Classifying the time period of a text

Vlaho Poluta, Sven Vidak

University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
{vlaho.poluta, sven.vidak}@fer.hr

Abstract

This paper proposes techniques for solving a problem introduced at SemEval-2015 competition. The problem is divided in two tasks which are conceptually similar. For both tasks the goal is to predict in which time period the given text belongs. Time periods for each task have different lengths but the main categories are fine, medium and coarse grained periods. The approach used in this paper is based only on machine learning methods without usage of any external source of information such as Wikipedia or various Google datasets used in most of the papers today. Even with this constraint, the results are comparable with results currently available online.

1. Introduction

Classifying the time period of a text is a task of identifying in which time period the given text was written.

All the world languages are constantly changing. Language change and development is happening in the same fashion as it was happening 500 years ago. There are a lot of circumstances which affect it. One of the recent driving factors of this language evolution are new technologies and ways of communication like Facebook and Twitter. On the other hand, the world is now more connected than it was 500 years ago, and languages tend to adopt words and phrases from other languages.

Besides the language change, time period can also be determined by entity names that are most frequent in a document. For example 70 years ago more documents would mention Nazi Germany, and today there are a lot of documents mentioning the European Union.

The task that is tackled in this paper was introduced at SemEval-2015 competition (Popescu and Strapparava, 2015a) to see if it is possible to automatically determine the time period to which a text belongs. For this purpose, the main task can be split in two tasks. The first task consists of a text that contains references to people from the past, past events and other clear "time anchors". The idea here is to use those "anchors" and try to extract time information from them in order to determine the time period to which those texts belong. The second task contains a text that is written in a specific time period and the idea here is to focus on how the words or sentences are formed in order to classify the text in the correct time period. Each of those tasks comes with three stages of granularity to further test the developed classification systems.

Goal of this paper is to try to solve those problems using only machine learning techniques and algorithms such as naive Bayes or support vector machines which are explained in more detail in Section 4..

2. Related work

In (Mihalcea and Nastase, 2012) the authors introduce the task of identifying word meaning changes over time. They formulate the task as the disambiguation problem where words are automatically classified to a certain time period depending on surrounding words. They are using three different epochs: 18th, 19th and 20th century, plus or minus 25 years, as well as words from books characteristic to those time periods. For their test set they chose a mixture of polysemous and monosemous words (mainly WordNet), words that occur frequently in all three epoch, and also words that are frequent to only one of those epochs. Using a set of 165 words they have succeeded to beat their baseline. Their baseline being classifier that chooses the epoch in which the selected word occurs most times. So they have shown there are significant differences in word meanings in sentences.

In (Wang et al., 2012) the authors are modeling the latent topics through a sequential collection of documents using continuous time dynamic topic (cDMT) models that use Brownian motion. That might sound too abstract to be related to this papers project, but the authors tested cDMT in time stamp prediction to date a document based on its content. They did it by finding its most likely location over the time line, and measured the error using same granularity at which the data is measured. In process they tested two approaches, flat and hierarchical (zooming in on predictions). From the two approaches the hierarchical outperforms the flat approach, and outperforms the baseline, where the baseline is an expectation of the error by randomly assigning a time.

3. Dataset

Since this problem is one of the problems given at this year's SemEval competition, dataset is freely available for download[1]. Here are two examples form a dataset:

```
<text id="800mr11411423">
<textF no="1700-1702" yes="1703-1705" ...
<textM yes="1702-1708" no="1709-1715" ...
<textC yes="1703-1715" no="1716-1728" ...
... text ...
</text>

<text id="123tu57234234">
<textF no="1699-1705" yes="1706-1712" ...
<textM yes="1703-1715" no="1716-1728" ...
```

[1] http://alt.qcri.org/semeval2015/task7/index.php?id=data-and-tools

```
<textC yes="1699-1719" no="1720-1740" ...
... text ...
</text>
```

For both tasks, some parts of the dataset are the same. *id* represents the unique identifier of an example, *textF, textM* and *textC* stand for fine, medium and coarse graded year intervals to which this example belongs and *text* is self-explanatory. For each example, there is only one time span marked as *yes* for all intervals. Difference between the two task is a length of time spans. For the first task, time spans are 2, 6 and 12 years and for the second task time spans are 6, 12 and 20 years, respectively. Years in which the texts belong are between 1700. and 2014.

After some early preprocessing in which some examples were removed due to errors in time span format or simply because there were some identical examples, what's left are 323 training and 267 test examples for the first task (which is not good at all) and 4217 training and 1040 test examples for the second task.

Another issue with the dataset is that starting years for each example are not aligned (e.g. in example above, starting year for the first example for *textF* interval is 1700 while for the second example is 1699). To solve this problem, a non-overlapping ranges were constructed with appropriate time spans (depending on the task).

Also, the distribution of years in the dataset is uneven so it is possible that some time spans have 0 or 1 text that belongs to it. In contrast with the issue above, this problem was not addressed in any way.

4. Methods

4.1. Features

The construction of non-overlapping time span ranges is pretty straightforward – find the intersection of a given and a custom time span and choose the custom time span for which the intersection is the largest. If there are more than one possible time span to choose from, the lowest year is chosen.

In classification, models are trained for lower and upper year in a time span and the final decision is made by averaging output from both classifiers effectively picking midpoint of a time span.

For the training, three types of features were used: bag-of-words, n-chars and part-of-speech (POS) tags.

Bag-of-words model is a logical choice for problems like this one because there are going to be a lot of words that are probably really good indicators for a certain time period. Also, there would probably be some use of bigrams (names, locations...) but the sparsity problem is too big of an issue so bigrams are not used.

On the other hand, n-chars are really great features for detecting time periods since they tend to change. As an example of what n-chars are let's say we have a word *"today"*. 2-chars that would be extracted are: *to, od, da,ay* while 3-chars are" *tod, oda, day*. It is obvious that the number of n-chars in a text can be huge so there must be some threshold for their frequency. In this paper this threshold is set to 20 as proposed in (Popescu and Strapparava, 2015b) and $n = \{2, 3\}$. That means that all n-char that have frequency less than 20 are discarded.

Table 1: Loss for each distance

intervals off	loss
0	0
1	0.1
2	0.15
3	0.2
4	0.4
5	0.5
6	0.6
7	0.8
8	0.9
$\geqslant 9$	0.99

Part-of-speech (POS) tags were used as the third type of features. POS tags are potentially good features because they can capture the usage of different parts of speech in a certain time period. This step was performed with Stanford Log-linear Part-Of-Speech Tagger (cite).

4.2. Algorithms

Algorithms that are used for a training are multinomial naive Bayes, logistic regression and support vector classifier with both linear and RBF (radial basis function) kernels. Multinomial naive Bayes is used as some kind of a baseline method, meaning it is expected that all other algorithms have better (or equal) results. Logistic regression is particularly interesting because it enables analysis of words (features) – words that got bigger weights are more important to the classifier than words that got smaller weights. Support vector classifier with linear kernel should probably give better (or equal) results than one with RBF kernel because there are a lot of features so hyperplane that separates classes is probably linear (or close to linear).

5. Results

Results in the following subsections are not presented in the usual way (accuracy, precision, recall, F-score) but in a form of score and precision computed using script given for this task. Precision is simply ratio between correctly classified texts and a total number of texts (this is actually accuracy) and the score is computed in a such way that it takes into account distance between the predicted and real interval. Loss for each distance is given in Table 1. If S denotes the score, $\boldsymbol{TLV}$ texts loss vector and TT the total number of texts, the score can be calculated as:

$$S = 1 - \frac{sum(\boldsymbol{TLV})}{TT}$$

where sum is the function that takes the vector and returns the sum of its elements.

5.1. The first task

As mentioned before, in the first task the training examples contain clear references to specific time anchors (persons, events, ...). Total number of features is around 4500. Bag-of-words model uses all words found in the texts even

Table 2: Parameters for the first task

Algorithm	C	γ
Linear SVM	2	–
RBF SVM	64	0.015625
LR	65536	–
Multinomial NB	–	–

Table 3: Results for the first task - fine granularity

Algorithm	Score	Precision
AMBRA	0.167	0.037
IXA	0.187	0.02
UCD	NA	NA
Linear SVM	0.042	0
RBF SVM	0.042	0
LR	0.042	0
Multinomial NB	0.042	0

Table 5: Results for the first task - coarse granularity

Algorithm	Score	Precision
AMBRA	0.554	0.074
IXA	0.557	0.090
UCD	NA	NA
Linear SVM	0.1886	0.0352
RBF SVM	0.1886	0.0352
LR	0.4016	0.0749
Multinomial NB	0.1886	0.0352

Table 6: Parameters for the second task

Algorithm	C	γ
Linear SVM	8	–
RBF SVM	16	0.0625
LR	16384	–
Multinomial NB	–	–

if they appear only once. There are roughly 900 n-chars and 36 POS tags. Hyperparameters for each algorithm are shown in Table 2 and the results are show in Tables 3, 4 and 5.

The first three algorithms are actually small systems described in (Popescu and Strapparava, 2015b). UCD is the one that is most similar to the one used in this project and the difference is that UCS is using Google syntactic n-grams database for date estimates and syntactic phase-structure rule occurrences. AMBRA is based on the learning-to-rank algorithms and IXA combines machine learning techniques and search for named entities or mentions of time in texts. Having that in mind, it is expected that relatively simple machine learning approach won't work as well as those systems and results confirm it. Also, algorithms are not able to learn a lot with only 350 training examples. However, logistic regression gives surprisingly good results (in comparison with other algorithms) for coarse grained intervals.

5.2. The second task

In this task the training examples are texts taken from old texts. In the essence, this is the same task as the first one, but features that are most important to the classifier are some archaic words or grammatical constructions and not named entities from the past. For this task, the number of features is around 12000. Bag-of-words model uses only words that appear at least two times in all the texts. The parameters are shown in Table 6 and the results in Tables 7, 8 and 9.

In most cases, the precision outperforms some algorithms presented in (Popescu and Strapparava, 2015b) while the score is not that great. The reason for this is that there is a large portion of the predictions that are more than 100 years off, but there is also a large number of predictions that are correct. For this problem, the UCD works really good probably because it uses the additional features mentioned before.

In general, the results are good because the features that are used are basic features used in most papers as a starting point and a better result can be achieved with some additional features (e.g. for ones used for the UCD system). Also, it is worth mentioning that some results are repetitive and this probably happens because each algorithm learns to the point where it cannot learn anything new with the current feature set.

Table 4: Results for the first task - medium granularity

Algorithm	Score	Precision
AMBRA	0.367	0.071
IXA	0.375	0.041
UCD	NA	NA
Linear SVM	0.105	0
RBF SVM	0.105	0
LR	0.2595	0.0524
Multinomial NB	0.1053	0

Table 7: Results for the second task - fine granularity

Algorithm	Score	Precision
AMBRA	0.605	0.143
IXA	0.261	0.037
UCS	0.759	0.463
Linear SVM	0.416	0.1923
RBF SVM	0.416	0.1923
LR	0.294	0.0288
Multinomial NB	0.416	0.1923

Table 8: Results for the second task - medium granularity

Algorithm	Score	Precision
AMBRA	0.767	0.143
IXA	0.428	0.067
UCD	0.846	0.472
Linear SVM	0.433	0.1349
RBF SVM	0.433	0.1349
LR	0.433	0.1349
Multinomial NB	0.433	0.1349

Table 9: Results for the second task - coarse granularity

Algorithm	Score	Precision
AMBRA	0.868	0.292
IXA	0.622	0.098
UCD	0.91	0.542
Linear SVM	0.585	0.3302
RBF SVM	0.585	0.3302
LR	0.585	0.3302
Multinomial NB	0.585	0.3302

6. Conclusion

This paper described the classification of a text in time periods. By using relatively simple approach good results were obtained, but there is definitely space for improvements. The first thing that can be improved is the dataset, especially the size of the one used for the first task. As for the features, a good idea would be to build a dataset that would have typical words or grammatical constructs used in language throughout the time. Since this is a quite complex task that would take some time to complete, a simpler approach would be to use currently available datasets that are not used (Wikipedia or Google syntactic n-grams database). Also, maybe some grammar features or topics of a text can be used as features because it's not probable that texts from 200 years ago are mentioning computers or selfies (selfpotraits are not considered selfies).

References

Rada Mihalcea and Vivi Nastase. 2012. Word epoch disambiguation: Finding how words change over time. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 259–263. Association for Computational Linguistics.

Octavian Popescu and Carlo Strapparava. 2014. Time corpora: Epochs, opinions and changes. *Knowledge-Based Systems*, 69:3–13.

Octavian Popescu and Carlo Strapparava. 2015a. Semeval-2015, task 7. `http://alt.qcri.org/semeval2015/task7/`. Accessed: 2015-06-04.

Octavian Popescu and Carlo Strapparava. 2015b. Semeval-2015 task 7: Diachronic text evaluation. *Proceedings of SemEval*.

Chong Wang, David Blei, and David Heckerman. 2012. Continuous time dynamic topic models. *arXiv preprint arXiv:1206.3298*.