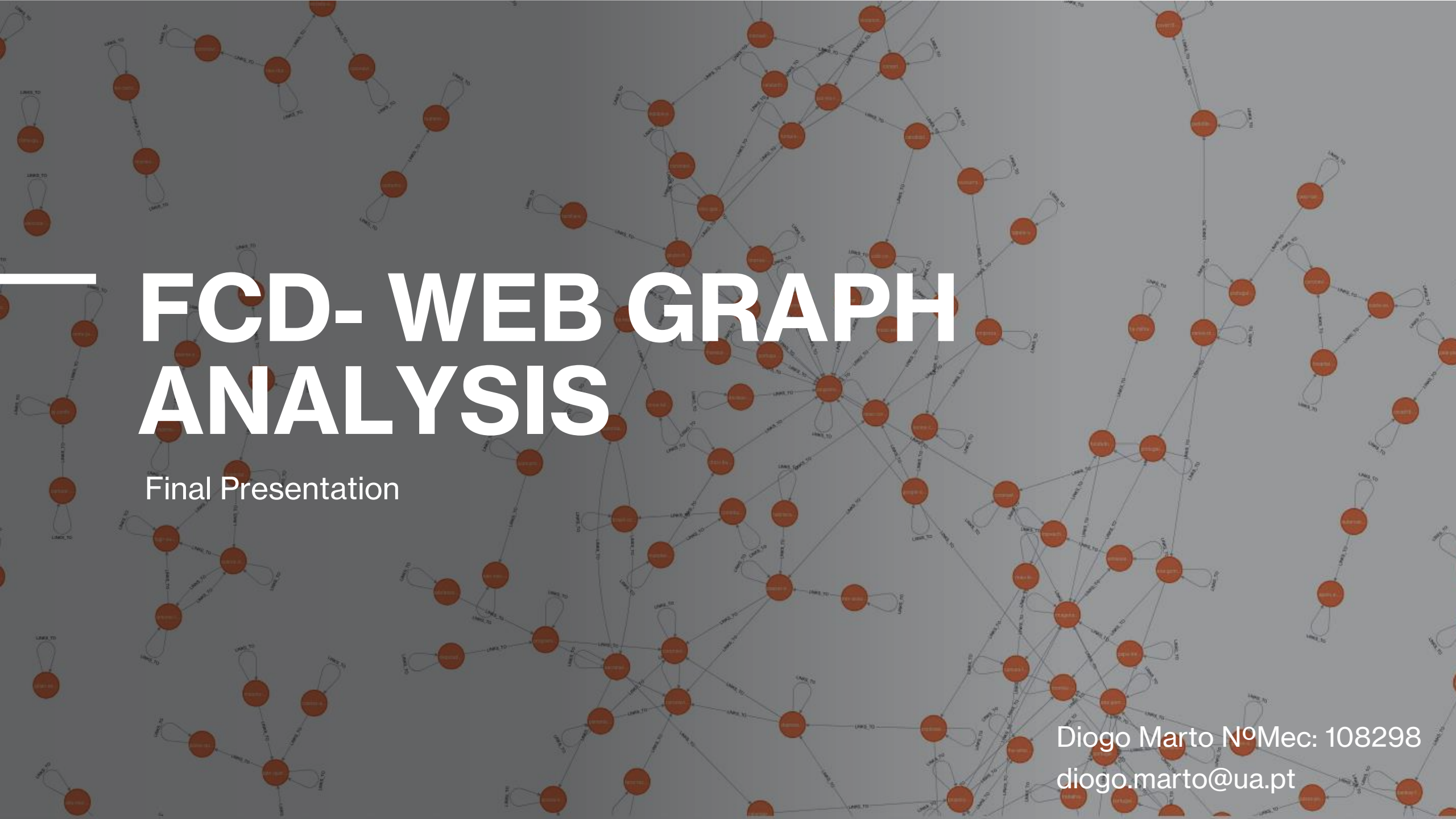


FCD- WEB GRAPH ANALYSIS

Final Presentation

Diogo Marto NºMec: 108298
diogo.marto@ua.pt

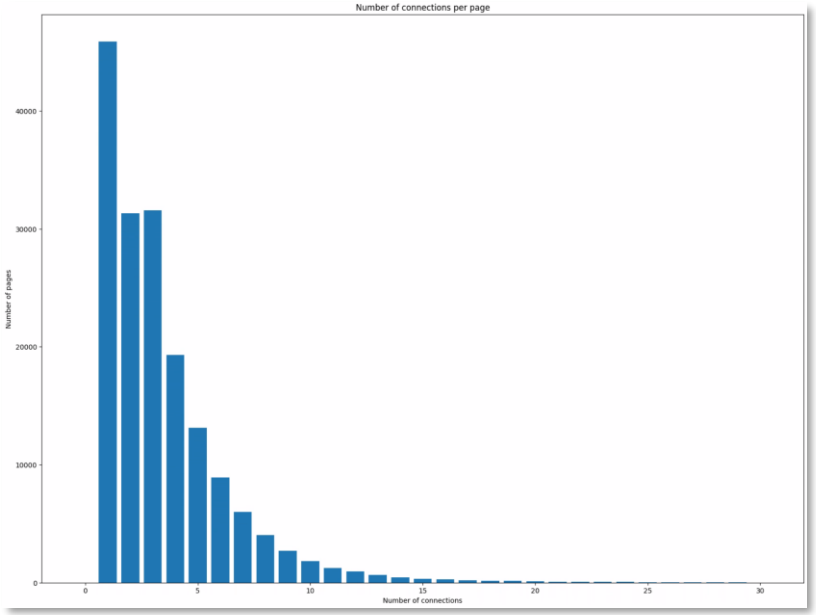
Index:

Previous improvements

- Moved files to another PC to prevent frying my laptop
- Fix missing connections caused by inconsistent arquive links

  `/noFrame/replay/` vs
  `https://arquivo.pt/noFrame/replay/`

- Fix problem with parsing category and converting to lower
- Speed up Neo4J insertion by 100x
- Better error handling when querying arquivo
- Sligthly change request to arquived page to reduce errors




```
url = f"https://arquivo.pt/noFrame/replay/{timestamp}/{url}"
 url = f"https://arquivo.pt/noFrame/replay/{timestamp}id_/{url}"
```

Text extraction

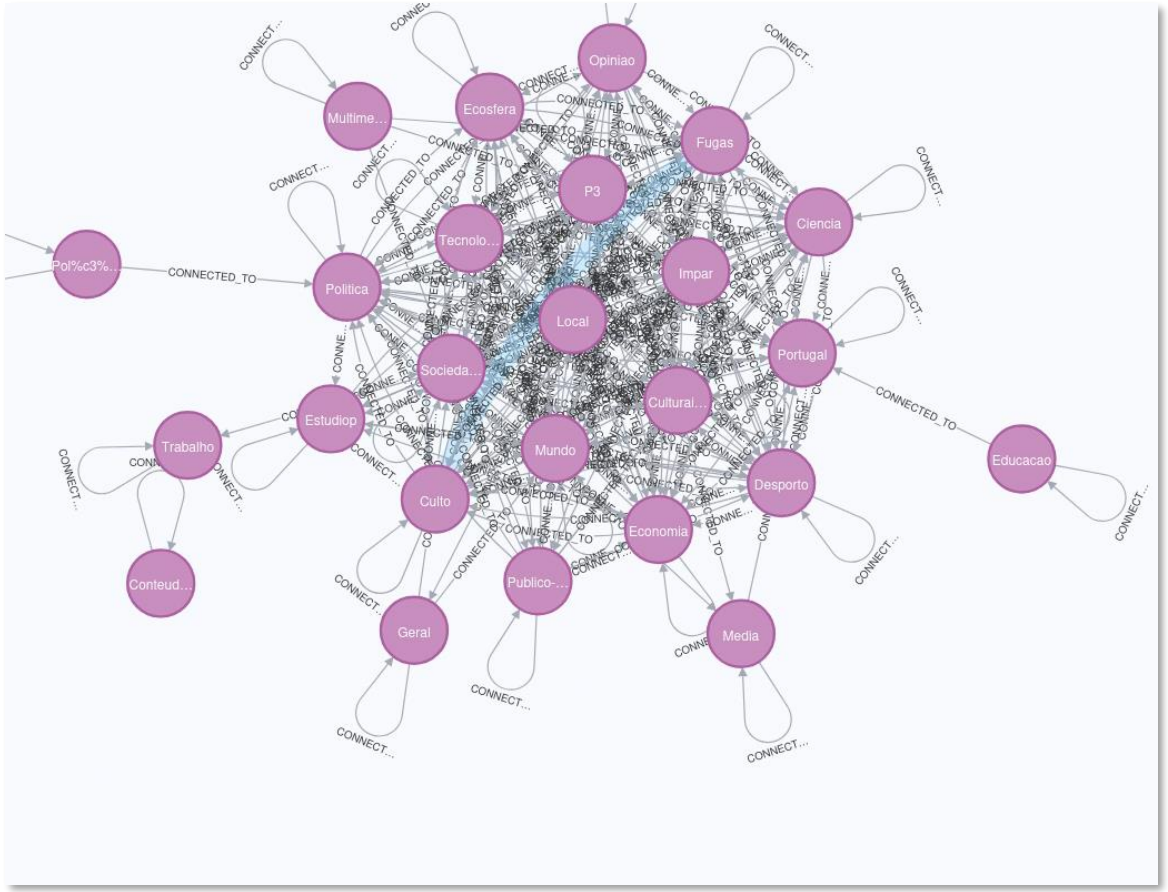
- BeautifullSoup4 to extract news body and title
- Using spacy and pt-sm-news-core
- Remove stop words and punctuation
- Lemmatization

Many news don’t have a lot of text since aruived pages have “Sign-in to read more”

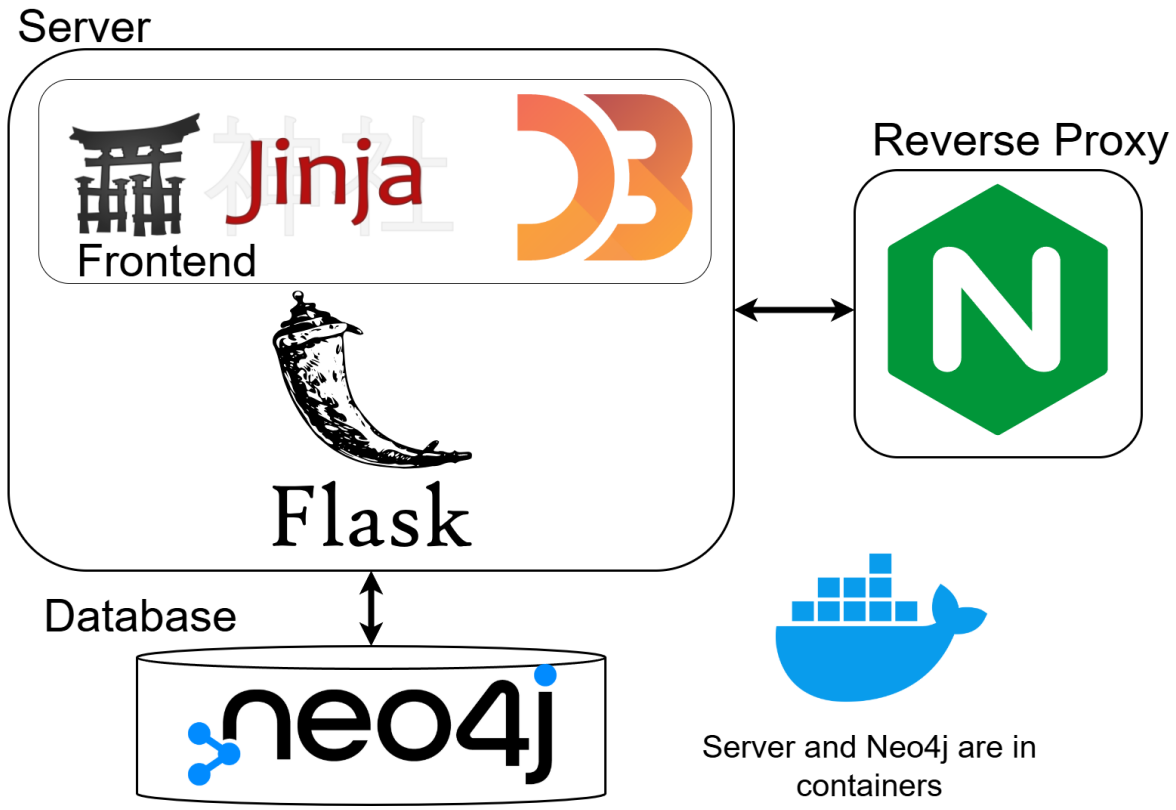
```python
def extract_text(content:str) -> str:
    soup = bs.BeautifulSoup(content, 'html.parser')
    # get text from only elements inside a div with the class "stor
    text = soup.find("h1", class_="story__headline").get_text()
    text = soup.find("div", class_="story__body").get_text() + text
    # replace newlines with spaces and multiple spaces with single
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'\d+', 'NUM', text)
    text = re.sub(r'[\uD800-\uDFFF]', '', text)
    return text

nlp = sp.load("pt_core_news_sm")
Codeium: Refactor | Explain | Generate Docstring | ×
def clean_text(text:str) -> str:
    # lemmatize and remove stop words and punct
    doc = nlp(text)
    text = " ".join(
        [token.lemma_ for token in doc
         if not token.is_stop and not token.is_punct])
    return text
```

Website

- Provide a way to visualize the graph and parts of the part in user firendly manner.
- Visualize the information pertaining to each node including the web page.
- Provide a way to extract graphs such as time series data and histograms
- Provide a way to visualize aggregate graph by category.

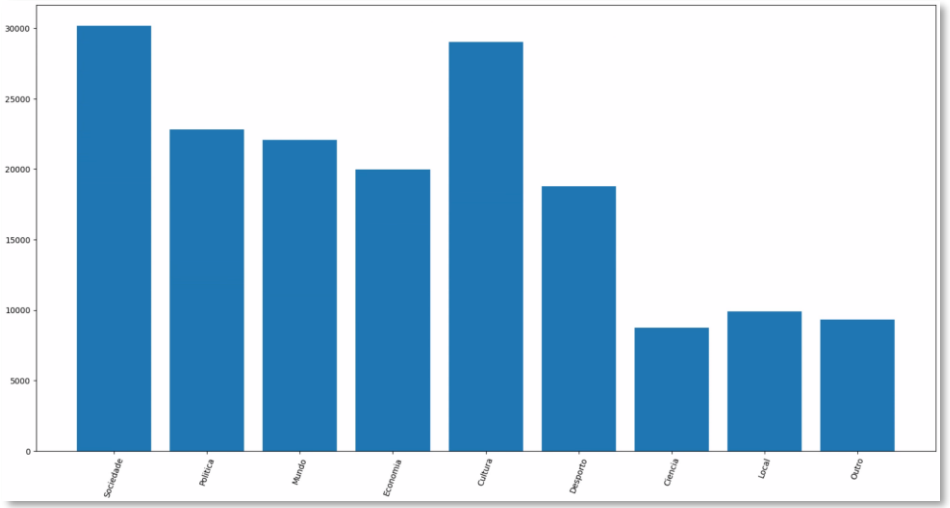


Website - Technology

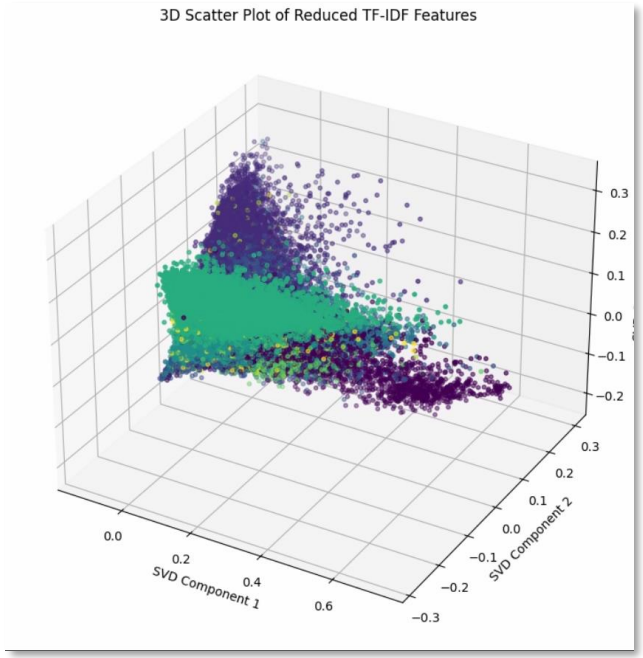


Website - Demo

Model – Feature extraction



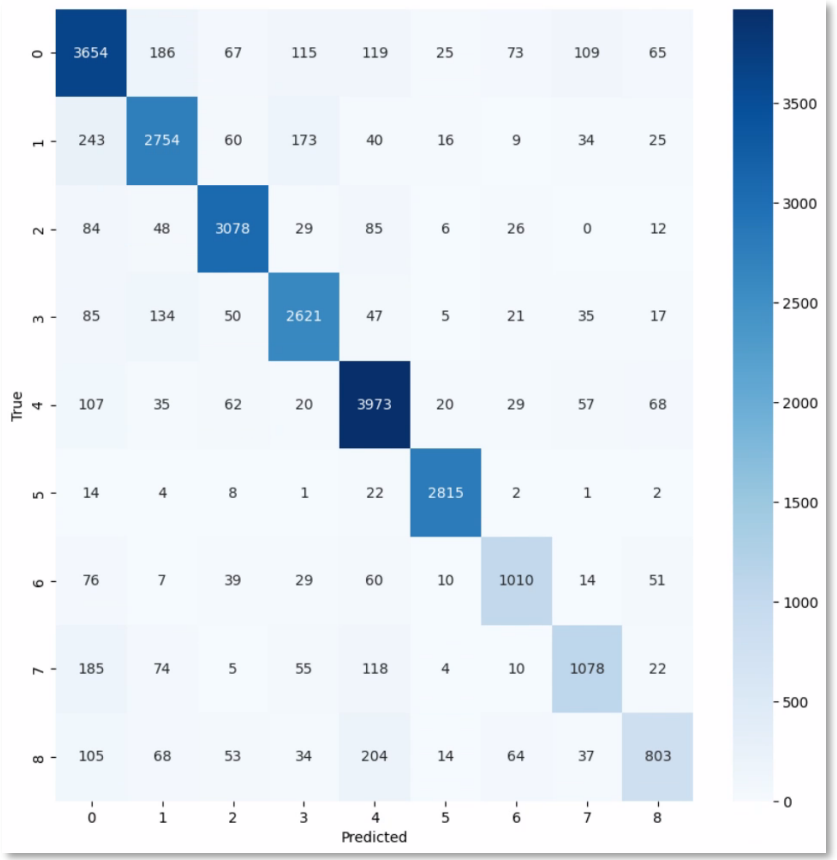
- Use tf-idf on our dataset
- Our labels are the categories from web scrapping but map to a reduced set
- Class distribution is not uniform
- PCA on tf-idf matrix shows that data is more or else clustered
- Train-Test split of 0.15%.



Model – Logistic Regression

- Test Accuracy: 0.85 and train Accuracy: 0.91
- Recall: 0.85 and Precision: 0.85 and F1-score:0.85

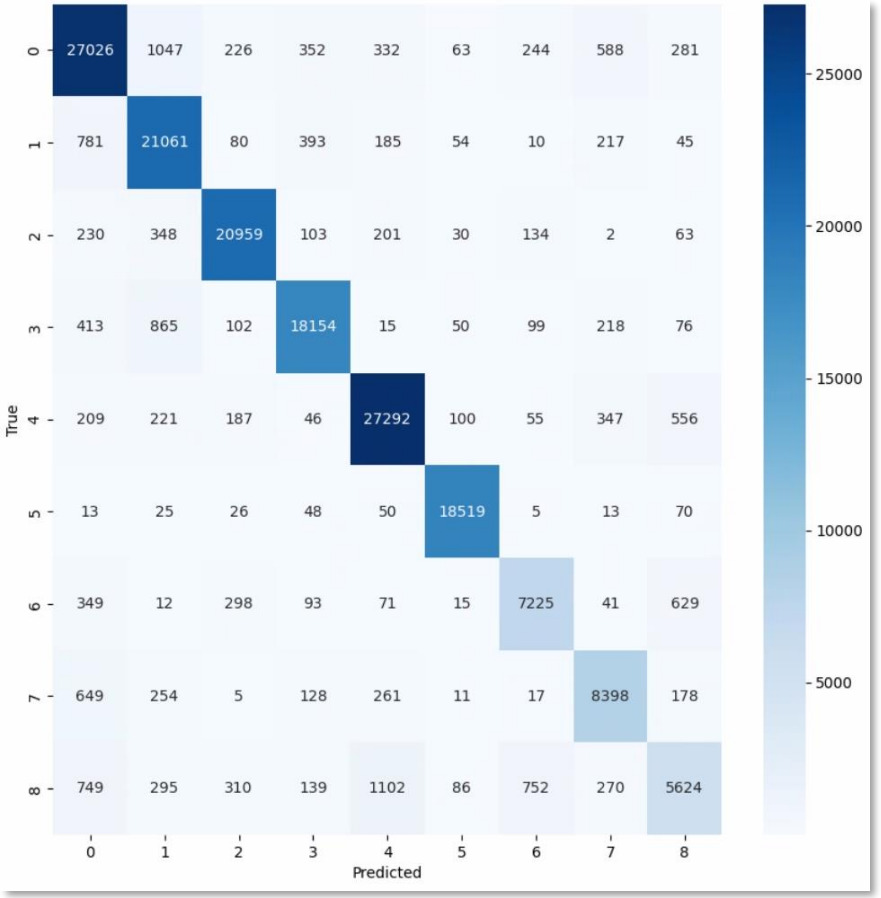
```
LogisticRegression
LogisticRegression(C=5, max_iter=5000, multi_class='ovr', random_state=42)
```



Model - LSTM

- Input raw text data not tf-idf
- Acc: 0.90 and Recall: 0.90 and Precision: 0.90 and F1-Score: 0.90
- More expensive to run takes around 1 min to classify the whole dataset

```
model = Sequential()
model.add(Embedding(vocab_size, embed_size, input_length=maxlen))
model.add(Bidirectional(LSTM(32, return_sequences=True)))
model.add(LSTM(32, return_sequences=True))
model.add(GlobalMaxPooling1D())
model.add(Dense(16, activation='relu' , kernel_regularizer='l2'))
model.add(Dense(16, activation='relu' , kernel_regularizer='l2'))
model.add(Dense(len(label_names), activation='softmax'))
model.summary()
```



Failed idea

- Use a large model to zero-shot classify the news based on a set of options
- Classify only a small part of the dataset with zero-shot then use that has train data to build a simpler model and classify rest of documents
- The model I used was giving poor results for zero-shot maybe because the news were in Portuguese

```
def multiple_choice(dataset):
    global classifier
    if classifier is None:
        print("GPU Device Name:", torch.cuda.get_device_name(0) if torch.cuda.is_a
        classifier = pipeline(
            "zero-shot-classification",
            model="facebook/bart-large-mnli",
            device=0 if torch.cuda.is_available() else -1  # Use GPU if available
        )
    prompt = lambda title, text , cat: f"Try to label the category of this news ar
    prompts = [
        prompt(title,text_,cat) for _ ,title, text_ , cat in dataset
    ]
    results = classifier(prompts, OPTIONS)
    return [(dataset[i][0],results[i]["labels"][0]) for i in range(len(results))]
```

Future improvements

- Better UX for website
- Employ a mechanics to build a classifier based on a set provided by user and then show on website
- CI/CD pipeline
- Use other sites besides publico.pt

Conclusion

- The main objectives of the work were accomplished and overall, I am happy with the result.
- I would like to have more time to polish certain areas of the work and generate more visualizations.
- I also learned a lot of things on this project and actually had fun making it.

THANK YOU