Machine Learning Engineer Nanodegree
Capstone Project Proposal

by Pablo Campos Viana

I - Domain Background

Tag annotations have become a useful and powerful feature to facilitate item search in many websites and web applications. In many cases, the majority of tags assigned to items are supplied by users, a task which is time consuming and may result in annotations that are subjective and lack precision[1]. Therefore, finding good ways to perform automatic tagging is crucial in these scenarios and in recent years Machine Learning (ML) approaches have arisen.

Stack Overflow, according to its official site, *is the largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers.* In the context of the Stack Overflow site, a tag is a word or phrase that describes the topic of the question and according to the help page about tagging of the site, they are a means of connecting experts with questions they will be able to answer by sorting questions into specific, well-defined categories, and they can be used to help users to identify questions that are interesting or relevant to them. This way, automated tagging is an important feature of the site since it improves user experience and fosters user engagement.

In 2012, Stack Overflow released a competition to build ML models to predict which new questions asked on its site will be closed, along with data that can be used for other kind of tasks beyond the goal of the competition, for example: identifying tags from question text, predicting whether questions will be upvoted or downvoted based on its text, or predicting how long questions will take to answer. In this project I'm interested in the former one. In this context since each question can be associated with more than one tag, from a ML perspective, the problem can be approached as a multi-label classification one.

At a personal level, this problem motivates me because of its degree of applicability. There are a wide range of websites and web applications that are currently using or can potentially use ML powered solutions to perform automatic tag generation, using text or other sources of data (image, video, etc.). More specifically, I'm interesting in using text to perform automatic tagging because in my current work I have been developing and deploying text categorization engines that predict multiple categories from short financial text and I find this problem to be closely related.

---

[1] Zhijie Shen, Sakire Arslan Ay, Seon Ho Kim, and Roger Zimmermann. 2011. Automatic tag generation and ranking for sensor-rich outdoor videos. In Proceedings of the 19th ACM international conference on Multimedia (MM '11). ACM, New York, NY, USA, 93-102. DOI: https://doi.org/10.1145/2072298.2072312

II - Problem Statement

The problem to be solved can be stated as **automatic tag generation**, where, given text of questions, we want to predict their most probable tags. As mentioned before, the problem can be approached as a multi-label classification one.

Furthermore, since this is an instance of a supervised learning problem and since we can generate both training and validation sets from the available data, the problem is quantifiable (since we can use standard classification metrics). One potential solution is to perform multi-label text classification using BlazingText.

III - Datasets and Inputs

The dataset considered for this project is that of StackSample. According to Kaggle's official documentation, the StackSample is a dataset with the text of 10% of questions and answers from the Stack Overflow programming Q&A website.

This dataset is organized as three tables:

- **Questions** contains the title, body, creation date, closed date (if applicable), score, and owner ID for all non-deleted Stack Overflow questions whose Id is a multiple of 10.
- **Answers** contains the body, creation date, score, and owner ID for each of the answers to these questions. The ParentId column links back to the Questions table.
- **Tags** contains the tags on each of these questions.

However, for the tag prediction problem I want to solve, I need to use only the Questions and Tags tables. At a high level, the tag prediction problem is a multi-label classification problem, i.e. a Supervised ML problem where the multi-labels are given by the tags of the questions, and the features will be obtained from the Questions table; specifically, I will be working only with the title (raw text to be further processed) of the questions.

This dataset can be obtained programmatically by means of the Kaggle's Public API via CLI, so the data acquisition can be done easily and the project can be reproducible from the beginning.

IV - Solution Statement

As mentioned before, one potential solution is to perform multi-label text classification using BlazingText. Another approach would be to train a word embedding and build a LSTM model (in the same architecture or not) to predict the most probable labels for a given text query, using a Deep Learning framework (specifically PyTorch).

Since we can generate predictions for any of the solutions described, and therefore we can compute classification metrics, the solutions are quantifiable. Also, since: 1) data is publicly available, 2) BlazingText and PyTorch are available in Amazon SageMaker, and 3) we can set seeds for any method with potential pseudorandom behaviour, the solutions are replicable.

V - Benchmark Model

An existing solution to the problem has been described by the developer Yi Ai in a Medium post. His solution uses fastText and I have noticed that is replicable. However, no hyperparameter tuning is performed and no metrics are reported. Nevertheless, with the settings described in his solution, I can replicate his model to use as baseline and evaluation metrics chosen for this project can be computed to use as benchmarks. At this moment, I have not found any other published solution for this particular problem and dataset.

VI - Evaluation Metrics

The evaluation metrics to be used are the standard classification metrics such as precision and recall, and their extensions to multi-label settings, this is, precision@k and recall@k, where *k* is the number of predictions requested for a given text query.

VII - Project Design

The project workflow can be summarized as follows:

a) **Data Ingestion**
   Download data using Kaggle's Public API via CLI.

b) **Data Preprocessing**
   Perform the following in a programmatic, modular and efficient way:
   Get text title from questions table, perform text cleaning and other standard NLP preprocessing.
   Match questions titles text (features) and tags text (labels) in one single table, and leave it with the format needed by both fastText and BlazingText.
   Split table pseudorandomly into training and validation sets, and upload them to an Amazon S3 bucket.

c) **Build a benchmark model**
   i) Replicate model following approach described in section V to use as benchmark.
   ii) Get evaluation metrics on validation data on validation data via a batch job.

d) Build two proposed models

i) BlazingText: Hyperparameter tuning and choice of best model.
Get evaluation metrics on validation data via a batch job.

ii) LSTM network: Architecture selection, based on literature review.
Hyperparameter tuning and choice of best model.
Get evaluation metrics on validation data via a batch job.

e) Hosting & Inference

i) Choose a model to deploy.

ii) Setup a Lambda function via AWS Lambda and setup an API via Amazon API Gateway.

iii) Build a web application and use it to perform real time inferences: given a question provided by the user, display its most probable tags according to the deployed model.