CS 4740 Project 2

Gi Yoon Han, Rui Liang, Xiaoyun Quan

10/21/2016

Kaggle team name: random_name

1 Overview

Our goal in this project is to tag uncertain words/sentences in the test text data given some training data. We start with preprocessing the training data appropriately, then compare a few potentially powerful methods in parallel in order to select the best training model by conducting different experiments.

2 Preprocessing

Before implementation, we preprocessed our data by following steps:

- filter out the lines which do not have exactly 3 columns (these are the errors and end of paragraph tag i.e. empty line)
- replace all tags that doesn't contain "CUE" with "O" tag
- replace the first CUE-n tag with B-CUE, $n = 1, 2, 3, ...$
- replace the rest of CUE-n tag with I-CUE, $n = 1, 2, 3, ...$

Now that the data has been 8i formatted, we can run an experiment with baseline system. Our baseline is a dictionary consisting of words tagged with `B-CUE` or `I-CUE` from the first 1% of the entire training data. We will tag any word in the test file whenever it appears in the dictionary.

3 Methods and theories

To foramlize, this supervised sequence tagging problem is that given training pairs $(w_1, t_1), ..., (w_n, t_n)$ where w_i is the i-th word in the sequence with t_i being its tag, find the most likely $t_1, ...t_n$ given some test text $w_1, ..., w_n$ by learning a function $f : \Omega \to T$ where Ω is the space of all possible sequences of words and T is the space of all possible tag sequences.

The first machine learning model we come up with is Hidden Markov Model (HMM). It is a generative method that predicts the most-likely tags of a sequence of n words $(w_1 w_2 ... w_n)$ by maximizing the conditional probability of the sequence of tags $P(t_1 t_2 ... t_n | w_1 w_2 ... w_n)$ based on the bayes theorem

$$P(t_1, ..., t_n | w_1, ..., w_n) = \frac{P(t_1, ..., t_n) \cdot P(w_1, ..., w_n | t_1, ..., t_n)}{P(w_1, ..., w_n)}$$

Having been successful in the past (Nguyen & Guo, 2007), it is a model worth of attempt because as a generative model it considers the joint distribution of (w_1^n, t_1^n) instead of directly estimating the conditional distribution $p(t_1^n | w_1^n)$, where for simplicity w_1^n refers to $(w_1 w_2 ... w_n)$ and likeweise for t_1^n. There are a few details to specify here:

- two strong assumptions are made in HMM: conditional independence among words such that $P(w_1^n|t_1^n) = \prod_i P(w_i|t_i)$; and $P(t_1^n)$ is approximated by n-gram language models;
- we decide to use bigram HMM due to computing economy.

Since HMM is making two strong assumptions, we are motivated to also experiment with models that have relaxing assumptions and allow non-independent features. As a result, we implemented Conditional Random Fields (CRF) and Structured Perceptrons (SP) as well for model comparisions. CRFs are famous for its ability to incorporate a rich set of overlapping and non-independent features, whilst keeping HMM's good tradition of discriminative training where the conditional probability of the tags are maximized given the inputs. On the other hand, as an alternative to CRFs, SP is another algorithm that combines with iterative simple additive updates which has shown improvements over MEMMs and CRFs (Collins, 2002).

4 Implementation

To find the best possible model, we conducted 6 experiments with HMM, CRF and SP and each of them is dividing the data in the same way: first 1067 (90%) of the training text files in `train` folder as training data, the rest 117 (10%) of the training text files as evaluating data to test the method on.

4.1 Experiment 1: Single feature

To start with, we simply consider only one single feature: the word itself but with every letter in lowercase, so a model will not distinguish if the word is capitalized or not. The results show that SP is so far the best.

4.2 Experiment 2: Single feature + Unknown words smoothing

Since the test data consists of 1000 files (`test-public` and `test-private`) whereas we only have 1184 training files, we wonder if unknown words will be a big issue so that unknown words smoothing will be helpful. Here we incorporate "<unk>" for words that only appear once, and the results turn out to be terribly bad. So we determine not to apply unknown words smoothing in other experiments.

4.3 Experiment 3: Single feature + Downsampling (Exclude data files with no "B/I-CUE")

Since we have an extremely imbalanced data where "O"s are way over-represented compared to the tags of interest "B-CUE" and "I-CUE", we feel a strong urge to resample the data so that it is *not* naturally favoring a random word to be tagged with "O". We planned to realize this via two steps:

- step 1: exclude all the files in the training data that has no "B-CUE" or "I-CUE" in it;
- step 2: for the remaining files, we truncate them by removing sentences which has only "O"s

After step 1, we are left with 666 files out of 1067 in the training data set. However, the second step unfortunately has a persistent bug that we are unable to fix. Despite that, the population of "O"s is already significantly reduced and the data is way more balanced than original.Consequently, the results favor SP the best. But improvements are seen in both CRF and SP. So we keep in mind that we should add unknown words smoothing in the final model if the model is either SP or CRF.

4.4 Experiment 4: Single feature + Different tagging system

Meanwhile, we are curious if a different tagging system will make a whole lot difference. Also in the spirit of balancing the data, we try this alternative tagging system where all "I-CUE" tags are replaced by "B-CUE".

4.5 Experiment 5: Multiple features + downsampling + ignoring all features that only appear once

Starting from this experiment, we will consider multiple features and run experiments with varyring one of them. (Note that we are only able to define multiple features in CRF but not SP or HMM (HMM doesn't even allow feature).) Meanwhile, we will keep downsampling as in Experiment 3 and also ignore all features that only appear once. In this experiment, we consider four features:

- word itself with all letters lowercased
- POS tags that are provided in the data
- first letter capitalized (rest lower case)
- k consecutive words in a sequence with the current word in the middle of its neighbour words

To find the best k, we tried $k = 3, 5, 7$. And the results are summerized in the table. We decide to use $k = 7$ since it has the best performance compared to smaller k's, and a larger k will make the computation too expensive.

4.6 Experiment 6: Multiple features

In this experiment, we drop the third feature "first letter capitalized (rest lower case)". So the features we include in our current model consist of:

- word itself with all letters lowercased
- POS tags that are provided in the data
- 7 consecutive words in a sequence with the current word in the middle of its neighbour words

And the result is worse than the one obtained from Experiment 5 (with $k = 7$).

Consequently, we pick up the model in Experiment 5 with $k = 7$ as the best model

Experiment		HMM	CRF	SP
1	Single feature	0.11	0.24	0.43
2	Single feature + UNK	0.00	0.00	0.02
3	Single feature + downsampling	0.08	0.35	0.44
4	Single feature + BO tagging	0.06	0.24	0.42
5	multiple features + downsampling + excl. 1-occurrence + k=3		0.54	
5	multiple features + downsampling + excl. 1-occurrence + k=5		0.53	
5	multiple features + downsampling + excl. 1-occurrence + k=7		0.57	
6	Experiment 6		0.56	

5 Error analysis

To further improve our model, we print out the false predictions (both false positives and false negatives) which can be found in the appendix below. We aim to spot any certain words/phrases that will always be missed or falsely tagged. The suspicious words that appear quite often in the false predictions include "may", "might", "many" and "some". However, all of them are not absolutely false positives or false negatives. So there is no enough strong evidence to forcibly set a word/phrase as a "B-CUE" or "I-CUE" or "O" always. Hence we do not add more features to the current model.

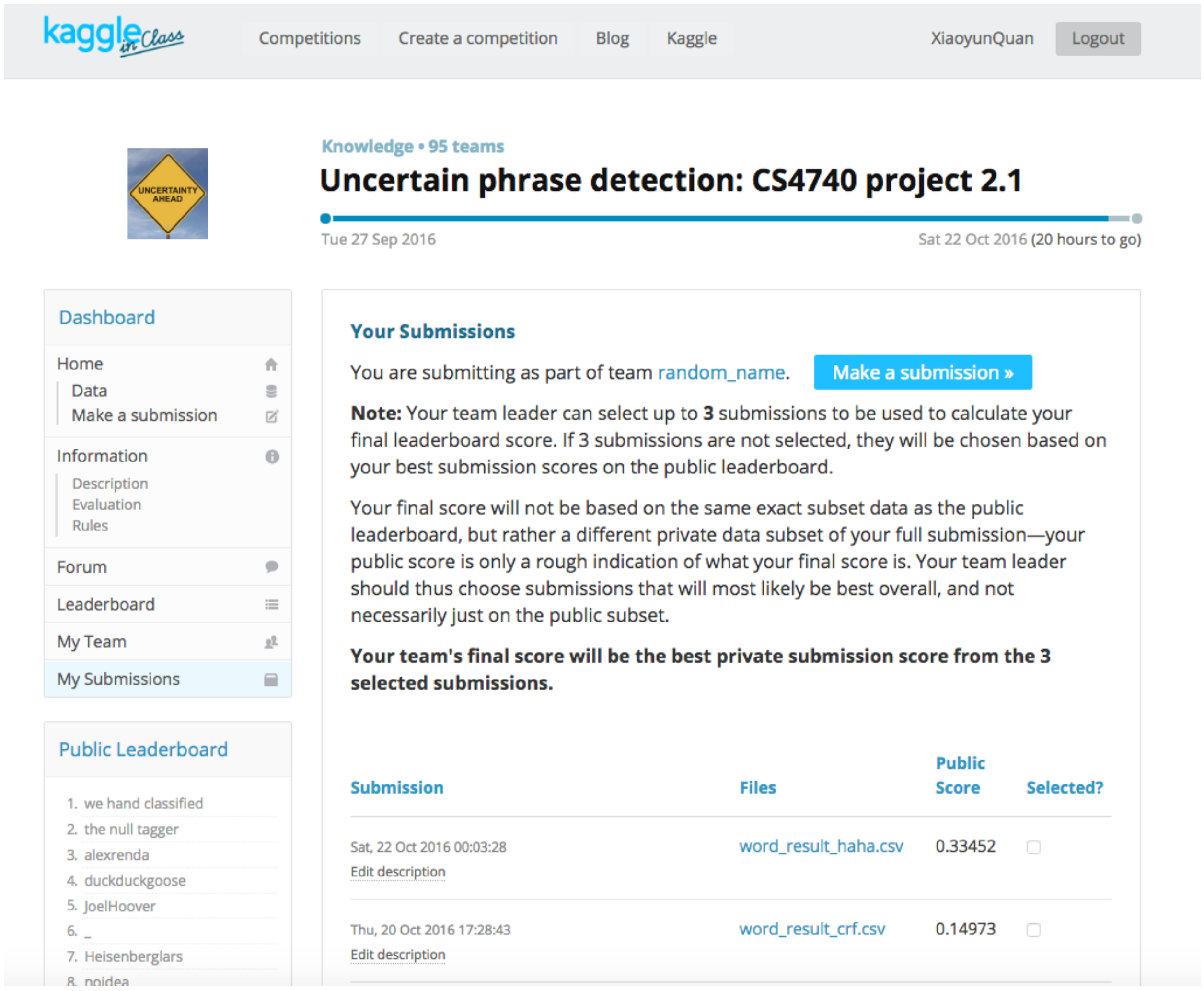
6 Sumamry

In conclusion, we implemented 3 methods: HMM, CRF, and SP, all done via packages `nltk`, `scikitlearn` and 'python-crfsuite'. Motivations of these methods can be found in section 3. We did preprocessing as described in section 2. We did post-processing analysis in section 5. We did two extensions: implementing CRF and SP in addition to HMM, but we think it might be better organized to put the implementation details of extension methods in the same section of HMM for easier comparisons (section 5). The extension method CRF with multiple features outperform all other models.

The entire team has been doing work together. We are glad that each member is actively involved and contributing to this project. The amount of work has been broken down fairly with willingness. Rui and Gi Yoon did the coding part, whilst Xiaoyun did theories research and report writing. Without any of us, the project couldn't been finished.

7 Code instruction + Kaggle best result snapshot

We have separate scripts for all the experiments: experiment 1-4 are done by `single_feature_model.py`; and experiment 5-6 are done by `multi_feature_model.py`. Note that the code of `multi_feature_model.py` is inspired by https://github.com/tpeng/python-crfsuite/blob/master/examples/CoNLL%202002.ipynb



Knowledge • 95 teams

Uncertain phrase detection: CS4740 project 2.1

Tue 27 Sep 2016 — Sat 22 Oct 2016 (20 hours to go)

1. we hand classified
2. the null tagger
3. alexrenda
4. duckduckgoose
5. JoelHoover
6. _
7. Heisenberglars
8. noidea

Your Submissions

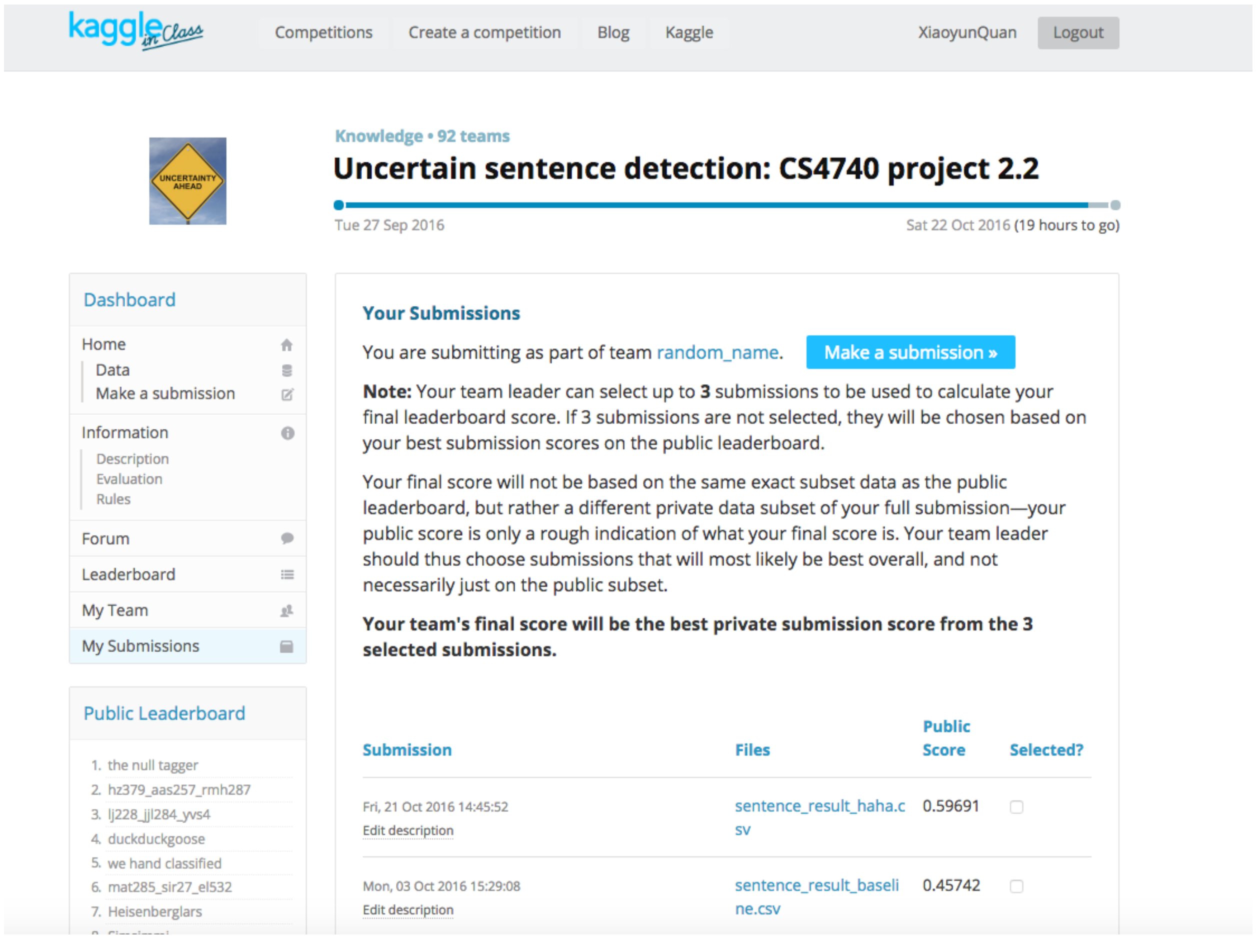
You are submitting as part of team random_name.

Note: Your team leader can select up to **3** submissions to be used to calculate your final leaderboard score. If 3 submissions are not selected, they will be chosen based on your best submission scores on the public leaderboard.

Your final score will not be based on the same exact subset data as the public leaderboard, but rather a different private data subset of your full submission—your public score is only a rough indication of what your final score is. Your team leader should thus choose submissions that will most likely be best overall, and not necessarily just on the public subset.

Your team's final score will be the best private submission score from the 3 selected submissions.

Submission	Files	Public Score	Selected?
Sat, 22 Oct 2016 00:03:28	word_result_haha.csv	0.33452	☐
Thu, 20 Oct 2016 17:28:43	word_result_crf.csv	0.14973	☐



Uncertain sentence detection: CS4740 project 2.2

Tue 27 Sep 2016 — Sat 22 Oct 2016 (19 hours to go)

Public Leaderboard

1. the null tagger
2. hz379_aas257_rmh287
3. lj228_jjl284_yvs4
4. duckduckgoose
5. we hand classified
6. mat285_sir27_el532
7. Heisenberglars

Your Submissions

You are submitting as part of team random_name. Make a submission »

Note: Your team leader can select up to **3** submissions to be used to calculate your final leaderboard score. If 3 submissions are not selected, they will be chosen based on your best submission scores on the public leaderboard.

Your final score will not be based on the same exact subset data as the public leaderboard, but rather a different private data subset of your full submission—your public score is only a rough indication of what your final score is. Your team leader should thus choose submissions that will most likely be best overall, and not necessarily just on the public subset.

Your team's final score will be the best private submission score from the 3 selected submissions.

Submission	Files	Public Score	Selected?
Fri, 21 Oct 2016 14:45:52 Edit description	sentence_result_haha.csv	0.59691	☐
Mon, 03 Oct 2016 15:29:08 Edit description	sentence_result_baseline.csv	0.45742	☐

8 Reference

Collins, M. 2002. "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms". *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, July 2002, pp. 1-8.

Nguyen, N., Guo, Y. 2007. Comparisons of Sequence Labeling Algorithms and Extensions. *Proceedings of the 24 th International Conference on Machine Learning*, Corvallis, OR, 2007.