ECL Exercise 4

Topic: Shallow and Distributional Semantics

Submission Deadline: Wednesday, 22 November 2023, 18:00

Notes on the submission:

- **Submission format:** a pdf with your solutions and screenshots.
- Filename the pdf **using your OLAT names**, in the following format: *username1_username2_ecl1_exerciseNo.pdf*
- **Languages:** You can write either in German or in English.
- Please number the tasks on the submission sheet in the same way as on the task sheet.
- **Learning partnerships in pairs** are **required**. Only one person submits the assignment on OLAT.
- Please state your first and last names on the submission sheet.
- Submit your file via the exercise module on OLAT. The submission window closes on 22 November, at 18:00.
- Good luck and have fun!

1 Mandatory Reading with Quizzes

Read Chapter 6 "Vector Semantics and Embeddings" in Jurafsky & Martin on OLAT. You can skip subsections 6.6 and 6.7 and the 'Bibliographical and Historical Notes' at the end (although they are interesting, for the curious ones among you)[1]. Then, answer the following questions.

1. **Explain in your own words how relational meaning can be expressed through word embeddings (cf. parallelogram model).** Give at least two examples to illustrate your explanation.

2. **Which of the following statement(s) about the architecture of embeddings is/are true? (one or more correct answers possible).** Briefly explain why True/False.

   (a) For SGNS algorithms we tend to use more negative than positive examples.

   (b) For the tf-idf weighted vectors, a low idf means that the term is important and should be weighted more.

   (c) Sparse vectors, as they are created by tf-idf and PMI algorithms, are better at capturing synonymy and other semantic relations than dense vectors.

   (d) The size of the context window can have an influence on what type of information the embedding stores.

3. **Create your own multiple-choice question about the text.** Your question must have 3-4 possible answers of which at least one must be correct. Best submissions will be posted on the OLAT forum as additional pre-exam practice tasks.

Submission: Your answers to questions 1 and 2 and your multiple-choice question. Please indicate the correct answer(s) to your multiple-choice test.

2 Hands-on Word Embeddings

In the previous section you read about word embeddings. Now, you have the chance to apply your knowledge and deepen your understanding using an online tool. Go to this page[2] and work through the following steps:

1. (a) Select some short text with 80-100 words. You can use Word Counter[3] to speed up the calculation.

   (b) Put the text into the respective field and then generate the dataset.

   (c) Describe what you can see in the section "**View the input and output vectors to the model**". How does it relate to the "**Tokens**" and "**Training examples**" listed above?

   (d) Then, klick on `Train model` and afterwards on `Run t-SNE`. You do not need to change any of the parameters for either step (NB: If you refresh the page, the perplexity might change, make sure to always use the same value).

[1] If you are a visual learner and want to look at some more colourful graphics, also have a look at this blog: https://jalammar.github.io/illustrated-word2vec/

[2] https://remykarem.github.io/word2vec-demo/

[3] https://wordcounter.net/

(e) Take a screenshot of the outcome and describe what you see.

2. Work the process again using the same text as in 1., but this time changing the **window size** to five. Leave out steps (a) and (c).

3. Now, take a longer text of 180-250 words and repeat what you did in part 1. Leave out steps (a) and (c).

4. What are your observations with these models and visualizations? What was unexpected? What was exactly as you expected?

Submission: Your observations and screenshots to all variations. No need to write a long text, but make sure to answer all the questions and mention your observations.

3 Named Entity Recognition and Linking

In this section, you will gain a hands-on experience with manuad, automated and neural NER (Named Entity Recognition) and NEL (Named Entity Linking). Working with 2 most popular NLP libraries spaCy and NLTK, you will see how information extraction can be done with a few lines of code, without having to code yourself.

This part of the exercise involves working in a Jupyter Notebook. Since most of you take PCL1, you should already be familiar with its basic functionalities. For those of you completely new to working with Jupyter Notebooks, this is a good and easy opportunity to learn it. **Don't worry, you are not expected to code in this exercise. Your task is to follow the script made for you, observe, experiment, and document what you see.**

3.1 Manual Annotation

1. Go to your favourite online English-language newspaper and select any article you would like to explore, preferably from the domains of politics, economics, or public life (celebrity gossip is fine too).

2. Copy any passage of 200-300 words into e.g. your submission document. You can use Word Counter to avoid the pain of counting the words manually.[4]

3. Find all the instances of named entities and add the corresponding labels. For an example of how this can be annotated, take a look at Chapter 8.3 of Jurafsky and Martin: https://web.stanford.edu/~jurafsky/slp3/8.pdf

   Example: ”Goodbye sleepy [$_{TIME}$ **Friday**], hello [$_{ORG}$ **UZH**]”.

3.2 Annotation with NLP Tools

You have three options for working with Jupyter Notebook we have prepared. They are all equivalent, so pick whichever works best for you. We have included all the packages in the code for your convenience, so it should work without additional installations. The installation and

[4] https://wordcounter.net/

import cells might take several minutes to execute - this is normal, don't panic and go make some tea.

- **Option 1: Binder**. Follow this link to Simon's Binder. Once there, navigate to `ecl1-notebooks / ex04 / ICL1_ex04.jpynb` and open it. Follow the instructions in the notebook. **Note:** We are aware that the Binder sometimes takes long to load, sometimes it does not load at all, and sometimes the kernel dies in the middle of the process. Should that be the case, you have 2 back-up plans :)
- **Option 2: Google Colab**. For this, you will need to create a Google account. Once logged in, you can upload the notebook and start working with it online. Go to Google Colab, click on `open colab`, then `upload`, and upload the file. Eeeeasy.
- **Option 3: VSC**. Go to the Jupyter Notebook you have downloaded from OLAT (same zip as the exercise sheet). Open it with VSC Visual Studio Code. Follow the instructions.

For this part of the exercise, you should use the **same text** you have used for the manual annotation in the previous section. All the tasks you need to do are marked with a **TODO** sign. **NOTE** sign signals lines that need your attention.

1. Annotate your text with Spacy (ENG and GER).
2. Annotate your text with NLTK (ENG and GER).
3. Compare your own annotation, as well as annotations done by SpaCy and NLTK. What differences have you found? Do the systems use the same tags for the same types of entities? **Copy and paste the NER output from SpaCy and NLTK here.**
4. Analyse how the entities were linked to the Wikipedia database by SpaCy and NLTK. Any differences? Any surprises? **Copy and paste the NEL output from SpaCy and NLTK here.**
5. **Optional task (bonus):** In the Notebook, you will see sections for the annotation of German texts. Choose any different language[5] of your liking and try to modify the code so that it works with the language of your choice. We have left **NOTE** comments whenever you need to be attentive. Remember to test it on a text in that language. Explore and enjoy!

Submission: Your answers to questions 3 and 4, including screenshots, comments + saved Jupyter Notebook with executed cells. If you are doing the bonus task with a different language, briefly comment on what you observed (2-5 sentences)

[5] `https://spacy.io/usage/models - find here the full list of available languages and instructions.`

3.3 Annotation with Neural Models

1. Ask ChatGPT to annotate the Named Entities in your text.[6]

2. Ask Google Bard to do the same.[7]

Submission: Briefly analyse the results and note differences (if any). Include screenshots with your prompts and the relevant chat output.

3.4 Bonus Task: Real-life Applications

This is an optional task. You can use it to make extra points to compensate for errors in other parts of the submission. Also, it happens to be a really nice read offering perspectives on your professional future.

Read the article **How the Guardian approaches quote extraction with NLP** (https://explosion.ai/blog/guardian) and answer the following questions:

1. What is the relation between NER (Named Entity Recognition), NEL (Named Entity Linking) and the Guardian's quest to extract quotations?

2. What is Prodigy and what do they use it for?

Submission: Brief answers to the questions (2-4 sentences each).

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

[7] https://bard.google.com/chat

4 Feedback

How was it?

- What did you like most about the assignment?
- What did you dislike most about it?
- What was the most useful part?
- Which exercise was the most irritating?
- Do you have any suggestions for improvements?