Group number 21:

Arjun D'Cunha (CS14BTECH11039)
Bhavana Jain (CS14BTECH11008)
Yash Chandarana (CS14BTECH11040)

Project title:

Tag suggestion for Stack Overflow

Dataset:

Step 1: We obtained question IDs from the StackLite dataset on kaggle.
Step 2: To obtain question body and tags corresponding to each question id we used the StackAPI. The code to fetch the complete dataset is in `fetch_dataset.py`

We have attached the raw dataset with question id, body and tags. Please find it in file `dataset.txt`

Following is a snippet from the raw dataset:

```
193896
;;;;;
c linux reverse-engineering decompiling assembly
;;;;;
<p>I am searching for a decompiler for a C program. The binary is a 32-bit x86 Linux executable. Objdump works
fine, so basically I am searching for something which attempts to reconstruct the C source from the asm source.</p
>

;;;;;

|||||
194121
;;;;;
php deprecated
;;;;;
<p>At the team with which I work, we have an old codebase using PHP's ibase_* functions all over the code to
communicate with database. We created a wrapper to it that would do something else beside just calling the
original function and I did a mass search-replace in the entire code to make sure that wrapper is used instead.</p
>

<p>Now, how do we prevent usage of ibase_* functions in the future?</p>

<p>Preferably, I'd like to still have them available, but make it throw a NOTICE or WARNING when it is used.</p>

<p>A solution in pure PHP (not needing to compile a custom version of PHP) is preferred.</p>

;;;;;

|||||
```

We have used `;;;;;` to separate categories (id, tags, body) within a post and `|||||` to separate posts.

We have implemented two methods to predict tags for Stack Overflow posts:

Deep model approach

The training part of this approach is done using `train_model.py`
To prepare data for training, we need to separate documents and ground truth labels in two files - `processed_docs.txt` and `labels.txt`
The code for this is available in `preprocess.py`
How do we process the raw posts?

1. Select top 118 tags based on their occurrence count in the tags dictionary and only consider posts which has at least one tag belonging to the set. Furthermore, for each post we only consider the top tags and discard the rest.
2. Use BeautifulSoup to filter out all elements containing code.

```
question_body = categories[2]
soup = BeautifulSoup(question_body, 'html.parser')

# Remove all tags with a class or id containing the word snippet
# Later use these snippets to predict the programming language ^_^
for snippet_tag in soup.find_all(attrs={'class': re.compile('snippet')}):
    snippet_tag.decompose()
for snippet_tag in soup.find_all(attrs={'id': re.compile('snippet')}):
    snippet_tag.decompose()

# Remove all the <pre> ... </pre> tags
for extra in soup('pre'):
    extra.extract()
```

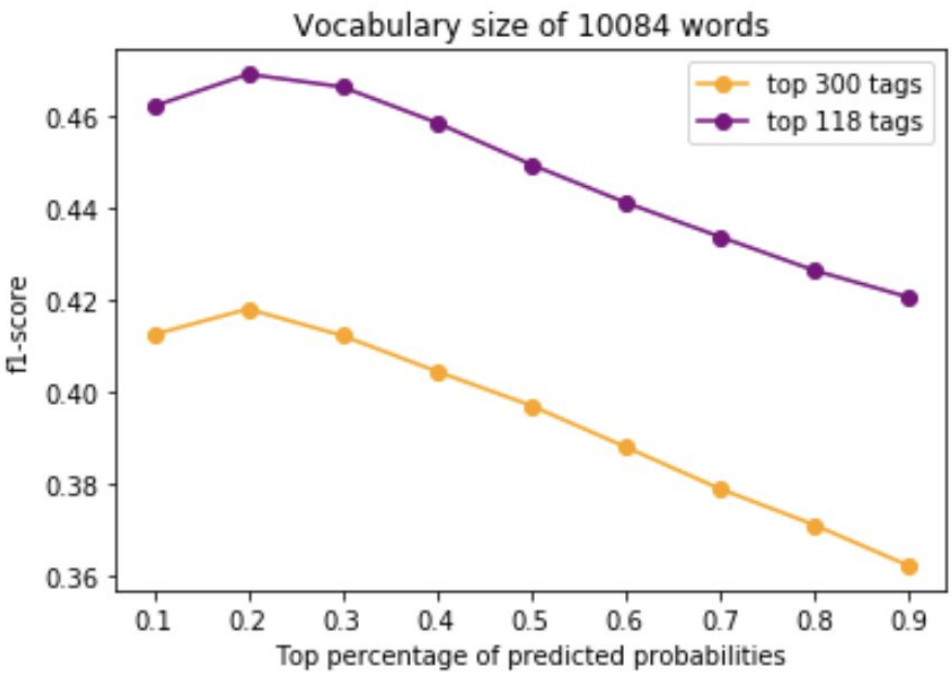
3. Filter out numbers, non-alphanumeric characters, punctuations appended to tokens and stopwords.

In `train_model.py` we convert processed docs to tf-idf vectors:

```
data = open('./processed_docs.txt', 'r').read().split('\n')[:-1]
vectorizer = TfidfVectorizer(input='content', token_pattern=r'\S+', analyzer='word', min_df=0.0002)
X = vectorizer.fit_transform(data).todense()
```

We input the obtained tf-idf vectors to a deep model which outputs a binary vector of size number of total tags (118 in our case). We use `softmax activation` in the last layer and `categorical_crossentropy loss` to train our model. More details can be found in `train_model.py`

Results from the deep model approach:



We get a maximum `F1-score` of `0.48` when we consider top 118 tags and threshold the top 20% of the values in the last layer of the neural network to 1.

Supervised LDA approach

Here we approach the problem as one of Topic Modelling. We preprocess our data in a similar manner as above but we store it in the following format:

```
[label1, label2, ...] document tokens
```

Sample processed dataset looks like:

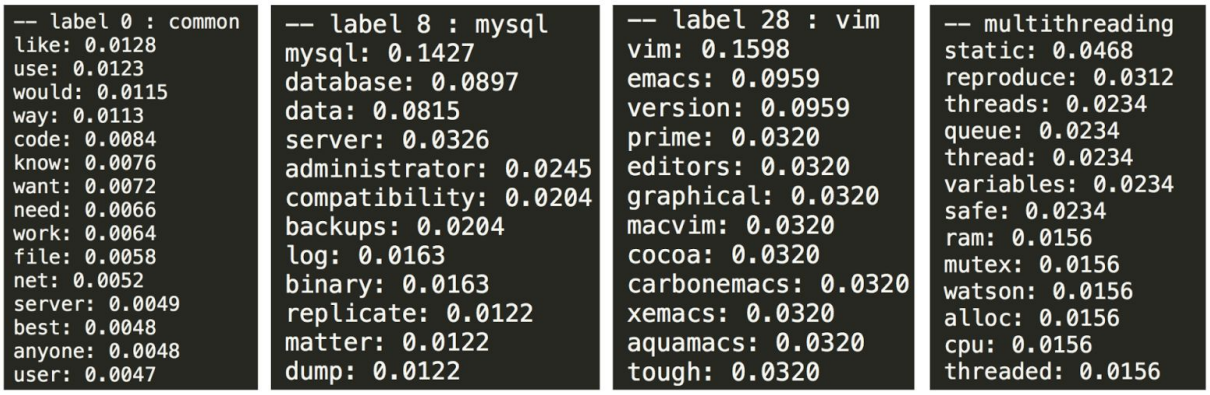
```
[sql,database,version-control,oracle]often run following problem work changes project require new tables
columns database make database modifications continue work usually remember write changes replicated live
system however don't always remember i've changed don't always remember write make push live system get big
obvious error newcolumnx ugh regardless fact may best practice situation version control system databases don't
care specific database technology want know one exists happens work ms sql server great
[sql,asp.net,xml]anyone got experience creating sql-based asp.net site-map providers i've got default xml file
web.sitemap working properly menu sitemappath controls i'll need way users site create modify pages dynamically
need tie page viewing permissions standard asp.net membership system well
```

We ran the Supervised LDA code on 800 posts since it was not able to scale to the full dataset of 35k posts. The processed dataset can be found in `800_llda_dataset.txt`

The code is present in `llda.py`

Results:

Here we are showing the top words corresponding to certain topics.



Instructions to run code:

- **Deep model**
  a. `python3 preprocess.py`
     This will run on `dataset.txt` and create the `processed_docs.txt` and `labels.txt`
  b. `python3 train_model.py`
  c. `python test_model.py`

  [**Extra:** `plot.py` to visualise results]

- **Supervised LDA model**
  a. `python llda.py -f 800_llda_dataset.txt -k 171`
     `-f dataset filename`
     `-k number of tags`

  (For the given dataset file, number of tags is 171)