Similar Offer Match via Text Input

Efficient solution to a user-end offer search

Anthony Jiang

Oct. 2023

Executive Summary

Highlights

- **Fast Respond**: normal response time 1~2 seconds
- **Accurate Entity Extraction**: spaCy-transformer NER model with 91.5 F score, understanding

Major Insights

- **Basic goal** of this project is to analyze what are the searched category/brand/retailer and what are the most similar offers
- **Some important assumptions**:
  - users are entering keywords or short sentence (by the nature of searching)
  - search input might not be crystal clear due to spelling errors or ambiguous wording (e.g. coke for Pepsi or Coca-cola)
  - use product design and avoid using algorithm to solve everything.When failing to find relevant offers, don't calculate similarity of search input to all available offers; rather, provide guide information for users to refine input
- **Deployment***: App is deployed on Hugging Face. For diagnostic reason, users now can chose similarity score threshold. In future, it can be replaced by user-friendly filtering options
- **Future Improvement**: fuzzy search to handle spelling errors in search input, more drop-down options in search menu to either narrow down search category or recommend keywords of relevant category/offers
- **Scalability**: as mentioned above, the current solution is lightweight and flexible to deploy on production pipeline; a fuzzy search component can be incorporated as well; to large-scale data (e.g. million-level search), you might consider Apache Spark, Parallelization, and Elasticsearch

Notes: *Github Repo at https://github.com/hjiangAnthony/retailer_nlp_challenger/tree/main and Hugging Face Space at https://huggingface.co/spaces/hjianganthony/fetch_ner

Problem Statement

User case: When a user searches for offers about a category/brand/retailer, we should return the most similar offers sponsored on our platform

- **Input**: text (could be keywords or short sentences)
- **Output**: search results (most similar offers and similarity score for diagnostic reason)

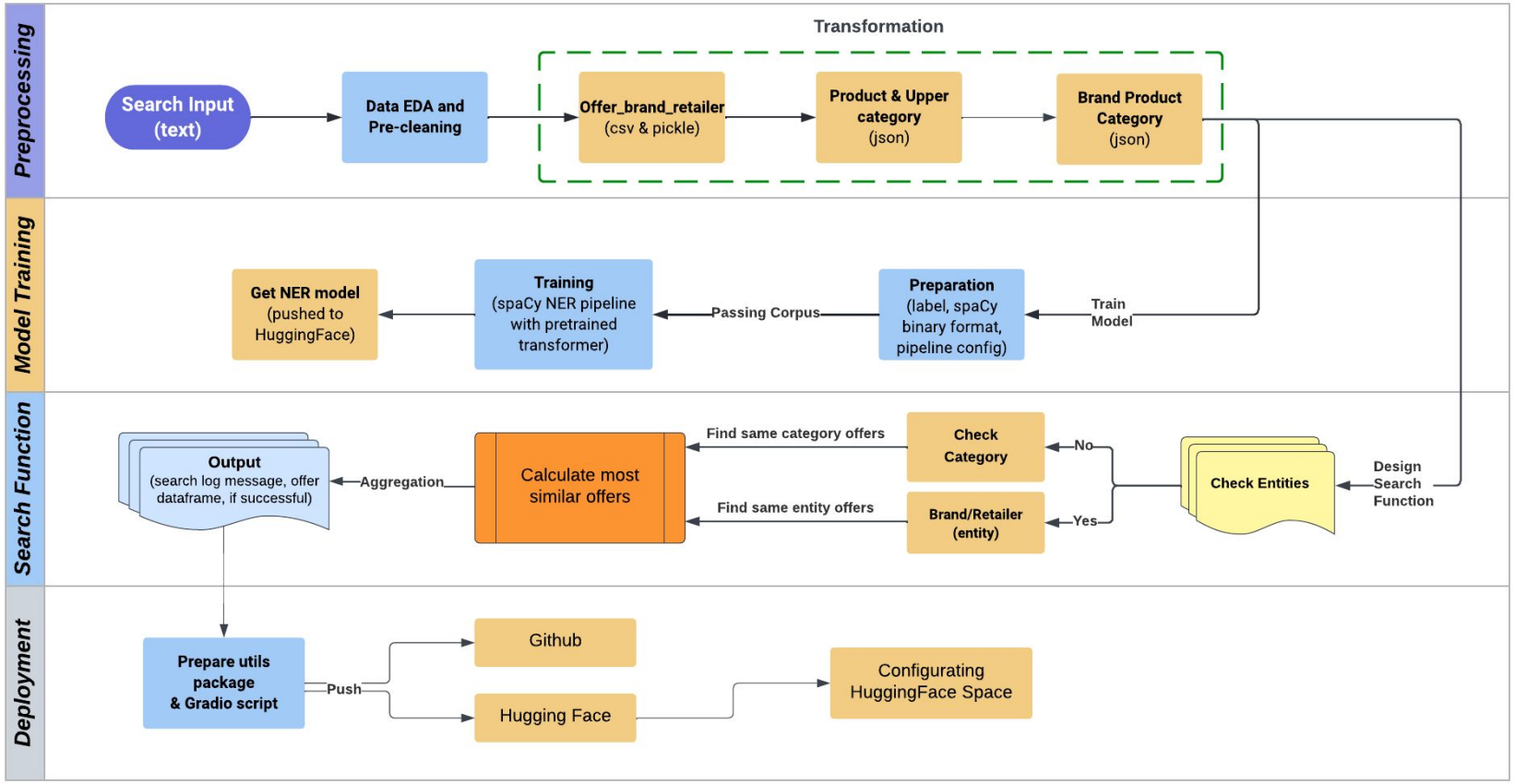
	Example Input	Process	Example Output
Category*	Beer, Diaper, Frozen Pizza	Identity category, and find offers under this category	Alcohol**, Baby & Toddler, Frozen Food
Brand	Back to Roots Soil	Identity brand, and find offers related to this brand	Back to the Roots Garden Soil, 1 cubic foot, at Lowe's Home Improvement
Retailer	Costco Member subscription	identify retailer, and find offers related to this retailer	Join Costco as Gold member … (Members Only)

Notes: * Although we have multiple examples in the category input, I assume that normal users will give one category per search; multi-category input will still work, but having accuracy decay
** In the dataset, `Alcohol` contains more than alcoholic drinks `[Beer, Malt Beverages, Hard Seltzers, Sodas, Waters, Lemonades & Teas, Hard Ciders, Wine, Spirits]`. I personally believe it should be either renamed or given more categories for better differentiation. However, I didn't change any category names.

Methodology: Project Workflow & Pipeline Design

Design rationale & objectives

- **Basic goal** of this project is to analyze (1) what are the searched category/brand/retailer and (2) what are the most similar offers
- **Why spaCy**? Because it is computationally efficient to carry out the task in step (1). Also, I still utilized the transformer-based layer
- **Why don't we use LLMs**? Too much processing time and limited labeled data. Task (1) only needs NER part. A tradeoff between customer satisfaction (speed) and search quality (search accuracy)
- **How to match search to offers**? See next slides
- **Similarity score**. Jaccard or Cosine. You can choose it by your own in demo



Major components: (a) Data EDA, preprocessing, **(b)** spaCy NER model with transformer embedding, **(c)** Development of Search Functionality and Gradio interface, and **(d)** Packaging and deploying to Hugging Face Space

Methodology: Data EDA, Pre-cleaning, and Data Transformation

Data files

- **categories.csv**: product categories and the upper categories (Beer - Alcohol)
- **offer_retailer.csv**: offers (OFFER, RETAILER) that supported on platform
- **brand_category.csv**:brands and the product categories they respond to

Exploratory results

- Basics info*:
  - 118 product categories & 23 upper categories (with a few overlaps)
  - 384 offers and 238 retailers
  - 8501 brands and 118 corresponding product category
- Some overlaps between retailers and brands (e.g. Subway), but such retailed-owned brands will not be sold by other retailers
- Other findings (e.g. entity token length distribution) that helped my cleaning progress can be found in notebooks, or explained further if I were to move on to next stage of this interview

Data Cleaning

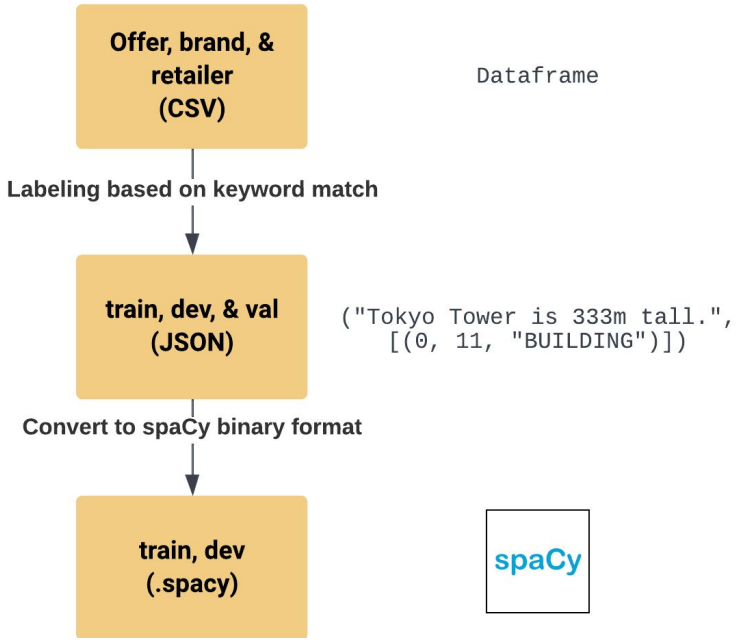
- Removed NaN values in offer_retailer table
- Remove punctuations, but kept alphanumeric and special characters like ["&", ".", ",", "-"] in offers and retailer names
- Removed offers that do not have brand retailer name -> noises labels

Data Transformation

- New files created:
- **offer_brand_retailer**: csv and pickle files about sponsored offers and their brands & retailers
- **corpus**: training materials for spaCy NER model, such as train.spacy
- **brand_belong_category_dict.json**: dict, key is category and value is the associated brands

Notes: *all in unique counts

Methodology: Labeling and Preparing Corpus



Goal: Using spaCy pipeline to train a new spaCy NER model to identify brand/retailer

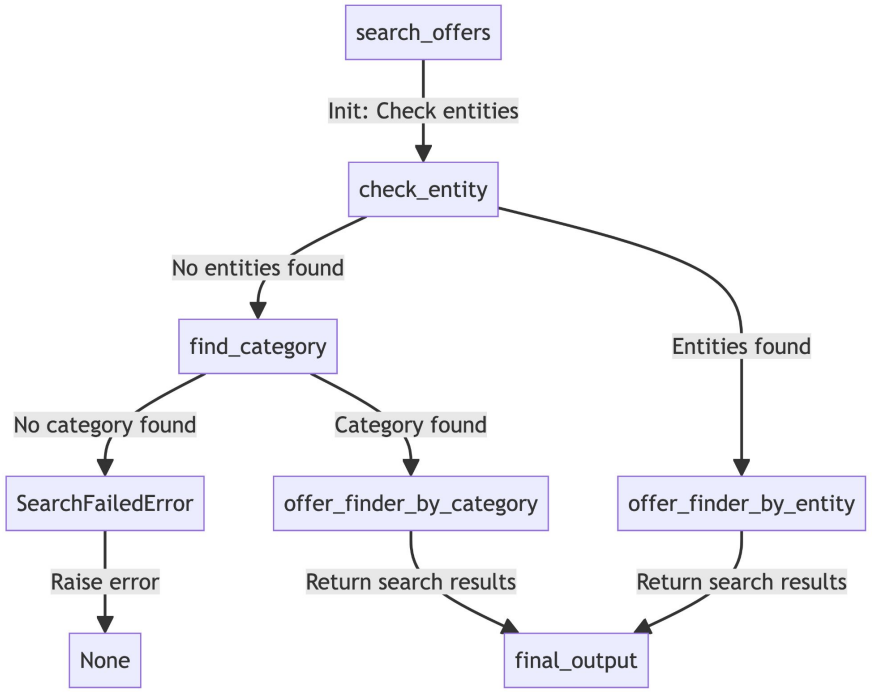
1. Use the offer_retailer.csv and clean the table (e.g. removing special characters & pure-number names, removing offers that don't have any information about its retailer or brand)
2. Use pandas and pd.Find() to extract labels and start/end positions
3. Convert to JSON format and write json_parser() to convert to spaCy binary format
4. Write spaCy configuration files and train on Colab (T4 GPU); about why using spaCy transformer, it is a choice about embedding, please refer to Appendix

Examples:

```
('DOVE Chocolate, select sizes, buy 1', [[0, 14, 'BRAND']]),
('The Rustik Oven bread', [[4, 15, 'BRAND']]),
('When you join Costco as an Executive Member New Members Only',
 [[14, 20, 'BRAND']]),
```

Notes: *Why removing pure-number names in labeled data? Because this is vague about what the input means–it could be a typo or a false search. Also, we know brands like Forever 21, but it is rare to have many brands named as pure number (you might even find it difficult to register a trademark like '123' or '99'). To deal with this special case, it might be better to add some special brand/retailer name dictionary, rather than just feeding it as labeled data

Methodology: Search Function



When inputting a text search (e.g. keyword or short sentence), here are the steps taken into account:

First, check if there's entity in the searched text

Case#1: If input has entities, then function will filter the available offers on the platform based on brands or retailers; it can handle multiple entities and then extract all relevant offers for similarity calculation (filtered by a customizable score threshold)

- If there are no matches of brand/retailer in available offers, it will search based category

Case#2: If no entities found, check if there's any category.

- Normally results for category check is a tuple (product_category, upper_category). **If such tuple is returned**, the function will filter available offers–through the mapping from brand to category. **If no categories found**, it will request the user to search again
- If mapping failed, it will recommend the user with other product categories under the same upper category. But it won't calculate similar offers due to computational limitations. Instead, the user will search again

Case#3: If none of the above searches have results, then return None and a message to request a refined search

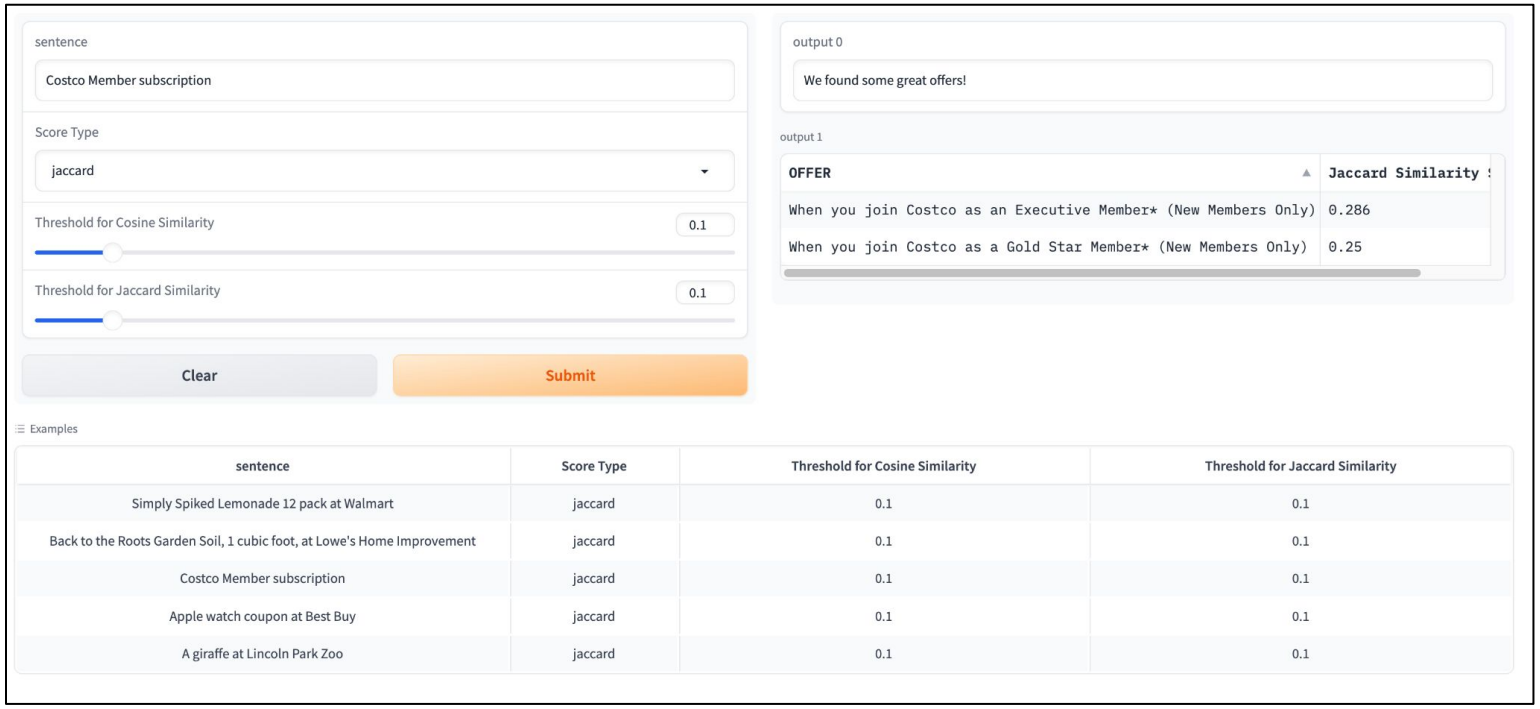
Step/Function Explanation:

- **`search_offers`**: The main function that takes in a search input and decides whether it can find entities or not. Then executes the appropriate functions
- **`check_entity`**: A function that checks if there are any entities in the search input
- **`find_category`**: A function that checks the category of the search input
- **`offer_finder_by_category`**: A function that finds similar offers based on the category of the search input
- **`offer_finder_by_entity`**: A function that finds similar offers based on the entities in the search input
- **`final_output`**: A function that sorts the search results by score and returns the final output
- **`SearchFailedError`**: An error that is raised when no entities or categories are found in the search input

Methodology: Text Similarity, Jaccard vs Cosine

	Jaccard Similarity	Cosine Similarity
How it works	Jaccard similarity measures the similarity between two sets. For text data, each sentence or document is treated as a set of words (or tokens)	Cosine similarity measures the cosine of the angle between two vectors. In text similarity, sentences or documents are represented as vectors (using techniques like TF-IDF or word embeddings).
Pros	Good for situations where the order of words doesn't matter. **Search inputs are usually a bunch of keywords**	Takes into account the frequency of words, which can provide a more nuanced similarity measure. Can be used with vector representations of text, which can capture semantic meanings of words.
Cons	May not be as effective if there are **synonyms** or **different ways of expressing** the same thing (since it relies on exact word matches, like soda can be baking soda/drinks); same flaws apply to the case of spelling errors	More computationally intensive, especially if using advanced word embeddings. The order of words is not considered (unless you're using more advanced embeddings like BERT).
Treatment & Issues	• Clean text, including removal of punctuations and stopwords • Currently using CountVectorizer due to computation limitation; as mentioned above, we can consider using better embedding methods to improve cosine similarity calculation • But the actual cases of user search might be more often with a bunch of keywords (e.g. Diapers discount Target) • Spelling is a big disturbance, which is why context is important–otherwise we have to do heavy cleaning if going further with Jaccard; we might use fuzzy search in further to address this issue	
Takeaways	If the offers in your database are phrased in very different ways, using a method that captures semantic similarity (like Cosine similarity with word embeddings) might give you better results, as it can understand that different words might have similar meanings. However, if the offers are relatively straightforward and use a consistent vocabulary, Jaccard similarity might be sufficient and would be faster to compute. Additionally, considering the commercial nature of the texts, you might want to implement some form of preprocessing to normalize product names or categories, which could improve the performance of either similarity measure.	

Demo & Deployment



Demo available on Hugging Face

If you want to run it locally, follow the instructions on my Github Repo

Appendix - Comparison Offer Similarity Scores

In [10]:

```
# test = "Sara Lee bread, select varieties, buy 2 at Walmart"
test = "Back to the Roots Garden Soil, 1 cubic foot, at Lowe's Home Improvement"

main(test, df_offers_brand_retailer, category_dict, brand_belong_category_dict, 'cosine', 0.3)
```

```
Function main | Search input: Back to the Roots Garden Soil, 1 cubic foot, at Lowe's Home Improvement
Found 2 entity object(s) in the search input.
Function offer_finder_by_entity | Found entity: Back to the Roots with label: BRAND
Function offer_finder_by_entity | Found 15 offer(s) for the brand/retailer: Back to the Roots
Function offer_finder_by_entity | Found entity: Lowe's Home Improvement with label: RETAILER
Function offer_finder_by_entity | No offer is found for the brand/retailer: Lowe's Home Improvement
```

Out[10]:

	OFFER	Cosine Similarity Score
0	Back to the Roots Garden Soil, 1 cubic foot, at Lowe's Home Improvement	1.000000
1	Back to the Roots Potting Mix, 1 cubic foot, at Lowe's Home Improvement	0.777778
2	Back to the Roots Soils, select varieties and sizes, at Lowes	0.377964
3	Back to the Roots Soils, select varieties, at Walmart or Lowes	0.377964
4	Back to the Roots Grow Kits at Walmart or The Home Depot	0.377964
5	Back to the Roots Dry Plant Food, 5 pounds, at The Home Depot	0.353553
6	Back to the Roots Seeds, at Walmart	0.333333
7	Back to the Roots Raised Bed Gardening Kit with Soil, Seeds and Plant Food, at Target	0.301511

In [11]:

```
test = "Back to the Roots Garden Soil, 1 cubic foot, at Lowe's Home Improvement"

main(test, df_offers_brand_retailer, category_dict, brand_belong_category_dict, 'jaccard', 0.1)
```

```
Function main | Search input: Back to the Roots Garden Soil, 1 cubic foot, at Lowe's Home Improvement
Found 2 entity object(s) in the search input.
Function offer_finder_by_entity | Found entity: Back to the Roots with label: BRAND
Function offer_finder_by_entity | Found 15 offer(s) for the brand/retailer: Back to the Roots
Function offer_finder_by_entity | Found entity: Lowe's Home Improvement with label: RETAILER
Function offer_finder_by_entity | No offer is found for the brand/retailer: Lowe's Home Improvement
```

Out[11]:

	OFFER	Jaccard Similarity Score
0	Back to the Roots Garden Soil, 1 cubic foot, at Lowe's Home Improvement	1.000
1	Back to the Roots Potting Mix, 1 cubic foot, at Lowe's Home Improvement	0.636
2	Back to the Roots Soils, select varieties and sizes, at Lowes	0.231
3	Back to the Roots Soils, select varieties, at Walmart or Lowes	0.231
4	Back to the Roots Grow Kits at Walmart or The Home Depot	0.231
5	Back to the Roots Dry Plant Food, 5 pounds, at The Home Depot	0.214
6	Back to the Roots Seeds, at Walmart	0.182
7	Back to the Roots Raised Bed Gardening Kit with Soil, Seeds and Plant Food, at Target	0.176
8	Back to the Roots Organic Kits and Planters, at Target	0.154
9	Back to the Roots Soils, select varieties, at Walmart	0.154
10	Back to the Roots Soils, select varieties and sizes, at Walmart	0.143
11	Back to the Roots Microgreens Grow Kit OR Seed Refill at Walmart	0.133
12	Back to the Roots Grow Hydroponic Grow Kit OR Refill Bundle at Walmart	0.125
13	Back to the Roots Organic 3-In-1 Seed Starting Mix 12 quart, at Walmart or Target	0.125
14	Back to the Roots Grow Seed Starting Pots OR Germination Trays, at Walmart or Target	0.118

Both scores provide good rank ordering (that is, separation of offers that can be easily grouped). However, Cosine Similarity Score seem to have a smother distribution of offers by score, allowing more flexible cuttoff with threshold value

Appendix - spaCy, boosting performance with transformers

spaCy NER training with token2vec

```
[2023-10-21 17:49:01,372] [INFO] Initialized pipeline components: ['tok2vec', 'ner']
✔ Initialized pipeline

============================= Training pipeline =============================
ℹ Pipeline: ['tok2vec', 'ner']
ℹ Initial learn rate: 0.001
E    #       LOSS TOK2VEC  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  ------------  --------  ------  ------  ------  ------
  0       0          0.00     49.33   13.43    8.49   32.14    0.13
 13     200         35.81   1490.48   78.90   81.13   76.79    0.79
 28     400          0.00      0.00   80.73   83.02   78.57    0.81
 48     600          2.80      1.93   84.11   88.24   80.36    0.84
 72     800         13.31      6.00   86.24   88.68   83.93    0.86
101    1000          0.01      0.00   83.02   88.00   78.57    0.83
136    1200         17.42      5.06   83.33   86.54   80.36    0.83
179    1400        101.81     21.93   81.90   87.76   76.79    0.82
229    1600         25.86      4.08   85.19   88.46   82.14    0.85
293    1800          0.00      0.00   84.11   88.24   80.36    0.84
361    2000          0.00      0.00   84.11   88.24   80.36    0.84
461    2200          0.00      0.00   84.40   86.79   82.14    0.84
561    2400          0.00      0.00   83.33   86.54   80.36    0.83
✔ Saved pipeline to output directory
```

spaCy NER training with base-Bert

```
============================= Training pipeline =============================
ℹ Pipeline: ['transformer', 'ner']
ℹ Initial learn rate: 0.0
E    #       LOSS TRANS...  LOSS NER  ENTS_F  ENTS_P  ENTS_R  SCORE
---  ------  -------------  --------  ------  ------  ------  ------
  0       0         576.22    453.42   13.94    9.48   26.32    0.14
100     200       21203.24  23940.32   88.89   88.31   89.47    0.89
200     400           0.00    316.27   90.07   90.67   89.47    0.90
300     600           1.58    308.27   88.74   89.33   88.16    0.89
400     800          30.78    338.28   91.39   92.00   90.79    0.91
500    1000           6.35    292.41   92.21   91.03   93.42    0.92
600    1200           1.00    276.68   92.21   91.03   93.42    0.92
700    1400           3.93    272.53   90.79   90.79   90.79    0.91
800    1600           0.00    249.77   90.79   90.79   90.79    0.91
900    1800           0.00    238.44   90.79   90.79   90.79    0.91
1000   2000           0.00    223.54   90.79   90.79   90.79    0.91
1100   2200           0.00    208.17   90.79   90.79   90.79    0.91
1200   2400          21.46    217.55   91.50   90.91   92.11    0.92
1300   2600           2.49    176.41   93.42   93.42   93.42    0.93
1400   2800           1.16    156.54   91.50   90.91   92.11    0.92
1500   3000           0.00    136.26   90.20   89.61   90.79    0.90
1600   3200           0.00    117.72   91.50   90.91   92.11    0.92
1700   3400           0.00     99.77   92.11   92.11   92.11    0.92
1800   3600           0.00     81.52   91.50   90.91   92.11    0.92
1900   3800           0.00     65.55   91.50   90.91   92.11    0.92
2000   4000           0.00     50.48   91.50   90.91   92.11    0.92
2100   4200           0.00     37.89   91.50   90.91   92.11    0.92
```

Does embedding helps to improve performance?

I believe there's minor boosting effect when I trained spaCy NER with transformers. Because user-generated search inputs might be vague and require context. For example:

- "Coca-cola 12 packs discount at Walmart" ➡️Pretty clear meaning
- "Coke 12 packs Walmart" ➡️Challenging because coke is not a clear brand name–it can be Pepsi or Coca-cola; but with context of Walmart, I know at least it is a purchasable brand at walmart, which is highly like a soft drink

About me

Han Jiang

- MS in Data Science@Uchicago (graduation: 12/2023)
- BA in Economics @ UW Madison
- Data Scientist Intern at CreditNinja
- NLP research intern (capstone) at KYC2020
- Data Analyst and Economic Intern at United Nations @UNCDF and @UNESCAP