KYOTO at the NTCIR-12 Temporalia Task: Machine learning approach for Temporal Intent Disambiguation Subtask

Tomohiro Sakaguchi
Kyoto University, Japan
sakaguchi@nlp.ist.i.kyoto-u.ac.jp

Sadao Kurohashi
Kyoto University, Japan
kuro@i.kyoto-u.ac.jp

ABSTRACT

This paper describes the Kyoto system for Temporal Intent Disambiguation (TID) subtask in the NTCIR-12 Temporal Information Access (Temporalia-2) challenge. The task is to estimate the distribution of temporal intents (*Past, Recency, Future, Atemporal*) of a given query. We took a supervised machine learning approach, using features of bag of words, POS and word vectors. We also incorporated knowledge about temporal and holiday expressions. Our system resulted in a competitive performance.

Team Name

kyoto

Subtasks

Temporal Intent Disambiguation (TID) Subtask (English)

Keywords

Supervised machine learning, Temporal knowledge, Word vector

1. INTRODUCTION

Temporal information is crucial for text understanding. There have been many researches about extracting temporal expressions, and they are important in the view of NLP applications such as text summarization and question answering.

In Information Retrieval (IR), estimating temopral intent of queries is important. For example, systems have to understand that people who submit a query *'weather in Kyoto'* would want to ask 'current' weather in Kyoto, and *'value of silver dollars 1976'* would want to ask the value of silver dollars 'in 1976'. There are also temporal independent expressions like *'brownie cake recipe'* and *'hydrogen energy'*.

A shared task, Temporalia in NTCIR-11 (we call it Temporalia-1 in this paper)[4], was held to deal with these problems. When a query is given, systems classify it into one of the following temporal intent classes: *past, recency, future* and *atemporal*. Since queries sometimes have multiple temporal intents, the task was expanded to predict temporal distribution in Temporalia-2. Please see the task overview[5] for the detail.

We implemented four supervised machine learning models, Support Vector Machine, Support Vector Regression, Feedforward Neural Network and Convolutional Neural Network. Features we used are as follows: bag of words, POS, words vectors features, knowledge about temporal and holiday expressions. The outputs were evaluated with two measures, the average of absolute loss (AvgAbsLoss) and the average of cosine (AvgCos). Among our three formal-run submissions, the averaged results of several classifers was the best: 16.33 AvgAbsLoss and 85.19 AvgCos. However, by additional experiments we found that a feedforward neural network model with a slightly modifed hidden layer configuration performed much better than our formal-run submissions, 14.57 AvgAbsLoss and 86.41 AvgCos.

query	past	rec	fut	atemp
uk 2009 balance of payments	1.0	0.0	0.0	0.0
Twitter Stock Price	0.0	1.0	0.0	0.0
PS4 Release Date	0.0	0.0	1.0	0.0
Annual Free Credit Report	0.3	0.0	0.1	0.6
price hike in bangladesh	0.0	0.545	0.091	0.364
credit score canada	0.1	0.4	0.0	0.5
Vacation in Maldives	0.0	0.1	0.1	0.8

Table 1: Queries and their class distribution. Queries are all issued in May 1, 2013.

2. DATA

The organizer provided a set of 93 queries for training and 300 queries for testing, which is labeled the probability distributions of four temporal intent classes (*Past, Recency, Future, Atemporal*) and their issuing times. The data format is as follows.

```
<query_string>Annual Free Credit Report</query_string>
<query_issue_time>May 1, 2013 GMT+0</query_issue_time>
<probabilities>
    <Past>0.3</Past>
    <Recency>0.0</Recency>
    <Future>0.1</Future>
    <Atemporal>0.6</Atemporal>
</probabilities>
```

Some query examples are shown in Table 1. All queries are issued in May 1, 2013.

Besides the given queries, we also used 300 queries released in Temporalia-1. Since the queries in Temporalia-1 are labeled only one temporal intent class, these distributions are regarded as 1.0 for the labeled class, and 0.0 for the unlabeled classes.

Investigating the queries in Temporalia-2, the class distribution for most queries is skewed (one class is predominant). Figure 1 shows the distribution of probability of predomi-

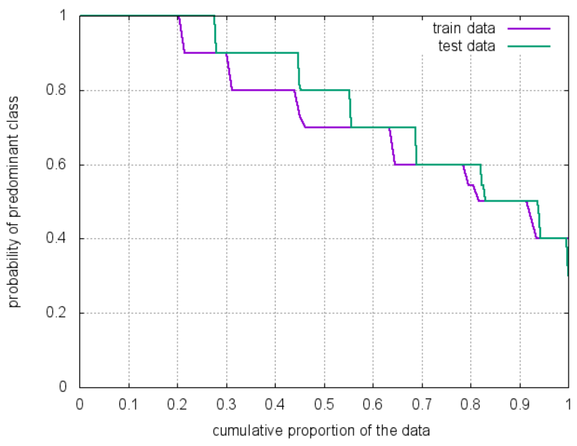

Figure 1: Distribution of probability of predominant class.

nant class, and in 90% of the queries, the probability of the predominant class is more than 0.5. In our experiments, we compared the following two inputs to the models, one uses the predominant class with the probability 1.0, and the other uses all the classes with their probabilities.

3. METHODOLOGY

We take a supervised machine learning approach. We extract six types of features from each query and trained classifiers.

3.1 Pre-processing

First, all queries are transformed to lower case. Next, by using Stanford CoreNLP toolkit[7], the queries are tokenized, and POS tags and temporal information are extracted.

3.2 Features

We designed six types of features which can be classified roughly into two groups. While features 1,2,3 correspond to lexical features, features 4,5,6 correspond to temporal features.

Feat.1 Unigram and POS

Some words have implicit temporal intent. For example, a word 'history' in a query *'history of volleyball'* and a word 'prediction' in a query *'NFL Playoffs Predictions'* can be a clue for identifying *past* and *future* respectively. We use pairs of word lemma and its POS. For example, from a query 'comet coming in 2013', 'comet_NN', 'come_VBG', 'in_IN', '2013_CD' are used as features.

In addition, since verb tense is important in temporal aspect, verb tense is also used as a feature. We prepare a binary verb tense vector and if a query has verbs, flags are set to the correspond elements.

Feat.2 Bigram

Some phrases also have implicit temporal intent. For example, phrases such as 'When was' are an important clue for identifying *past*. We use bigram of lemmas to get the phrases information. To recognize the head and tail of queries, 'BOS' and 'EOS' tags are added at the beginning and the end of queries. For example, from a query 'comet coming in 2013', '(BOS, comet)', '(comet, come)', '(come, in)', '(in, 2013)', '(2013, EOS)' are used as features.

Feat.3 Word vector

Although similar words may behave in the similar way such as 'Tokyo' and 'Kyoto', they are associated with the different dimensions in the unigram and bigram features, and thus the similar behavior of the similar words cannot be captured. To cope with this problem, the word vectors are used. The vectors of similar words are represented as similar vectors. We train the word vectors whose dimension is 200 using 69 million sentences of the New York Times.

Though the length of queries is different, the length of feature vector must be constant for all the queries. Since the simple addition of word vector weakens the effect of important words, a vector of a head word, which is generally important in a sentence, is used as a feature.

Feat.4 Time gap

Temporal expressions in a query play an important role in estimating temporal intent. Since temporal intent depends on the gap between temporal expressions in a query and query issuing time, we used it as a feature. First, we recognize temporal expressions and its value using Stanford CoreNLP (SUTime[1]). For example, the value of 'august' in *'drudge report august'* is 'XXXX-08' and 'future' in *'what would the future life be like'* is 'FUTURE_REF'. Next, we identify the gap between temporal expressions and query issuing time, and set PAST_FLAG, PRESENT_FLAG, FUTURE_FLAG corresponding to the gap.

Feat.5 Numerical temporal expressions

We prepare binary features to recognize numerical values of temporal expressions. A four-dimensional binary vector corresponding to each season, a twelve-dimensional binary vector corresponding to each month are used.

Feat.6 Holiday expressions

In training data, there are a lot of queries which include holiday expressions (e.g. *'martin luther king day 2013'*, *'mother's day in 2013'*). Since SUTime cannot recognize these expressions, we extract holiday information from Wikipedia. We collected all infobox information in Wikipedia, and extracted 908 infoboxes which have holiday tag. Since some holidays' date are different by year (e.g. 'Martin Luther King Day' is the third Monday of January), the month of holidays are used as a feature. We prepare twelve-dimensional binary feature vector, and if the query includes a holiday expression, a flag is set in the corresponding month.

3.3 Classifiers

In this task, we use four classifiers, Support Vector Machine, Support Vector Regression, Feedforward Neural Network and Convolutional Neural Network. The classifiers can be divided into two groups. SVM is a classifier which uses only one class of four classes. Since one class is predominant in most of the queries, we use the most predominant class as correct data and train to assign the highest probability to the most predominant class. The other classifiers can consider all the classes and train to be similar to the given distributions.

The details are as following.

1. **Support Vector Machine (SVM)**
   We use LIBSVM[2] as a four class classifier, and normalized scores are outputted in each class.

2. **Support Vector Regression (SVR)**
   We use LIBSVM and train the class distribution. The normalized scores are outputted in each class.

3. **Feedforward Neural Network (FFNN)**
   We built three types of feedforward neural network models shown in Figure 2. A feature vector is inputted into the network, and the 4 units in the output layer output the distribution. Type(a) has four layers and the two hidden layers have 1000 units. Type(b) also has four layers, but the hidden layers have 1000 and 100 units. Type(c) has three layers and the hidden layers have 100 units. We used the type(a) in the submitted runs.

4. **Convolutional Neural Network (CNN)**
   Convolutional neural network model extracts features which do not depend on a position. Though it is originally invented for conputer vision, it has been applied to NLP tasks and achieved good performance[3].

   Our model is based on the model proposed by Kim[6] which classified sentences by using CNN, and gained fine performance. Figure 3 shows our architecture. Though the inputs of other classifiers are six types of features extrated from each query, the input in this model is a set of vectors which corresponding to input words. We used the vector which concatenated vectors of a 200-dimensional word vector and a binary vector corresponding to feature 4 and 5. As filters, unigram, bigram and trigram are used.

4. EXPERIMENTS

We describe the experimental evaluation and our results.

4.1 Evaluation

In this task, predictions are evaluated with two measures, the average of absolute loss and the average of cosine.

The average of absolute loss (AvgAbsLoss) evaluates the difference of distributions. It calculates average of per-class absolute loss. When we denote the correct distribution of a query q as $P = \{p_1, p_2, p_3, p_4\}$ and the output distribution as $W = \{w_1, w_2, w_3, w_4\}$, we can calculate it as follow.

$$\frac{1}{4}\sum_{i=1}^{4}|w_i - p_i|$$

The average of cosine (AvgCos) evaluates the similarity of distributions. It calculates cosine similarity of two distributions.

$$\frac{P \cdot W}{|P||W|} = \frac{\sum_{i=1}^{4}|p_i * w_i|}{\sqrt{\sum_{i=1}^{4} p_i^2}\sqrt{\sum_{i=1}^{4} w_i^2}}$$

	AvgAbsLoss	AvgCos
SVM	16.41	83.56
SVR	17.06	84.26
FFNN(a) (Run1)	17.03	82.69
FFNN(b)	**14.57**	**86.41**
FFNN(c)	15.15	84.90
CNN	19.87	79.22
SVM+SVR+FFNN(a) (Run2)	16.33	85.19
SVM+SVR+FFNN(b)	15.65	86.30
SVM+SVR+FFNN(a)+CNN (Run3)	18.52	83.42

Table 2: Results of each classifier

4.2 Results

We predicted the distributions of 300 test queries, and submitted 3 results. We submitted the result of FFNN(a) (Run1), the averaged result of SVM, SVR, FFNN(a) (Run2) and the averaged result of SVM, SVR, FFNN(a), CNN (Run3). Table 2 shows the result. Results of each classifer are calculated after the organizer released the correct distributions.

Run2 performed the best in our submitted runs, whose AvgAbsLoss is 16.33 (2nd in all submitted runs) and AvgCos is 85.19 (1st in all submitted runs). After additional experiments, it was found that FFNN(b) was the best. Its AvgAbsLoss and AvgCos of FFNN(b) are 14.57 and 86.41.

Table 3 shows the best and worst 15 results of submitted run2, measured with AvgCos. Queries which includes year expressions (e.g. '2012', '2014'), implicit temporal word (e.g. 'history', 'forecast') and holiday expressions (e.g. 'Election Day') achieved excellent performance. Question format queries which include the past verb tense (e.g. 'when was . . . ', 'how did . . . ') also gained good results.

5. DISCUSSION

5.1 Effective features

To understand which features were effective, we experimented leaving one feature out for each feature using SVM and FFNN(b). Table 4 shows that all the features worked well. In the six types of features, Feat.3 (word vector of head word) and Feat.4 (Time gap) contributed greatly.

5.2 Confusion matrix

Focused on the most predominant classes of correct and output distributions, confusion matrix is described as Table 5. The confusion matrix shows that the major errors are confusing *future* as *atemporal* and confusion between *recency* and *atemporal*.

Classified future as atemporal:

23 *future* queries were misclassified into *atemporal*. The examples are *'PS4 Release Date'* (the correct distribution is *past*: 0, *recency*: 0, *future*: 1, *atemporal*: 0), *'when does big brother start'* (0.2, 0, 0.7, 0.1), *'movie releases'* (0, 0, 0.8, 0.2) and *'When Is Thanksgiving'* (0, 0, 0.9, 0,1), which issued in 1st May 2013.

In the first two cases, we need the temporal knowledge of events. Since the release date of PS4, a home video game console, is November 2013 in US, the first case should be *future*. The second case is more difficult. Big Brother, a famous TV series, has 18 versions from season1 in 2000 to season18 in present. Systems have to estimate the tempo-

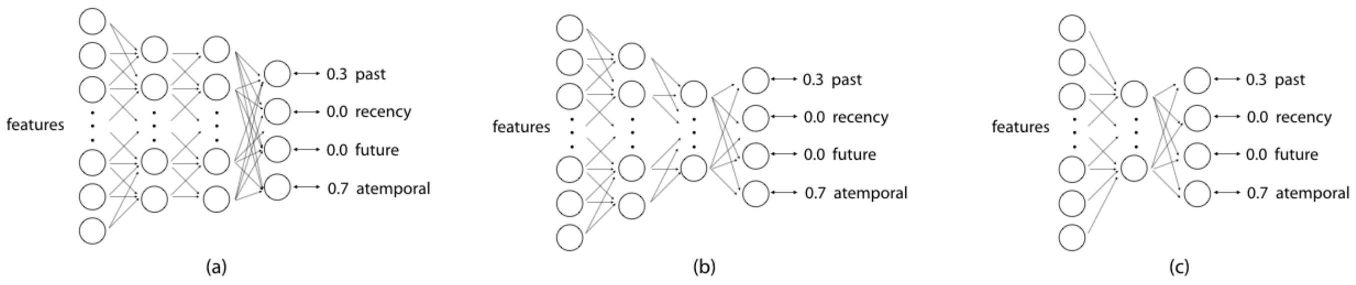

Figure 2: Three types of FFNN model.

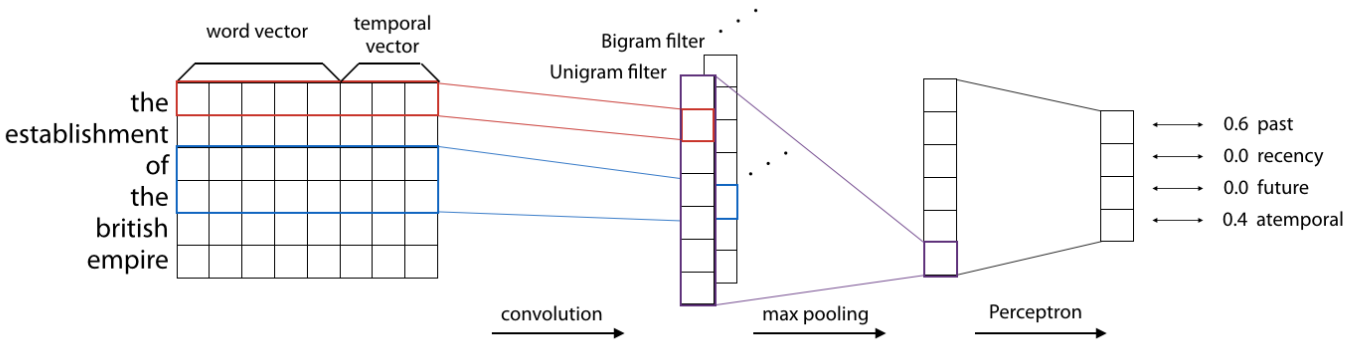

Figure 3: The architecture of CNN model.

ral distribution with a full understanding of versions and verb tense. To solve the third case, temporal commonsense in web search is essential. In the fourth case, though the system recognizes 'Thanksgiving' as a holiday expression in November, the interpretation between *past* and *future* is difficult because November and May (query issuing month) differ just six months. *'November Calendar Printable'* in Table 3 also faced with the similar problem. In this case, 'printable' is another clue. To cope with the problem, recognizing clues in implicit temporal words are essential.

Classified recency as atemporal:
16 *recency* queries were misclassified into *atemporal*. The examples are *'time in australia'* (0, 1, 0, 0), *'sydney bus timetable'* (0, 0.6, 0, 0.4), *'baseball regionals results'* (0.3, 0.6, 0.1, 0), *'NCAA Baseball Regionals Scores'* (0.2, 0.6, 0, 0.2), which issued in 1st May 2013. Since time and scores of sports change frequently, users issue these queries when they want to get current information.

Classified atemporal as recency:
15 *atemporal* queries were misclassified into *recency*. The examples are *'delicious food in spring festival'* (0, 0.1, 0.2, 0.7), *'city transportation in modern China'* (0, 0.3, 0, 0.7), *'Trends In Biochemical Sciences'* (0, 0.3, 0, 0.7), which issued in 1st May 2013. Since the contents of these queries do not change day by day, *atemporal* should be more dominant than *recency*.

5.3 Structure of FFNN

We tested three types of FFNN. Figure 2 shows that type(b) got the best result. Though the models with two hidden layers have greater representation ability than the model with one hidden layer, since the training data set is relatively small, type(a) seems to overfit to the data.

5.4 4 classes vs 1 class

Though the SVM model has to be trained using the most predominant class, the other models can be trained using either the most predominant class or all four classes. We compared the model using one class with the model using all classes. Table 6 shows that using all classes is better in almost all classifiers. It also revealed that FFNN(b) and (c) models got better results than SVM model in the same condition.

6. CONCLUSION

In this paper, we described our approach of NTCIR-12 Temporal Intent Disambiguation Subtask. We tackled the problem with a supervised machine learning approach. As features, POS, verb tense, word vectors and knowledge about temporal and holiday expressions are used. We built four classifiers, Support Vector Machine, Support Vector Regression, Feedforward Neural Network and Convolutional Neural Network. Though the averaged results of SVM, SVR, FFNN was the best in our submitted runs, the additional experiments showed that a feedforward neural network model with two hidden layers performs much better. To improve the model, knowledge of implicit temporal words and event dates would be needed.

Corrected queries and their AvgCos scores		Mistook queries and their AvgCos scores		sys class	ans class
1.000	The original building was built in 1710	0.513	PS4 Release Date	atemp	fut
1.000	Price of Hay	0.508	life in the future	atemp	fut
1.000	history of volleyball	0.492	NCAA Baseball Regional Predictions	atemp	fut
1.000	history of rap	0.474	season 4 pretty little liars	past	fut
1.000	NFL Playoffs Predictions	0.467	TV Schedule NASCAR Racing	rec	fut
1.000	when was electricity invented	0.451	NFL Schedules	atemp	fut
1.000	When to File 2014 Taxes	0.400	ncaa baseball regionals results	atemp	rec
1.000	Howard Stern Jesse Ventura 2016	0.395	news crew attacked los angeles	atemp	rec
0.999	susan miller 2012	0.379	Cost of Iraq War	rec	past
0.999	when did elvis die	0.359	NCAA Baseball Regionals Scores	atemp	rec
0.999	Price of Poured Terrazzo Flooring	0.353	what the date is today	atemp	rec
0.999	History of Rap Jimmy Fallon	0.350	famous events in the 20th century	rec	past
0.998	what was the stamp act	0.334	When Is Thanksgiving	atemp	fut
0.998	St. Louis Breaking News	0.260	full moon may	rec	fut
0.998	the difference between a brain and a computer	0.144	November Calendar Printable	past	fut

Table 3: Best and worst 15 examples measured with AvgCos in the submitted run2. In the worst examples, most predominant classes of system and answer are described.

	SVM		FFNN(b)	
features	AvgAbsLoss	AvgCos	AvgAbsLoss	AvgCos
all	16.41	83.56	14.57	86.41
leave out Feat.1	16.64	83.84	16.63	82.80
leave out Feat.2	16.62	82.79	15.39	84.03
leave out Feat.3	17.13	83.33	17.17	82.42
leave out Feat.4	17.21	82.52	16.36	83.14
leave out Feat.5	17.18	83.48	15.91	84.71
leave out Feat.6	16.68	83.82	16.63	83.29

Table 4: Experiments using SVM and FFNN(b) leaving out one feature.

	Classified as:			
	past	recency	future	atemporal
past	**33**	2	0	7
recency	0	**16**	0	16
future	3	4	**28**	23
atemporal	4	15	1	**148**

Table 5: Confusion Matrix in the submitted run2.

	4 classes		1 class	
	AvgAbsLoss	AvgCos	AvgAbsLoss	AvgCos
SVM	-	-	16.41	83.56
SVR	17.06	84.26	17.51	84.06
FFNN(a)	17.03	82.69	21.13	70.28
FFNN(b)	**14.57**	**86.41**	15.03	84.01
FFNN(c)	15.87	84.94	15.80	84.41
CNN	19.87	79.22	19.29	78.52

Table 6: Comparison between using 4 classes and 1 class.

7. REFERENCES

[1] A. X. Chang and C. D. Manning. Sutime: A library for recognizing and normalizing time expressions. In *In LREC*, 2012.

[2] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, Nov. 2011.

[4] H. Joho, A. Jatowt, R. Blanco, H. Naka, and S. Yamamoto. Overview of NTCIR-11 temporal information access (temporalia) task. In *Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies*, 2015.

[5] 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.

[6] Y. Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014.

[7] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.