Report on assignment 3

Name: Roman

Surname: Voronov

Email: ro.voronov@innopolis.university

Nickname on codalab: romashka

Github: link

Solutions

Solution using Spacy

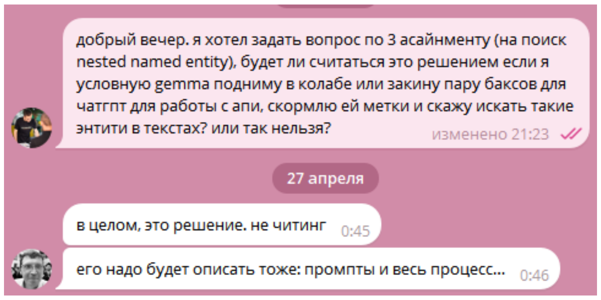
Spacy model of language consists of many components. One of them is **ner** that is responsible for named entity recognition. Its problem is that this component cannot deal with **nested** named entities. Therefore, I tried to train a custom pipeline **spancat** to retrieve relevant spans up to **5** words from the text and categorize them. The prediction is taken into account if the confidence is very high (**>0.85**)

This solution did not lead to good results. The result on dev set was almost **0** for every model I trained. I used different learning rates and different number of iterations.

Solution using ChatGPT

Here I decided to check, can **ChatGPT** solve the problem or not, without seeing the train set, or not.

The first problem to overcome was how to use ChatGPT from Russia if you do not have credit card from not Russia. The solution is to use **Proxy API service**.



добрый вечер. я хотел задать вопрос по 3 асайнменту (на поиск nested named entity), будет ли считаться это решением если я условную gemma подниму в колабе или закину пару баксов для чатгпт для работы с апи, скормлю ей метки и скажу искать такие энтити в текстах? или так нельзя? изменено 21:23

27 апреля

в целом, это решение. не читинг 0:45

его надо будет описать тоже: промпты и весь процесс... 0:46

This service allows using **rubles** to pay for API usage. It does not use the account on OpenAI or something like this. Moreover, it even does not require using additional libraries when writing code. You need to just use other endpoint when using official OpenAI API. Other staff is the same entirely.

Now we have access to API of **gpt-3.5-turbo**. To use the model, the prompt is needed. I send the description of the task as the context. The context is the following:

> *Пользователь даст текст. Тебе нужно найти все сущности в нем. Список сущностей на английском: AGE, AWARD, CITY, COUNTRY, CRIME, DATE, DISEASE, DISTRICT, EVENT, FACILITY, FAMILY, IDEOLOGY, LANGUAGE, LAW, LOCATION, MONEY, NATIONALITY, NUMBER, ORDINAL, ORGANIZATION, PENALTY, PERCENT, PERSON, PRODUCT, PROFESSION, RELIGION, STATE_OR_PROVINCE, TIME, WORK_OF_ART. Тебе нужно ответить без лишних комментариев в следующей форме: [["Газпром", "ORGANIZATION"], ["Роман", "PERSON"]]. Например, для текста "Я работаю учителем", тебе нужно ответить [["учителем", "PROFESSION"]], потому что в тексте есть упоминание профессии.. Пиши как было в тексте, не меняй падеж и прочее!*

This context specifies the task of **NNER**, lists the entities available, describes the output format and provides the example. The output of the model is the list of entities, where the model specifies the entity itself and its type. For example, **[”Roman”, “PERSON”]**.

As the user input, the text is passed.

Then, for every entity the model found, we search for them in the text to determine indices of text where this entity appears. Since the model saves the case, we can do such action. So, for **[”Roman”, “PERSON”]**, we look for **“Roman”** in the text and determine that my name appears from **5 to 9**, and save the new record as **[5, 9, “PERSON”]**.

Usually, the output of the model requires some post-processing. Sometimes it includes new entity classes, puts line separators, and so on, but such problems are easily fixable.

The best score I could achieve is **0.25**. It perfectly finds persons, locations and other staff, but obviously struggles with more complicated things like **"EVENT"** that should be trained.

Additionally, I tried to ask the model to search for entities directly with their location. However, the model could not determine the location of entities it finds. And the score of such solution was **0.**

Finally, I added an element of training in context. I gave some examples of entities from one text in train set to show how **EVENT, DATE, IDEOLOGY** and other entities can look like. This raised the score to **0.26** 👍

So, the best solution is the use of ChatGPT to find entities with sending a part of training set that contains important entities as the context that gives f1=0.26

Note on code

The folder with the second solution with ChatGPT contains two files.

The first file is the **NNER_model** with function **extract_entities.** Anyone who want to use the same functionality on his/her own text can use this function for this.

The second file is the **.ipynb notebook** with interactive step-by-step solution with explanations. Also, notebook provides the example of using the function from **NNER_model**