Dauphine | PSL
UNIVERSITÉ PARIS

Master 2 Intelligence Artificielle, Systèmes, Données en apprentissage

Projet NLP

Mohamed Aziz AYARI & Zakaria YBEGGAZENE

26 Mai 2023

Table des Matières

Introduction

Dans le cadre de notre projet NLP, nous avons développé un système de classification automatique des articles par thèmes en utilisant les techniques de traitement du langage naturel et d'apprentissage automatique. Ce genre de système peut être utilisé pour le filtrage d'information, la recommandation personnalisée, la veille stratégique etc.

Nous avons développé le système de bout en bout en suivant les étapes clés suivantes :

- **Crawling & et Scraping :** Tout d'abord, nous procédons au scraping, à partir des sites web de différents médias et journaux réputés, des articles de presse de thèmes assez variés comme le sport, la politique, la science, l'économie etc. Cette étape nous permet de construire un corpus de données.

- **Annotation :** La deuxième étape consiste à réaliser l'annotation du corpus : il s'agit d'attribuer des étiquettes (labels) aux articles en fonction de leur thème. Nous avons exploré plusieurs techniques d'annotation tel que l'annotation manuelle, semi-automatique et automatique pour créer le dataset.

- **Apprentissage automatique :** Une fois que notre corpus est annoté, nous procédons à l'étape d'entraînement du modèle de classification multi-classe. Nous avons choisi d'utiliser le modèle BERT (Bidirectional Encoder Representations from Transformers) pour cette tache de classification et nous avons évalué ses performances sur un nouveau jeu d'articles.

Crawling & Scrapping

Dans cette section dédiée au crawling et au scraping, nous détaillons le processus d'extraction que nous avons mis en place pour collecter les articles à partir de plusieurs sources.

1 Beautiful Soup

Dans cette partie, on présentera le processus de scraping en utilisant la bibliothèque Beautiful Soup et requests. Tout d'abord la librairie requests nous permet de récupérer le contenu d'une page sous forme de code HTML, une fois le contenu est récupéré, la librairie Beautiful Soup nous permet d'analyser et parcourir la structure HTML des pages web à travers différentes commandes telles que $get - text()$ et $find()$. Elle permet également d'avoir accès aux différents éléments de la page en les identifiant à travers les balises et

leurs attributs.
Nous devons donc identifier les balises, les classes et les ID des éléments qui nous intéressent dans le code HTML de la page web.

```
[11] url = "https://www.dailymail.co.uk"

[35] request_dailymail = requests.get(url)
     page = request_dailymail.content

[36] soup1 = BeautifulSoup(page, 'html5lib')

[37] page_news = soup1.find_all('h2', class_='linkro-darkred')
```

FIGURE 1 – Scraping du site Daily Mail

La figure 2 montre le code Python que nous avons utilisé afin de "crawler" les URL des différents articles sur la page principale du site du Daily Mail et de "scraper" le contenu des pages web correspondantes qui servira comme corpus. Tout cela en utilisant les librairies requests et Beautiful Soup.

```
for n in np.arange(0, number_of_articles):

    # Récupération des liens des articles
    link = url + page_news[n].find('a')['href']
    # Extraction des titres
    title = page_news[n].find('a').get_text()
    # Extraction du contenu de l'article sous forme de bloc
    article = requests.get(link)
    article_content = article.content
    soup_article = BeautifulSoup(article_content, 'html5lib')
    body = soup_article.find_all('p', class_='mol-para-with-font')
    # Fusionner le contenu de l'article en un seul texte
    list_paragraphs = []
    for p in np.arange(0, len(body)):
        paragraph = body[p].get_text()
        list_paragraphs.append(paragraph)
        final_article = " ".join(list_paragraphs)
    # nettoyage du texte pour enlever les caractères spéciaux lors du scrapping
    title = re.sub("\n", "", title)
    final_article = re.sub("\\xa0", "", final_article)
    # Create a dictionary for the article
    article_data = {'Title': title, 'Text': final_article,'Link':link}
    # Append the article to the list
    articles.append(article_data)
    # Create a DataFrame from the list of articles
```

FIGURE 2 – Crawling et Scraping

Enfin, nous chargeons les données récupérées dans un DataFrame pandas contenant sur chaque ligne le titre d'un article, son URL et son contenu comme on peut le voir sur la figure 3.

	Title	Text	Link
0	Asteroid City explosion! Tom Hanks i...	Tom Hanks was notably absent from this morning...	https://www.dailymail.co.uk/tvshowbiz/article-...
1	Hammered! Texas Attorney General claims Speake...	Texas Attorney General Ken Paxton has accused ...	https://www.dailymail.co.uk/news/article-12119...
2	Does space hold key to treating CANC...	Scientists believe the key to treating cancer ...	https://www.dailymail.co.uk/sciencetech/articl...
3	Target PULLS Pride products includin...	Target has pulled some of the most controversi...	https://www.dailymail.co.uk/news/article-12119...
4	Can't stick to a weight-loss plan? T...	You've heard of eating smaller meals, cutting ...	https://www.dailymail.co.uk/health/article-121...

FIGURE 3 – DataFrame pandas résultant

Notre objectif est de récupérer un maximum d'articles de différentes sources pour avoir un corpus de qualité pour la suite de notre projet. Néanmoins, scraper différents sites web

devient rapidement très fastidieux car il faut inspecter les codes HTML de tous les sites pour identifier pour chaque site les éléments à récupérer. C'est la raison pour laquelle nous avons fait recours à la librairie **Newspaper3k** qui effectue le crawling et le scraping de façon automatisée à partir d'une liste d'URLs de sites.

2 Newspaper3k

Newspaper permet de faire du web scraping et fournit des fonctionnalités permettant d'extraire des articles de presse à partir de divers sites en fournissant les urls. La bibliothèque crée un objet à partir d'un site web fournissant les informations présente sur le site telles que les catégories, les articles, les auteurs et la description. Elle offre également une configuration pour gérer d'autres paramètres tels que la langue. Une fois notre objet source est crée, on peut extraire les articles les plus récents de ce site.

Les fonctionnalités de Newspaper permettent d'accéder aux URLs des articles, leurs auteurs ainsi que leurs contenus. Cette librairie garde les articles récupérés en cache pour éviter de scrapper deux fois le même article. Newspaper se base sur l'objet **Article** qui fournit plusieurs méthodes pour le téléchargement **Download()**, l'analyse **Parse()** qui permet d'accéder à l'auteur, au titre et au contenu et également des fonctionnalités de traitement du langage naturel (NLP) sur le contenu de l'article **NLP()** pour générer un résumé ou extraire des entités nommées.
Après le nettoyage des données récupérées, nous avons obtenu **5166** articles de presse (en Anglais) de différentes tailles.

```python
def scrape_articles(liste_urls):
    scraped_urls = set()  # Set to store the scraped article URLs
    articles = []  # List to store the scraped articles
    for base_url in base_urls:
        paper = newspaper.build(base_url, memoize_articles=True)
        for article in paper.articles:
            # Check if the article URL has already been scraped
            #The problem of scraped 2 times the same article is
            #also fixed by the parameter  memoize_articles=True
            if article.url in scraped_urls:
                continue
            # Add the article URL to the set
            scraped_urls.add(article.url)
            # Download and parse the article
            try:
                article.download()
                article.parse()

            except newspaper.article.ArticleException:
                continue
            # Extract article information
            title = article.title
            text = article.text
            # Create a dictionary for the article
            article_data = {'Title': title, 'Text': text,'Link':article.url}
            # Append the article to the list
            articles.append(article_data)
    # Create a DataFrame from the list of articles
    df = pd.DataFrame(articles)
    return df
```

FIGURE 4 – Scrapping avec Newspaper3k

3 Difficultés

Nous avons également été confrontés à divers problèmes lors de cette tache de scraping, tels que les mesures de protection anti-scraping des sites ou des limites de requêtes possibles. On a essayé de contourner ces difficultés en scrapant des sites qui autorisent cette procédure et également avec l'ajout des sleep après un certains nombre de requêtes pour éviter d'être bloqué par le site.

Annotation

La phase d'annotation est une étape très importante dans la construction d'un bon modèle de classification. Le modèle va apprendre à classifier en se basant sur l'annotation réalisée dans cette partie, ce qui influe énormément sur la qualité des résultats de notre modèle. Il existe plusieurs techniques pour annoter un corpus de données. Dans cette partie, on détaillera deux approche d'annotation le **Keyword mapping** et **l'apprentissage non supervisé**.

4 Keyword mapping

Cette approche se base sur des règles d'annotation. Les règles prédéfinies consistent à un mapping entre les thèmes et leurs champs lexical associé pour baliser le texte. Les règles peuvent également être basées sur des expressions régulières et des listes de mots.

```python
def annotate_article(row):
    theme_keywords = {
        'Politics': ['politics', 'political', 'government', 'policy', 'election', 'democracy', 'governance'],
        'Business and Finance': ['business', 'finance', 'economy', 'market', 'stock', 'investment', 'financial'],
        'Sports': ['sport', 'athletics', 'sportsmanship', 'competition', 'athlete', 'team', 'tournament'],
        'Science and Technology': ['science', 'technology', 'research', 'discovery', 'innovation', 'experiment', 'scientific'],
        'Entertainment and Celebrities': ['entertainment', 'celebrity', 'fame', 'star', 'film', 'music', 'celebrities'],
        'Health and Wellness': ['health', 'wellness', 'medical', 'well-being', 'disease', 'nutrition', 'fitness'],
        'Environment and Climate': ['environment', 'climate', 'weather', 'ecology', 'sustainability', 'carbon', 'green'],
        'Education': ['education', 'school', 'learning', 'students', 'teachers', 'university', 'knowledge'],
        'Travel and Tourism': ['travel', 'tourism', 'destination', 'vacation', 'adventure', 'explore', 'hotel'],
        'Art and Culture': ['art', 'culture', 'painting', 'music', 'theater', 'literature', 'exhibition'],
        'Fashion and Lifestyle': ['fashion', 'lifestyle', 'style', 'trends', 'clothing', 'design', 'beauty'],
        'Food and Cooking': ['food', 'cooking', 'recipe', 'restaurant', 'nutrition', 'ingredients', 'culinary'],
        'History and Heritage': ['history', 'heritage', 'past', 'historical', 'museum', 'monument', 'archaeology'],
        'Automotive and Transportation': ['automotive', 'transport', 'car', 'vehicle', 'road', 'driver', 'traffic']
    }

    article_lower = row['Text'].lower()
    article_tokens = nltk.word_tokenize(article_lower)

    stop_words = set(stopwords.words('english'))
    article_keywords = [word for word in article_tokens if word not in stop_words]

    for theme, keywords in theme_keywords.items():
        if any(keyword in article_keywords for keyword in keywords):
            return theme

    return 'Other'
```

FIGURE 5 – Keyword mapping annotation

5 Apprentissage non supervisé

Dans cette approche, nous avons utilisé le modèle **DistilRoBERTa**[1] (pré-entrainé sur du texte en Anglais et plus rapide à l'entraînement et à l'inférence que RoBERTa avec des performances presque similaire). L'idée est de faire du "Topic Modeling" en suivante ces étapes :

- Calculer les "embedding" de nos articles avec DistilRoBERTa.
- Les vecteurs d'embedding obtenus sont de dimension 768. On leur applique l'algorithme **UMAP**[2] (Uniform Manifold Approximation and Projection) pour réduire leur dimensionalité tout en gardant le maximum d'information sur les structures locales qu'on retrouve en grande dimension.
- L'algorithme **HDBSCAN**[3] (Hierarchical Density-Based Spatial Clustering of Applications with Noise) permettant de faire du clustering a démontré une belle compatibilité avec les vecteurs réduits par UMAP. Nous l'utilisons alors afin de détecter les différentes communautés formées par nos vecteurs dans l'espace réduit. On considère chaque communauté comme un thème pour l'ensemble des articles qui le composent.
- Afin d'avoir une meilleure compréhension du contenu de chaque groupe d'articles détecté, nous concaténons tous les articles qui représentent un thème donné et nous appliquons l'algorithme **TF-IDF**[4] (Term Frequency - Inverse Document Frequency) permettant de trouver les mots (ou tokens) les plus importants par rapport au document concaténé fourni en se basant sur la fréquence de chaque mot dans le document.
- Nous regardons enfin les mots les plus importants pour chaque cluster et nous donnons manuellement (ou en utilisant ChatGPT) un thème général pour l'ensemble des articles le composant.

Nous avons dû optimiser les paramètres des algorithmes comme UMAP et HDBSCAN pour trouver les meilleurs combinaisons qui permettent de trouver des cluster intéressant (on ne veut pas avoir 50 cluster). Et nous avons au final choisi de réduire la taille des vecteurs jusqu'à dimension 32, d'utiliser la "cosine similarity" comme métrique pour calculer les distances entre les vecteurs et d'utiliser les 15 plus proches voisins de chaque vecteur comme paramètres dans UMAP. Pour HDBSCAN, nous fixons la taille minimale d'un cluster à 32 et nous utilisons la norme euclidienne pour calculer la distance entre les vecteurs.

La figure6 montre une projection en 2 dimensions où nous avons obtenu 17 clusters :

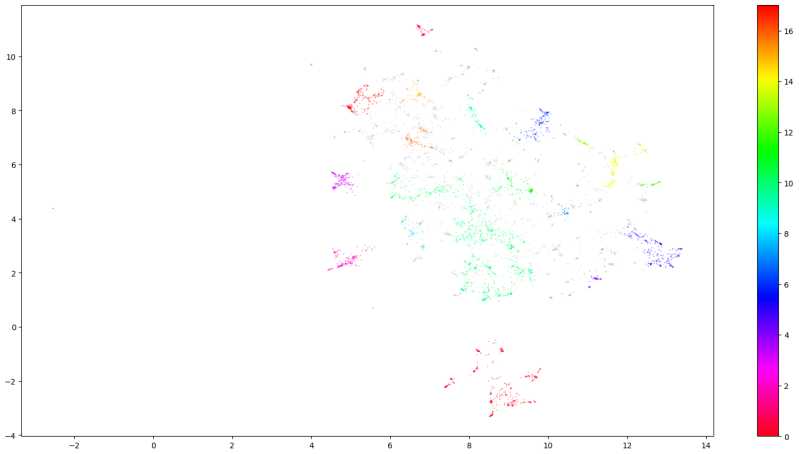

FIGURE 6 – Projection des cluster d'articles en 2D

Par exemple, nous pouvons voir que dans l'un des clusters, les mots les plus importants retrouvés avec TF-IDF sont les suivants : "zelda", "nintendo", "game", "kingdom", "codes", "characters", "mario" etc. Notre algorithme a pu donc regrouper avec succès les articles qui parlent du "gaming".

Apprentissage automatique

6 Entraînement et évaluation

Dans cette partie d'apprentissage, on a effectué au début quelques pré-traitements pour le texte qui s'adaptent avec ce modèle qu'on utilise pour l'entraînement. Nous avons utilisé le modèle BERT[5] pour effectuer notre classification multi-classes en utilisant les librairies de HuggingFace[6].

Après avoir entraîné le modèle, nous avons évalué sa performance à classifier le contexte d'un article à l'aide de quelques métriques, telles que l'accuracy, le recall, la précision et le F1 score. Dans cette évaluation notre métrique de référence est le F1 score car notre jeu de données présente un déséquilibre entre les différents thèmes présents dans les données. Le score F1 prend en considération à la fois la précision et le rappel du modèle, ce qui la rend adapté pour évaluer les performances dans les classes minorantes. Au bout de 3 époques (epoch), le modèle obtenu présente de très bonnes performances avec un F1 score de 0.81 7.

Epoch	Training Loss	Validation Loss	Accuracy	Recall	Precision	F1	Roc Auc
1	0.710300	0.955185	0.759690	0.759690	0.769508	0.749267	0.965846
2	0.381200	0.817441	0.813953	0.813953	0.827302	0.807068	0.972171
3	0.186600	0.925580	0.817829	0.817829	0.828667	0.812676	0.960988

FIGURE 7 – Métriques d'évaluation du classifieur sur le jeu de validation

7 Inférence

Dans la phase d'inférence, nous avons utilisé le modèle de classification des thèmes entraîné pour effectuer des prédictions sur de nouvelles données. Le modèle a montré une très bonne performance.
L'exemple suivant montre une classification réussi par le modèle :
Nous avons donné au modèle ce gros titre issue d'un article sur medium : *"I Asked Leading Covid Scientists — Off the Record — About the Virus's Origins and the 'Lab Leak'... Here's what they told me."* et le modèle a bien classifié ce titre dans le cluster contenant les mots importants suivants 8 :

```
[('sleep', 0.011461576405120416),
 ('covid', 0.011122103094328716),
 ('health', 0.010310192611306772),
 ('disease', 0.0071651007335370885),
 ('patients', 0.007098409555863594),
 ('virus', 0.006794551888643533),
 ('weight', 0.006550636170998724),
 ('drugs', 0.006028174100553724),
 ('symptoms', 0.005740529292799722),
 ('drug', 0.005660415647254097),
 ('vaccines', 0.0056566160565988365),
 ('hearing', 0.005550581047427769),
 ('apnea', 0.005540473481109756),
 ('vaccine', 0.005463078813393765),
 ('vitamin', 0.005433138206407731),
 ('cdc', 0.005064039220830946),
 ('treatment', 0.004901014045518031),
 ('loss', 0.0048707190884980465),
 ('risk', 0.004758677596571365),
 ('study', 0.004577120168315401)]
```

FIGURE 8 – Mots importants dans le cluster de "Health & Wellness"

D'autres exemples de classification peuvent être retrouvés sur le notebook Jupyter accompagnant ce rapport.

Conclusion

En conclusion, ce projet a été une occasion pour travailler sur un projet de traitement du langage naturel de bout en bout. Tout au long du projet, nous avons acquis des connaissances dans différents concepts, y compris le crawling, le scraping des sites web, et les modèles d'apprentissage automatique en NLP. Nous avons exploré et mis en œuvre les différentes techniques dans chacune des phases de ce projet ce qui nous a permis d'acquérir une compréhension approfondie des concepts clés de NLP.

Références

[1] Victor Sanh et al. "DistilBERT, a distilled version of BERT : smaller, faster, cheaper and lighter". In : *ArXiv* abs/1910.01108 (2019).

[2] Leland McInnes, John Healy et James Melville. *UMAP : Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv : 1802.03426 [stat.ML].

[3] Ricardo J. G. B. Campello, Davoud Moulavi et Joerg Sander. "Density-Based Clustering Based on Hierarchical Density Estimates". In : *Advances in Knowledge Discovery and Data Mining*. Sous la dir. de Jian Pei et al. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013, p. 160-172. isbn : 978-3-642-37456-2.

[4] "TF–IDF". In : *Encyclopedia of Machine Learning*. Sous la dir. de Claude Sammut et Geoffrey I. Webb. Boston, MA : Springer US, 2010, p. 986-987. isbn : 978-0-387-30164-8. doi : 10.1007/978-0-387-30164-8_832. url : https://doi.org/10.1007/978-0-387-30164-8_832.

[5] Jacob Devlin et al. "BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding". In : *CoRR* abs/1810.04805 (2018). arXiv : 1810.04805. url : http://arxiv.org/abs/1810.04805.

[6] Hugging Face. *Bert Model : . (French)*. 2019. url : https://huggingface.co/docs/transformers/tasks/sequence_classification.