ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

Εργασία 1

A. Tokens, Types, Zipf's Law

Δίνεται ένα αρχείο (wsj_untokenized.txt) που περιλαμβάνει κείμενα σύντομων ειδήσεων στα Αγγλικά από την εφημερίδα **Wall Street Journal**.

Εφαρμόστε tokenization στο σύνολο των κειμένων του αρχείου με βάση τις ακόλουθες μεθόδους:

- Χρήση του **nltk.word_tokenize()** που συμπεριλαμβάνεται στο NLTK[1].
- Χρήση του μοντέλου για την Αγγλική γλώσσα **en_core_web_sm** που συμπεριλαμβάνεται στο spaCy[2]. Το συγκεκριμένο μοντέλο παρέχει διάφορα επίπεδα ανάλυσης στο κείμενο εισόδου. Στα πλαίσια της εργασίας μας ενδιαφέρει μόνο ο τεμαχισμός του κειμένου σε tokens.
- Χρήση του **BertTokenizer** που συμπεριλαμβάνεται στο HuggingFace[3]. Να χρησιμοποιηθεί η έκδοση **bert-base-cased**[4]. Ο συγκεκριμένος tokenizer έχει προκαθορισμένο λεξιλόγιο και δημιουργεί subword tokens, παρόμοια με τον αλγόριθμο byte-pair encoding.

1. Συμπληρώστε τον παρακάτω πίνακα:

	NLTK	**spaCy**	**Bert**
#tokens			
#types			
TTR			
Hapax legomena			
Hapax dislegomena			

   Όπου το #tokens είναι το πλήθος των tokens που υπάρχουν στη συλλογή κειμένων, #types είναι το πλήθος των διαφορετικών tokens (types) που βρέθηκαν, TTR (type-token ratio) είναι ο λόγος types προς tokens, hapax legomena είναι το πλήθος των types που εμφανίζονται ακριβώς μία φορά και hapax dislegomena είναι το πλήθος των types που εμφανίζονται ακριβώς 2 φορές στη συλλογή.
   Σχολιάστε τα αποτελέσματα (Ποια μέθοδος tokenization δημιουργεί μεγαλύτερο λεξιλόγιο; Πώς κρίνετε το TTR των μεθόδων; Πώς συγκρίνονται τα hapax legomena/dislegomena με τις προβλέψεις του νόμου του Zipf;)

2. Επιλέξτε τυχαία μία πρόταση από τη συλλογή με τουλάχιστον 15 λέξεις και δείξτε την λίστα των tokens που παράγει η κάθε μέθοδος. Σχολιάστε τις διαφορές μεταξύ των παραγόμενων tokens.

3. Για την κάθε μέθοδο tokenization, αφού κατατάξετε τα types ανά πλήθος εμφανίσεων, εμφανίστε σε ένα κοινό πίνακα τα πιο συχνά types των οποίων το άθροισμα των εμφανίσεών τους στη συλλογή να αντιστοιχεί στο 30% των συνολικών tokens της συλλογής. Πόσα από τα

[1] https://www.nltk.org/
[2] https://spacy.io/
[3] https://huggingface.co/docs/transformers/main_classes/tokenizer
[4] https://huggingface.co/bert-base-cased

πιο συχνά types που βρέθηκαν υπάρχουν στη λίστα και των 3 μεθόδων tokenization που εξετάστηκαν;

4. Σύμφωνα με τον Νόμο του Zipf, το γινόμενο της πιθανότητας ενός type επί την θέση του στη λίστα των types (ταξινομημένη κατά φθίνουσα σειρά συχνότητας) είναι ίσο με μια σταθερά A. Βρείτε ποια τιμή της σταθεράς A (εξετάστε τις επιλογές: 0.1, 0.2, 0.3, …, 1.0) ταιριάζει καλύτερα με το συγκεκριμένο σύνολο κείμενων. Δημιουργήστε ένα διάγραμμα όπου ο άξονας x είναι η θέση ενός type σε φθίνουσα σειρά συχνότητας και ο άξονας y είναι η συχνότητα του type στην συλλογή. Και οι 2 άξονες να ακολουθούν την λογαριθμική κλίμακα. Δείξτε στο ίδιο διάγραμμα με διαφορετικά χρώματα (i) τις προβλέψεις του νόμου του Zipf (με την καλύτερη τιμή που βρέθηκε για τη σταθερά A) και (ii) τις πραγματικές μετρήσεις που έγιναν στα κείμενα.

5. Χρησιμοποιήστε το **ChatGPT**[5] για να δημιουργήσει τον κώδικα στο ερώτημα 4. Συγκρίνετε τα αποτελέσματα που προκύπτουν με την δική σας απάντηση, εντοπίστε τυχόν λάθη στον κώδικα του ChatGPT και διορθώστε τα. Αναφέρετε τα prompts που χρησιμοποιήσατε για τη δημιουργία του κώδικα και ποια λάθη εντοπίστηκαν.

B. N-gram Language Models

Σε αυτό το τμήμα της εργασίας θα χρησιμοποιήσουμε και πάλι κείμενα σύντομων ειδήσεων στα Αγγλικά από την εφημερίδα **Wall Street Journal**. Πιο συγκεκριμένα, στο NLTK υπάρχουν διαθέσιμα 199 αρχεία ειδήσεων τα οποία είναι επεξεργασμένα σε διάφορες μορφές. Εμείς θα χρησιμοποιήσουμε το επίπεδο ανάλυσης στο οποίο τα κείμενα έχουν ήδη χωριστεί σε tokens και προτάσεις. Παρακάτω φαίνεται ένα παράδειγμα σε Python για το πώς μπορείτε να δείτε το πρώτο κείμενο της συλλογής σε αυτήν την μορφή:

```
>>> from nltk.corpus import treebank
>>> files=treebank.fileids()
>>> len(files)
Out[1]: 199
>>> treebank.sents(files[0])
Out[2]: [['Pierre', 'Vinken', ',', '61', 'years', 'old', ',',
'will', 'join', 'the', 'board', 'as', 'a', 'nonexecutive',
'director', 'Nov.', '29', '.'], ['Mr.', 'Vinken', 'is', 'chairman',
'of', 'Elsevier', 'N.V.', ',', 'the', 'Dutch', 'publishing',
'group', '.']]
```

Θα πρέπει να δημιουργήσετε κώδικα python για την εκπαίδευση και αξιολόγηση γλωσσικών μοντέλων ν-γραμμάτων με εφαρμογή της μεθόδου add-k smoothing.

Τα πρώτα 170 αρχεία της συλλογής να χρησιμοποιηθούν για την εκπαίδευση των γλωσσικών μοντέλων και τα υπόλοιπα 29 αρχεία να χρησιμοποιηθούν για την αξιολόγηση του μοντέλου.

Στη φάση της εκπαίδευσης του μοντέλου να αντικαταστήσετε όλα τα tokens που εμφανίζονται στο σύνολο των κειμένων εκπαίδευσης λιγότερες από 3 φορές με το ειδικό token <UNK>. Τα υπόλοιπα

[5] https://chat.openai.com/

tokens θα συμπεριλαμβάνονται στο λεξιλόγιο L. Στη φάση της αξιολόγησης, όσα tokens στα κείμενα αξιολόγησης δεν συμπεριλαμβάνονται στο L θα πρέπει να αντικατασταθούν με <UNK>.

Τα n-grams θα πρέπει να εξάγονται στα πλαίσια της κάθε πρότασης (δηλ. να μην υπάρχουν n-grams που περιλαμβάνουν το τέλος μιας πρότασης και την αρχή της επόμενης). Προσθέστε δύο ειδικά tokens (<BOS>, <EOS>) που να υποδηλώνουν την αρχή και το τέλος της κάθε πρότασης. Π.χ., η πρώτη πρόταση από το πρώτο κείμενο της συλλογής που δίνεται παραπάνω:

```
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join',
'the', 'board', 'as', 'a', 'nonexecutive', 'director', 'Nov.', '29',
'.']
```

θα μετατραπεί σε:

```
['<BOS>', 'Pierre', 'Vinken', ',', '61', 'years', 'old', ',',
'will', 'join', 'the', 'board', 'as', 'a', 'nonexecutive',
'director', 'Nov.', '29', '.', '<EOS>']
```

και στην περίπτωση των bigrams θα εξαχθούν τα ακόλουθα:

```
('<BOS>', 'Pierre'),
('Pierre', 'Vinken'),
('Vinken', ','),
(',', '61'),
…
('Nov.', '29'),
('29', '.'),
('.', '<EOS>')
```

1. Συμπληρώστε τον παρακάτω πίνακα με το perplexity[6] των γλωσσικών μοντέλων της υλοποίησής σας:

Language model	Original text	Lowercase	Abstract digits
Bigrams (k = 1)			
Bigrams (k = 0.01)			
Trigrams (k = 1)			
Trigrams (k = 0.01)			

Όπου η τιμή του k αναφέρεται στη μέθοδο add-k smoothing, η στήλη Original text περιλαμβάνει τα μοντέλα που προέκυψαν από τα κείμενα της συλλογής χωρίς κάποια προ-επεξεργασία, η στήλη Lowercase περιλαμβάνει τα μοντέλα που προέκυψαν μετά την μετατροπή όλων των χαρακτήρων σε πεζούς και η τελευταία στήλη αναφέρεται στα μοντέλα που προέκυψαν μετά την αντικατάσταση όλων των ψηφίων με το σύμβολο # (για παράδειγμα, τα '33', '8.4%', και '$352.7' θα μετατραπούν στην αφηρημένη μορφή '##', '#.#%' και '$###.#' αντίστοιχα).

6

$$Perplexity = e^{-\frac{1}{N}\sum_i^N \ln p(g_i)}$$

όπου g_i είναι ένα bigram/trigram και N είναι το συνολικό πλήθος των bigrams/trigrams στα κείμενα αξιολόγησης.

Σχολιάστε τα αποτελέσματα που προκύπτουν (Ποια τάξη μοντέλου (bigrams/trigrams) φαίνεται να είναι πιο αποτελεσματική; Ποια τιμή του k είναι πιο κατάλληλη; Πώς επηρεάζει η διαδικασία lowercase τα αποτελέσματα; Ποια η επίδραση της χρήσης αφηρημένων αριθμών στην επίδοση των μοντέλων;)

2. Με βάση το κάθε μοντέλο ν-γραμμάτων που βασίστηκε στα αρχικά κείμενα (χωρίς καμία προ-επεξεργασία), δημιουργήστε 3 νέες προτάσεις. Οι νέες προτάσεις θα πρέπει να ξεκινούν με <BOS> (και πρώτες λέξεις της δικής σας επιλογής) και να τερματίζουν σε <EOS>. Η κάθε επόμενη λέξη να επιλέγεται τυχαία αναλογικά με την πιθανότητα του κάθε ν-γράμματος (όσο πιο μεγάλη η πιθανότητα του ν-γράμματος τόσο μεγαλύτερη η πιθανότητα να επιλεγεί η λέξη). Δεν θα πρέπει να περιλαμβάνονται tokens <UNK> στις παραγόμενες προτάσεις. Σχολιάστε την ποιότητα των παραγόμενων προτάσεων για το κάθε μοντέλο.

Χρήσιμα Links:

https://nlp.stanford.edu/IR-book/html/htmledition/zipfs-law-modeling-the-distribution-of-terms-1.html

https://www.nltk.org/book/ch03.html

https://web.stanford.edu/~jurafsky/slp3/3.pdf

https://devopedia.org/n-gram-model

Οδηγίες υποβολής:

Η εργασία είναι **ατομική**. Πρέπει να υποβάλλετε μία αναφορά με τις απαντήσεις στα ερωτήματα της εργασίας καθώς τον κώδικα που χρησιμοποιήσατε. Συστήνεται η χρήση **Jupyter Notebook**[7].

[7] https://jupyter.org/