Automated Tag Classification and Question Routing on AI Stack Exchange Using Classical NLP Techniques

Bojan Dimovski[1], Tomi Nikoloski[1]

[1]Faculty of electrical engineering and information technologies

Outline

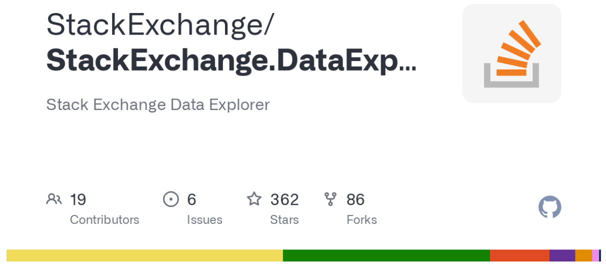
Outline

Introduction

Definition

Natural Language Processing (NLP) is a branch of artificial intelligence that enables machines to understand, interpret, and generate human language. In the context of specific applications, NLP is used for automatic translation, recommendation systems, and text data classification.

The goal of this project is to create a model that accurately predicts tags for AI-related questions on the AI Stack Exchange platform using classical NLP techniques, using vectorization.



Initial raw text cleaning

- XML files (https://dn720201.ca.archive.org/0/items/stackexchange)
- Parsing XML files using xml.etree.ElementTree
- Cleaning Posts.xml with BeautifulSoup, html, regexes

```python
import html

df_raw['Body'] = df_raw['Body'].apply(lambda x: html.unescape(x) if isinstance(x, str) else "")
df_raw['Title'] = df_raw['Title'].fillna("")

df_raw['Content'] = df_raw['Title'] + " " + df_raw['Body']

df_raw.drop(columns=['Title','Body'])

df_raw = df_raw[['OwnerUserId','Content','Tags']]

df_raw.head(10)
```

✓ [8] 93ms

10 rows 10 rows × 3 columns

	OwnerUserId	Content	Tags
0	8	What is "backprop"? <p>What does "backprop" mean? Is the "ba…	[neural-networks, backpropagation, terminology, definitions]
1	8	How does noise affect generalization? <p>Does increasing the…	[neural-networks, machine-learning, statistical-ai, generali…
2	8	How to find the optimal number of neurons per layer? <p>When…	[neural-networks, hyperparameter-optimization, artificial-ne…
3	29	Are humans intelligent according to the definition of an int…	[philosophy, definitions, intelligent-agent]
4	26	Why does Stephen Hawking say "Artificial Intelligence will k…	[agi, superintelligence, singularity, ai-safety, ai-takeover]
5	8	What is fuzzy logic? <p>I'm new to A.I. and I'd like to know…	[deep-neural-networks, terminology, fuzzy-logic]
6	38	Can a single neural network handle recognizing two types of …	[neural-networks, image-recognition]
7	9	Is the Turing Test, or any of its variants, a reliable test …	[turing-test, agi, intelligent-agent, narrow-ai]
8	8	What is "early stopping" in machine learning? <p>What is <a …	[deep-learning, definitions, overfitting, regularization, ea…
9	55	What is the concept of the technological singularity? <p>I'v…	[philosophy, definitions, agi, superintelligence, singularit…

Multilabel classification - performance metrics

$$\text{hamming\_loss} = \frac{\sum_l \sum_i P_i \oplus T_i}{|L| \cdot |I|}$$

$$\text{recall\_per\_instance} = \frac{\text{CPPL}}{\text{CPPL} + \text{IPAL}}$$

$$\text{precision\_per\_instance} = \frac{\text{CPPL}}{\text{CPPL} + \text{IPPL}}$$

$$F_1 = \frac{2}{\frac{1}{\text{RPI}} + \frac{1}{\text{PPI}}}$$

- Macro, micro
- Subset accuracy

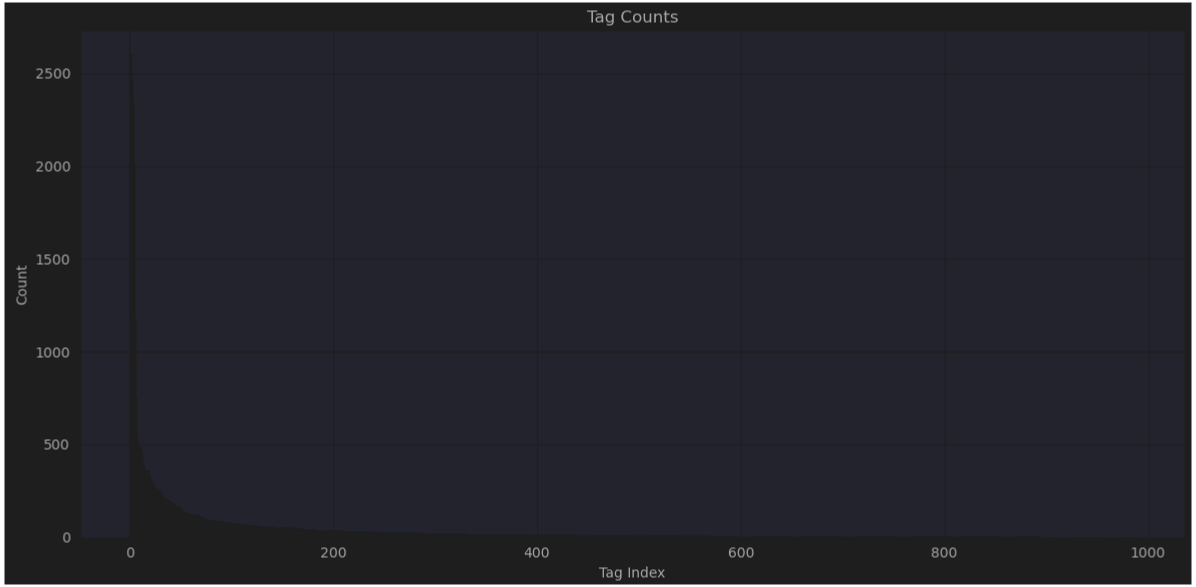
Tag selection

📊 986 tags

	Tag	Count
0	neural-networks	2600
1	reinforcement-learning	2462
2	machine-learning	2354
3	deep-learning	2021
4	convolutional-neural-networks	1179
...	...	...
981	gini-impurity	1
982	next-frame-video-prediction	1
983	video-generation	1
984	mistral	1
985	accelerators	1

Tag selection

- 📊 Tag frequency decreases exponentially



Tag selection

- 📊 Let's see how many posts are covered only by, say, 10 tags:

Tag selection

- Let's see how many posts are covered only by, say, 10 tags:

```python
NUM_TAGS_STOP = 10

top_ten_tags = set(df_tag_counts['Tag'].head(NUM_TAGS_STOP).to_numpy())
count = 0
for instance in df_raw['Tags']:
    if set(instance) & top_ten_tags:
        count+=1

print(f"Vkupno {count} postovi sodrzat barem eden od top {NUM_TAGS_STOP} tagovite :)")
```

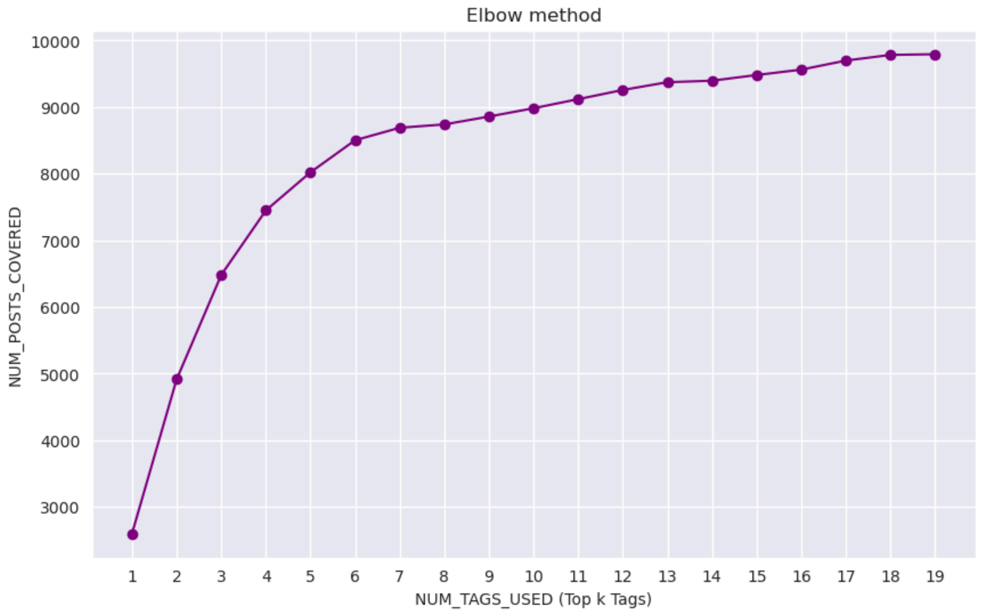
✓ [17] 17ms

```
Vkupno 8981 postovi sodrzat barem eden od top 10 tagovite :)
```

- Elbow method (?)

Number of tags

- Cut-off at 7? 10? 13?



- Top 10 tags: covers ≈9000 out of 12000 posts.
- Intersect: set1 & set2

Exploratory data analysis

Basic statistics:

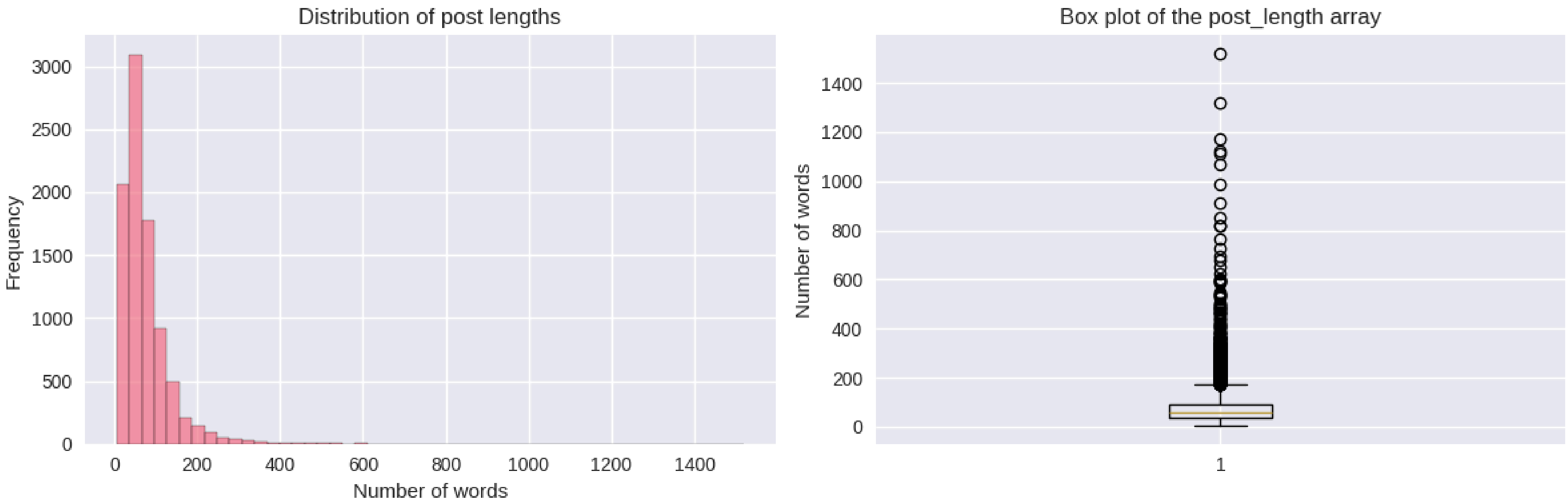
- 684,383 words across 8,981 posts
- Appearances of top 10 tags:

	Tag	Count
0	neural-networks	2600
1	reinforcement-learning	2462
2	machine-learning	2354
3	deep-learning	2021
4	convolutional-neural-networks	1179
5	natural-language-processing	759
6	computer-vision	536
7	deep-rl	510
8	training	489
9	classification	484

- Disbalance with multilabel classification problems

Exploratory data analysis

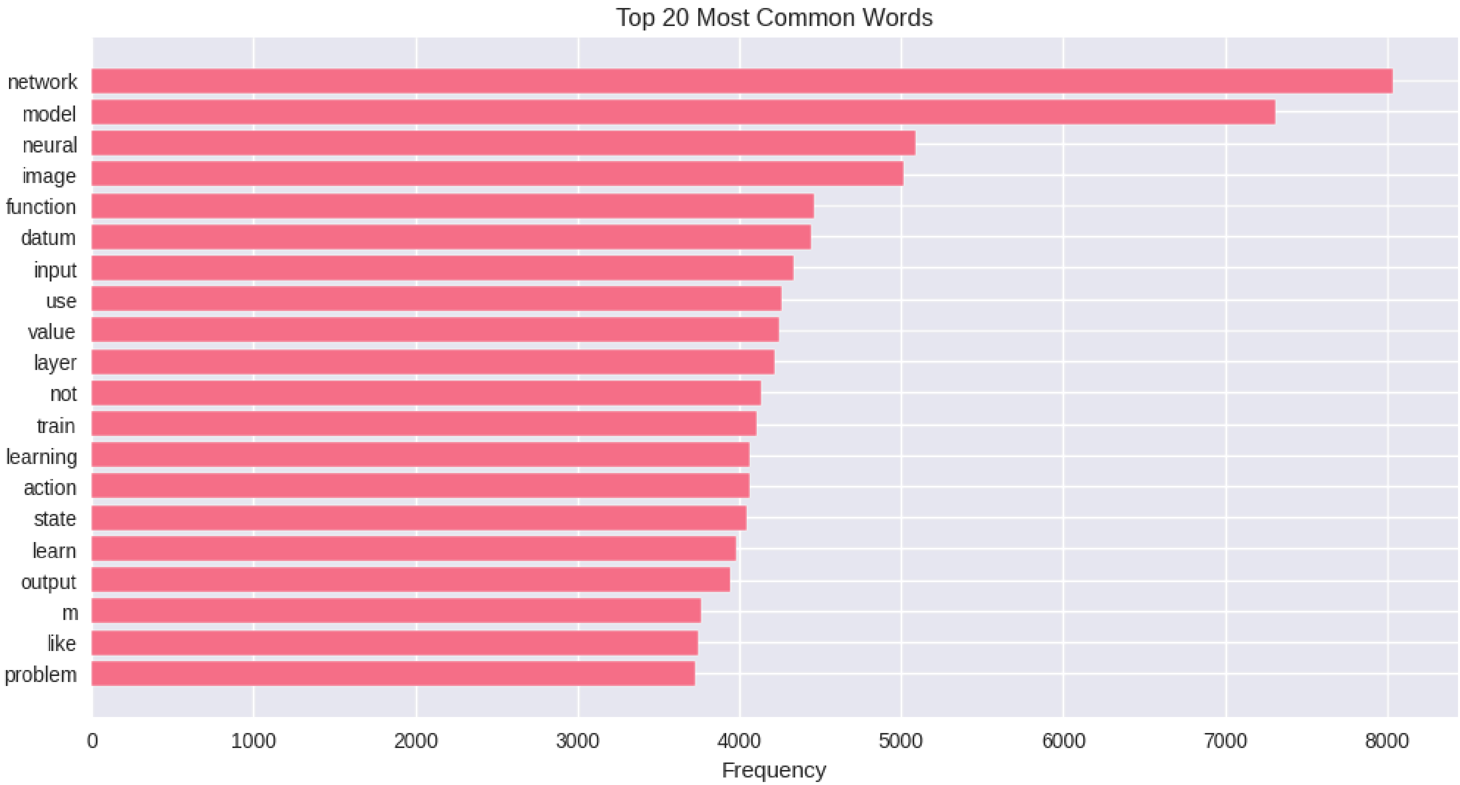
- Post length analysis



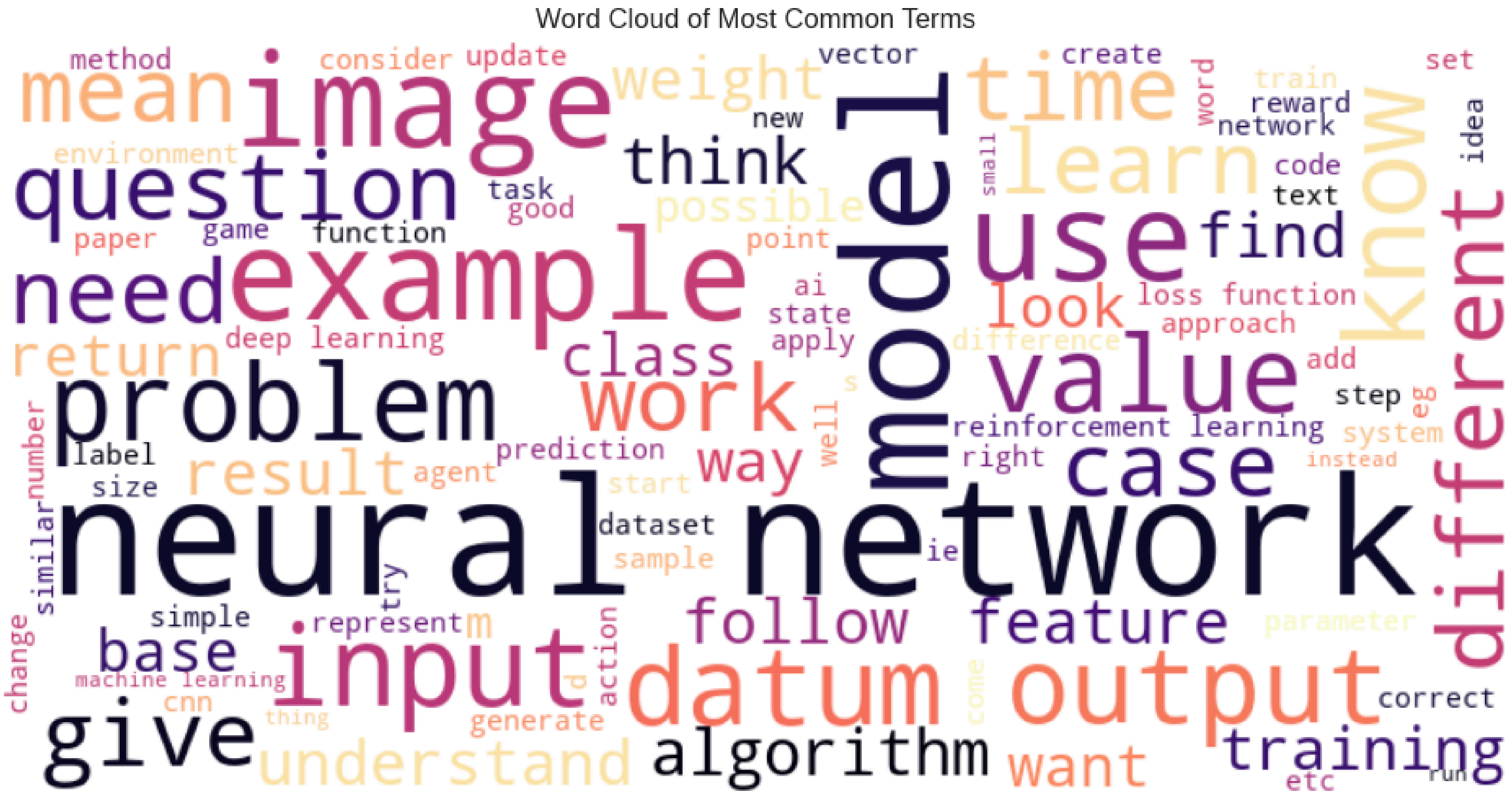
- Cosine-similarity as a measure.

Exploratory data analysis

Most frequent words



Word cloud


Word Cloud of Most Common Terms
neural network
model
example
image
question
problem
mean
use
know
different
input
datum
output
value
need
time
learn
think
work
give
understand
algorithm
case
follow
feature
training
want
weight
possible
find
look
class
result
return
way
base
method
consider
update
vector
create
set
train
word
reward
network
idea
new
small
code
text
environment
task
good
paper
game
function
point
ai
state
apply
loss function
approach
difference
deep learning
add
step
eg
reinforcement learning
system
prediction
right
instead
label
number
agent
start
size
well
dataset
sample
ie
similar
try
simple
represent
parameter
change
machine learning
cnn
thing
generate
action
come
correct
etc
run

Max features & Zipf's law

📊 Lemmatization > stemming

```python
def preprocess_text(text):
    # nlp e gotova f-ja koja
    doc = nlp(text)
    # ekstrahiraj
    processed_tokens = []
    for token in doc:
        if not token.is_stop and token.is_alpha: #ako ne e stopword i e so bukvi
            processed_tokens.append(token.lemma_)
    # spoj gi tokenite vo edinstven string za ponatamu tekstot da se vektorizira
    return ' '.join(processed_tokens)

# na Content kolonata ja primenuvame preprocess procedurata
df_raw['LemmatizedContent'] = df_raw['Content'].apply(preprocess_text)

# kreirame i corpus za ponatamosna analiza.
corpus = df_raw['LemmatizedContent']
```
✓ [24] 5m 40s

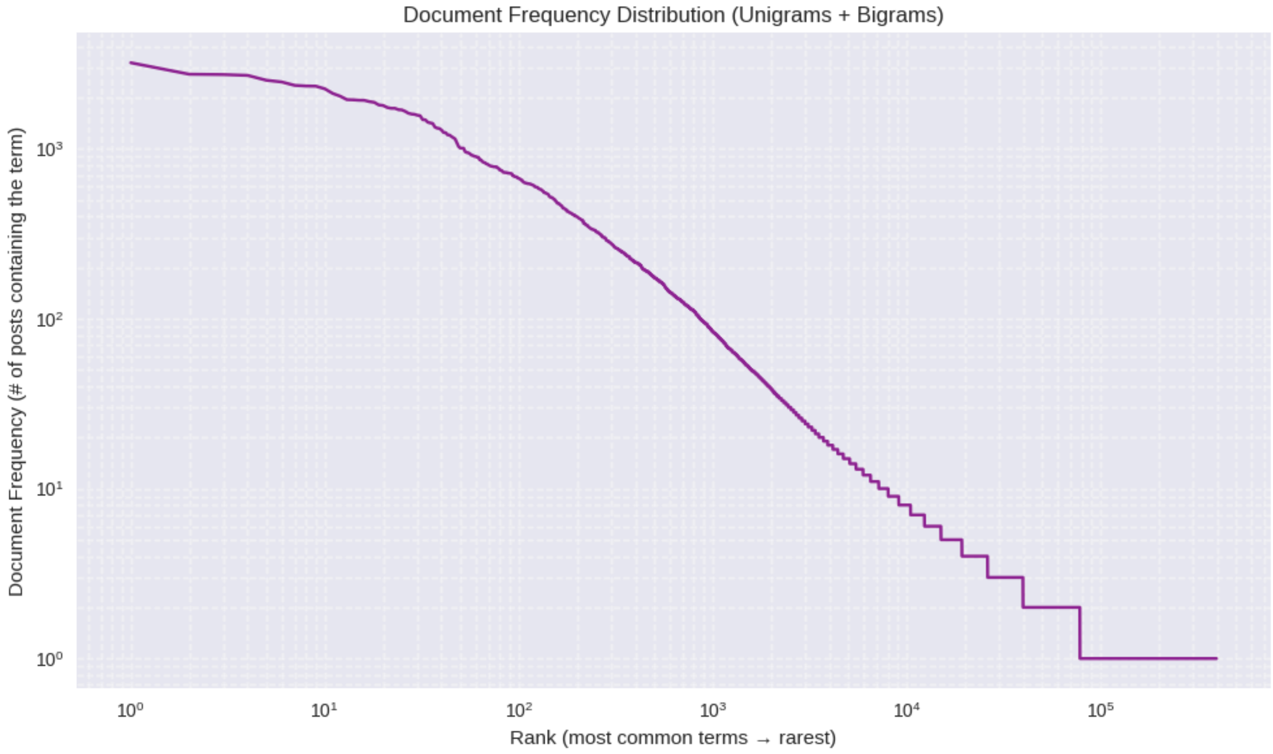
```python
corpus
```
✓ [25] < 10 ms

1-10 Length: 8981, dtype: object

	LemmatizedContent
0	backprop backprop mean backprop term basically backpropagati…
1	noise affect generalization increase noise datum help improv…
2	find optimal number neuron layer write algorithm know neuron n

Max features & Zipf's law

- 📊 We opt for 2000 features (unigrams+bigrams)



Base loss

- Predict no tags

For the hamming loss, we can calculate a base dummy classifier hamming loss to compare to.

```python
df_tag_counts['cum_sum'] = df_tag_counts['Count'].cumsum()

NUM_TAGS = 10
total_preds = df_raw.shape[0] * NUM_TAGS # rows * tags_per_row

positives = df_tag_counts.iloc[NUM_TAGS-1,df_tag_counts.columns.get_loc('cum_sum')]

base_loss = positives/total_preds

print(f"Base Hamming loss: {base_loss}")
```

✓ [34] 13ms

```
Base Hamming loss: 0.14913706714174368
```

- base_loss = 0.15

Onto our models

Vectorization and different feature vectors

📊 Bag of words representation

- CountVectorizer()
- StandardScaler()

📊 TF-IDF representation

- TfidfVectorizer()

📊 SVD representation

- on TF-IDF
- Dimensionality reduction technique, better for sparse matrices
- RobustScaler() - dividing by IQR preserves outliers which sometimes contain info

Models - Binary relevance method

📊 Binary relevance method - assumes independence of labels.

	f1_macro	f1_micro	avg_recall	avg_prec	hamming	subset_acc
NB_BoW	0.3968	0.5288	0.3308	0.5452	0.1149	0.2955
NB_TFIDF	0.3326	0.5114	0.2721	0.5905	0.1087	0.3200
NB_SVD	0.4221	0.4626	0.5566	0.3521	0.2026	0.1124
RF_BoW	0.4016	0.5906	0.3417	0.7510	0.0960	0.3756
RF_TFIDF	0.3978	0.5866	0.3367	0.7224	0.0959	0.3806
RF_SVD	0.2002	0.3723	0.1499	0.5395	0.1176	0.2270
KNN_BoW	0.2776	0.3936	0.2130	0.4960	0.1391	0.2176
KNN_TFIDF	0.3878	0.5008	0.3283	0.5432	0.1243	0.3116
KNN_SVD	0.1957	0.3217	0.1709	0.4149	0.1544	0.2020

Models - classifier chain method

- Keep NB_SVD, drop other SVD's
- Binary relevance assumes independence of labels, which may be faulty if co-dependence exists $\Rightarrow \varphi$ coefficient heatmap

Models - classifier chain method

	f1_macro	f1_micro	avg_recall	avg_prec	hamming	subset_acc
NB_BoW	0.4170	0.5448	0.3605	0.5348	0.1147	0.3038
NB_TFIDF	0.3547	0.5331	0.3009	0.5882	0.1087	0.3383
NB_SVD	0.3797	0.4149	0.5539	0.3932	0.2304	0.0484
RF_BoW	0.4091	0.5987	0.3478	0.7388	0.0948	0.3845
RF_TFIDF	0.3934	0.5868	0.3373	0.6973	0.0963	0.3784
KNN_BoW	0.2797	0.4140	0.2380	0.4675	0.1452	0.2732
KNN_TFIDF	0.3849	0.5116	0.3456	0.5036	0.1275	0.3400

📊 No significant improvements - slightly lesser hamming loss, sacrificing 1-2% precision

Models - χ^2 feature selection on top of binary relevance

- Instead of Vectorizer(max_features=500), employ feature selection from top 2000 features with χ^2 criterion on validation subset of training set.
- 2×2 contingency tables word(present/absent) vs. tag(present/absent) - $\chi^2 \upuparrows$ relevance & will generate 500 features that differ slightly and perform *slightly* better

	f1_macro	f1_micro	avg_recall	avg_prec	hamming	subset_acc
NB_BoW	0.3588	0.5011	0.2873	0.5504	0.1126	0.2844
NB_TFIDF	0.3080	0.4817	0.2415	0.5725	0.1087	0.2866
NB_SVD	0.4682	0.5103	0.5786	0.4038	0.1718	0.1664
RF_BoW	0.4448	0.6127	0.3860	0.6251	0.0959	0.3990
RF_TFIDF	0.4333	0.6068	0.3766	0.6411	0.0958	0.3918
KNN_BoW	0.3940	0.5092	0.3262	0.5445	0.1220	0.3072
KNN_TFIDF	0.4084	0.5222	0.3502	0.5489	0.1208	0.3261

- Sacrifices precision - slightly improves subset accuracy - most rigid metric

Use-case scenario (Routing our own sample post)

The struggle was real

3. Use-case scenario

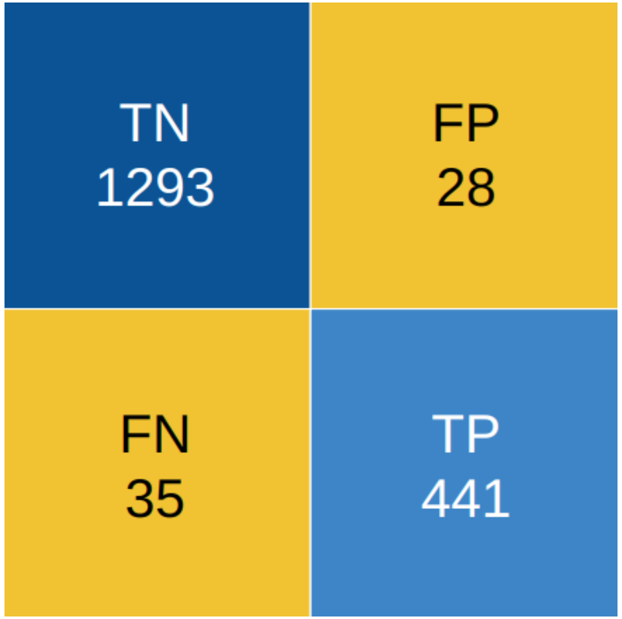
```python
example_post = "My model is performing poorly. I want to train it better but training it makes me so tired. Perhaps deep learning models could perform better for such tasks."

example_vec = bow_vectorizer.transform([example_post])

rf_bow = OneVsRestClassifier(RandomForestClassifier(n_estimators=100, random_state=42, n_jobs=-1))
X_train, X_test, Y_train, Y_test = train_test_split(X_bow.drop(columns=['UserId']), Y, test_size=0.2, random_state=42)
rf_bow.fit(X_train, Y_train)

predicted_tags_binary = rf_bow.predict(example_vec)

predicted_tags_array = predicted_tags_binary.toarray()[0] if hasattr(predicted_tags_binary, "toarray") else predicted_tags_binary[0]

predicted_tags = [label_names[i] for i, val in enumerate(predicted_tags_array) if val == 1]

print("Predicted tags: ", predicted_tags)
```

✓ [121] 57s 10ms

```
Predicted tags:  ['deep-learning']
```

A pleasant surprise

- The models actually predict the presence of the 'train' tag really well (to the right: RF-TFIDF)
- Also has good predictions for 'computer-vision' and some other tags
- Disbalance and inability to oversample or stratify make it underperform.



Concluding remarks (why classical NLP techniques fall short)

🧠 **No Context**
Bag-of-Words & TF-IDF ignore word order and meaning.

📐 **Sparse Features High-dimensional vectors** = hard to generalize, especially with imbalanced data.

🔍 **Surface-Level Semantics**
Can't handle polysemy, rare tags, or deeper understanding of questions.

⚠️ **Dimensionality Reduction ≠ Magic**
SVD improved recall, but sacrificed other metrics and loses important interpretability.

✅ **Works as a baseline**
But struggles to capture the complexity of real multilabel problems

Link to project

Link to GitHub repository: https://github.com/andWeirdFishes/Automated-Tag-Classification-on-AI-Stack-Exchange-Using-Classical-NLP-Techniques

Any questions?