Udacity Capstone Proposal: Argumentation-focused Machine Learning

Julio Cezar Silva

Semptember 17, 2020

1 Introduction

The current project proposal aims for building an ML intelligence capable of partially understanding the concept of argumentation robustness, that is, how strong is a given claim, towards or against a given thesis. When passed a claim about a specific thesis as input, the model should then output a score for that argument's robustness pertaining the thesis. Below lay the specifics about how such project would be design and implemented.

2 Domain background

In terms of the Machine Learning strategies, the project is based on Natural Language Processing (NLP) and regression. NLP, being comprised of a vast array of techniques around the intelligent/dynamic understanding of textual context, and regression a statistical analysis technique, to be used in this project to fit the robustness function.

3 Problem Statement

Debate resolution helps shape social and scientific progress since the dawn of time. In today's era this is not only factual offline - it's also one of the backbones of the internet:

- StackOverflow and over a hundred siblings
- a vast portion of Reddit
- Quora

... and the many forums in between. Even if it was resumed to the above most popular examples, Q&A was made to thrive on the resolution of various rarely one-sided debates. The goal of This project's goal is to help lead to intelligence that can receive a **debate as input**, and **output the answer** that's most robust - and likely right.

This way, if after submitted as Capstone Project, the project presents promising consistency, its contextual grasp may be transferable to debates that are external to the original dataset's source, unto one of the mentioned Q&A platforms, as a dynamic way to determine robustness in argumentations in general.

4 Datasets and Inputs

The dataset for this project is custom, and has been scraped from Kialo's APIs by myself - the code for this is inside the `kialo_scraping.py` file, and can be tested through `discussions = get_discussions(); scrape_into(discussions)`.

In this dataset, each discussion has a list of `dicts`, which are the claims around what the discussion: one main claim, the root node of the discussion, also called thesis. The thesis contains `pros` and `cons`, ID's of its child nodes the either support or oppose it. In fact, all claims have that behavior, as well as the `text` property, holding the full textual content of a claim, as well as `created`, the datetime (in seconds since epoch) indicating when the claim was first posted. Other metadata, such as user account IDs, was discarded from the API for not being directly related to the purpose of the project.

5 Solution Statement

The steps planned to be followed for solving this problem are:

- Representing a given debate as a tree data structure (as explained above, already done in the dataset)
- Defining (or choosing from literature, if available) a formula for the weight of a claim based on its ratings done by users and its proportion of pro and con child nodes. The formula should also be involved when calculating the robustness of a thesis (which is nothing more than a node that comes before all others in the discussion tree).
- Preprocessing data and adding custom metrics to calculate the weight of a given claim, and cumulatively, the positive/negative weight of the thesis.
- Natural language understanding of context within individual argumentations (given BERT [1] pre-trained model's embeddings as input)
- Propagating PCA-reduced embeddings of theses to their associated argumentations for broader contextual input
- Evaluating per-node predicted voting weights vs actual calculated weights, to determine RMSE within an `LGBMRegressor`

BERT's embeddings will be used to represent all of the claims' texts, including the thesis', which will be reduced by PCA.

6 Evaluation Metrics

The evaluation metric is going to be RMSE [2]:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Predicted_i - Actual_i)^2}{N}} \tag{1}$$

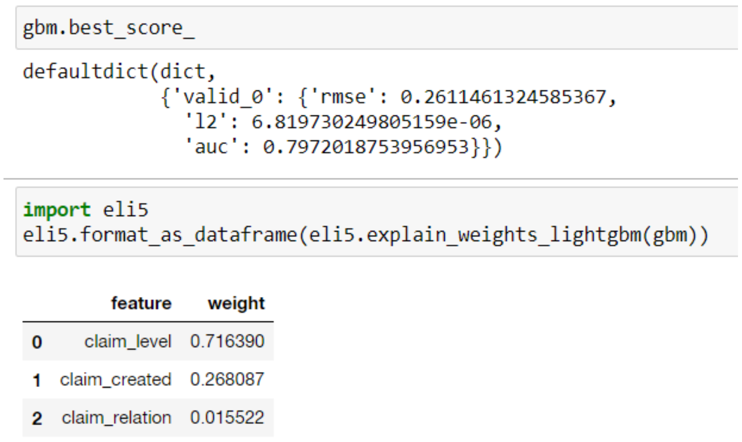
The squaring of the difference between predictions and labels helps trim the biggest predicted discrepancies on w_x, then stopping its propagation onto calculating R_T from predicted argumentation weights.

7 Benchmark Model

I trained a baseline `LGBMRegressor` model, without embedding features, based on just the features:

- `created` (creation datetime)
- `level` (at which depth within the tree was the claim posted)
- `relation` (-1 if the current claim is opposing its parent claim, 1 if it's supporting)

For how simple and standard (in terms of parameters) the model is, it should serve as a baseline for assessing if the embedding, PCA and engineered features are actually improving the scores:



```
gbm.best_score_
```

```
defaultdict(dict,
            {'valid_0': {'rmse': 0.2611461324585367,
              'l2': 6.819730249805159e-06,
              'auc': 0.7972018753956953}})
```

```
import eli5
eli5.format_as_dataframe(eli5.explain_weights_lightgbm(gbm))
```

	feature	weight
0	claim_level	0.716390
1	claim_created	0.268087
2	claim_relation	0.015522

Figure 1: Metrics (focus currently on RMSE) for baseline LGBMRegressor.

The focused metric in comparison is the RMSE, that can be interpreted as *roughly an averaged distance* of the model towards the aimed results (labels) [2]. If, in the final project version, the models hold a better RMSE score (ideally, 10 times smaller than the baseline's 0.26 score), such models will be considered better than the baseline. The feature importances, considering the introduced embeddings, PCA, and other engineered features[1], should also reflect the progress over baseline, by having a drastically difference top-3 feature list, or at least having high-ranked contributions of the introduced features. That would mean that not only has the model improved, but that evaluated improvement is largely due to the new insights on data.

[1]e.g. a time delta from the creation of a discussion to the creation of the current claim

8 Project Design

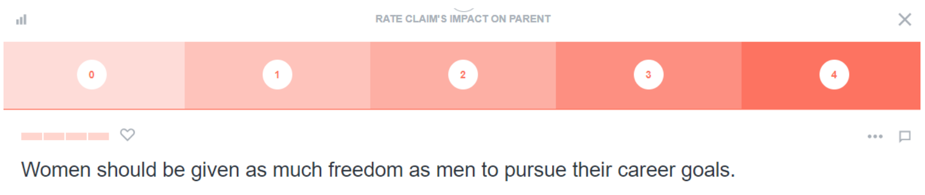
There are two main blocks to this project: the mathematical modeling of metrics, and the implementation itself. Before the project implementation, an EDA round is plausible: analysing the distribution of votes for discussions over time, or comparing custom metrics to other metrics coming from the Kialo API itself. After that, the project implementation can be summed by the following pseudo-code:

```
for discussion in discussions
    for claim in discussion
        claim.clean_text = preprocess(claim.text)

        if claim is thesis
            thesis_weight = weight_based_on_all_children_of(thesis)
        else
            weight = normal_claim_weight(claim)

        claim.embeddings = BERT.embeddings(claim.text)
        claim.pca_features = PCA(BERT.embeddings(thesis.text))

predicted_weight = regression_model(some_claim)
loss = predicted_weight - actual_weight
regression_model.learn(loss)
```

Yet, the first moments must be thoroughly focused on the research and not only implementation: to asses that, the following will be define what is needed for calculating - and labeling - individual argumentation weights.

8.1 Claim Rating System

By design, Kialo uses a 5 point Likert scale as the rating system for all claims therein. Users can rate a claim's impact on parent (be it a claim or the thesis), in a scale of No Impact (0), Low Impact (1), Medium Impact (2), High Impact (3), Very High Impact (4), as seen below.



Women should be given as much freedom as men to pursue their career goals.

Figure 2: Impact rating from a con-node for the Should women stay at home to raise children? discussion.

8.2 Calculating of individual weights

The robustness of a single argumentation node will be weighed by pro/con votes traversing the tree downwards from the root node. Within the discussion tree, the impact weight w_x for a single argumentation node x will be

$$w_x = \frac{\sum_{i=1}^{s} r_i}{r_t * r} + \frac{l}{l_x}, \qquad (2)$$

where:

r_i	= number of impact ratings for the claim, by type (”No Impact”, ”Low Impact”, ...)
s	= number of rating types in the scale (equals 5 for Kialo’s 5 point scale)
r	= total number of ratings in the discussion
l	= total number of levels in the discussion
l_x	= the level x is in; $d \geq 1$
w_x	= impact of x on its parent node

Both r and d are normalization variables, making a thesis with more activity (and thus more votes and levels) comparable to one that’s less active or just new.
By also depending on d_x instead of just r_i, w_x balances the weight of nodes that are farther down in the tree, since those nodes are less visible in the interface and thus more likely to have zero votes. If w_x was just dependent on votes, it’d result in an unrealistic weight of 0 for arguments lacking any vote. At the bottom of the tree, top arguments get broken down into increasingly specific nodes, which are essential proofs of smaller parts of the thesis itself. V_x is inversely proportional to d_x because those smaller, essential arguments prove generally less about the thesis as a whole.

References

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018. 2

[2] W. Wang and Y. Lu, “Analysis of the mean absolute error (MAE) and the root mean square error (RMSE) in assessing rounding model,” *IOP Conference Series: Materials Science and Engineering*, vol. 324, p. 012049, Mar. 2018. 2, 3