ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Απαλλακτική Εργασία Μαθήματος: «Επεξεργασία Φυσικής Γλώσσας»

Όνομα Φοιτητή – Αριθμός Μητρώου	Αδάμ Δέσποινα – Π22003
	Γκίκα Δήμητρα – Π22034
	Παλιακάση Ειρήνη – Π22127
Ημερομηνία παράδοσης	24/09/2025

Πίνακας Περιεχομένων

Εισαγωγή

Η σημασιολογική ανακατασκευή αναφέρεται στη διαδικασία κατά την οποία ένα σύστημα επιχειρεί να αναδομήσει το νόημα ενός κειμένου, μιας φράσης ή μιας πρότασης, πέρα από τη συντακτική ή λεξική τους μορφή. Δεν αρκεί δηλαδή να αναγνωριστούν οι λέξεις και η γραμματική δομή, αλλά χρειάζεται να αποδοθεί η πραγματική σημασία τους, λαμβάνοντας υπόψη το πλαίσιο, τις σχέσεις μεταξύ εννοιών και την πρόθεση του ομιλητή ή συγγραφέα.

Η σημασιολογική ανακατασκευή είναι σημαντική γιατί συμβάλλει στην αποσαφήνιση του αρχικού νοήματος ενός κειμένου. Παράλληλα διευκολύνει την κατανόηση του πλαισίου, λαμβάνοντας υπόψη στοιχεία όπως πρόθεση, ειρωνεία, συνδηλώσεις ή μεταφορές. Επιπλέον, υποστηρίζει κρίσιμες εφαρμογές, όπως η αυτόματη μετάφραση, η περίληψη κειμένων, τα συστήματα ερωταποκρίσεων, οι φωνητικοί βοηθοί και τα συστήματα σύστασης.

Η Επεξεργασία Φυσικής Γλώσσας (NLP) παρέχει τα απαραίτητα εργαλεία και μεθόδους για να πραγματοποιηθεί αυτή η ανακατασκευή. Αυτό γίνεται μέσα από τη λεξιλογική και συντακτική ανάλυση (όπως tokenization, POS tagging και parsing) για την αναγνώριση των βασικών δομών, καθώς και τη σημασιολογική ανάλυση (με τεχνικές όπως word embeddings, contextual embeddings και semantic role labeling) για την κατανόηση εννοιών και ρόλων.

Σημαντικό ρόλο παίζει επίσης η αποσαφήνιση της σημασίας λέξεων (Word Sense Disambiguation), που συνδέει τη λέξη με το σωστό της νόημα, καθώς και η ανάλυση συναισθήματος και πραγματολογίας για την κατανόηση της πρόθεσης. Τέλος, η χρήση μοντέλων deep learning, όπως οι μετασχηματιστές (Transformers) τύπου BERT και GPT, δίνει τη δυνατότητα εκμάθησης σημασιολογικών αναπαραστάσεων από μεγάλα σώματα κειμένων.

Μεθοδολογία

Στρατηγικές Ανακατασκευής

Παραδοτέο 1 : Α

Για την ανακατασκευή των 2 προτάσεων που επιλέξαμε από τις παραγράφους που μας δόθηκαν, χρησιμοποιήσαμε τις εξής στρατηγικές ανακατασκευής, αφού κάναμε χειρωνακτικά την συντακτική ανάλυση και διαπιστώσαμε ποια είναι τα πιθανά επιθυμητά αποτελέσματα ανακατασκευής των αρχικών προτάσεων:

Αρχικά inputs:

```
INPUT: Hope you too, to enjoy it as my deepest wishes
```

```
INPUT: I got this message to see the approved message
```

Στρατηγικές μετασχηματισμού(edit_ops):

1. **Εισαγωγές** → insert_I (προσθέτει τη λέξη "i" στην αρχή της πρότασης).
2. **Διαγραφές** → delete_first_too (αφαιρεί την πρώτη εμφάνιση της λέξης " too"), delete_to_before_enjoy (αν υπάρχει "to enjoy" αφαιρεί το "to")
3. **Μετακινήσεις** → move_too_after_it (αν υπάρχει "too" μετακινείται ώστε να εμφανίζεται μετά από το "it")
4. **Αντικαταστάσεις** → sub_as_to_with (αντικαθιστά το "as" με "with"), sub_this_to_your (αντικαθιστά το "this" με "your"), sub_see_to_review (αντικαθιστά το "see" με "review"), sub_last_message_to_content (αντικαθιστά την τελευταία εμφάνιση της λέξης "*message*" με "*content*"

Χρήση μετασχηματισμών:

Για κάθε υποψήφια πρόταση:

- Γίνονται όλοι οι δυνατοί συνδυασμοί αυτών των μετασχηματισμών (μέχρι και $2^8 = 256$ πιθανές παραλλαγές).
- Κάθε πρόταση βαθμολογείται με βάση τρία κριτήρια:
    1. **structure_score** : Συμβατότητα με μια απλοποιημένη γραμματική PCFG (συντακτική ορθότητα).
    2. **bigram_score** : Συμβατότητα με bigram language model (στατιστική φυσικότητα ακολουθιών λέξεων).
    3. **levenshtein** : Απόσταση επεξεργασίας από το αρχικό input (ποινή για πολύ μεγάλες αλλαγές).
- Με τις υπολογιστικές τεχνικές που περιγράφονται παρακάτω, το αυτόματο καταλήγει στις 6 καλύτερες ανακατασκευές:

```
Top 6 candidates (detokenized) with scores and posterior:
  -10.4262  ed=5  p_struct=0.024 p_bigram=0.06744 posterior=0.6467 -> I hope you enjoy it too, with my deepest wishes.
  -10.9356  ed=4  p_struct=0.03 p_bigram=0.01457 posterior=0.3421 -> hope you enjoy it too, with my deepest wishes.
  -14.0693  ed=4  p_struct=0.024 p_bigram=0.000793 posterior=0.006807 -> I hope you enjoy it, with my deepest wishes.
  -14.5787  ed=3  p_struct=0.03 p_bigram=0.0001713 posterior=0.003601 -> hope you enjoy it, with my deepest wishes.
  -16.3040  ed=3  p_struct=0.048 p_bigram=1.907e-05 posterior=0.0004167 -> I hope you enjoy it as my deepest wishes.
  -16.8134  ed=2  p_struct=0.06 p_bigram=4.119e-06 posterior=0.0002205 -> hope you enjoy it as my deepest wishes.
```

```
Top 6 candidates (detokenized) with scores and posterior:
  -8.1429  ed=3  p_struct=0.04493 p_bigram=0.07135 posterior=0.9783 -> I got your message to review the approved content.
  -11.4123  ed=2  p_struct=0.0576 p_bigram=0.0009509 posterior=0.01643 -> I got your message to review the approved message.
  -12.8549  ed=4  p_struct=0.04493 p_bigram=0.001427 posterior=0.002707 -> I got your message to review the approved content too.
  -12.9821  ed=4  p_struct=0.04493 p_bigram=0.001257 posterior=0.002309 -> I I got your message to review the approved content.
  -15.3886  ed=2  p_struct=0.0576 p_bigram=1.783e-05 posterior=0.000114 -> I got your message to see the approved content.
  -15.6482  ed=2  p_struct=0.04493 p_bigram=1.764e-05 posterior=8.243e-05 -> I got this message to review the approved content.
```

- Και επιλέγουμε τις καλύτερες από αυτές:

```
BEST (deterministic argmax): I hope you enjoy it too, with my deepest wishes.
```

```
BEST (deterministic argmax): I got your message to review the approved content.
```

Παραδοτέο 1 : Β

Για την ανακατασκευή των αρχικών παραγράφων που μας δόθηκαν, χρησιμοποιήσαμε τα εξής pipelines ώστε να τις ανακατασκευάσουμε με 3 διαφορετικούς τρόπους :

Αρχικά inputs:

```
Original Text:
Today is our dragon boat festival, in our Chinese culture, to celebrate it with all safe and great in our lives. Hope you too, to enjoy it as my deepest
wishes. Thank your message to show our words to the doctor, as his next contract checking, to all of us. I got this message to see the approved message. In
fact, I have received the message from the professor, to show me, this, a couple of days ago. I am very appreciated the full support of the professor, for
our Springer proceedings publication
```

```
Original Text:
During our final discuss, I told him about the new submission — the one we were waiting since last autumn, but the updates was confusing as it not included
the full feedback from reviewer or maybe editor? Anyway, I believe the team, although bit delay and less communication at recent days, they really tried best
 for paper and cooperation. We should be grateful, I mean all of us, for the acceptance and efforts until the Springer link came finally last week, I think.
Also, kindly remind me please, if the doctor still plan for the acknowledgments section edit before he sending again. Because I didn't see that part final
yet, or maybe I missed, I apologize if so. Overall, let us make sure all are safe and celebrate the outcome with strong coffee and future targets
```

1. **Μοντέλο humarin/chatgpt_paraphraser_on_T5_base:** Βασισμένο στο μοντέλο T5, ειδικό για παράφραση κειμένου. Ορίσαμε ως task text2text - generation και προσαρμόσαμε ελαφρώς την θερμοκρασία (temperature = 0.8), ώστε η παράφραση να είναι αρκετά κοντά στις αρχικές παραγράφους . (https://huggingface.co/humarin/chatgpt_paraphraser_on_T5_base)

   Ανακατασκευασμένα κείμενα:

   ```
   Corrected Text:
   Our Chinese culture celebrates the dragon boat festival today to ensure the safety and prosperity of all involved. May you also have a good time, as my thoughts
    are with you. We appreciate your message to convey our thanks to the doctor for sharing our thoughts as they prepare for their next contract check. I was
    notified to view the sanctioned message. The professor sent me a message to show me this a few days ago. The professor's complete backing has been greatly
    appreciated for publishing our Springer proceedings.
   ```

   ```
   Corrected Text:
   During our last discussion, I shared with him the news of the new submission we had been waiting for last autumn, but the changes were unclear as they did not
    include full feedback from the reviewer or editor. Despite the delay and reduced communication, the team made efforts to produce papers and collaborate
    effectively. The acceptance and efforts made until the Springer link finally arrived last week should be acknowledged by everyone. Kindly inform me if the doctor
     has time to revise the acknowledgments section before resending. Is it possible that I didn't witness the final segment or was I missing something? I'm sorry if
     this appears. In the end, we must ensure that everyone is secure and mark the occasion with potent coffee and future objectives.
   ```

2. **Μοντέλο stanford-oval/paraphraser-bart-large:** Βασισμένο στο μοντέλο BART, αρκετά καλό για παράφραση κειμένου. Ορίσαμε ως task text2text - generation και επιλέξαμε να αφήσουμε τις παραμέτρους του στις default τιμές, καθώς έδινε αρκετά καλά αποτελέσματα (λόγω της εντροπίας με βάση την οποία λειτουργεί το συγκεκριμένο μοντέλο). (https://huggingface.co/stanford-oval/paraphraser-bart-large)

   Ανακατασκευασμένα κείμενα:

```
Corrected Text:
Today is our Dragon Boat Festival. In Chinese culture, we celebrate this with everything safe and beautiful in our lives. I hope you enjoy this as much as I do,
 as is my deepest wish. Thank you for your report showing our words to the doctor, as well as his next contractual check-up, for all of us. I received this
 message so I could see the approved report. Actually, I got a text from the professor to show me this a few days ago. I really appreciate the professor's full
 support in our Springer publication.
```

```
Corrected Text:
During our last discussion, I told him about a new submission -- the one we had been waiting for since last fall, but the update was confusing because it didn't
 include full feedback from a reviewer or even an editor. Anyway, I trust that the team, even if a little late and less communicative in the last few days, has
 really tried its best in terms of paper and cooperation. I think we should all be grateful, I mean all of us, for accepting and trying until the Springer
 connection finally came through last week. Also, please remind me if the doctor is still planning to edit the acknowledgements section before sending it again.
 Since I haven't seen the final part yet, or maybe I missed it, I'm sorry if I did. All in all, let's make sure everyone is safe and celebrate the results with
 strong coffee and future goals.
```

3. **Μοντέλο tuner007/pegasus_paraphrase:** Βασισμένο στο μοντέλο Pegasus, σχετικά καλό για παράφραση κειμένου. Ορίσαμε ως task text2text - generation και προσαρμόσαμε τις παραμέτρους του (temperature = 1.1, num_beams=10, repetition_penalty=2.0, top_k=30, top_p=0.8), ώστε η παράφραση να είναι αρκετά κοντά στις αρχικές παραγράφους και ταυτόχρονα να υπάρχουν κάποιες διαφοροποιήσεις. Το ρίσκο του συγκεκριμένου μοντέλου είναι η απώλεια πληροφορίας κατά την ανακατασκευή, καθώς το μοντέλο έχει σχεδιαστεί κυρίως για παράφραση κειμένου που επικεντρώνεται στην σύνοψη (summarization) του αρχικού κειμένου
(https://huggingface.co/tuner007/pegasus_paraphrase)

   Ανακατασκευασμένα κείμενα:

```
Corrected Text:
Today is the dragon boat festival in our Chinese culture and we should celebrate it with great joy. Hope you enjoy it as my deepest wishes. Thank you for your
 message, it will be shown to the doctor as his next contract checking. I received this message to see the approved one. I received the message from the professor
  a couple of days ago. The professor was very supportive of the Springer proceedings publication.
```

```
Corrected Text:
During our final discuss, I told him about the new submission we were waiting for since last autumn, but it was confusing as it did not include the full feedback
 from reviewer or editor. I believe the team tried their best for paper and cooperation despite delay and less communication recently. We should be thankful for
 the acceptance and efforts until the Springer link came last week, I think. If the doctor still plans for the acknowledgments section to be edited before he
 sends again, please remind me. I apologize if I missed that part final because I didn't see it yet. Let us make sure all are safe and celebrate the outcome with
 coffee and targets.
```

Ανάλυση υπολογιστικών τεχνικών

Παραδοτέο 1 : Α

Βιβλιοθήκες που χρησιμοποιήθηκαν : math, random

Επεξήγηση Κώδικα:

1. Δημιουργήσαμε την συνάρτηση *tokenize(sent)* που :
   a. Λαμβάνει ως όρισμα την πρόταση (sent) που θέλουμε να κάνουμε tokenize
   b. Προσθέτει σε όλα τα κόμματα κενά στην αρχή και στο τέλος τους
   c. Δημιουργεί μια λίστα από λέξεις (toks) και για κάθε λέξη που προστίθεται στην λίστα:
      i. Αφαιρεί τα περιττά κενά (strip)
      ii. Μετατρέπει όλα τα γράμματα της λέξης σε πεζά (lower)

iii. Αγνοεί τα κόμματα

d. Επιστρέφει την λίστα με τις λέξεις

2. Δημιουργήσαμε την συνάρτηση *detokenize(tokens)* που :

   a. Λαμβάνει ως όρισμα μια λίστα(tokens) με λέξεις

   b. Αν η λίστα είναι κενή τότε επιστρέφεται η κενή συμβολοσειρά

   c. Αν η λίστα περιλαμβάνει tokens, τότε σχηματίζουμε την πρόταση s ενώνοντας όλα τα tokens με την κενή συμβολοσειρά. Για λόγους αισθητικής, αν στην πρόταση βρεθεί το token «i» το μετατρέπουμε σε «I» και αντικαθιστούμε το token «with» με «, with»

   d. Τελικά, επιστρέφουμε την ανακατασκευασμένη πρόταση

3. Δημιουργήσαμε την συνάρτηση *levenshtein(a, b)* που:

   a. Λαμβάνει ως ορίσματα 2 συμβολοσειρές (a,b)

   b. Αφού βρούμε το μήκος των 2 συμβολοσειρών, ορίζουμε έναν πίνακα dp (len(a)+1) × (len(b)+1) σε κάθε θέση (i,j) του οποίου αποθηκεύεται κάθε φορά η απόσταση μεταξύ των πρώτων i χαρακτήρων του string a και των πρώτων j χαρακτήρων από το string b. Ο υπολογισμός των αποστάσεων γίνεται ως εξής:

      i. Για την πρώτη γραμμή, για την οποία ισχύει i=0 (χαρακτήρες του a) χρειαζόμαστε j εισαγωγές (χαρακτήρων του b)

      ii. Για την πρώτη γραμμή, για την οποία ισχύει j=0 (χαρακτήρες του b) χρειαζόμαστε i διαγραφές (χαρακτήρων του a)

      iii. Για τις υπόλοιπες γραμμές και στήλες υπολογίζουμε την απόσταση Levenshtein λαμβάνοντας υπόψη: (i) την τοπική ομοιότητα των χαρακτήρων και (ii) το κόστος εισαγωγής, διαγραφής ή αντικατάστασης χαρακτήρα. Αν το σύμβολο i είναι ακριβώς ίδιο με το σύμβολο j, τότε το κόστος της τοπικής ομοιότητας γίνεται 0 ενώ σε διαφορετική περίπτωση γίνεται 1 . Στην συνέχεια, υπολογίζουμε την απόσταση υπολογίζοντας τα κόστη εισαγωγής χαρακτήρα, διαγραφής χαρακτήρα και αντικατάστασης χαρακτήρα (ή μη αν το κόστος της τοπικής ομοιότητας είναι 0).

   c. Τελικά, επιστρέφουμε τον πίνακα αποστάσεων levenshtein των συμβολοσειρών a, b

4. Ορίσαμε την λίστα *VOCAB* που περιλαμβάνει όλες τις λέξεις που θα χρησιμοποιηθούν για βελτίωση/αντικατάσταση των αρχικών προτάσεων

5. Ορίσαμε το λεξικό *BIGRAM* που περιλαμβάνει για τις 2 προτάσεις που θέλουμε να ανακατασκευάσουμε τις πιθανότητες εμφάνισης 2 λέξεων στην σειρά, καθώς και τις πιθανότητες έναρξης μιας πρότασης με μια συγκεκριμένη λέξη. Για τα tokens που δεν περιλαμβάνονται στο συγκεκριμένο BIGRAM, ορίσαμε ένα default bigram score (BIGRAM_DEFAULT)

6. Ορίσαμε το λεξικό PCFG_PROBS που περιλαμβάνει τους πιθανοτικούς κανόνες της γραμματικής μας.
7. Ορίσαμε το add-k smoothing, προκειμένου στην διαδικασία ανακατασκευής να ελέγχονται όλοι οι κανόνες.
8. Ορίσαμε το temperature scaling της κατανομής των πιθανοτήτων. Στην περίπτωση μας αφήνουμε την κατανομή όπως είναι και δεν αλλάζουμε τις πιθανότητες.
9. Ορίσαμε την συνάρτηση *generate_candidates(tokens)* που:
    a. Λαμβάνει ως όρισμα έναν πίνακα με λέξεις(tokens)
    b. Αρχικοποιούμε μια λίστα(edit_ops) με τις βασικές λειτουργίες που θα γίνουν στις προτάσεις.
    c. Έπειτα, ορίζουμε την εσωτερική συνάντηση apply_flags(toks, flags) η οποία:
        i. Δέχεται σαν όρισμα μια λίστα με tokens και ένα λεξικό που κάνει δυναμικά map τις λειτουργείες edit_ops σε true ή false με βάση αν σε έναν έλεγχο χρησιμοποιείται μια λειτουργία ή όχι
        ii. Παίρνει ένα αντίγραφο από τη λίστα με τα tokens
        iii. Κάνει τις αντικαταστάσεις, τις εισαγωγές και τις διαγραφές
        iv. Εισάγει το “i” στην αρχή εάν είναι απαραίτητο
        v. Επιστρέφει την νέα λίστα με τα tokens που έχει φτιάξει.
    d. Αρχικοποιήσουμε το λεξικό candidates, το οποίο:
        i. Ελέγχει 256 συνδυασμούς για τις οκτώ πιθανές αλλαγές
        ii. Για κάθε mask, φτιάχνει το λεξικό με τις κατάλληλες τιμές από τα flags, ανάλογα με το ποια bits είναι ενεργά.
    e. Εφαρμόζουμε αυτές τις αλλαγές στην λίστα με τα tokens και αποθηκεύουμε την τροποποιημένη σειρά αυτών σαν κλειδί στο λεξικό candidates (συνδεδεμένο με το flag set που το παρήγαγε).
    f. Ελέγχουμε ότι το αρχικό input περιέχεται στην επιστρεφόμενη λίστα ακόμη και αν δεν εφαρμόστηκαν αλλαγές.
    g. Επιστρέφουμε την αλλαγμένη πρόταση ως λίστα από tokens, και το λεξικό με τα flags που εφαρμόστηκαν.
10. Δημιουργούμε ένα unigram backoff distribution από το ήδη υπάρχον bigram, ως εξής:
    a. Ορίζουμε ως μεταβλητές ένα λεξικό το οποίο θα περιέχει το άθροισμα από όλα τα bigram σκορ που ξεκινούν με κάθε λέξη και τον αθροιστή από όλα τα bigram σκορ από το λεξικό.

b. Για κάθε ζεύγος από bigrams στο αρχικό προσθέτει το σκορ όλων των bigrams που ξεκινούν με a στον αντίστοιχο αθροιστή και τελικά υπολογίζει το συνολικό άθροισμα από όλα τα bigram scores.

c. Τελευταίο βήμα είναι η κανονικοποίηση του λεξικού που ορίσαμε στην αρχή με τον αθροιστή. Αυτό θα μας δώσει μια κατανομή η οποία προκύπτει ως συνάρτηση της συχνότητας εμφάνισης κάθε λέξης ως την πρώτη λέξη ενός bigram.

11. Ορίσαμε την συνάρτηση *bigram_cond_prob(prev, w, vocab_list=VOCAB)* που:

a. Υπολογίζει την δεσμευμένη πιθανότητα μια λέξης δεδομένης μια άλλης.

b. Παίρνει το μη κανονικοποιημένο σκορ από το bigram, (prev, w), από το αρχικό bigram, αν δεν υπάρχει τότε το θέτει μηδενικό.

c. Στην συνέχεια υπολογίζει προσεγγιστικά όλα τα bigram scores που το prev τους είναι η πρώτη λεξη. Δεν χρησιμοποιεί όλα τα πιθανά σενάρια, μόνο αυτά που βρίσκονται στο vocab_list.

d. Χρησιμοποιεί add-k smoothing για να αποφύγει μηδενικές πιθανότητες, έτσι ακόμη και αν το (prev, w) δεν παρατηρηθεί το smoothed θα είναι μεγαλύτερο του μηδενός.

e. Ελέγχουμε αν το prev εμφανίζεται στο μοντέλο του bigram. Αν ναι χρησιμοποιείται η backoff πιθανότητα του, αλλιώς χρησιμοποιεί την καθολική.

f. Υπολογίζουμε την πιθανότητα παρεμβολής, η οποία είναι η ανάμειξη της δεσμευμένης εκτίμησης από το bigram και το unigram backoff. Με το βάρος παρεμβολής alpha, που ισούται με 0.15, να σημαίνει ότι έχουμε 85% πιθανότητα από το bigram (μετά την εξομάλυνση) και 15% από την backoff πιθανότητα.

12. Ορίσαμε την συνάρτηση *bigram_score(tokens, temp=TEMPERATURE)* που:

a. Υπολογίζει ένα πιθανοτικό σκορ για μια σειρά από tokens χρησιμοποιώντας ένα μοντέλο bigram με χειρισμό smoothing & temperature.

b. Ελέγχουμε αν η λίστα με τα tokens είναι άδεια, εφόσον είναι τότε επιστρέφουμε το ουδέτερο σκορ που είναι η μονάδα.

c. Αρχικοποιούμε την λογαριθμική πιθανότητα (logp) στο μηδέν και ως προηγούμενη κατάσταση θέτουμε την '<s>', που είναι η αρχή.

d. Για κάθε token στην σειρά που έχουμε, υπολογίζουμε την πιθανότητα αυτού δεδομένης της προηγούμενης κατάστασης χρησιμοποιώντας το smoothed bigram μοντέλο, προσθέτουμε τον λογάριθμο της πιθανότητας αυτής στο logp, σιγουρεύοντας ότι δεν θα έχουμε το log(0), και θέτουμε το παρόν token την νέα προηγούμενη κατάσταση.

e. Εφαρμόζουμε temperature scaling στο logp

f. Επιστρέφουμε την λογαριθμική πιθανότητα αφού την κανονικοποιήσουμε.

13. Ορίσαμε την συνάρτηση *structure_score(tokens)* που:

    a. Προσεγγίζει μια συντακτική πιθανότητα για μια δεδομένη σειρά από tokens, βασισμένη σε μια ευριστική που είναι σαν PCFG. Δίνει τα υψηλά σκορ σε σειρές που μοιάζουν με τις γραμματικές δομές σύμφωνα με απλοποιημένο, πιθανοτικό μοντέλο (PCFG_PROBS).

    b. Επιστρέφουμε μια πάρα πολύ μικρή τιμή εάν η λίστα με τα tokens είναι άδεια.

    c. Ελέγχουμε με τι token ξεκινά η πρόταση, εάν ξεκινά με (i, you, we , it) τότε εικάζουμε ότι είναι της μορφής S -> NP VP και παίρνουμε με την σειρά τις πιθανότητες για τις επόμενες μορφές "φράσεων". Αλλιώς εικάζουμε ότι είναι της μορφής S -> VP NP.

    d. Ελέγχουμε επαναληπτικά εάν ποια tokens είναι γενικές λέξεις ή ρήματα, για να αυξήσουμε την πιθανότητα αντίστοιχα.

    e. Ως επόμενο βήμα έχουμε τον έλεγχο εμφάνισης δύο φράσεων στην πρόταση. Εάν αναγνωριστεί η πρώτη περίπτωση τότε ενισχύουμε το σκορ χρησιμοποιώντας τον κανόνα VP-> V NP ADV PP, ενώ αν είναι η δεύτερη τότε χρησιμοποιεί μια σταθερά, υποδεικνύοντας ότι το περιεχόμενο είναι σημαντικό.

    f. Τέλος επιστρέφουμε το τελικό σκορ δομής, έχοντας ως κάτω όριο μια πολύ μικρή τιμή.

14. Ορίσαμε την συνάρτηση *softmax_from_logits(logits, temp=1.0)* που:

    a. Μετατρέπει απλά σκορ σε πιθανοτική κατανομή.

    b. Ως πρώτο βήμα βρίσκει το μέγιστο logit.

    c. Σε κάθε logit αφαιρούμε το μέγιστο που βρήκαμε προηγουμένως (mx), του εφαρμόζουμε temperature scaling και μετά το κάνουμε exponentiate, δηλαδή χρησιμοποιούμε το αποτέλεσμα ως δύναμη του e.

    d. Τέλος, διαιρούμε κάθε τιμή από τη λίστα exps με το άθροισμα αυτών για να έχουμε μια σωστή πιθανοτική κατανομή.

15. Ορίσαμε την συνάρτηση *rank_and_sample_candidates(scored_candidates, sample_n=0, sample_temp=1.0)* που:

    a. Δέχεται ως όρισμα μια λίστα λεξικών (scored_candidates),που το καθένα από αυτά περιέχει μια αριθμητική τιμή, το πλήθος των candidates που θα δειγματοληπτούμε (sample_n) και το μέγεθος του temperature scaling (sample_temp) που θα χρησιμοποιηθεί κατά τη διάρκεια της δειγματοληψίας.

    b. Φτιάχνει την λίστα με τα σκορ, logits, και υπολογίζει τις κανονικοποιημένες πιθανότητες.

    c. Ορίζονται οι πιθανότητες που υπολογίστηκαν ως νέο κλειδί για κάθε candidate λεξικό και ταξινομείται η σειρά των candidates από το καλύτερο στο χειρότερο.

d. Για κάθε δείγμα, παίρνουμε έναν τυχαίο αριθμό από το [0,1) και για κάθε candidate συγκεντρώνουμε τις πιθανότητες του (posterior).

e. Επιλέγουμε το πρώτο candidate που η συνολική πιθανότητα του ξεπερνά τον τυχαίο αριθμό που πήραμε.

f. Τέλος, επιστρέφουμε την οργανωμένη λίστα με τα candidates (sorted_cands) και την, sampled, λίστα με τα candidates που δειγματοληπτήσαμε με βάση το posterior.

16. Ορίσαμε την συνάρτηση *rewrite_sentence(src_sentence, alpha=1.0, beta=1.0, gamma=1.0, temp=TEMPERATURE, sample_n=0, sample_temp=1.0)* που:

    a. Λαμβάνει την πρόταση που της δίνουμε(src_sentence), δημιουργεί, αλλά και βαθμολογεί, πολλές παραλλαγές της και επιστρέφει την καλύτερη παραλλαγή της αρχικής πρότασης, όλες τις βαθμολογημένες, υποψήφιες, αλλαγμένες προτάσεις που προκύπτουν και προαιρετικά δείγματα από τις posterior κατανομές.

    b. Χωρίζει την αρχική πρόταση σε λίστα από tokens και έπειτα φτιάχνει όλα τα είδη αλλαγών που μπορούν να γίνουν στην πρόταση (π.χ. εισαγωγή, διαγραφή κλπ.).

    c. Δημιουργούμε την λίστα scores και για κάθε candidate υπολογίζουμε το σκορ που έχει με βάση τη δομή της, το bigram score που χρησιμοποιείται για την προσέγγιση του fluency και βάζει πέναλτι σε candidates που απέχουν πολύ από την αρχική.

    d. Υπολογίζουμε το συνολικό σκορ (combined_log), που είναι η λογαριθμική πιθανότητα ενός candidate δεδομένης της δομής (alpha), του fluency από λέξη σε λέξη (beta) και της ομοιότητας της ως προς την αρχική (gamma).

    e. Κάθε candidate γίνεται ένα λεξικό με metadata και στη συνέχεια κατατάσσονται τα λεξικά με βάση τα σκορ

    f. Τελικά, επιλέγεται το πρώτο από την λίστα με τα διατεταγμένα candidates και επιστρέφεται μαζί με την λίστα με όλα τα υπόλοιπα candidates και ένα τυχαία επιλεγμένο υποσύνολο

17. Ορίσαμε την συνάρτηση *main()* που:

    a. Αρχικοποιείται μια λίστα(inputs) με τις δυο προτάσεις που θέλουμε να ανακατασκευάσουμε

    b. Ρυθμίζουμε τις default τιμές παραμέτρων που επηρεάζουν την συντακτική ορθότητα (alpha), την φυσικότητα του λόγου (beta), την ομοιότητα με τις αρχικές προτάσεις (gamma), καθώς και τις τιμές των υποψηφίων (sample_n, sample_temp) κατά τέτοιον τρόπο, ώστε η ανακατασκευή να έχει μια σχετική ελευθερία στην παράφραση και ταυτόχρονα να μην χάνεται το νόημα των αρχικών προτάσεων.

    c. Έπειτα ορίζουμε μια λίστα (outputs) και για κάθε πρόταση στην λίστα inputs:

        i. Βρίσκουμε το καλύτερο candidate(best), την λίστα με τα candidates(ranked) και την λίστα με το υποσύνολο(sampled) με την βοήθεια της συνάρτησης

rewrite_sentence(src, alpha=alpha, beta=beta, gamma=gamma, temp=temp, sample_n=sample_n, sample_temp=sample_temp)

ii. Εκτυπώνουμε την καλύτερη ανακατασκευή και τις 6 υποψήφιες ανακατασκευές

iii. Αποθηκεύουμε τις καλύτερες ανακατασκευές στο αρχείο outputs_of_A

Παραδοτέο 1 : Β

Βιβλιοθήκες που χρησιμοποιήθηκαν : transformers.pipeline (Για την εισαγωγή των τριών pipelines από το huggingface)

Επεξήγηση Κώδικα:

1. Δημιουργήσαμε την συνάρτηση *paraphrase_text(include_paraphrase, include_dot, cut_at_question_mark, pipeline, input_text, **pipeline_args) που :*
    1. Λαμβάνει ως ορίσματα:
        i. *include_paraphrase => Καθορίζει σε ένα μοντέλο αν θα υπάρξει η διευκρίνηση " paraphrase: (κείμενο)" πριν επεξεργαστεί το κείμενο από αυτό*
        ii. *include_dot => Καθορίζει αν στο τέλος μιας πρότασης πρέπει να προστεθεί τελεία*
        iii. *cut_at_question_mark => Καθορίζει αν μια πρόταση πρέπει να χωριστεί από την επόμενη στο ερωτηματικό(αν υπάρχει)*
        iv. *pipeline => Καθορίζει το transformer pipeline που χρησιμοποιείται για την παράφραση*
        v. *input_text => Το κείμενο που θα παραφραστεί*
        vi. ***pipeline_args => Προαιρετικά keyword arguments για το pipeline (π.χ. do_sample, temperature, num_beams, top_k, top_p, repetition_penalty).*
    2. Διαιρεί το αρχικό κείμενο σε μεμονωμένες προτάσεις με βάση το σημείο στίξης στο οποίο τελειώνουν(«.» ή «?»).Στην περίπτωση που το σημείο στίξης είναι «?» αντικαθίσταται από «.». Στην συνέχεια σπάει το κείμενο σε προτάσεις με βάση τις τελείες και τις αποθηκεύει σε μια λίστα προτάσεων.
    3. Έπειτα το pipeline παραφράζει κάθε πρόταση που περιλαμβάνεται στην λίστα των προτάσεων (εκτυπώνοντας ένα progress bar που υποδεικνύει τον χρόνο που χρειάζεται η ανακατασκευή της κάθε πρότασης) και οι παραφρασμένες προτάσεις αποθηκεύονται σε μια λίστα corrected sentences.
    4. Στην συνέχεια, ενώνονται οι παραφρασμένες προτάσεις και - αφού εκτυπωθεί- επιστρέφεται στο κύριο πρόγραμμα το ανακατασκευασμένο κείμενο.
2. Στο κύριο πρόγραμμα:
    1. Ορίζουμε τα 3 pipelines καθορίζοντας για κάθε ένα από αυτά :
        i. Το task παράφρασης (text2text-generation)
        ii. Το μοντέλο παράφρασης που θα χρησιμοποιηθεί
        iii. Τον tokenizer (= τρόπο σπασίματος των προτάσεων σε λέξεις) που χρησιμοποιεί το συγκεκριμένο μοντέλο

2. Ορίζουμε τα 2 κείμενα που θέλουμε να παραφράσουμε και δημιουργούμε μια κενή λίστα στην οποία θα συμπεριλάβουμε τα ανακατασκευασμένα κείμενα
3. Για κάθε pipeline, χρησιμοποιούμε την συνάρτηση paraphrase_text που ορίσαμε παραπάνω 2 φορές (μια για το πρώτο και μια για το δεύτερο κείμενο) προσαρμόσαμε τις εξής **pipeline_args παραμέτρους, ώστε να επιτύχουμε καλύτερα αποτελέσματα:

ΓΙΑ ΤΟ ΠΡΩΤΟ ΚΕΙΜΕΝΟ

Pipeline 3**(tuner007/pegasus_paraphrase)**:

i. do_sample = True, ώστε να γίνεται τυχαία δειγματοληψία των tokens για την ανακατασκευή των προτάσεων
ii. temperature=1.1 ώστε η ανακατασκευή να είναι κοντά στο αρχικό κείμενο, αλλά ταυτόχρονα να υπάρχει και ελευθερία στην επιλογή των tokens παράφρασης
iii. num_beams=10, ψάχνουμε 10 "διαδρομές" για καλύτερο αποτέλεσμα
iv. repetition_penalty=2.0, αυστηρή ποινή που αποθαρρύνει το μοντέλο από την επανάληψη tokens
v. top_k=30, ώστε να περιορίσουμε επιλογές στα 30 καλύτερα tokens
vi. top_p=0.8, ώστε να περιορίζουμε επιλογές στα tokens που αθροίζουν τουλάχιστον σε πιθανότητα ίση με 0.8

ΓΙΑ ΤΟ ΔΕΥΤΕΡΟ ΚΕΙΜΕΝΟ

Pipeline 1**(humarin/chatgpt_paraphraser_on_T5_base)**:

i. do_sample = True, ώστε να γίνεται τυχαία δειγματοληψία των tokens για την ανακατασκευή των προτάσεων
ii. temperature=0.8 ώστε η ανακατασκευή να είναι πιο κοντά στο αρχικό κείμενο και να περιοριστεί η ελευθερία στην επιλογή των tokens παράφρασης

Pipeline 3**(tuner007/pegasus_paraphrase)**:

1. do_sample = True, ώστε να γίνεται τυχαία δειγματοληψία των tokens για την ανακατασκευή των προτάσεων
2. temperature=1.1 ώστε η ανακατασκευή να είναι κοντά στο αρχικό κείμενο, αλλά ταυτόχρονα να υπάρχει και ελευθερία στην επιλογή των tokens παράφρασης,
3. num_beams=10, ψάχνουμε 10 "διαδρομές" για καλύτερο αποτέλεσμα
4. repetition_penalty=2.0, αυστηρή ποινή που αποθαρρύνει το μοντέλο από την επανάληψη tokens
5. top_k=30, ώστε να περιορίσουμε επιλογές στα 30 καλύτερα tokens
6. top_p=0.8, ώστε να περιορίζουμε επιλογές στα tokens που αθροίζουν τουλάχιστον σε πιθανότητα ίση με 0.8

4. Τέλος, αποθηκεύουμε όλες τις εκδοχές παράφρασης των αρχικών κειμένων στο αρχείο inputs_of_B

Επεξήγηση Παραδοτέου 1 : C

Βιβλιοθήκες που χρησιμοποιήθηκαν : re (για επεξεργασία των προτάσεων), nltk.PCFG (για την δημιουργία γραμματικής με πιθανότητες), nltk.ViterbiParser (για ανάλυση των προτάσεων με βάση τον αλγόριθμο Viterbi)

Προτάσεις που αναλύθηκαν : οι 2 προτάσεις που δημιουργήθηκαν στο Παραδοτέο 1A, καθώς και οι προτάσεις και των τριών transformers του Παραδοτέου 1B που αντιστοιχούν σημασιολογικά στις προτάσεις του Παραδοτέου 1A .

Επεξήγηση Κώδικα:

1. Δημιουργήσαμε μια πιθανοτική γραμματική που να εκφράζει και τις 8 προτάσεις που δημιουργήθηκαν:

ΕΙΔΗ ΠΡΟΤΑΣΕΩΝ/ΦΡΑΣΕΩΝ ΓΡΑΜΜΑΤΙΚΗΣ	
S	Βασική πρόταση
NP, N1	Ονοματική φράση
VP	Ρηματική φράση
INF	Απαρεμφατικές προτάσεις
SBAR	Δευτερεύουσες/Συγκριτικές προτάσεις
PP	Προθετικές φράσεις

ΓΡΑΜΜΑΤΙΚΗ ΑΝΑΛΥΣΗ ΛΕΞΕΩΝ	
Prep	Πρόθεση
Pron	Προσωπική/Δεικτική/Αριθμητική Αντωνυμία
Det	Προσδιοριστής(Μπροστά από ουσιαστικό)
INF	Απαρεμφατικές προτάσεις
Adj	Επιθετικός προσδιορισμός
N	Ουσιαστικό
V	Ρήμα
Modal	Τροπικό Ρήμα
Aux	Βοηθητικό Ρήμα (Χρονικό)
Adv	Επίρρημα

ΚΑΝΟΝΕΣ ΓΡΑΜΜΑΤΙΚΗΣ	
S →	NP VP | VP | Modal NP VP | Modal VP | NP
NP →	Pron | Det N1 | N | Det Adj N | Det N | NP PP
N1 →	Adj N | N | N INF | N SBAR
VP →	V NP | V | V NP PP | V NP INF | V S | Aux V NP | Aux V INF | Modal V NP | Modal V INF | Adv VP | V NP Adv | V NP SBAR | V NP SBAR SBAR | V NP Adv PP | Aux PP
INF →	'to' VP
PP →	Prep NP

*Οι υπόλοιποι κανόνες επικεντρώνονται στην κατηγοριοποίηση των λέξεων των προτάσεων που θέλουμε να αναλύσουμε με κριτήριο την γραμματική τους ανάλυση.

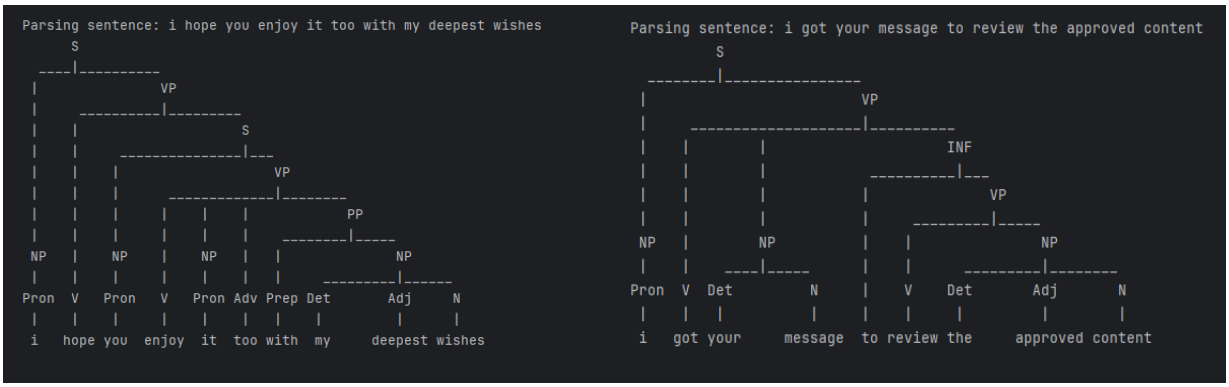
2. Αφού διαβάσουμε τις προτάσεις του αρχείου outputs_of_A και βρούμε τις αντίστοιχες προτάσεις του αρχείου outputs_of_B που θέλουμε να αναλύσουμε, αποθηκεύουμε της 8 προτάσεις σε μια λίστα('sentences') ώστε να τις επεξεργαστούμε.

3. Η επεξεργασία των προτάσεων γίνεται ως εξής :
    a. Για κάθε πρόταση στην λίστα sentences αφαιρούμε τις τελείες και τα κεφαλαία γράμματα
    b. Σπάμε την πρόταση σε λέξης (tokens)
    c. Αναλύουμε την πρόταση δημιουργώντας το grammar tree που της αντιστοιχεί με βάση τους κανόνες της γραμματικής

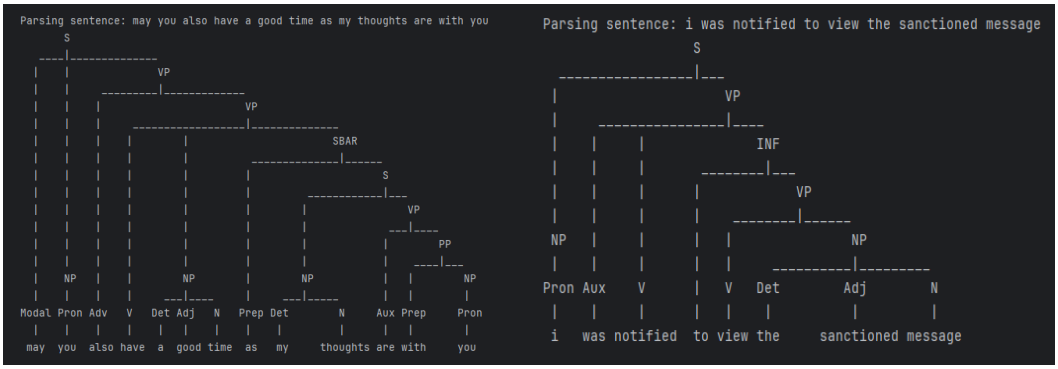
Σύγκριση των 8 Προτάσεων:

**Η παρακάτω σύγκριση γίνεται λαμβάνοντας υπόψη και τις αρχικές προτάσεις : "Hope you too, to enjoy it as my deepest wishes" (10 tokens) και "I got this message to see the approved message"(9 tokens)*

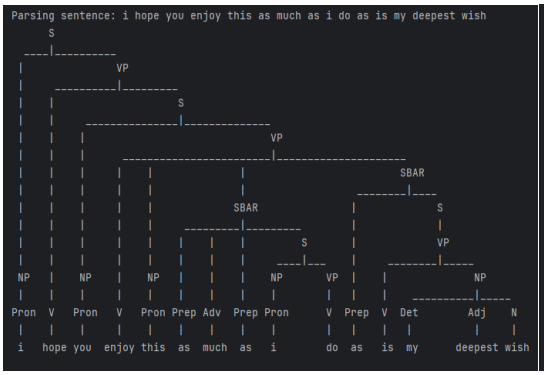
Προτάσεις 1 && 2 (από custom ανακατασκευή): Καλή ανακατασκευή, οι προτάσεις είναι σημασιολογικά και συντακτικά ορθές, χωρίς περιττή πληροφορία και χωρίς απώλεια πληροφορίας, καθώς ο αριθμός των tokens των νέων προτάσεων είναι ο ίδιος με τον αριθμό των tokens των αρχικών προτάσεων

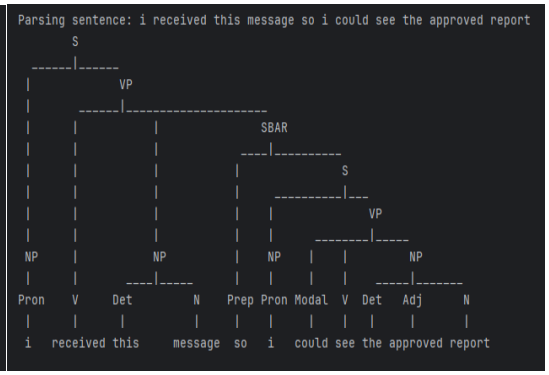


<u>Προτάσεις 3 && 4 (από ανακατασκευή του humarin/chatgpt_paraphraser_on_T5_base):</u> Καλή ανακατασκευή, οι προτάσεις είναι συντακτικά και σημασιολογικά ορθές, μικρή απόκλιση στον αριθμό των tokens σε σχέση με τις αρχικές προτάσεις (3 tokens για την 1η και 1 token για την 2η). Η παράφραση γίνεται με χρήση περισσότερων γραμματικών όρων σε σχέση με την ανακατασκευή του custom μοντέλου.

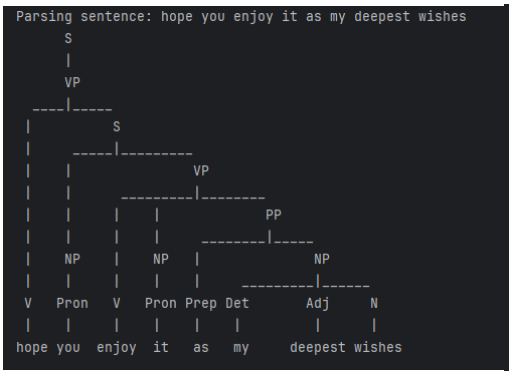


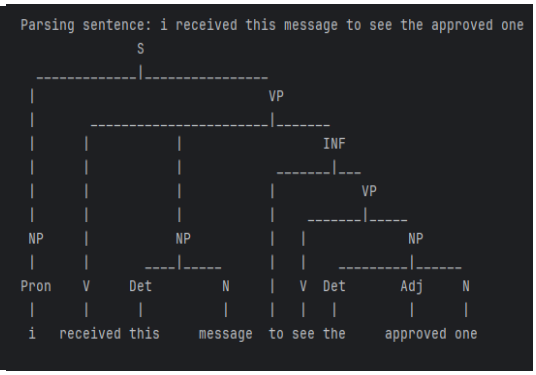
<u>Προτάσεις 5 && 6 (από ανακατασκευή του stanford-oval/paraphraser-bart-large):</u> Όχι τόσο καλή ανακατασκευή, οι προτάσεις είναι σημασιολογικά ορθές, αλλά δεν είναι συντακτικά ορθές. Μεγαλύτερη απόκλιση στον αριθμό των tokens σε σχέση με τις αρχικές προτάσεις (5 tokens για την 1η και 2 tokens για την 2η), σε αντίθεση με τα προηγούμενα μοντέλα. Η παράφραση του συγκεκριμένου μοντέλου προσθέτει περιττές λεπτομέρειες σε αντίθεση με τις προηγούμενες προσεγγίσεις(πχ. στην πρώτη πρόταση καθορίζει τον βαθμό απόλαυσης με την χρήση του ‘as much as’ σε αντίθεση με την αρχική εκδοχή που δεν διευκρινίζεται ο βαθμός της απόλαυσης του υποκειμένου ‘you’).





Προτάσεις 7 && 8 (από ανακατασκευή του tuner007/pegasus_paraphrase): Όχι τόσο καλή ανακατασκευή, οι προτάσεις είναι σημασιολογικά ορθές, αλλά η πρώτη πρόταση δεν είναι συντακτικά ορθή. Δεύτερη καλύτερη απόκλιση στον αριθμό των tokens σε σχέση με τις αρχικές προτάσεις (2 tokens για την $1^{η}$ και κανένα για την $2^{η}$). Υπάρχει απώλεια πληροφορίας στην παράφραση της πρώτης πρότασης, καθώς αφαιρείται το 'too' της αρχικής πρότασης που υποδήλωνε την συμμετοχή του υπονοούμενου υποκειμένου ' I ' .





Παραδοτέο 2

Ανάλυση Μεθόδων Ενσωμάτωσης

Προεπεξεργασία (συνάρτηση tokenize)

1. Κανονικοποίηση σε πεζά και αντικατάσταση «’» εισαγωγικών.
2. Αφαίρεση μη αλφαριθμητικών χαρακτήρων εκτός από το απόστροφο «'».
3. Διάσπαση σε tokens με whitespace.

Μοντέλα

Word2Vec

Το Word2Vec είναι μία τεχνική εκπροσώπησης λέξεων βασισμένη σε νευρωνικά δίκτυα που μαθαίνει συμμετρικές σχέσεις χρησιμοποιώντας τοπικά συμφραζόμενα για να προσεγγίσει τη σημασιολογική σχέση μεταξύ λέξεων. Το pretrained μοντέλο **word2vec-google-news-300** που χρησιμοποιούμε προέρχεται από μεγάλα συλλεγμένα corpora ειδησεογραφίας και web δεδομένων και περιέχει σταθερές στατιστικές συσχετίσεις που προέκυψαν από γενικά κείμενα. Οι

ενσωματώσεις από το Word2Vec χρησιμοποιούνται με μέσο όρο των ενσωματώσεων των μεμονωμένων λέξεων για να παράγουν ένα χαρακτηριστικό διάνυσμα για ολόκληρη τη φράση/κείμενο, έτσι ώστε να συγκριθούν τα αρχικά και τα ανακατασκευασμένα κείμενα στο ίδιο σημασιολογικό χώρο.

FastText

Το FastText επεκτείνει την ιδέα του Word2Vec ενσωματώνοντας πληροφορία από subwords. Η βασική ιδέα είναι ότι κάθε λέξη αναπαρίσταται ως άθροισμα ενσωματώσεων για τα n-gram της λέξης, κάτι που επιτρέπει την κατανομή πληροφορίας σε επίπεδο μορφολογίας και να παράγει χρήσιμες ενσωματώσεις για σπάνιες ή άγνωστες λέξεις. Το pretrained μοντέλο **fasttext-wiki-news-subwords-300** έχει εκπαιδευτεί σε ευρύ κείμενο και επωφελείται από αυτόν τον υπολεξικό χειρισμό, οπότε δίνει πιο ανθεκτικές μέσες ενσωματώσεις όταν υπάρχουν out of vocabulary tokens ή κλιτικές/παραλλαγές λέξεων που δεν καλύπτονται καλά από στατικά λεξικά. Στο συγκεκριμένο σύστημα, ο μέσος όρος των token-level ενσωματώσεων του FastText προσφέρει συνήθως καλύτερη συμπεριφορά σε ρητορικά και μορφολογικά διαφοροποιημένα κείμενα σε σχέση με το κλασικό Word2Vec.

GloVe

Το GloVe βασίζεται σε στατιστικά co-occurrence matrix και μαθαίνει στατικές ενσωματώσεις που κωδικοποιούν global στατιστικά συσχετίσεων μεταξύ λέξεων. Το **glove-wiki-gigaword-50** είναι ένα προεκπαιδευμένο μοντέλο που προέρχεται από μεγάλες συλλογές κειμένων και χρησιμοποιείται εδώ με την ίδια λογική όπως τα άλλα λεξιλογικά μοντέλα: υπολογίζεται ο μέσος όρος των ενσωματώσεων των λέξεων κάθε τεκμηρίου για να παραχθεί ένα διάνυσμα εγγράφου. Το πλεονέκτημα του GloVe είναι ότι αποτυπώνει ισχυρές, παγκόσμιες στατιστικές εξαρτήσεις· το μειονέκτημα είναι, όπως και στο Word2Vec, ότι πρόκειται για στατικές ενσωματώσεις και μπορεί να χάνει εκφάνσεις που εξαρτώνται ισχυρά από το context.

SBERT

Το SBERT είναι μια μέθοδος που παράγει ενσωματώσεις για ολόκληρες προτάσεις ή κείμενα, προσαρμόζοντας αρχιτεκτονικές τύπου Transformer ώστε τα output τους να είναι ικανοποιητικά για συγκρίσεις προτάσεων με χρήση απλών μέτρων. Το μοντέλο **all-MiniLM-L6-v2** που χρησιμοποιείται είναι προεκπαιδευμένο και fine-tuned για να δίνει ποιοτικά sentence embeddings αμέσως, χωρίς να απαιτείται επιπλέον pooling μετρήσεων πέρα από την κλήση encode. Στο pipeline σε αυτό το μοντέλο αντί να συγκεντρώνει στατικές λέξεις, παίρνει υπόψη του το συνολικό νόημα της πρότασης/κειμένου, και για αυτό συχνά δίνει πιο αξιόπιστες μετρικές ομοιότητας μεταξύ πλήρων προτάσεων.

Custom

Αρχικά κατασκευάζεται λεξιλόγιο από το corpus, δημιουργούνται ζεύγη target-context με παράθυρο συμφραζομένων 5 και εφαρμόζεται negative sampling με κατανομή unigram^0.75 για την επιλογή ψευδώς αρνητικών δειγμάτων. Τα embeddings εκπαιδεύονται με Adam optimization πάνω σε batches, και στο τέλος τα embeddings των λέξεων χρησιμοποιούνται για να υπολογιστεί ένας μέσος όρος ανά κείμενο ώστε να παραχθούν οι εκπροσωπήσεις εγγράφου. Το όφελος της προσέγγισης είναι ότι προσαρμόζεται πλήρως στο συγκεκριμένο data distribution και μπορεί να πιάσει ιδιαιτερότητες του dataset που τα γενικά dictionaries δεν αποτυπώνουν.

Μέτρηση Ομοιότητας

Η μέτρηση ομοιότητας γίνεται με το cosine similarity μεταξύ των ενσωματώσεων που αντιπροσωπεύουν την αρχική και την ανακατασκευασμένη εκδοχή κάθε κειμένου. Ο κώδικας την υπολογίζει για κάθε ζεύγος (πρωτότυπο, ανακατασκευή) για κάθε μέθοδο (SBERT, Word2Vec, FastText, GloVe και το custom μοντέλο) και συγκεντρώνει τα αποτελέσματα σε ένα DataFrame το οποίο αποθηκεύεται ως similarity_results.csv. Αυτή η κατανομή επιτρέπει άμεση σύγκριση του πόσο καλά κάθε μέθοδος διατηρεί το νόημα μετά την ανακατασκευή: τα sentence-level embeddings τείνουν να είναι πιο ευαίσθητα σε μικρές σημασιολογικές αλλαγές, ενώ οι μέσοι όροι λέξεων μπορεί να εξομαλύνουν διαφορές αλλά και να χάνουν λεπτομέρειες.

PCA / t-SNE

Για την οπτικοποίηση, πρώτα εφαρμόζεται κανονικοποίηση στα embeddings και στη συνέχεια μειώνονται οι διαστάσεις με PCA σε δύο διαστάσεις ώστε να παραχθεί ένα γραμμικό projection. Οι θέσεις των αρχικών και των ανακατασκευασμένων κειμένων στον χώρο απεικονίζονται με διαφορετικά σύμβολα και τα αντίστοιχα ζεύγη ενώνονται με βέλη προκειμένου να καταδειχθεί η «μετατόπιση» στον σημασιολογικό χώρο. Επιπλέον, εφαρμόζεται t-SNE για μη γραμμική μείωση διάστασης ώστε να αποκαλυφθούν τοπικές, μη γραμμικές σχέσεις που μπορεί να αποκρύπτονται από το PCA. Σημειώνεται ότι τα NaN αντικαθίστανται με μηδενικά και ελέγχεται αν όλα τα διανύσματα είναι μηδενικά πριν από την απεικόνιση, ώστε να αποτραπούν σφάλματα απεικόνισης.

Ανάλυση Αποτελεσμάτων

<u>Σημείωση:</u>
Ανακατασκευές A = Ανακατασκευές του humarin/chatgpt_paraphraser_on_T5_base
Ανακατασκευές B = Ανακατασκευές του stanford-oval/paraphraser-bart-large
Ανακατασκευές C = Ανακατασκευές του tuner007/pegasus_paraphrase

Ανακατασκευή [ ]_r1 = Ανακατασκευή κειμένου 1
Ανακατασκευή [ ]_r2 = Ανακατασκευή κειμένου 2

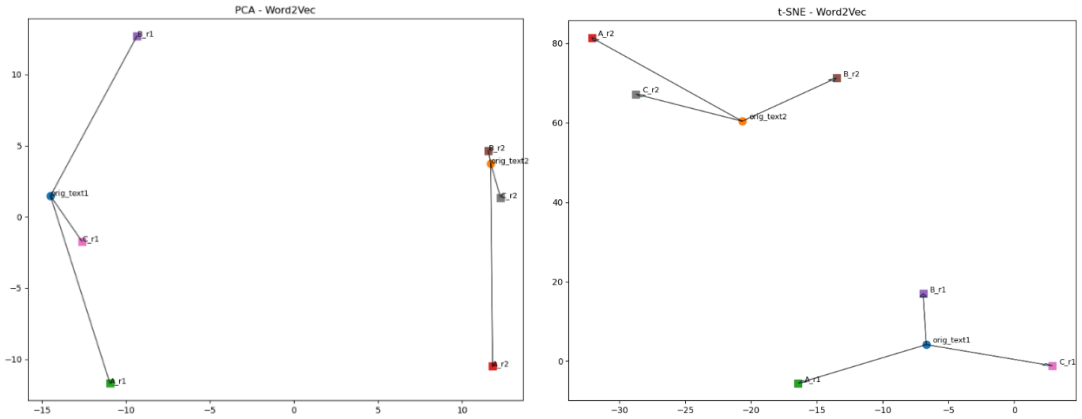
Μέτρηση Ομοιότητας

Βάσει των αποτελεσμάτων ομοιότητας, το GloVe ξεχωρίζει ως το πιο αποτελεσματικό μοντέλο για τη σύγκριση των αρχικών κειμένων με τις ανακατασκευασμένες εκδοχές τους, επιτυγχάνοντας συνεπώς υψηλές βαθμολογίες ομοιότητας. Το FastText ακολουθεί σε δεύτερη θέση με επίσης εξαιρετικές επιδόσεις, ενώ το Word2Vec και το Custom μοντέλο παρουσιάζουν ικανοποιητικές αλλά χαμηλότερες επιδόσεις. Το SBERT παρουσιάζει τις χαμηλότερες βαθμολογίες ομοιότητας μεταξύ όλων των μοντέλων, κάτι που μπορεί να υποδηλώνει ότι η προσέγγισή του στην κωδικοποίηση προτάσεων δεν αποτυπώνει με την ίδια ακρίβεια τις σημασιολογικές ομοιότητες που εντοπίζουν τα άλλα μοντέλα.

	orig	recon	sim_sbert	sim_w2v	sim_fasttext	sim_glove	sim_custom
1	orig_text1	A_r1	0.9551653861999512	0.9685013890266418	0.991058349609375	0.996619462966919	0.9404689073562622
2	orig_text2	A_r2	0.909416139125824	0.9639269113540649	0.9865648746490479	0.9949365854263306	0.9539297819137573
3	orig_text1	B_r1	0.9464226961135864	0.972284197807312	0.9799154996871948	0.9968956708908081	0.9804058074951172
4	orig_text2	B_r2	0.9353756904602051	0.9865474104881287	0.9950743913650513	0.9981726408004761	0.988091230392456
5	orig_text1	C_r1	0.9578958749771118	0.9801369905471802	0.9898777008056641	0.9981393814086914	0.9848912358283997
6	orig_text2	C_r2	0.9743707180023193	0.9949347972869873	0.998002290725708	0.9992904663085938	0.9936044216156006

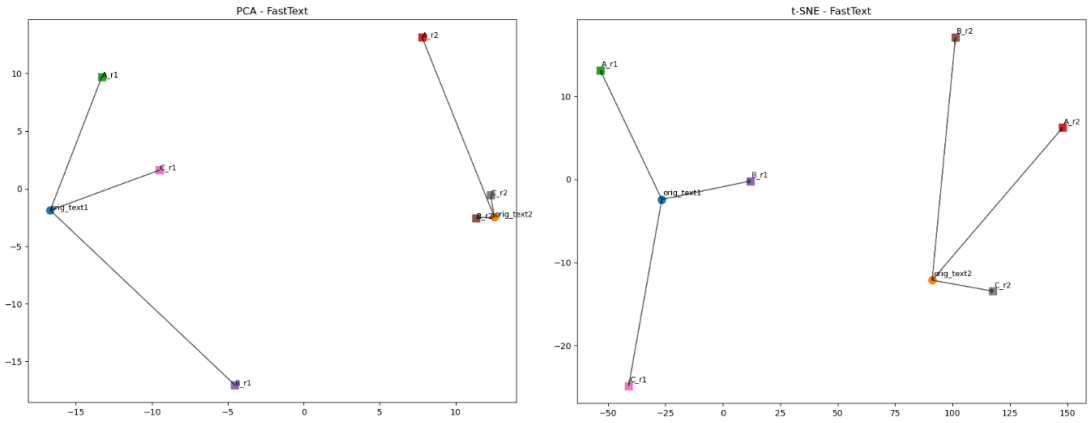
Word2Vec

Παρατηρούμε στο PCA ότι οι ανακατασκευές του orig_text1 κατέχουν μια πιο διαφοροποιημένη κατανομή από αυτές του orig_text2. Επίσης, το B_r1 και A_r2 εμφανίζονται σε σημαντική απόσταση από τις υπόλοιπες ανακατασκευές, υποδηλώνοντας διαφορετικές σημασιολογικές ιδιότητες. Το t-SNE αποκαλύπτει μια σχετικά πιο συμπαγή ομαδοποίηση των ανακατασκευών που προέρχονται από το orig_text2 ενώ οι ανακατασκευές από το orig_text1 εμφανίζονται πιο διασκορπισμένες.



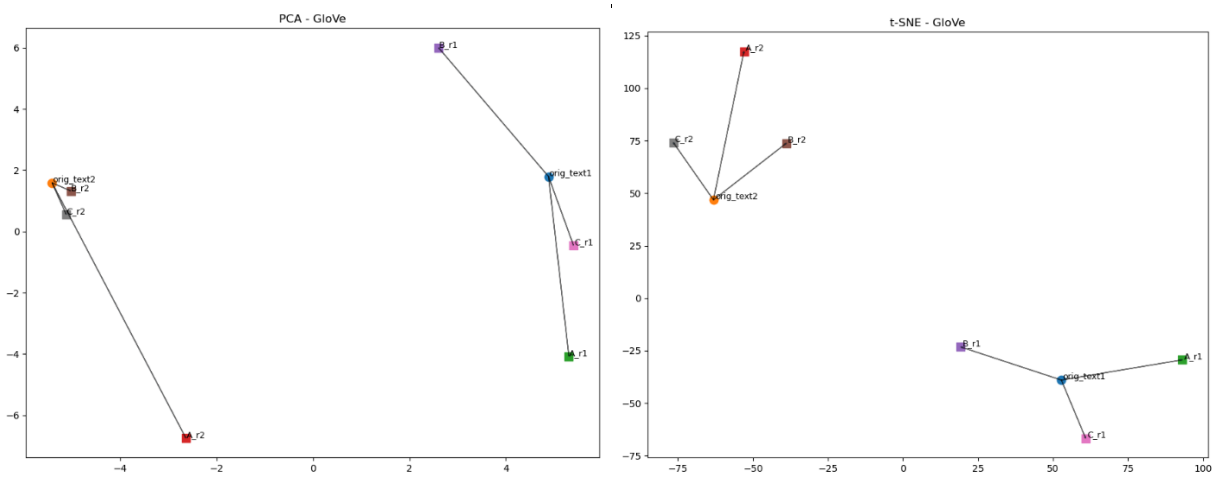
FastText

Το FastText παρουσιάζει παρόμοια αποτελέσματα με το Word2Vec στο PCA, με τις κατασκευές του orig_text2 να είναι ποιο συμπαγώς κατανεμημένες από αυτές του orig_text1. Επίσης, η B_r1 και η A_r2 εμφανίζονται ξανά σε σημαντική απόσταση από τις υπόλοιπες ανακατασκευές. Οι ομαδοποιήσεις στο t-SNE, όμως, παρουσιάζονται ποιο διασκορπισμένες, με αυτή την φορά η C_r1 και η B_r2 να έχουν τις σημαντικότερες διαφορές από τις υπόλοιπες ανακατασκευές.



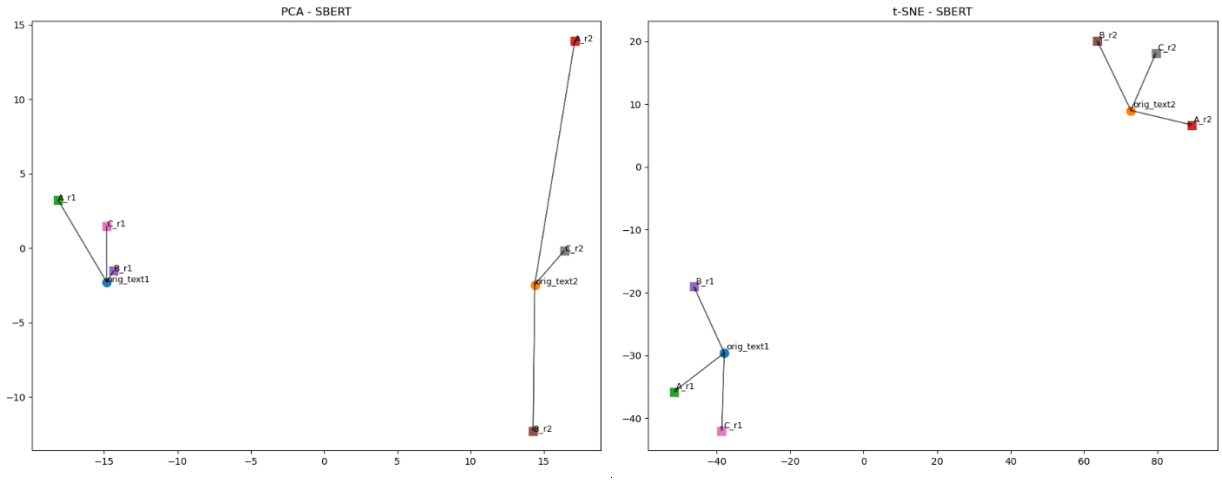
GloVe

Οι αποστάσεις ανάμεσα στα original και τα reconstructions στην PCA οπτικοποίηση του GloVe φαίνονται να είναι σχετικά μικρότερες. Όμως, και πάλι, οι κατασκευές του orig_text2 να είναι ποιο συμπαγώς κατανεμημένες από αυτές του orig_text1, και οι B_r1 και A_r2 εμφανίζονται ξανά σε σημαντική απόσταση από τις υπόλοιπες ανακατασκευές. Το t-SNE αποκαλύπτει μια πιο συμπαγή ομαδοποίηση των ανακατασκευών από το orig_text2, ενώ οι ανακατασκευές από το orig_text1 εμφανίζονται πιο κοντά στα αρχικά κείμενα, υποδεικνύοντας καλύτερη διατήρηση των σημασιολογικών χαρακτηριστικών.



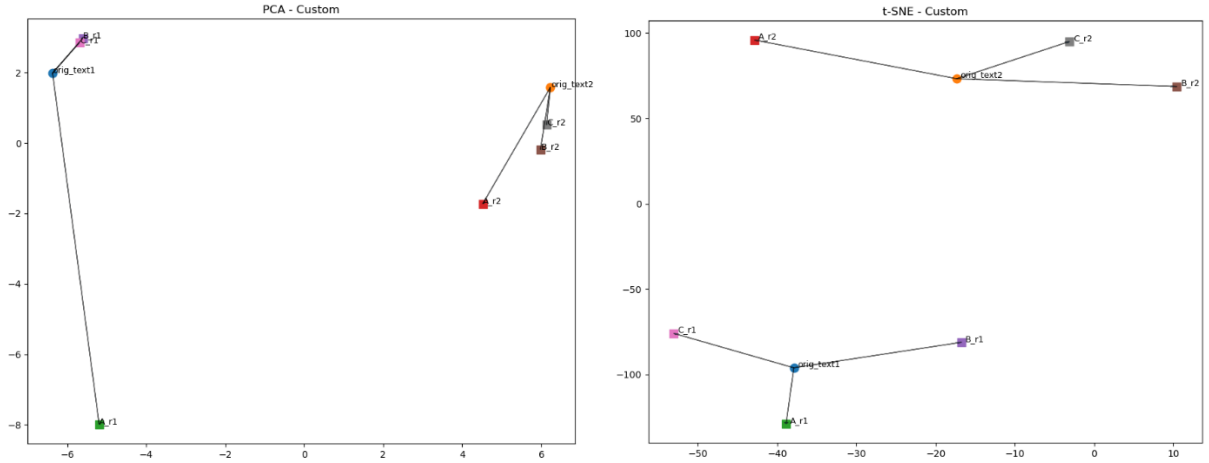
SBERT

Το SBERT δημιουργεί μια ιδιαίτερα διαφορετική διάταξη στο PCA, όπου οι ανακατασκευές του orig_text1 είναι ποιο συμπαγώς κατανεμημένες και απέχουν σημαντικά λιγότερο μεταξύ τους και από το original σε σύγκριση με το orig_text2. Το t-SNE παρουσιάζει ποιο κοντινές ομαδοποιήσεις από όλα τα υπόλοιπα t-SNE γραφήματα των υπόλοιπων μοντέλων.



Custom

Το PCA του custom μοντέλου παρουσιάζει μια αρκετά διαφορετική εικόνα, όπου οι ανακατασκευές B και C έχουν πολύ μικρές αποστάσεις μεταξύ τους και από τα original κείμενα τους, σε αντίθεση με τις ανακατασκευές του A, όπου απέχουν σημαντικά περισσότερο. Το t-SNE παρουσιάζει παρόμοια συμπεριφορά με το t-SNE του Word2Vec.



Βιβλιογραφία

1. https://huggingface.co/models?search=paraphrase
2. https://thales.cs.unipi.gr/modules/document/file.php/TMC113/Coding_notesNlp.pdf
3. https://huggingface.co/docs/optimum/main/en/intel/ipex/models#transformers
4. https://huggingface.co/docs/transformers/v4.56.0/en/model_doc/auto#natural-language-processing
5. https://www.geeksforgeeks.org/nlp/text2text-generations-using-huggingface-model/
6. https://github.com/dimitris1pana/nlp_lab_unipi/tree/main
7. https://www.geeksforgeeks.org/nlp/why-is-nlp-important/
8. https://www.geeksforgeeks.org/machine-learning/understanding-tf-idf-term-frequency-inverse-document-frequency/
9. https://www.geeksforgeeks.org/python/python-word-embedding-using-word2vec/\
10. https://www.geeksforgeeks.org/python/normalizing-textual-data-with-python/
11. https://www.geeksforgeeks.org/python/normalizing-textual-data-with-python/
12. https://www.geeksforgeeks.org/nlp/nlp-gensim-tutorial-complete-guide-for-beginners/
13. https://www.geeksforgeeks.org/python/overview-of-word-embedding-using-embeddings-from-language-models-elmo/
14. https://www.geeksforgeeks.org/machine-learning/seq2seq-model-in-machine-learning/
15. https://www.geeksforgeeks.org/nlp/Glove-Word-Embedding-in-NLP/