Ingredio Application

User Manual

May 2021

Contents

1 Introduction

Ingredio application is a natural processing language (NLP) application that offers a pipeline of 3 separate NLP services related to biomedical text. The application heavily relies on machine learning models and algorithms trained on different datasets for each specific task that stem from a corpus of abstract from biomedical papers. The application is able to classify biomedical text based on certain features of its content, extract compound names and infer causal relations from the text, however it is experimental and is not meant to replace human curation. It's main use is to showcase how this can be used as a high-throughput and high precision language filtering software for large scale biomedical data. The codebase of the application is found here. The machine learning task that we are trying to tackle falls into the category of language inference.

2 General Usage

While each stage can be used independently, the application however facilitates the sequential usage in its three stages (1. Classification, 2. Entity Extraction and 3. Causality Inference). Each stage involves the input of some text and running through the respective stage in the application back-end, if the query is successful and bears results, the input is forwarded to the next stage filling all the required information for the submission of the next stage.

3 Classification

3.1 development

- Tokenization. Tokenizing separates text into units such as sentences or words. It gives structure to previously unstructured text. eg: Dioxine can be dangerous--¿ 'Dioxine','can','be','dangerous'().
- Preprocessing Data: Lemmatizing. Lemmatizing derives the canonical form ('lemma') of a word. i.e the root form. It is better than stemming as it uses a dictionary--based approach i.e a morphological analysis to the root word.eg: Entitling, Entitled--¿Entitle. In Short, Stemming is typically faster asit simply chops off the end of the word, without understanding the context of the word. Lemmatizing is slower and more accurate as it takes an informed analysis with the context of the word in mind.
- Splitting the dataset: Using the Scikit--learn train_test_split method we split the dataset to 70% astraining setwhich was used to train the machine learning algorithm, 20% as validation set which was used for hyperparameter tuning and 10% as test set in order to evaluate the trained model.

- Vectorizing Data. Vectorizing is the process of encoding text as integers i.e. numeric form to create feature vectors so that machine learning algorithms can understand our data.Here, we used the "Bag--Of--Words" approach TF--IDF Vectorizer. It computes "relative frequency" that a word appears in a document compared to its frequency across all documents. It is more useful than "term frequency" for identifying "important" words in each document (high frequency in that document, low frequency in other documents).This approach is very useful for document clustering suchas the present project.

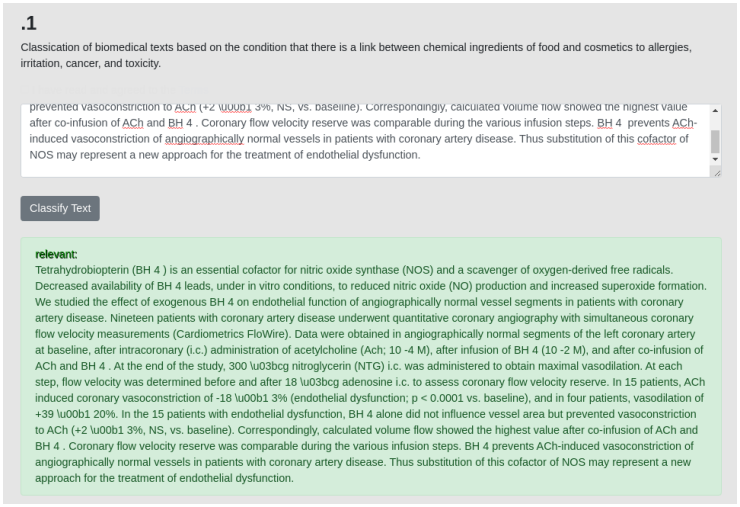
- Remove stopwords. Stopwords are common words that will likely appear in any text. They don't tell us much about our data so we remove them. e.g: heavy metals aredangerous in food --¿ heavy, metals, dangerous, food.

- Vectorizing Data: N--Grams. N--grams are simply all combinations of adjacent words or letters of length n that we can find in our source text. Ngrams with n=1 are called unigrams. Similarly, bigrams (n=2), trigrams (n=3) and so on can also be used.Unigrams usually don't contain much information as compared to bigrams and trigrams. The basic principle behind n--grams is that they

3.2 Usage

The classification stage of the application employs different machine learning models that were trained independently with the aim to be able to classify biomedical text according to its relevance with toxicity of compounds found in foods and cosmetics. This stage is based on the combining four different ML algorithms [1, 2, 3, 4] to reach a consensus regarding the classification of the text. The data used to train the model for this task, are compiled using the PubChem IDs from the Ingredio dataset to find papers from PubMed, that at the same time are associated to a compound and are categorized by PubChem as related to Foods and/or Cosmetics and other relevant categories using api requests to the PubChem database. The PubMed IDs of the papers are then collected and are used to download the respective papers.

The user can input the query text into the 1st stage text area input field and press the submit button. Upon submission, two possible outcomes can occur, either the text is classified as relevant or as irrelevant.



4 Entity Extraction

4.1 Info & Development

The entity extraction stage, is able to find names of chemical compounds in biomedical text, classifying stretches of characters as entities that denote compound names. this stage is based on the bert model [5, 6, 7]. the user again similarly with the first stage inputs its text to the text area input field and the model. Named-entity recognition (NER) or entity extraction is a subtask of information extraction that seeks to locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations etc. The BERT model (Bidirectional Encoder Representation from Transformers), is a neural network built for natural language processing (NLP) tasks. BERT is pre-trained in an unsupervised manner on a large corpus of unlabeled texts. BERT learns bidirectional representations, by jointly conditioning on both left and right context. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create a state of the art model. In our case a biomedical pre-trained BERT model is used, and fine-tuned for our dataset and task. BERT is based on the transformer model architecture, instead of LSTMS. Attention mechanism is applied to understand relationships between all words in a sentence, regardless of their respective position. BERT is a language model which is bidirectionally trained (this is also its key technical innovation). This means we can now have a deeper sense of language context and flow compared to the single-direction language models. A machine learning model was constructed and trained that takes as input a positively classified article from the OpenAIRE data and outputs best candidate words that represent chemical compounds. To implement this, Bidirectional Encoder Representations from Transformers (BERT) was used. To fine-tune the

BERT model a set of peer-reviewed articles was created, that contain chemical ingredients and annotated with part of speech tags. This task focuses on applying named entity recognition to the text. Specifically, a machine learning model was trained in a way that it is able to understand the context of a sentence and extract the compound names existing in it. After the completion of the training phase the model receives as input the documents classified in Objective 1 and outputs compound names found in those documents.

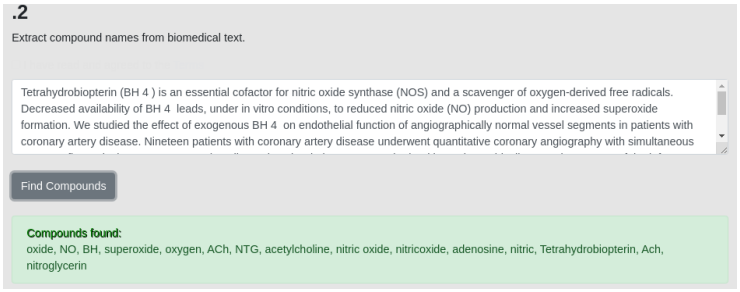
To train the model for this section, we augmented our corpus with part of speech (POS) annotation. Each word in the corpus is assigned a POS tag, as well as a vector denoting the position of the compound names in the text. As a next step we vectorize the text itself, ending with a dataset that for each sample (text) contains: the encoded text, POS tag vectors, and compound vectors. Our goal is translated to predicting the positions of the compound names in the texts given the first two vectors.

4.2 Usage

if compound names are identified:

- the compounds are shown in the output below the input field.
- the text is fed into the next stage text area input field.
- the compounds names are fed into the compounds field of the next stage.

if no compound is found, a message that says *no compounds found* is shown.



5 Causality Inference

5.1 Info & Development

The causality inference stage is able to determine if compounds found in a biomedical text cause any adverse effect defined by the user. To address the issue of inferring the compound toxicity, we transformed the problem to a classification one. We built an auxiliary dataset that contains biomedical sentences that are labeled positively when a sentence is of causal nature i.e. *The -compound A - is associated to adverse effect B*. The dataset is attempted to include various

forms that this transitive effect appears in English. As a next step we trained and validated an ML model on this dataset. We can presume that if for a compound exists in multiple sentences that are positively classified as causal, while containing the name of the adverse effect in question we can be almost certain of the positive causal relation that the compound has on the effect, due to the exponentially diminishing probabilities of not being the case for each occurrence in the corpus. This stage is also based on the bert model.

5.2 Usage

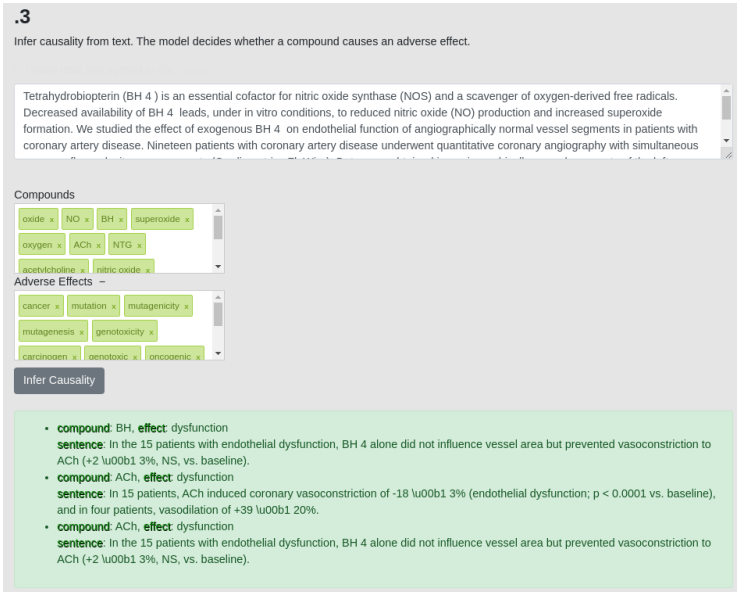
There are three inputs in this stage:

- the input text.
- a list of compound names.
- a list of adverse effects.

the list of compound names can be filled automatically by running the second stage, while the list of adverse effects is filled with some sensible defaults. the adverse effects can be expanded or removed by the user. the output of this stage is of the form :

- compound: ach, effect: dysfunction
  sentence: in 15 patients, ach induced coronary vasoconstriction of $-18 \pm 3\%$ (endothelial dysfunction; p ¡ 0.0001 vs. baseline), and in four patients, vasodilation of $+39 \pm 20\%$.

the list of compound names can be filled automatically by running the second stage, while the list of adverse effects is filled with defaults. the adverse effects can be expanded or removed by the user, by pressing the + button.



6 Bibliography

References

[1] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

[2] McCullagh, Peter, and John A. Nelder.(1989). Generalized linear models. Vol. 37. CRC press.

[3] Ruder, S. (2016). An overview of gradient descent optimization algorithms. ArXiv Preprint ArXiv:1609.04747.

[4] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297.

[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805

[6] Google AI Blog

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N. & Polosukhin, I. (2017). Attention is all you need. arXiv 2017. arXiv preprint arXiv:1706.03762.