C++ Parallel application for Traveling Salesman Problem resolution by genetic algorithm

Parallel and distributed system: paradigms and models
2019 / 2020

Berti Stefano

Abstract

The aim of this project is to implement a genetic algorithm for the resolution of the Traveling Salesman Problem and to parallelize it with two different mechanism:

- C++ standard threads and mutex
- Fastflow library

1 Problem description

The genetic algorithm mimin the natural evolution and selection. In order to do that, we need a population, a way to understand which elements of the population will surivive (and so will reproduce), how they reproduce and how mutations can occour.

- **Population**: is a matrix (vector of vector), where each row contains a sequence of number, and each number represent a node
- **Affinity**: for each element x is the probability that x will be selected for reproduction, and is calculated as the inverse of the path length, normalize w.r.t. all other elements in order to obtain a proper probability
- **Reproduction**: 2 paths A and B can create a new path C by generating 2 random numbers i and j and putting the subsequence B[i, j] in A
- **Mutation**: a mutation occour with a certain probability to every generated path, and it is a random shuffle of two position in the paths. The probability that a mutation does not occour is 0.9.

To calculate the length of each path, I use a random generated city, where for each node number we have the relative coordinates. The city can also be plotted for real time problem resolution visualization, but this has heavy consequences on the program performances.

id, topic, y, review

where

- **id** is the id of the review
- **topic** is one element in $['cleanliness','amenities','value','wifi','location','staff','other']$
- **y** in ACP task, this refers to the sentiment of the review towards that topic and it is an element in $['positive','negative','neutral','mixed']$, in ACD task this refers if the topic is dealt in the review or not and it is an element in $['positive','negative']$
- **review** is the raw review

Table 1: element for class

data	**positive**	**neutral**	**mixed**	**negative**
train	4942	0	173	3797
test	2080	0	64	1757

Since I didn't have a single element for neutral review neither in train set nor test set, I decided to remove it from the possible classes. I also had to create negative samples for the ACD train and test set, since I only had positive samples. In order to do so, for each review and for each topic I added that review with that topic with and *negative* as y with a certain probability, that I set to value 0.2. The metrics used in this competition were $micro - precision$, $micro - recall$ and $f1 - score$.

2 Possible models

These tasks like others sentiment analysis tasks can be approached with statistical models and a correct preprocessing of the text. LinearSVC combined with lemmatization is very good in understanding which topic is dealt inside a review, and other statistical approaches can give us good results in sentiment analysis, but here we are aiming to something more complex: our model can also understand if a review contains mixed sentiment, which in a single review can help us understand more information. Also BERT fine tuning could be good, but it is not topic-related, so it would give us information about the entire review, without paying attention to the different topics and the different sentiments towards them.

3 Description of model chosen

The model chosen is the one who wins the *Semeval*2017, which tasks were very similar to this one. It is a deep-learning model with context-aware attention:

- It embeds each word of the sentences and the topic using the same word embeddings.

- It feeds each embedding in a LSTM. It does the same with the topic using the same weights in order to try to get meaningful representation.
- It concatenates each word representation with the topic representation
- It uses a context-aware attention mechanism, which tries to understand which part of the reviews contribute more to understand better sentiment/references towards the topic
- It feeds those representations in a dense layer with a single sigmoid neuron for task ACD and 3 softmax neurons for task ACP

4 Experiments

4.1 ACP

Initially I only considered the positive and negative classes because of the very few mixed samples, by assigning to mixed reviews a random sentiment, and, although this gives nice results, I moved from a one sigmoid output neuron for positive and negative class, to 3 softmax output neurons for positive, negative and mixed class. I did lose some accuracy, around 5%, but theoretically this way is more consistent to the task. I used class weights, which help dealing with imbalanced datasets by weighting more the misclassified prediction of a class with a limited number of example. In my case the class weights used were 1.26 for negative, 1.0 for positive and 8.14 for mixed. Embeddings are given in a txt file, but after the first loading, they are formatted as dictionary and saved in a pickle file to speed up following loading. Initially I made some experiments using a word embedding whose dimension was 128, but this lead to a slow and limited training, that couldn't overtake the accuracy score of 0.55, which was very bad. So I moved to a more informative embeddings which size is 300, and although the big dimension of 2.4GB and the poor quality of most of the entries, it gives good results. I didn't do a lot of preprocessing of the reviews, since the embedding size is big (667564 words one removed the non-ascii ones) and it could cover the 85% of the words, but looking better at the words that were indexed as $< unk >$, a lot of them were words with a capital letter. So by setting all chars lowercase, without losing much information because no proper name was present in the dataset, and just by removing all **l'**, the coverage of ward embeddings increased to 95%. This can be increased more correcting typo using edit distance for example, but I did not investigating further since this improved coverage improved my metrics by more or less 0.04, that was what I needed.

4.2 ACD

Even if this model was not designed for topic-detection analysis, I experimented a way to understand its potential and limit. So I proceed modifying the dataset in such a way to only detect if a topic is dealt in a review, and not in which

way it is dealt. The idea is that the context-aware attention should learn which words refers to a specific topic. So I tried various probability to add a negative sample (see ACD dataset description) since I obtain both class weights around 1.0, hoping that a balanced dataset could give better results.

5 Results

I had the official gold test set during training, but I didn't use it for model selection since that would have been incorrect. So the model selection is based on the highest validation recall. I used the official evaluation_absita script for calculate the scores.

- **ACD results**: the obvious and only way to predict the presence of topic in a sentence, is to try each topic for each sentence and collect the positive results only.

Table 2: independent results for ACP task

model	**Micro-Precision**	**Micro-Recall**	**Micro-F1-score**
absita best model	0.8397	0.7837	0.8108
my model	0.6832	0.8204	0.7455

- **ACP results**: this results can be calculated in two ways: using the output of the previous task as input, or create a new input based on the right presence of topics in sentences. In the first way, the results obviously is dependent from the result of the first task, in particular the score cannot be higher than what we obtain from the ACD task. In the second way,

Table 3: DEPENDENT results for ACP task

model	**Micro-Precision**	**Micro-Recall**	**Micro-F1-score**
absita best model	0.8264	0.7161	0.7673
my model	0.3675	0.4366	0.3991

  we test the model with correct topic, leading to a better analysis of this task

Obviously the second way is not possible if we don't have the gold test data, so for completeness i considered both cases.

Table 4: INDEPENDENT results for ACP task

model	**Micro-Precision**	**Micro-Recall**	**Micro-F1-score**
absita best model	0.8264	0.7161	0.7673
my model	0.9277	0.9162	0.9219

6 Conclusion

Deep-Learning approaches with context-attention has difficulty to understand the topics dealt in reviews. This could be due to a small dataset, which contains few element for each topic class. Simpler statistical approaches like LinearSVC gives us better results, arund 90% for each class, thanks to lemmatization which reduces the vocabulary and has fewer elements to analyze. Nevertheless, the model is very good at understanding the sentiment towards a certain topic, and it can also understand if the sentiment towards a topic are mixed, and in different topics have different sentiments in the same review.

7 Other plots