fdd_list):
    T, P = DetectPeriod(tdd_list, fdd_list)
    if len(P) > 0:
        time_gap = ComputeMaximumPoint(tdd_list, T)
    elif mean(fdd_list.spec) < FDD_MEAN_THLD and
         mr(tdd_list.freq) > TDD_MR_THLD:
        time_gap = ComputePeakPoint(tdd_list)
    return time_gap

DetectPeriod (tdd_list, fdd_list)
    P = []
    for item in fdd_list:
        if item.f > FDD_FREQUENCY_THLD and
           item.spec > FDD_SPEC_THLD:
           P.append(item)
    T = 1/P[0].f
    return T, P

ComputeMaximumPoint(tdd_list, T)
    Q = tdd_list[-T:0] ∪ tdd_list[-2T:-T] ∪ tdd_list[-
3T:-2T]
    phase = mean(argmax(q.freq).week - q[0].week for
q in Q)
    if PMQ_UP <= phase:
        time_gap = T - phase
    elif PMQ_DOWN <= phase <= PMQ_UP:
        time_gap = -phase
    elif phase < PMQ_DOWN:
        time_gap = 0
    return time_gap

ComputePeakPoint(tdd_list)
    peak_point = argmax(tdd_list)
    time_gap = tdd_list[-1].week-peak_point.week
    return time_gap
```

Figure 3. The algorithm to compute implicit time gap. Input: TDD and FDD of a query. Output: time gap of a query.

3.3 Classification Model

We use Scikit-learn machine learning package[2] in Python to train our model. Five classifiers are applied in the dry run including SVM with linear kernal, SVM with Gaussian kernel, logistic regression, random forest, and decision tree, while logistic regression based method is submitted in the formal run.

4. EXPERIMENTS & DISCUSSION

4.1 Data

The training data includes 93 samples from the dry run of NTCIR-12 Temporalia and 300 samples from the formal run of NTCIR-11 Temporalia. As samples in NTCIR-12 are tagged as temporal intent probability distribution, we transfer them into the NTCIR-11-style-like format as the input of the classifiers. The trained models are tested on the 300 queries of formal run in NTCIR-12.

4.2 Runs

We submitted three runs in the formal run by employing logistic regression model. In RUN1, four groups of features are used including trigger word, word POS, explicit time gap, word temporal probability. In RUN2, we add feature rule-based time gap based on RUN1. In RUN3, we replace feature rule-based time gap with time-series statistics.

Table 2 shows the results of the three runs. We can see there is no significant difference among the three runs, which indicates the feature time-series statistics and rule-based time gap do not take much effect here. But this is against our intuition because when manually tagging the formal run samples we often need to refer to the search frequency data of Google Trends in which the periodical pattern of the data conforms to the real world with high probability. The reason may be that the two features are so rough that they cannot catch the hidden periodical patterns well. All in a nutshell, it is a challenge but a chance to improve time-series features in the future.

Table 2. Average cosine and average absolute loss for RUN1, RUN2, and RUN3

	AvgCosine	AvgAbsLoss
RUN1	0.728	0.208
RUN2	0.732	0.210
RUN3	0.727	0.212

Table 3. Confusion matrix for RUN1, RUN2, RUN3 and manual results compared with the standard result

		STANDARD			
		P	R	F	A
RUN1	P	**34**	2	10	25
	R	4	**18**	6	25
	F	1	4	**39**	14
	A	5	10	8	**95**
RUN2	P	**33**	3	11	32
	R	4	**17**	6	33
	F	1	4	**39**	13
	A	6	10	7	**81**
RUN3	P	**34**	2	11	32
	R	4	**17**	5	25
	F	1	4	**40**	14
	A	5	11	7	**88**
MANUAL	P	**42**	3	3	15
	R	0	**23**	6	8
	F	0	3	**37**	6
	A	2	5	17	**130**

To further investigate the results generated from different runs and our manually tagged result, we calculate the confusion matrix as shows in Table 3. The confusion matrix is calculated by mapping the probability distribution to the category of highest probability and counting the category-category pair numbers respectively. The baseline is the manually tagged result. The bold numbers are queries right classified, and the numbers of grey shadow are wrongly classified. We can see that human tagged result highly conforms to the standard reference, indicating TID task is reasonable and meanwhile there is still improvement for automatic TID task. However, human also misunderstand atemporal as past, or future as atemporal. This problem is even worse for the three runs. Besides, the three runs tend to classify future as past, recency as atemporal. We can see three out of four of the major errors are related to atemporal category, inspiring us to classify atemporal

[2] http://scikit-learn.org/

from temporal in the first step and handle the rest three categories later.

Table 4. Classification errors in RUN1

	System				Standard			
Query	**P**	**R**	**F**	**A**	**P**	**R**	**F**	**A**
wind turbine	0.04	0.02	**0.89**	0.05	0.00	0.00	0.00	**1.00**
civil action	**0.91**	0.03	0.02	0.03	0.00	0.00	0.00	**1.00**
what time is inauguration	**0.73**	0.05	0.03	0.19	0.00	**0.40**	**0.40**	**0.20**
When Is Thanksgiving	**0.79**	0.02	0.13	0.06	0.00	0.00	**0.90**	0.10
famous events in the 20th century	0.09	**0.50**	0.12	0.29	**1.00**	0.00	0.00	0.00
life in the future	0.09	**0.51**	0.16	0.24	0.00	0.00	**0.70**	0.30

We list some samples with the cosine value less than 0.25 in Table 4. "wind turbine" and "civil action" are wrongly classified because of the misleading value of feature **word temporal probability.** The training data is so sparse that it makes the probability distribution of "wind" and "civil" biased to specific samples like "weather for tomorrow wind" and "when did civil war end" in training data, whereas "turbine" and "action" are **out-of-vocabulary** words. For "what time is inauguration", feature *trigger_word_atemporal* is right assigned "TRUE", *verb_tense* right assigned "VBZ", and for "When Is Thanksgiving", feature *explicit_time_gap_future* is right assigned "TRUE", but the results are also affected by **stop words** "in" and "is" etc. in feature word temporal probability. The situation is similar to the rest two examples "famous events in the 20th century" and "life in the future". It indicates that word temporal probability is a high-quality feature only on the condition of large samples and removing stop words.

5. CONCLUSION

We participated in the TID subtask of the NTCIR-12 Temporalia task and submitted three runs. The temporal intent probability distribution of the four categories of the 300 test queries are predicted through logistic regression in all the three runs. In RUN1, four groups of features are used including trigger word, word POS, explicit time gap, temporal probability of words. Implicit time gap is added in the form of rule-based time gap in RUN2 and in the form of time-series statistics in RUN3. RUN2 performs slightly better than the rest two runs with AvgCosine of 0.732 and AvgAbsLoss of 0.210.

We discovered two promising features for TID, i.e. temporal probability of words and query search time-series. The future work may lie in three directions. The first direction is to discriminate the atemporal queries with temporal ones in the early step and then to classify the rest three one. Currently, temporal probability of words is limited by small training data, causing too much bias to training samples and too many out-of-vocabulary words. We may solve the problem by extending the probability distribution of a word to that of its synonyms using methods like WordNet or word embedding. Finally, the time-series data of queries need to be preprocessed such as removing long-term trends and seasonal changes.

6. ACKNOWLEDGMENTS

This work was supported by the National Nature Science Foundation of China (No. 61173100 61272375) and National Social Science Foundation of China (No. 15BYY175).

7. REFERENCES

[1] Nunes, S., Ribeiro, C., & David, G. 2008. Use of temporal expressions in web search. In: *ECIR 2008*, pp.580-584.

[2] Joho, H., Jatowt, A., Blanco, R., Yu, H., & Yamamoto, S. 2016. Overview of NTCIR-12 Temporal Information Access (Temporalia-2) Task, In *Proceedings of the NTCIR-12 Conference on Evaluation of Information Access Technologies*.

[3] Joho, H., Kishida, K. 2014. Overview of NTCIR-11. In *Proceedings of the 11th NTCIR Conference*, Tokyo, Japan.

[4] Campos, R., Dias, G., Jorge, A. M., & Jatowt, A. 2015. Survey of temporal information retrieval and related applications. *ACM Computing Surveys (CSUR)*, *47*(2), 15.

[5] Kanhabua, N., Blanco, R., & Nørvåg, K. 2015. Temporal information retrieval. *Foundations and Trends in Information Retrieval*, 9(2), 91-208.

[6] Yu, H., Kang, X. & Ren, F. 2014. TUTA1 at the NTCIR-11 Temporalia Task. In *Proceedings of the 11th NTCIR Conference*, Tokyo, Japan.

[7] Hasanuzzaman, M., Dias, G. & Ferrari, S. 2014. HULTECH at the NTCIR-11 Temporalia Task: Ensemble Learning for Temporal Query Intent Classication. In *Proceedings of the 11th NTCIR Conference*, Tokyo, Japan.

[8] Shah, A., Shah, D., & Majumder, P. 2014. Andd7 @ NTCIR-11 Temporal Information Access Task. In *Proceedings of the 11th NTCIR Conference*, Tokyo, Japan.

[9] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)* (pp. 55-60).

[10] Cryer, J. D., Chan, K. 2008. *Time series analysis with applications in R (second edition)*.Springer-Verlag New York.

[11] Aho, A. V., & Corasick, M. J. 1975. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, *18*(6), 333-340.

[12] Ren, P., Chen, Z., Song, X., Li, B., Yang, H., & Ma, J. 2013. Understanding temporal intent of user query based on time-based query classification. In *Natural Language Processing and Chinese Computing* (pp. 334-345). Springer Berlin Heidelberg.