TD 2 BONUS

Youna Froger, Alice Gyde, Mathis Quais

13 Novembre 2024

1 Exercice du TP

1.1 Explication du code

```python
def read_corpus(file_path):
    with open(file_path, "r") as infile:
        content = infile.read()
        content = content.lower()
        content = re.sub(r"[^\w\s']",' ',content)
        content = re.sub(r'\s+',' ',content)
    return content

def split_corpus(content, lang):
    return nltk.tokenize.word_tokenize(content, language=lang)

def content_to_dict(splitted_content):
    content_dict = {}
    for word in splitted_content:
        content_dict[word] = content_dict.get(word, 0) + 1
    return content_dict

def content_to_list(content_dict):
    return sorted(content_dict.items(), key=lambda x: x[1], reverse=True)
```

Les premières fonctions sont reprises des précédents TD et servent de prétraitement aux textes.

```python
def compute_p_at_20(most_frequent_words, stopwords_list):
    top_20_frequent = set(most_frequent_words[:20])
    top_20_stopwords = set(stopwords_list[:20])
    common_words = top_20_frequent.intersection(top_20_stopwords)
    return len(common_words) / 20
```

Cette fonction permet l'identification de la langue d'un texte. Elle compare les 20 mots les plus fréquents du texte avec la liste des stopwords (mots vides) de différentes langues. Plus le résultat du P@20 d'une liste de stopwords comparé à la liste des mots les plus fréquents est élevé, plus il est probable que la langue cherchée corresponde.

```python
def identify_language_for_folder(folder_path):
    expected_language = os.path.basename(folder_path)
    print(f"Processing folder: {folder_path} (expected language: {expected_language})")

    files = glob.glob(f"{folder_path}/*.txt")
    languages = ['english', 'french', 'spanish', 'greek', 'finnish', 'dutch']

    stopwords_count = {lang: 0 for lang in languages}

    for file_path in files:
        print(f"\nFile: {file_path}")

        content = read_corpus(file_path)
        splitted_content = split_corpus(content, expected_language)
        content_dict = content_to_dict(splitted_content)
        most_frequent_words = [word for word, _ in content_to_list(content_dict)]

        best_p_at_20 = 0
        best_stopwords_language = None

        for stopwords_lang in languages:
            stopwords_list = stopwords.words(stopwords_lang)
            p_at_20 = compute_p_at_20(most_frequent_words, stopwords_list)
            print(f"P@20 for stopwords in {stopwords_lang}: {p_at_20}")

            if p_at_20 > best_p_at_20:
                best_p_at_20 = p_at_20
                best_stopwords_language = stopwords_lang

        if best_stopwords_language:
            print(f"Best P@20 score for file {file_path} is with stopwords '{best_stopwords_language}': P@20 = {best_p_at_20}")
            stopwords_count[best_stopwords_language] += 1
        else:
            print(f"No stopword list had a significant P@20 score for file {file_path}")

    print("\nStopwords language count across all files in the folder:")
    for stopwords_lang, count in stopwords_count.items():
        print(f"{stopwords_lang}: {count}")

identify_language_for_folder('data/dutch')
identify_language_for_folder('data/english')
identify_language_for_folder('data/finnish')
identify_language_for_folder('data/french')
identify_language_for_folder('data/greek')
identify_language_for_folder('data/spanish')
```

Cette fonction reprend les précédentes et l'applique sur un dossier avec plusieurs fichiers textes. La sortie de la fonction est un print de combien de fichiers correspondent le mieux à chaque langue pour l'ensemble du dossier en entrée.

1.2 Matrice de confusion

Actual \ Predicted	english	french	spanish	greek	finnish	dutch	null
english	60	0	0	0	0	0	0
french	0	58	0	0	0	0	2
spanish	0	0	60	0	0	0	0
greek	0	1	0	0	0	1	58
finnish	0	1	0	0	41	0	18
dutch	0	0	0	0	0	60	0
null	0	0	0	0	0	0	0

English :

Précision :

$$\frac{60}{60+0} = 1$$

Rappel :

$$\frac{60}{60+0} = 1$$

F1 :

$$\frac{2*1*1}{1+1} = 1$$

French :

Précision :

$$\frac{58}{58+0} = 1$$

Rappel :

$$\frac{58}{58+2} = 0.96$$

F1 :

$$\frac{2*1*0.96}{1+0.96} = 0.98$$

Spanish :

Précision :

$$\frac{60}{60+0} = 1$$

Rappel :

$$\frac{60}{60+0} = 1$$

F1 :

$$\frac{2*1*1}{1+1} = 1$$

Greek :

Précision :

$$\frac{0}{0+2} = 0$$

Rappel :

$$\frac{0}{0+58} = 0$$

F1 :

$$\frac{2*0*0}{0+0} = 0$$

Finnish :

Précision :

$$\frac{41}{41+1} = 0,98$$

Rappel :

$$\frac{41}{41+18} = 0,69$$

F1 :

$$\frac{2*0,98*0,69}{0,98+0,69} = 0,81$$

Dutch :

Précision :

$$\frac{60}{60+0} = 1$$

Rappel :

$$\frac{60}{60+0} = 1$$

F1 :

$$\frac{2*1*1}{1+1} = 1$$

Accuracy :

$$\frac{60 + 58 + 60 + 41 + 60}{360} = 0,775$$

2 Conclusion

Pour conclure sur les résultats, la méthode est plutôt efficace pour l'anglais, le francais, l'espagnol et le néerlandais car elles ont les valeurs les plus hautes pour le score F1. Pour le finnois, la réussite est moins marquée, avec ici un score F1 de 0,81. Dans le cas du grecque, nous supposons une différence d'alphabet entre la liste de stop-words et les textes sur lesquelles nous avons travaillés car les résultats sont vides et ne pointent aucune langue. Au final, l'accuracy nous donne quand même l'indication d'une bonne efficacité du code avec une valeur 0,775.