КУРСОВ ПРОЕКТ

Анотиране на понятия в текст чрез Укипедия

**Факултет по Математика и Информатика**  
**Студент:** Борислав Стоянов Марков  
**Факултетен номер:** **0MI3400048**  
**Учебен план:** **Изкуствен Интелект (редовно, магистър)**  
Курс: **Курс 1**; Група: **Група 1**  
**Активен период**: 2021/2022 летен, магистри  
**Дисциплина**: Откриване на знания в текст

Дата: 28.06.2022г.

# 1. Съдържание

[1. Съдържание 2](#_Toc106298978)

[2. Увод 2](#_Toc106298979)

[3. Средата „Акробот“[3] в Gym 2](#_Toc106298980)

[4. Алгоритъм Актьор-критика 4](#_Toc106298981)

[5. Реализация на проекта 6](#_Toc106298982)

[5.1 Анализ на резултатите 7](#_Toc106298983)

[6. Идеи за бъдещо развитие и подобрения 9](#_Toc106298984)

[7. Източници и използвана литература 9](#_Toc106298985)

[Приложения 9](#_Toc106298986)

[1. Сорс код (Source code) 9](#_Toc106298987)

# 2. Резюме

В този проект съм се опитал да открия понятия в текст само с един пас през текста, определяйки дължината на споменаването автоматично. За анотиран текст съм използвал част от Уикипедия на български език.

# 3. Въведение

В деншно време особено важно е да се анотира текста на новини, за да може да се търси по семантични връзки. За целта имайки за вход един текст, ние искаме да разберем какви споменавания има на обекти от реалния свят, да анотираме текста. Могат да се поставят и линкове съответно към Уикипедия. Анотирането искаме да стане на един пас, без POS тагиране и разпознаване части на речта а само на базата на близост между вектори. Тъй като анотирането е скъпа операция, Уикипедия е чуесен източник на безплатни ръчно анотирани данни.

За вход имаме текст на български език, на изхода имаме анотиран текст или разпознати словосъчетания и към какво сочат в българската Уикипедия. Разпознаването става по следния начин: първо за всяка дума се определя контекст (5 думи напред и 5 думи назад). Изваденият контекст се векторизира с модел на трансформър невронна мрежа и векторизация с HuggingFace, който поддържа бъгарски. След това вектора се подава на невронна мрежа от 3 линейни слоя и Softmax класификатор на изхода. Мрежата е предварително обучена за този проект и на изхода се получава само един клас, отговарящ на дадена страница от Уикипедия.

Обучението на данните е по идея на статия от Гугъл[1], но доста неща не беше ясно как са направени и съответно за имплементация реших да сложа по-лесно и реалистично решение. Избраното решение е само демонстративно и се нуждае от подобрения. В оригиналната статия имаме косинус близост, а тук съм заложил невронна мрежа, която ни дава съответствието на M -> E, където М е споменаването(mention), а Е е обекта от Уикипедия (Entity). Съответно разчитам, че невронната мрежа ще генерализира и подобни споменавания ще сочат към подобни обекти.

# 4. Преглед на областта

Последните работи в областта са фокусирани върху трениране на невронни мрежи кандидати и определяне на ранк (Francis-Landau et al., 2016; Eshel et al., 2017; Yamada et al., 2017a; Gupta et al., 2017; Silet al., 2018). Като цяло тези работи изследват полезни контекстни фийчъри и използват нови подходи и архитектури, комбинирайки страната на споменаването и страната на обекта. Разширенията включват съвместно разпознаване на всички обекти в документ (Ratinov et al., 2011; Globerson et al., 2016; Ganea and

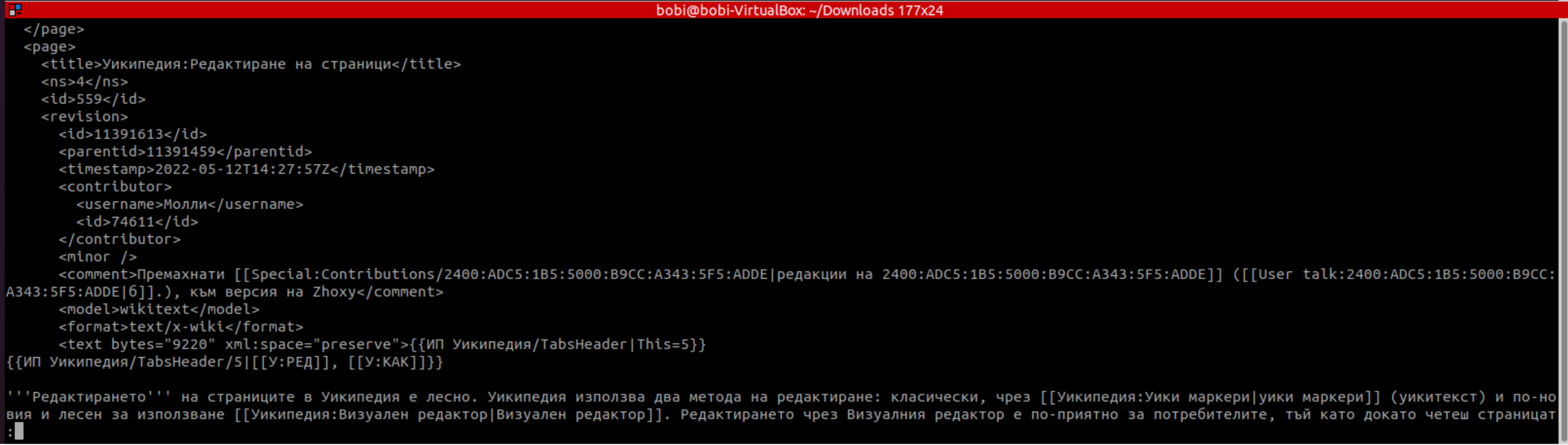
Hofmann, 2017), съвместно моделиране с релативни задачи като подобие на текст (Yamada et al., 2017b; Barrena et al.,2018) и междуезиково моделиране (Sil et al., 2018). Също в областта има и разработки с Reinforcement Learning, моделирайки Reinforcement агента като невронна мрежа с вход от контекста на споменаване, кандидат обекти и предишни решения (Joint Entity Linking with Deep Reinforcement Learning, Zheng Fang et al. 2019).

# 5. Данни и характеристики

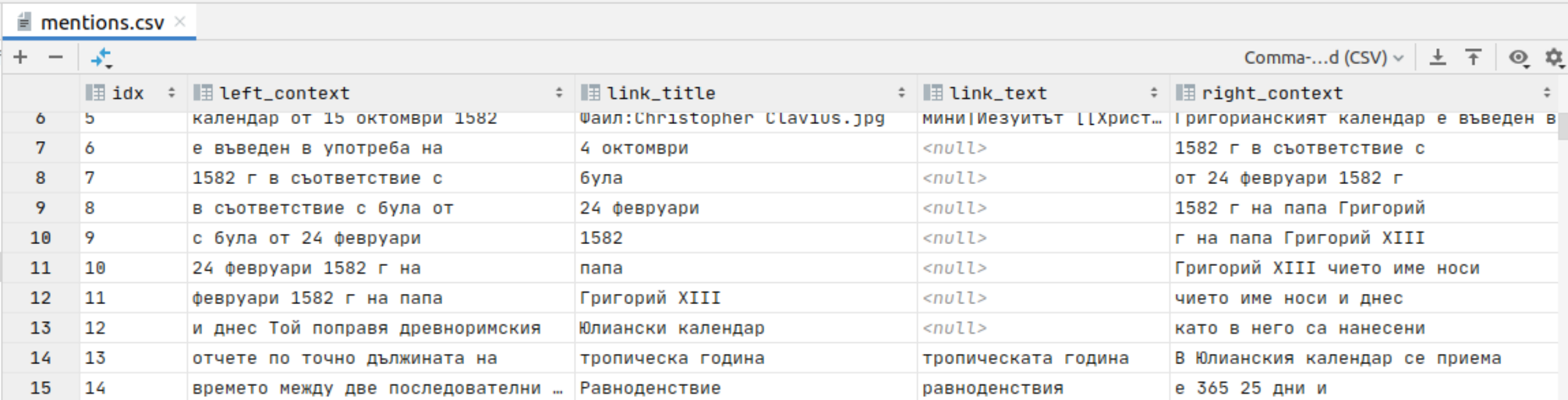
Уикипедия е чудесен източник на анотирани данни, защото текста е изпълнен с вътрешни препратки и споменаванията са всъщност текста на връзката. Използвал съм част от българската уикипедия, свалена от <https://dumps.wikimedia.org/bgwiki/20220620/> . Архивът е **bgwiki-20220620-pages-articles.xml.bz2** 373.8 МБ. След разархивиране е около 3.5 ГБ. Като оправна точка за парсване на статиите съм ползвал собствен скрипт, който има заимствани методи от <https://github.com/google-research/google-research/tree/master/dense_representations_for_entity_retrieval/> . Идеята на скрипта „parse\_wiki.py“ е да обходи статиите и за всяка страница от Уикипедия да намери споменаванията, а страницата да запамети като обект. Споменаванията са ограничени до 20 на страница. Извикването на скрипта става по следният начин:

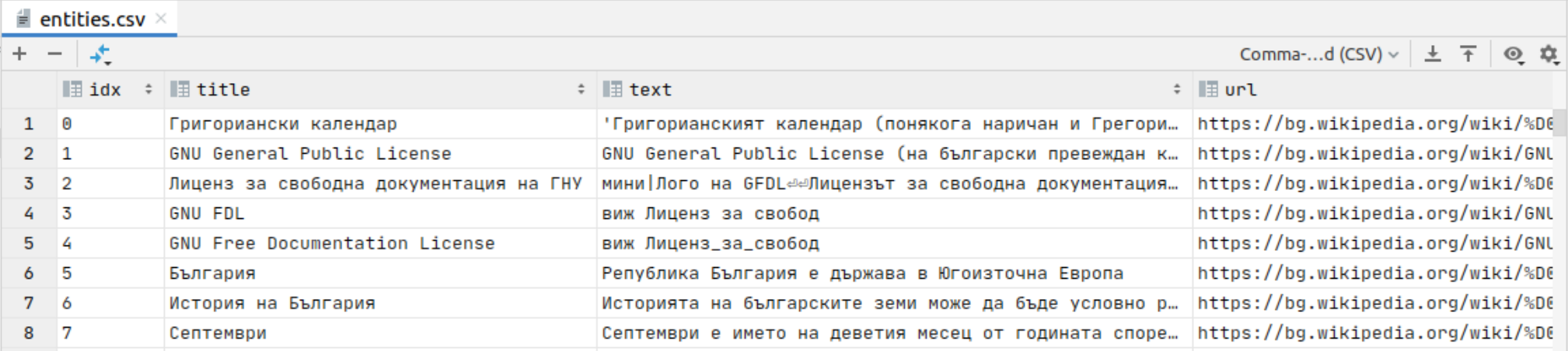
|  |
| --- |
| python parse\_wiki.py --bgwiki\_archive=/home/bobi/Downloads/bgwiki-20220620-pages-articles.xml --max\_pages=1000 --limit\_mentions\_per\_page=20 |

Ето и суровият вид на файла на Уикипедия:



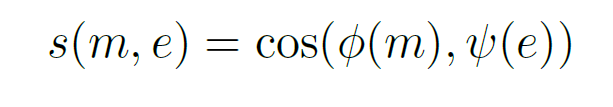
Скриптът произвежда 2 csv файла, съответно **mentions.csv** и **entities.csv**.

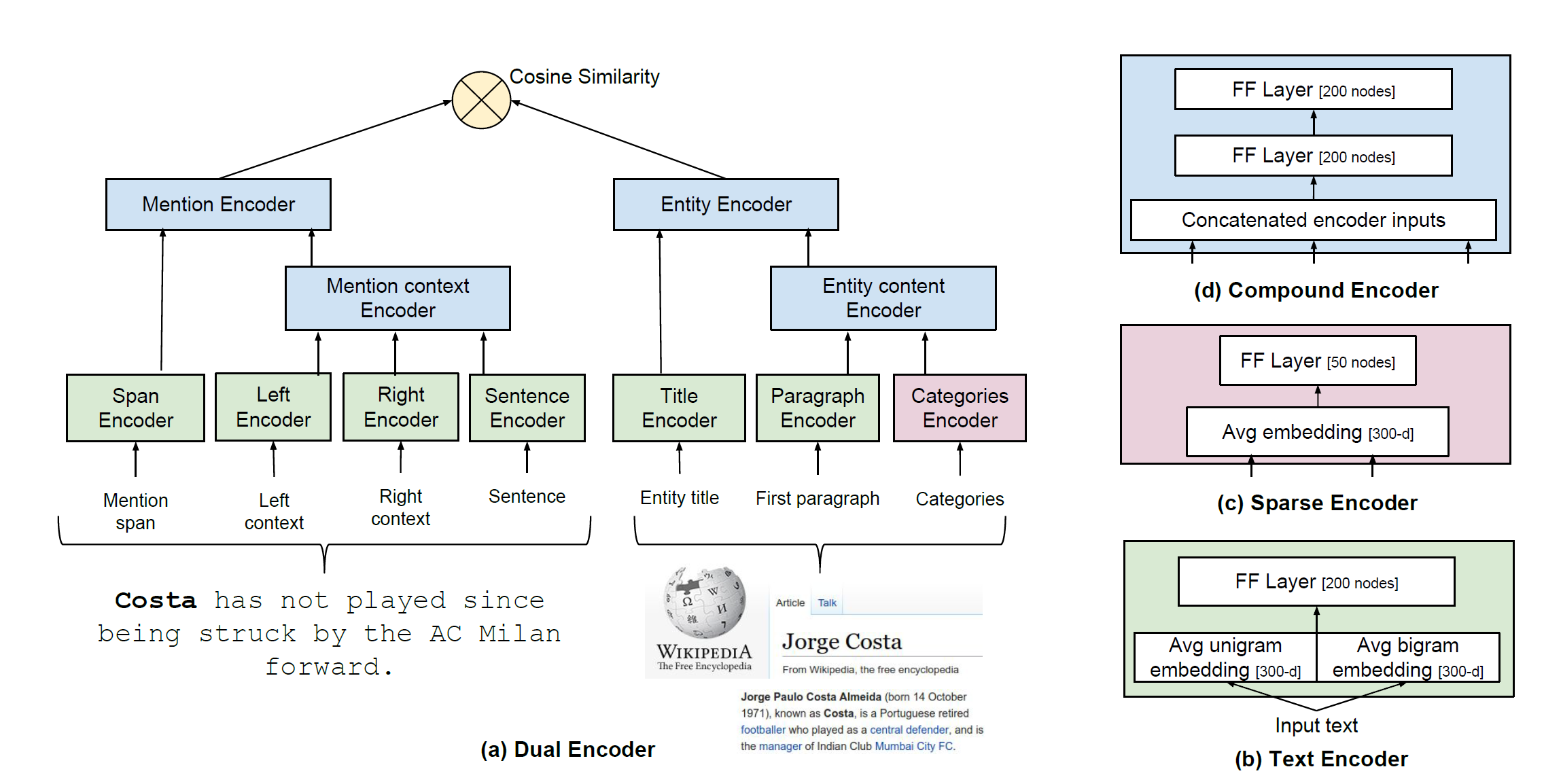




# 6. Методи

Първоначалната идеа на проекта е според статия [1] използваща косинус близост между кодирано споменаване и кодиран обект. Двойният енкодер научава кодирането на споменаването φ и научава кодиране на обекта ψ. След това смята косинус близост.



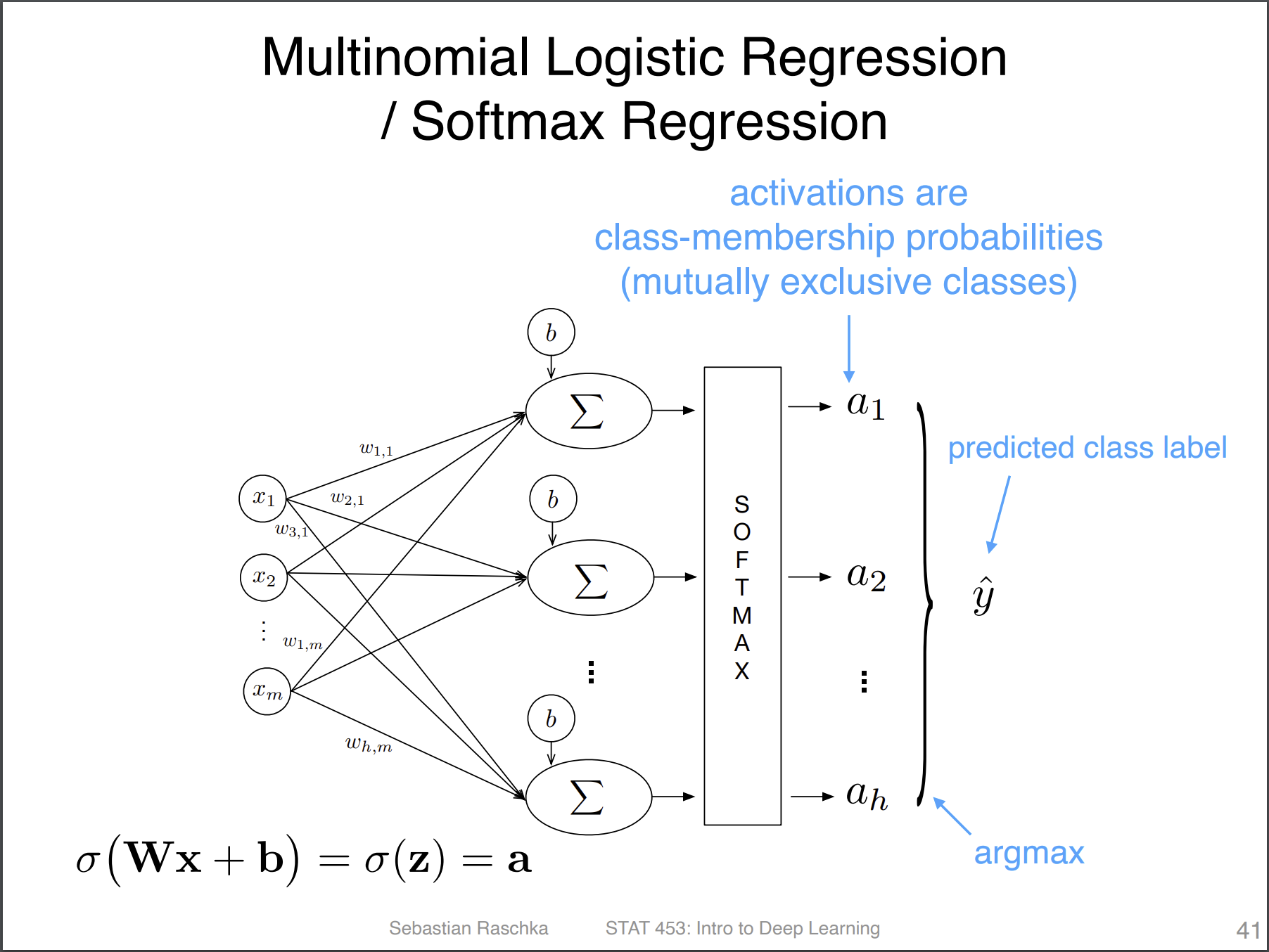


Поради недостатъчно обяснени данни и обяснения как работи оригиналният алгоритъм, реших да подменя някои компоненти и да направя собствена имплементация използвайки невронна мрежа и готово кодиране на споменаването „m“ (mention) и обекта „е“(entity). Кодирането на вектори по даден текст правим с готови научени модели на трансформъри, поддържащи български език. Можете да намерите пример на адрес <https://huggingface.co/sentence-transformers/all-MiniLM-L12-v1> .

|  |
| --- |
| from sentence\_transformers import SentenceTransformer  sentences = ["This is an example sentence", "Each sentence is converted"]  model = SentenceTransformer('sentence-transformers/all-MiniLM-L12-v1')  embeddings = model.encode(sentences)  print(embeddings) |

И така задачата за анотиране на текст се свежда до следното. По зададено „m“, което включва контекст по 5 думи от ляво и 5 думи от дясно и самият текст на връзката, ние да намерим такава функция „f“, която да ни дава съответното „е“. Векторизацията на „m“ ще означим „hugging\_face(.)“ .По-формално имаме:

Невронната мрежа на изхода има функция на активация softmax(.) , която ни осигурява избор на само един клас от много възможни. Можете да научите повече от слайдовете на Себастиан Рашка [4]



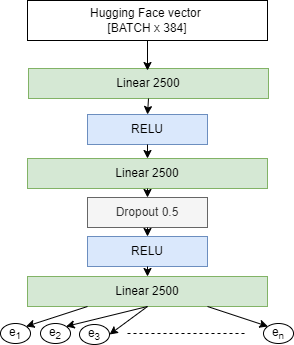
Sebastian Raschka, STAT 453: Intro to Deep Learning [4]

Softmax активационна функция:

|  |
| --- |
|  |

Sebastian Raschka, STAT 453: Intro to Deep Learning [4]

Даваме структура на невронна мрежа, която ще ни представлява функцията на следващата фигура.



# 7 . Експерименти/Резултати/Дискусия

# 8 . Заключение

# 8. Библиография

[1] Learning Dense Representations for Entity Retrieval, Daniel Gillick et. al., 2019 [PDF]

<https://aclanthology.org/K19-1049.pdf?fbclid=IwAR0vPi8AQrc5rPeAF2sLjZ9lg5K5WUTSwmhFPuxx436kmi-SN0RZ8OtdLLo>

[2] End-to-End Retrieval in Continuous Space, Daniel Gillick et al. 2018

[3] Deep Learning for Coders with fastai & PyTorch, Jeremy Howard & Sylvain Gugger, O’Reilly, 2020

[3] Pytorch documentation, <https://pytorch.org/docs/stable/torch.html>

[4] Logistic Regression and Multi-class classification, Lecture 08, Sebastian Raschka, <https://sebastianraschka.com/pdf/lecture-notes/stat453ss21/L08_logistic__slides.pdf>

# Приложения

## 1. Сорс код (Source code)

<https://github.com/borkox/uni-sofia-nlp-wiki-entity-linking>