ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Επεξεργασία Φωνής και Φυσικής Γλώσσας
Χειμερινό Εξάμηνο 2019-20

1η Εργαστηριακή Άσκηση: Μέρος πρώτο: Ορθογράφος

ΠΕΡΙΓΡΑΦΗ

Σκοπός αυτού του μέρους της 1ης εργαστηριακής άσκησης είναι η εξοικείωση με τη χρήση μηχανών πεπερασμένων καταστάσεων για επεξεργασία γλώσσας. Συγκεκριμένα θα χρησιμοποιήσουμε απλά γλωσσικά μοντέλα και απλούς μετασχηματισμούς για τη δημιουργία ενός ορθογράφου. Θα γίνει χρήση της βιβλιοθήκης openfst. Μπορείτε να βρείτε εδώ ένα script για την εγκατασταση
https://github.com/georgepar/python-lab/blob/master/install_openfst.sh
Το documentation βρίσκεται εδώ:
http://www.openfst.org/
μαζί με παραδείγματα
http://www.openfst.org/twiki/bin/view/FST/FstExamples

Επίσης θα γίνει μια εισαγωγή στο word2vec. Για αυτό το λόγο θα χρησιμοποιήσετε τη βιβλιοθήκη gensim.

Σας παρέχεται και ένα παράδειγμα δημιουργίας fst με συνδυασμό python και bash scripting:
https://gist.github.com/georgepar/4cc8793e270c3486bff94993a215fb2c
(Η συνάρτηση format_arc διαμορφώνει μια γραμμή από το αρχείο περιγραφής του fst και είναι προς υλοποίηση)

ΠΡΟΣΟΧΗ: Χρησιμοποιείστε την έκδοση 1.6.1 του Openfst. Επόμενες εκδόσεις έχουν σημαντικό bug στη σύνθεση FSTs.

ΕΚΤΕΛΕΣΗ

Βήμα 1: Κατασκευή corpus

Σε αυτό το βήμα θα δημιουργήσουμε ένα μετρίου μεγέθους corpus από διαθέσιμες πηγές στο διαδίκτυο.
α) Το project Gutenberg είναι μια πηγή για βιβλία που βρίσκονται στο public domain και μία καλή πηγή για γρήγορη συλλογή δεδομένων για κατασκευή γλωσσικών μοντέλων. Κατεβάστε ένα βιβλίο της επιλογής σας από το project gutenberg σε plain .txt μορφή. Ενδεικτικά προτείνουμε τα http://www.gutenberg.org/ebooks/1661 και http://www.gutenberg.org/ebooks/36.
β) Αυτά τα corpora θα χρησιμοποιηθούν για την εξαγωγή στατιστικών κατά την κατασκευή γλωσσικών μοντέλων. Μπορείτε επίσης να ενώσετε πολλά βιβλία για την κατασκευή ενός μεγαλύτερου corpus. Ποια είναι το πλεονεκτήματα αυτής της τεχνικής (εκτός από το μέγεθος των δεδομένων;) Αναφέρετε τουλάχιστον 2.

Βήμα 2: Προεπεξεργασία corpus
Αφού έχετε το .txt αρχείο του corpus πρέπει να το διαβάσετε με την κατάλληλη προεπεξεργασία
α) Γράψτε μια συνάρτηση identity_preprocess που διαβάζει ένα string και γυρνάει τον εαυτό του.
β) Γράψτε μια συνάρτηση Python η οποία δέχεται σαν όρισμα το path του αρχείου και μια συνάρτηση preprocess και διαβάζει το αρχείο γραμμή προς γραμμή σε μία λίστα, καλώντας την preprocess σε κάθε γραμμή. (Χρησιμοποιήστε την identity_preprocess του πρώτου ερωτήματος σαν default όρισμα για την preprocess).
γ) Γράψτε μια συνάρτηση tokenize η οποία δέχεται σαν όρισμα ένα string s και: α) καλεί την strip() και lower() πάνω στο s, β) αφαιρεί όλα τα σημεία στίξης / σύμβολα / αριθμούς, αφήνωντας μόνο αλφαριθμητικούς χαρακτήρες, γ) αντικαθιστά τα newlines με κένα, δ) κάνει split() τις λέξεις στα κενά. Το αποτέλεσμα είναι μια λίστα από lowercase λέξεις. Αυτό το βήμα λέγεται tokenization και εμείς υλοποιήσαμε μια απλή εκδοχή που αναγνωρίζει σαν tokens μόνο lowercase λέξεις.
δ) Μπορείτε να πειραματιστείτε με τους διαθέσιμους tokenizers στη βιβλιοθήκη nltk και να συγκρίνετε τα αποτελέσματα

Βήμα 3: Κατασκευή λεξικού και αλφαβήτου
Εδώ θα κατασκευάσουμε 2 λεξικά που θα χρησιμεύσουν στην κατασκευή των FSTs, ένα λεξικό για τα tokens (λέξεις) και ένα για τα σύμβολα (αλφάβητο).
α) Δημιουργήστε μια λίστα που περιέχει όλα τα μοναδικά tokens που βρίσκονται στο corpus, σύμφωνα με τον tokenizer του Βήματος 2
β) Δημιουργήστε μια λίστα που περιέχει το αλφάβητο του corpus (μοναδικούς χαρακτήρες).

Βήμα 4: Δημιουργία συμβόλων εισόδου/εξόδου
Για την κατασκευή των FSTs χρειάζονται 2 αρχεία που να αντιστοιχίζουν τα σύμβολα εισόδου (ή εξόδου) σε αριθμούς.
Γράψτε μια συνάρτηση Python που διαβάζει το αλφάβητο και αντιστοιχίζει κάθε χαρακτήρα σε ένα αύξοντα ακέραιο index. Το πρώτο σύμβολο με index 0 είναι το <epsilon> (ε). Το αποτέλεσμα πρέπει να γράφεται σε ένα αρχείο chars.syms με αυτή τη μορφή
http://www.openfst.org/twiki/pub/FST/FstExamples/ascii.syms
Σε όλα τα επόμενα βήματα θα χρησιμοποιήσουμε το chars.syms για τα σύμβολα εισόδου και εξόδου.

Βήμα 5: Κατασκευή μετατροπέων FST
Για τη δημιουργία του ορθογράφου θα χρησιμοποιήσουμε μετατροπείς βασισμένους στην απόσταση Levenshtein (https://en.wikipedia.org/wiki/Levenshtein_distance). Θα χρησιμοποιήσουμε 3 τύπους από edits: εισαγωγές χαρακτήρων, διαγραφές χαρακτήρων και αντικαταστάσεις χαρακτήρων. Κάθε ένα από αυτά τα edits χαρακτηρίζεται από ένα κόστος. Σε αυτό το στάδιο θα θεωρήσουμε ότι όλα τα πιθανά edits έχουν το ίδιο κόστος w=1.

α) Κατασκευάστε ένα μετατροπέα με μία κατάσταση που υλοποιεί την απόσταση Levenshtein αντιστοιχίζοντας: α) κάθε χαρακτήρα στον εαυτό του με βάρος 0 (no edit), β) κάθε χαρακτήρα στο <epsilon> με βάρος 1 (deletion), γ) το <epsilon> σε κάθε χαρακτήρα με βάρος 1 (insertion), δ) κάθε χαρακτήρα σε κάθε άλλο χαρακτήρα με βάρος 1. Τι κάνει αυτός ο μετατροπέας σε μια λέξη εισόδου αν πάρουμε το shortest path;
β) Αυτός είναι ένας αρκετά αφελής τρόπος για τον υπολογισμό των βαρών για κάθε edit. Αν είχατε στη διάθεσή σας ότι δεδομένα θέλατε πώς θα υπολογίζατε τα βάρη;

Βήμα 6: Κατασκευή αποδοχέα λεξικού
α) Κατασκευάστε έναν αποδοχέα με μία αρχική κατάσταση που αποδέχεται κάθε λέξη του λεξικού από το βήμα 3α. Τα βάρη όλων των ακμών είναι 0. Αυτό είναι ένας αποδοχέας ο οποίος απλά αποδέχεται μια λέξη αν ανήκει στο λεξικό.
β) Καλέστε τις *fstrmepsilon*, *fstdeterminize* και *fstminimize* για να βελτιστοποιήσετε το μοντέλο. Τι κάνουν αυτές οι συναρτήσεις;

Βήμα 7: Κατασκευή ορθογράφου
α) Συνθέστε τον Levenshtein transducer με τον αποδοχέα του ερωτήματος 6α παράγοντας τον min edit distance spell checker. Αυτός ο transducer διορθώνει τις λέξεις χωρίς να λαμβάνει υπόψιν του κάποια γλωσσική πληροφορία, με κριτήριο να κάνει τις ελάχιστες δυνατές μετατροπές στην λέξη εισόδου. Αναλύστε τη συμπεριφορά αυτού του μετατροπέα α) στην περίπτωση που τα edits είναι ισοβαρή, β) για διαφορετικά βάρη των edits.
Hint χρησιμοποιήστε την fstcompose
β) Ποιες είναι οι πιθανές προβλέψεις του min edit spell checker αν η είσοδος είναι η λέξη cit? (Hint: Μπορεί να χρειαστεί να καλέσετε την fstarcsort στις εξόδους του transducer ή/και στις εισόδους του acceptor)

Βήμα 8: Αξιολόγηση ορθογράφου
α) Κατεβάστε αυτό το σύνολο δεδομένων για το evaluation
https://raw.githubusercontent.com/georgepar/python-lab/master/spell_checker_test_set
β) Η βέλτιστη διόρθωση προβλέπεται με βάση τον αλγόριθμο ελαχίστων μονοπατιών στο γράφο του μετατροπέα του βήματος 7.
Χρησιμοποιείστε το μετατροπέα για να διορθώσετε κάποιες από τις λέξεις του test set. Σε αυτό το σημείο επιλέξτε 20 λέξεις στη τύχη και σχολιάστε το αποτέλεσμα.
Μπορείτε να χρησιμοποιήσετε σαν βάση αυτόν τον κώδικα
https://gist.github.com/georgepar/f6d14e6ba32b29be78b48dd8cf8e21fc

Βήμα 9: Εξαγωγή αναπαραστάσεων word2vec

Μπορείτε να βασιστείτε σε αυτόν τον κώδικα για τα επόμενα ερωτήματα https://gist.github.com/georgepar/7d1fd391f182024bca48983ad4bf12c2

α) Διαβάστε το κείμενο σε μια λίστα από tokenized προτάσεις με τον κώδικα του βήματος 2γ.
β) Χρησιμοποιείστε την κλάση Word2Vec του gensim για να εκπαιδεύσετε 100 διαστατα word2vec embeddings με βάση τις προτάσεις του βηματος 9α. Χρησιμοποιείστε window=5 και 1000 εποχές. (παραδείγματα: https://radimrehurek.com/gensim/models/word2vec.html)
γ) https://gist.github.com/Επιλέξτε 10 τυχαίες λέξεις από το λεξικό και βρείτε τις σημασιολογικά κοντινότερες τους.
Είναι τα αποτελέσματα τόσο ποιοτικά όσο περιμένατε; Βελτιώνονται αν αυξήσετε το μέγεθος του παραθύρου context / τον αριθμό εποχών; Γιατί;

ΠΑΡΑΔΟΤΕΑ

(1) Σύντομη αναφορά (σε .html ή .pdf) που θα περιγράφει τη διαδικασία που ακολουθήθηκε σε κάθε βήμα, καθώς και τα σχετικά αποτελέσματα.
(2) Κώδικας, ο οποίος περιέχει και τις εντολές του OpenFst (συνοδευόμενος από σύντομα σχόλια).

Συγκεντρώστε τα (1) και (2) σε ένα .zip αρχείο το οποίο πρέπει να αποσταλεί μέσω του mycourses.ntua.gr εντός της καθορισμένης προθεσμίας.

Προτείνουμε να κάνετε χρήση notebook στα οποία μπορείτε να γράψετε τόσο διακριτά τμήματα κώδικα όσο και τον απαραίτητο σχολιασμό. Μπορείτε στη συνέχεια να κάνετε export του notebook (.ipynb) τόσο σε .html όσο και σε .py μορφή και να μας αποστείλετε και τα τρία αυτά αρχεία.