**ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ**
**ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

**Επεξεργασία Φωνής και Φυσικής Γλώσσας**
**Χειμερινό Εξάμηνο 2014-15**

# 1η Εργαστηριακή Άσκηση: Εισαγωγή στις γλωσσικές αναπαραστάσεις

## 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://raw.githubusercontent.com/georgepar/python-lab/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 δεδοδένων;) 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) 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;

***Τα παρακάτω βήματα δεν αποτελούν μέρος της προπαρασκευής.***

**ΠΕΡΙΓΡΑΦΗ**
Σκοπός αυτού του μέρους της 1ης εργαστηριακής άσκησης είναι να γίνει μια εισαγωγή σε διαφορετικές γλωσσικές αναπαραστάσεις και τη χρήση τους για γλωσσικά tasks.
Στο πρώτο μέρος θα εμπλουτίσουμε τον ορθογράφο που φτιάξαμε στην προπαρασκευή με character level και word level unigram γλωσσικά μοντέλα.
Στο δεύτερο μέρος θα κάνουμε μια εισαγωγή στις λεξικές αναπαραστάσεις bag-of-words και word2vec και θα τις χρησιμοποιήσουμε σε ένα απλό πρόβλημα ταξινόμησης.

## ΜΕΡΟΣ 1
## Ορθογράφος

Βήμα 10: Εξαγωγή στατιστικών

Εδώ θα κατασκευάσουμε 2 πηγές στατιστικών για τα γλωσσικά μας μοντέλα, μια word / token level και μία character level.
α) Εξάγετε την πιθανότητα εμφάνισης κάθε token (λέξης) και αποθηκεύστε τη σε ένα λεξικό με key το token και value την πιθανότητα εμφάνισής του.
β) Εξάγετε την πιθανότητα εμφάνισης κάθε χαρακτήρα και αποθηκεύστε τη σε ένα λεξικό με key το χαρακτήρα και value την πιθανότητα εμφάνισής του. Αυτό είναι η βάση για ένα unigram μοντέλο.

Βήμα 11: 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.
α) Υπολογίστε τη μέση τιμή των βαρών του word level μοντέλου που κατασκευάσατε στο Βήμα 10α. Αυτό είναι το κόστος των edits w.
β) 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 w (deletion), γ) το <epsilon> σε κάθε χαρακτήρα με βάρος w (insertion), δ) κάθε χαρακτήρα σε κάθε άλλο χαρακτήρα με βάρος w. Αυτός είναι ο μετατροπέας Levenshtein για το word level μονέλο.
γ) Επαναλάβατε για το unigram γλωσσικό μοντέλο του Βήματος 10β
δ) 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 12: Κατασκευή γλωσσικών μοντέλων

α) Κατασκευάστε έναν αποδοχέα με μία αρχική κατάσταση που αποδέχεται κάθε λέξη του λεξικού από το βήμα 3α. Αυτή τη φορά χρησιμοποιήστε ως βάρη τον αρνητικό λογάριθμο της πιθανότητας εμφάνισης κάθε λέξης -logP(w)
Hint: (η πρώτη ακμή εχει βάρος -logP(w) και οι υπόλοιπες 0)
β) Καλέστε τις fstrmepsilon, fstdeterminize και fstminimize για να βελτιστοποιήσετε το μοντέλο.
γ) Επαναλάβετε για το unigram γλωσσικό μοντέλο

Βήμα 13: Κατασκευή ορθογράφων

α) Επαναλάβετε το Βήμα 7 για να κατασκευάσετε τον ορθογράφο με το word-level γλωσσικό μοντέλο και το word-level μετατροπέα
β) Επαναλάβετε το Βήμα 7 για να κατασκευάσετε τον ορθογράφο με το unigram γλωσσικό μοντέλο και το word-level μετατροπέα
γ) Τι κάνουν αυτοί οι ορθογράφοι; Δώστε παραδείγματα αμφίσημων διορθώσεων (πχ cit -> cat or cut)

Βήμα 14: Αξιολόγηση των ορθογράφων

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. Γράψτε τον κώδικα αξιολόγησης που υπολογίζει την ακρίβεια κάθε spell checker για τις λέξεις του test set. Μπορείτε να χρησιμοποιήσετε σαν βάση αυτόν τον κώδικα https://gist.github.com/georgepar/f6d14e6ba32b29be78b48dd8cf8e21fc
γ) Σχολιάστε τα αποτελέσματα

Βήμα 15: Extra Credit

α) Φτιάξτε και αξιολογήστε έναν ορθογράφο που θα έχει ως βάση ένα bi-gram γλωσσικό μοντέλο. (Τις πιθανότητες εμφάνισης κάποιας δυάδας γραμμάτων)

## ΜΕΡΟΣ 2
## Χρήση σημασιολογικών αναπαραστάσεων για ανάλυση συναισθήματος

Στο πρώτο μέρος της άσκησης ασχοληθήκαμε κυρίως με συντακτικά μοντέλα για την κατασκευή ενός ορθογράφου. Εδώ θα ασχοληθούμε με τη χρήση λεξικών αναπαραστάσεων για την κατασκευή ενός ταξινομητή συναισθήματος . Ως δεδομένα θα χρησιμοποιήσουμε σχόλια για ταινίες από την ιστοσελίδα IMDB και θα τα ταξινομήσουμε σε θετικά και αρνητικά ως προς το συναίσθημα.

Βήμα 16: Δεδομένα και προεπεξεργασία

α) Κατεβάστε τα δεδομένα από το παρακάτω λινκ http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz
β) Διαβάστε και προεπεξεργαστείτε τα δεδομένα. Σας δίνεται για διευκόλυνση ο κώδικας ανάγνωσης και κάποιες απλές συναρτήσεις προεπεξεργασίας. Μπορείτε αντί αυτών να χρησιμοποιήσετε αυτές που φτιάξατε στο Βήμα 2
https://gist.github.com/georgepar/eba00343e7ddc995898a7f075dcfc445

Βήμα 17: Κατασκευή BOW αναπαραστάσεων και ταξινόμηση

Η πιο βασική αναπαράσταση για μια πρόταση είναι η χρήση Bag of Words. Σε αυτή την αναπαράσταση μια λέξη κωδικοποιείται σαν ένα one hot encoding πάνω στο λεξιλόγιο και μια πρόταση σαν το άθροισμα αυτών των encodings. Για παράδειγμα στο λεξιλόγιο [cat, dog, eat] η αναπαράσταση της λέξης cat είναι [1, 0, 0], της λέξης dog [0, 1, 0] κοκ. Η αναπαράσταση της πρότασης dog eat dog είναι [0, 2, 1]. Επιπλέον μπορούμε να πάρουμε σταθμισμένο άθροισμα των

one hot word encodings για την αναπαράσταση μιας πρότασης με βάρη TF-IDF (https://en.wikipedia.org/wiki/Tf–idf).
α) Εξηγείστε διαισθητικά το ρόλο των βαρών TF-IDF στις bag-of-words αναπαραστάσεις
β) Χρησιμοποιείστε τον transformer CountVectorizer του sklearn για την εξαγωγή μη σταθμισμένων BOW αναπαραστάσεων
γ) Εκπαιδεύστε τον ταξινομητή LogisticRegression του sklearn για να ταξινομήσετε τα σχόλια σε θετικά και αρνητικά.
δ) Επαναλάβατε χρησιμοποιώντας τον TfidfVectorizer για την εξαγωγή TF-IDF αναπαραστάσεων. Συγκρίνετε τα αποτελέσματα

Βήμα 18: Χρήση Word2Vec αναπαραστάσεων για ταξινόμηση

Ένας άλλος τρόπος για να αναπαραστήσουμε λέξεις και προτάσεις είναι να κάνουμε χρήση προεκπαιδευμένων embeddings. Σε αυτό το βήμα θα εστιάσουμε στα word2vec embeddings. Αυτά τα embeddings προκύπτουν από ένα νευρωνικό δίκτυο με ένα layer το οποίο καλείται να προβλέψει μια λέξη με βάση το context της (παράθυρο 3-5 λέξεων γύρω από αυτή). Αυτό ονομάζεται CBOW μοντέλο. Εναλλακτικά το δίκτυο καλείται να προβλέψει το context με βάση τη λέξη (skip-gram μοντέλο).
Τα word2vec vectors είναι πυκνές (dense) αναπαραστάσεις σε λιγότερες διαστάσεις από τις BOW και κωδικοποιούν σημασιολογικά χαρακτηριστικά μιας λέξης με βάση την υπόθεση ότι λέξεις με παρόμοιο νόημα εμφανίζονται σε παρόμοια συγκείμενα (contexts).
Μια πρόταση μπορεί να αναπαρασταθεί ως ο μέσος όρος των w2v διανυσμάτων κάθε λέξης που περιέχει (Neural Bag of Words).

α) Υπολογίστε το ποσοστό out of vocabulary (OOV) words για τις αναπαραστάσεις που υπολογίσατε στο Βήμα 9.
Hint: Για τα παρακάτω ερωτήματα θεωρείστε την αναπαράσταση μιας OOV λέξης το μηδενικό διάνυσμα
β) Χρησιμοποιήστε αυτές τις αναπαραστάσεις για την κατασκευή Neural Bag of Words αναπαραστάσεων για κάθε σχόλιο στο corpus και εκπαιδεύστε ένα Logistic Regression μοντέλο για ταξινόμηση. Σχολιάστε τα αποτελέσματα. Εξηγείστε τη χαμηλή επίδοση του μοντέλου.
γ) Κατεβάστε τα προεκπαιδευμένα GoogleNews vectors.
https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit.
δ) Φορτώστε τα με gensim και εξάγετε αναπαραστάσεις με βάση αυτά. Επαναλάβατε το ερώτημα 9γ για τις ίδιες λέξεις με χρήση των GoogleNews. Σχολιάστε τα αποτελέσματα
Hint: (model = KeyedVectors.load_word2vec_format('./GoogleNews-vectors-negative300.bin', binary=True, limit=NUM_W2V_TO_LOAD)
Χρησιμοποιείστε την παράμετρο limit για να μη γεμίσετε τη μνήμη.
ε) Εκπαιδεύστε ένα LogisticRegression ταξινομητή και συγκρίνετε τα αποτελέσματα
δ) Δημιουργείστε αναπαραστάσεις προτάσεων με χρήση σταθμισμένου μέσου των w2v αναπαραστάσεων των λέξεων. Ως βάρη χρησιμοποιήστε τα TF-IDF βάρη των λέξεων.
ε) Επαναλάβατε την ταξινόμηση με τις αναπαραστάσεις του ερωτήματος δ)

Βήμα 19: Extra credit

α) Πειραματιστείτε με άλλους ταξινομητές (SVM, KNN etc). Συγκρίνετε την επίδοση τους.
β) Πειραματιστείτε με άλλα word embeddings. https://github.com/facebookresearch/fastText. Συγκρίνετε τις διαφορές τους από τα w2v.

**ΠΑΡΑΔΟΤΕΑ**

(1) Σύντομη αναφορά (σε 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.