

GIR at the NTCIR-12 Temporalia Task

Long Chen
School of Computing,
University of Glasgow, UK
long.chen@glasgow.ac.uk

Haitao Yu
Faculty of Library, Information
and Media Science, University
of Tsukuba
yuhaitao@slis.tsukuba.ac.jp

Fajie Yuan
School of Computing,
University of Glasgow, UK
fajie.yuan@glasgow.ac.uk

Joemon M Jose
School of Computing,
University of Glasgow, UK
joemon.jose@glasgow.ac.uk

ABSTRACT

The GIR team participated in the NTCIR 12 Temporal Information Access (Temporalia) Task. This report describes our approach to solving the Temporal Intent Disambiguation (TID) problem and discusses the official results. We explore the rich temporal information in the labeled and unlabeled search queries. A semi-supervised linear classifiers is then built up to predict the temporal classes for each search query.

Team Name

GIR

Keywords

user intent, temporal, query, classification

Subtasks

Temporal Intent Disambiguation (English)

1. INTRODUCTION

The GIR team at University of Glasgow participated in the the NTCIR 12 Temporal Information Access (Temporalia) Task [3]. This minority report describes our approach to solving the Temporal Intent Disambiguation (TID) problem and discusses the official results. For detailed introductory information of this task, please refer to the task design paper [2] and the overview paper [3]. This task investigate the identification of temporal information over four categories, namely, "\past", "\recent", "\future", and "\atemporal". Since the average length of queries are very short, e.g., 4.2 words in dry-run phase, to find useful temporal features in queries and to explore the background information (especially for named entities) seem to be prominent in this subtask. For those search queries with explicit temporal expression or predicate verbs, we extract the "\time gap" and "\verb tense" features, separately. For those with no temporal information, we submit them to Google to collect temporal features for us to infer its temporal state.

2. APPROACH

Our system consists of two separate modules: (a) identifying temporal features in search queries, and (b) exploring

the contextual information in these queries. In this section, we detail the temporal information extraction for search queries, and explain the learning of semi-supervised classifier, based on the labeled and unlabeled training examples, for Temporal Query Intent Classification.

When it comes to query intent classification, unlabelled queries can be easily crawled from internet without making too much effort (everything can be made automated by a pre-defined program); on the other hand, it's not a trivial work for even experts (let alone people with no such expertise) to label the query manually especially when the required corpus is in a large scale.

Under such context, semi-supervised learning would be suitable to reduce the cost for training dataset construction. But the prerequisite of co-training - two sets of features working independently teach each other iteratively in a mutual iterative manner - should be satisfied in the first place [1]. Intuitively textual features play an important part and can fit into this context - they can analyze the query by understanding the semantics. In the same time, we find that there are also plenty of contextual information associated with each query available when submitting the query, such as verb tenses and temporal expressions returned by Google results, which provides different perspective to understand the query complementarily.

2.1 Textual Features

Textual pre-process is done by using Weka API. As for textual feature, we find that 3-gram characters and word have a better performance than other options. While 3-gram character can give a slightly better result, word is chosen as the textual representation as it is easier for people to have a better understanding of it, and can consequently allow us to fix up mistakes swiftly. The text features of a query are extracted from the bag-of-words content of the query after standard pre-processing steps (tokenization, lower-casing, stopword-removal, and stemming) [4]. Finally each query is represented as a vector of terms weighted by TF-IDF [4].

2.2 Contextual Features

A total of 3 contextual features are learned in order to build the classifier. To capture the temporal features, we resort to the temporal expressions and verb tenses of a query. A temporal expression in a query is a sequence of terms that represent a point in time, a duration or a frequency. For example, two pre-annotated temporal expressions are

extracted as:

1. <t val=201007> July 2010</t>
2. <t val=201605">the end of May</t>

Verb tense is a specific indicator that signals a situation takes place. In a search query, it is possible to observe multiple verbs with different verb tenses. For example, in the query "an experience that you enjoyed while learning something new", we can see two verbs, "VBD enjoy" and "VBG learn". In such scenarios, we can resort to the Stanford Parser library in Stanford CoreNLP pipeline, and choose the main predicate by picking the uppermost verb in the parse tree. The tense of the main predicate then can be used to represent the tense of the whole search query. Lastly, when there is no temporal expressions and verb tenses, each query is submitted to Google via Google Search API¹ to serve as the knowledge from external resources. The snippets of the top 10 results are used as query expansion if no temporal expressions and verb tense exist in the original query.

So the problem now lies in how to implement a co-training framework to put together the mentioned two perspectives. The co-training scheme using in our experiment is a version similar to that of [1], our algorithm can be detailed as follows:

Given:

- * F_Q and F_H are word-based feature (only from query) and contextual features from Google results respectively
- * C_Q and C_H represent classifiers trained using the above features, respectively
- * L_{train} is the labelled training dataset
- * L_{test} is the labelled testing dataset
- * U is the unlabelled training dataset

Loop:

- C_Q labels all instances from U by using F_Q
- C_Q picks out top K_Q instances with the highest confidence as dataset T_Q , whose distribution of each class is equal to that of L_{train}
- Remove T_Q from U and add to L_{train}
- C_H labels all instances from U by using F_H
- C_H chooses top K_H instances of the highest confidence as dataset T_H , whose distribution of each class is equal to that of L_{train}
- Remove T_H from U and add to L_{train}
- If result evaluated on L_{test} is below the expected threshold T or iterative times gone beyond threshold N , then jump out of the loop;

Notice that this task is essentially a multi-class problem, and there are couple of options to cope with it namely one-versus-one and one-versus-all. The former will construct 4 smaller binary classifiers first and then decide the class distribution by using a voting mechanism; the latter instead will set up only one generic classifier and choose the instance's class upon the margin distance. Considering the data size and time-complexity of the program, one-versus-one is adopted instead of one-versus-all.

¹<https://developers.google.com/custom-search/json-api/v1/overview>

3. EXPERIMENT

3.1 Datasets and Metrics

The Temporal Intent Disambiguation (TID) Subtask subtask provides a dry-run dataset of 100 search queries for developing the temporal classification models, and a formal-run dataset of 300 search query samples for examining the results. The details of the datasets and the metrics are described in [3].

3.2 Results

A number of machine learning algorithms implemented in Weka², including C4.5, Random Forest, Naive Bayes, k-Nearest-Neighbours, and Linear Support Vector Machine (SVM), have been tried out for semi-supervised learning (co-training). The one-vs-rest ensemble scheme was used to achieve multi-class classification. Linear SVM kept to deliver the best classification performance in our experiments, so we only report its results here. The system is evaluated against Averaged Per-Class Absolute Loss (APCAL) and Cosine Similarity. The formal run values our system achieved are 0.326 for APCAL and 0.417 for Cosine Similarity. We compare the following approaches:

1. Supervised Approach, which simply employ the original dry run queries as training instances.
2. Semi-Supervised Approach, which is the method that we described in Section 2, but without the enhancement of any external resources.
3. Semi-Supervised Approach with external knowledge, which is similar to the second one but is equipped with the knowledge learned from Google.

Given the optimal parameter values, the classification performances of those approaches on the formal run set with different sets of features, measured by precision, are reported in Table 1. Consistent to the observation in [5], adding the unlabelled instances brings substantial performance improvement to the supervised approach. More importantly, it is clear that semi-supervised approach coupled with external knowledge from Google achieves the best results. The Linear SVM parameters are set to default values except that the class weights are optimised for each query category by dry run setting.

It is obvious that using both text features and contextual features works better than using either kind of features alone, for all query types.

4. CONCLUSIONS AND FUTURE WORK

The main contribution of this work is double-fold.

First, we identify several contextual features which can be used together with standard text features by machine learning algorithms to classify queries according to their underlying temporal intent. Second, our experimental results demonstrate that it is better to exploit both text features and contextual features through the semi-supervised learning framework, co-training, rather than simply combining them in supervised learning, since the former can make use of a large amount of unlabelled data.

²<http://www.cs.waikato.ac.nz/ml/weka/>

Table 1: Table 1: The experimental results on formal-run dataset.

measures	Supervised	Semi-Supervised	Semi-Supervised with External Knowledge
Cosine Similarity	0.366	0.384	0.417
Absolute Loss	0.416	0.358	0.326

In the future, we would like to explore more sophisticated features, such as the name entities identified in the query, which can provide a great deal of semantic information about the original text. In addition, it will also be interesting to introduce more advanced semi-supervised learning algorithms.

5. REFERENCES

- [1] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory, COLT' 98*, pages 92{100, New York, NY, USA, 1998. ACM.
- [2] H. Joho, A. Jatowt, and R. Blanco. Ntcir temporalia: a test collection for temporal information access research. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 845{850. International World Wide Web Conferences Steering Committee, 2014.
- [3] H. Joho, A. Jatowt, R. Blanco, H. Yu, and S. Yamamoto. Overview of NTCIR-12 temporal information access (temporalia-2) task. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*, 2016.
- [4] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.
- [5] H. Yu, X. Kang, and F. Ren. Tuta1 at the ntcir-11 temporalia task. 2014.