Wertsching, Keith | Schoenleber, Brian | April 20, 2020

Named Entity Resolution for Vendor Database

An Analysis of Entity Name Resolution Using Deep Learning and Unstructured Text Data

Abstract

The goal of this project is to determine whether or not incorporating unstructured text data associated with an entity can improve the accuracy of a deep learning entity resolution model. The determination will be made by first building a model that can identify entity matches between a grounded source of entity names and an ungrounded source of entity names, then comparing the accuracy of a baseline model that uses only the entity names to the accuracy of an EntityPlus model that incorporates unstructured text data into the grounded corpus of knowledge. We will also perform an error analysis to determine each model's strengths and weaknesses in matching grounded entities with non-grounded entities.

Github Link: https://github.com/keithwerts1/Named_Entity_Resolution_for_Vendor_Database

Keywords: Entity Resolution, Deep Learning, Neural Networks, Entity Matching, Entity Grounding

Introduction

An entity is defined as a thing with distinct and independent existence. One of the largest challenges facing Entity Resolution (the process of matching entities between a grounded source and an ungrounded source) lies in the ambiguity of the entity concept, defined as a thing with distinct and independent existence. The process of determining what is and what isn't a thing with distinct and independent existence - part of Entity Grounding (Leidner, J., Sinclair, G., & Weber, B.) - can be highly subjective depending on the use case. The rigidity of academia has allowed for a wider adoption of Entity Grounding than the corporate world. Academic scientists have used the power of Machine Learning to associate and group proteins into family units (Bachman, J., Gyori, B., & Sorger, P.). This project will focus on entity to entity matching issues in accounting and financial databases. The goal of the project is to implement a flexible model with applicability to realistic business issues and report on the accuracy and risks associated with using the Entity Grounding program for entity to entity matching.

Entity Resolution in the Field

There are many different ways in which Entity Resolution can be used. Much of the field focuses on the act of matching any two given terms. Are they or are they not referring to the same thing? This concept can be applied pair-wise, or by clustering depending upon the application. A significant portion of this field is dedicated to the task of disambiguation, determining when homonyms or very similar titles are actually referring to two different things (Volk, M.). Record linkage is perhaps the task that best describes the objective task of our project, matching de-duplicated records between one database and another (Anastácio, I., Martins, B., & Calado, P., 1970). These techniques are broadly deployed in such tools as Wikipedia, Google Knowledge Graph, and Wikidata, as well as being used extensively by Intelligence agencies such as the CIA and NSA. The nature of our specific objective comes with some difficulties and some expediencies. For starters, we don't have to concern ourselves with de-duplication or disambiguation because one of our lists is grounded; and the existence of a grounded value means that we can focus on matching pairs between the lists rather than searching for clusters of similar titles. The primary added difficulty lies with the fact that we are performing matches based on nothing other than it's title.

Data Sources

The data source for grounded entities will rely on the U.S. Securities and Exchange Commission's EDGAR company filings (EDGAR, 2009). The database contains all companies that offer stock for sale that have not been granted exceptions from regular SEC reporting. The identifier for what is a separate entity is the Central Index Key (CIK number). An entity may have several different entity names associated with a single CIK number. For the purpose of our baseline model, the CIK number will be treated as the categorical value, and the list of entity names will be the baseline corpus of knowledge that the model will train against.

The data source for ungrounded entities is a list of the S&P 500 New York Stock Exchange company names with the CIK number already associated with it from the S&P 500 wikipedia page (List of S&P 500 companies, 2020). The evaluation dataset has 500 ungrounded entity names that are labeled with the CIK number for evaluation purposes. This dataset can be found in the Curated github folder as 'TEST_NAME.csv'.

We will only be training on the 500 entities that are represented in the S&P 500 list, which means that every entity in the evaluation dataset has exactly a matching CIK number in the grounded dataset. Because the SEC allows multiple entity names to be registered against a single CIK number, the grounded entity list has 820 entity names representing 500 unique, grounded entities. The training dataset can be found in the Curated github folder as 'TRAIN_NAME.csv'.

The EntityPlus model will incorporate the entire text of the wikipedia article associated with each of the 500 grounded entities. The association was done manually for the 500 grounded entities by a member of our team. This association can be found in the Curated github folder as 'TRAIN_WIKI.csv'.

Pre-Processing and Embedding

The automated matching process will be performed by a custom-built NLP deep learning python program using Keras and Tensorflow as a base (Chris, & Yang, 2020). The CIK labels will be encoded using sklearn's LabelEncoder package. We will then use Numpy's to_categorical package to create a Keras recognizable category for each CIK number so that we can use a categorical cross-entropy loss function. The unstructured text data will be split up by sentence with the associated CIK number categorical value to ensure that the model can cycle through and train on the data effectively. The entity names and unstructured text sentences will be encoded using sklearn's feature extraction package called CountVectorizer, which is commonly used for a 'Bag of Words' model. All available words from the grounded entity names, the unstructured text sentences, and the target matching entity names will be used to fit the CountVectorizer model, and the model will be used to transform each entry into a Keras readable sparse vector.

Model

We experimented with several different architectures of increasing complexity to find the layer structure that yields the best accuracy for the Baseline and EntityPlus Modes. We performed separate runs of each architecture on the Baseline and EntityPlus models to test out the impact of increasing complexity on the model's ability to match between the grounded entity list and the ungrounded entity list.

The Baseline model was trained on the set of 500 grounded entities, which is made up of 820 labeled entity names. The model will attempt to match each of the 500 ungrounded entity names with one of the categorical labels provided in the grounded entity dataset. Because we have at most 2-3 entity names for each category, we are going to train the model with a batch size of 1 and will run a number of simulations of different epochs to get the highest accuracy score before over-fitting kicks in.

The EntityPlus model will train on the 820 labeled entity name vectors and the labeled sentence vectors from the manually-associated wikipedia data. Since the number of records to train on is much larger, we will vary both the batch size and the number of epochs to get the highest accuracy score before over-fitting kicks in. We will then compare the accuracy scores and do a deep dive confusion matrix analysis to see what each model matched correctly and what each model was not able to match correctly.

The first architecture, which we are calling the 'Simple Model', starts as a Keras sequential model with an initial dense input layer equal to the dimension of the CountVectorizer fitted model (Real Python, 2018). As part of hyperparameter tuning, we experimented with different output dimensions for this dense layer, and found that setting the dimensionality equal to the number of categories gave the most promising results (compared to 10% and 50% of the number of categories). We chose relu activation to increase efficiency of the model and lower the overall processing time. The 'Simple Model' then concludes with a sigmoid layer for predictive purposes, which has an output dimensionality equal to the number of categories that the model is using for predictions. The model is then compiled using a categorical cross-entropy loss function, which is necessary for the model carry-out the matching process, where it predicts from a known set of categories to assign to the ungrounded evaluation set of data. We decided to use 'Adam' optimizer to find the ideal learning rate, which allowed us to focus on varying the architecture complexity and comparing the results between the Baseline and EntityPlus models. Given that the actual label for every value in the test set was unique within the test set, we had no need for a metric that normalizes for the frequency of any given label, and were therefore able to use accuracy.

After compiling our initial results, we implemented a more complicated architecture, which we are calling our 'Advanced Model'. The 'Advanced Model' includes everything from the baseline model, but after the first layer we included a Dropout layer with a 25% dropout rate, meant to add some randomization to the model and protect against over-fitting. After the Dropout layer, we included another dense layer with dimensionality equal to the number of categories to allow the model to recalibrate from the randomness introduced by the Dropout layer before it moves to the predictive Softmax layer. We kept the same layer parameters as the 'Simple Model' for the predictive Softmax layer, and again compiled the model using a categorical cross-entropy loss function, adam optimizer, and with accuracy as the key metric.

The performance of each type of architecture was evaluated by comparing the performance of the Baseline model across the layer structures, the performance of the EntityPlus model across the layer structures, and difference in performance between the Baseline and EntityPlus Model across the layer structures.

Hyperparameter Tuning

The first stage of Hyperparameter tuning was performed while determining the output size of the first dense layer of the 'Simple Model' architecture. After analyzing the performance of the model with varying output sizes, we saw significant increases in accuracy from a low-dimensionality output to an output size equal to the number of categories. Increasing the output dimensionality beyond the number of categories did not appear to improve the accuracy in any significant way. Our working theory is that setting the output dimensionality equal to the number categories allows the model to handle the complexity of the input data without over-generalizing too much information in a way that harms the ability for the predictive layer to assign the correct value.

Our strategy for finding the ideal number of epochs was to start with a few epochs and slowly increase the number of epochs until the evaluation accuracy started to decline. We considered the point of decline to be the tipping point where the model starts to overfit on the training data and becomes less able to generalize on the evaluation data. The accuracy at this point is what we are calling the maximum accuracy for that model, architecture, and hyperparameter values.

We also embarked on figuring out the ideal batch size that would lead to the highest overall accuracy score. For the Baseline model, we saw significant decreases in accuracy with a batch size over 1, likely due to the relative sparsity of records for each category (at most 2-3 entity names). For the EntityPlus model, we saw improvements in maximum accuracy as we increased the batch size, up to a certain amount, at which point the maximum accuracy started to decline. We took this point of decline to be the tipping point where over-generalization starts to occur as the model is unable to focus on the important information in the training data that would help it to predict the correct label. For the purposes of Evaluation, each model type and architecture type was assigned a final maximum accuracy that was the highest maximum accuracy for the best performing batch size.

Evaluation

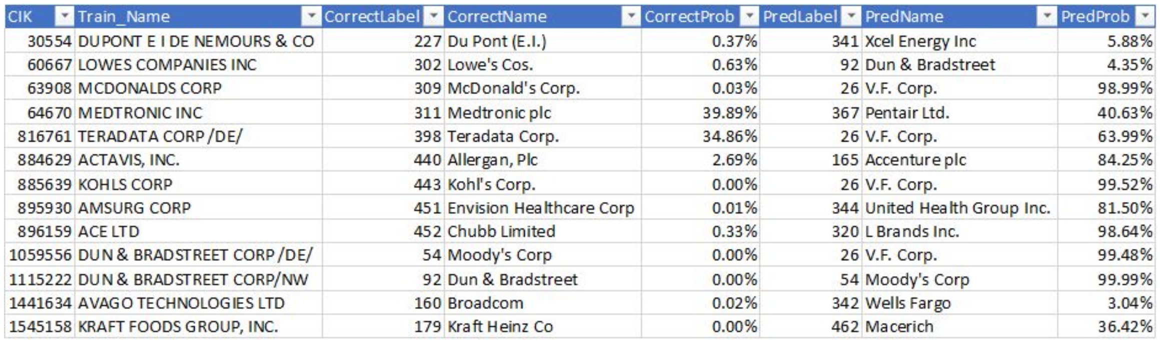
The 'Simple Model' architecture trained on the Baseline model performed quite well by most conventional standards. The model was able to train over 1000 epochs with batch size of 1 before showing signs of over-fitting, and ended with a maximum accuracy score of 73.4%. The same architecture trained on the EntityPlus dataset took much longer to train given the need to vary batch size and number of epochs, but produced a remarkably high accuracy rate of 93.8% with a batch size of 16 and 160 epochs.

The ‘Advanced Model’ architecture showed significant improvements over the ‘Simple Model’ architecture when comparing Baseline-Baseline, EntityPlus-EntityPlus, and the difference between Baseline and EntityPlus across the two architecture types. Though each Epoch took longer to train, the number of Epochs needed to reach the maximum accuracy score was much lower. The Baseline model produced a maximum accuracy score of 97.4%, an improvement of over 23% when comparing Baseline-Baseline accuracy. Similarly, the ‘Advanced Model’ architecture produced a maximum accuracy score of 98.4% for the EntityPlus model, an increase of 4.6% over the ‘Simple Model’. The gap between Baseline performance and EntityPlus performance was much lower for the ‘Advanced Model’, but still demonstrated that including the unstructured text data from the wikipedia articles helped to fill in some of the gaps created by the variations in entity names between the grounded dataset and the ungrounded dataset.

Evaluation

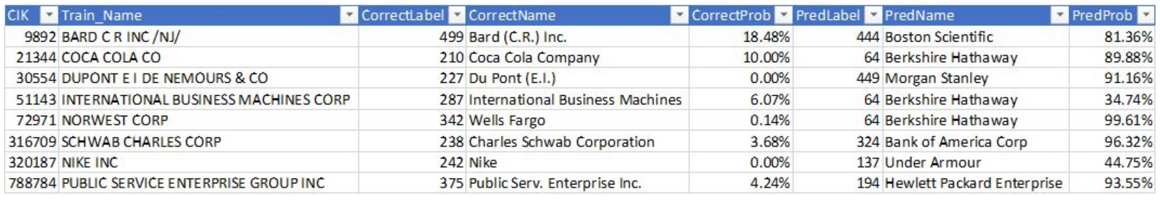
A comparison of the confusion matrices between the baseline model and the EntityPlus model showed the following results (analysis is done on the results of the ‘Advanced Model’ architecture due to the higher maximum accuracy scores across the board):

Baseline Model



CIK	Train_Name	CorrectLabel	CorrectName	CorrectProb	PredLabel	PredName	PredProb
30554	DUPONT E I DE NEMOURS & CO	227	Du Pont (E.I.)	0.37%	341	Xcel Energy Inc	5.88%
60667	LOWES COMPANIES INC	302	Lowe's Cos.	0.63%	92	Dun & Bradstreet	4.35%
63908	MCDONALDS CORP	309	McDonald's Corp.	0.03%	26	V.F. Corp.	98.99%
64670	MEDTRONIC INC	311	Medtronic plc	39.89%	367	Pentair Ltd.	40.63%
816761	TERADATA CORP /DE/	398	Teradata Corp.	34.86%	26	V.F. Corp.	63.99%
884629	ACTAVIS, INC.	440	Allergan, Plc	2.69%	165	Accenture plc	84.25%
885639	KOHLS CORP	443	Kohl's Corp.	0.00%	26	V.F. Corp.	99.52%
895930	AMSURG CORP	451	Envision Healthcare Corp	0.01%	344	United Health Group Inc.	81.50%
896159	ACE LTD	452	Chubb Limited	0.33%	320	L Brands Inc.	98.64%
1059556	DUN & BRADSTREET CORP /DE/	54	Moody's Corp	0.00%	26	V.F. Corp.	99.48%
1115222	DUN & BRADSTREET CORP/NW	92	Dun & Bradstreet	0.00%	54	Moody's Corp	99.99%
1441634	AVAGO TECHNOLOGIES LTD	160	Broadcom	0.02%	342	Wells Fargo	3.04%
1545158	KRAFT FOODS GROUP, INC.	179	Kraft Heinz Co	0.00%	462	Macerich	36.42%

EntityPlus Model



CIK	Train_Name	CorrectLabel	CorrectName	CorrectProb	PredLabel	PredName	PredProb
9892	BARD C R INC /NJ/	499	Bard (C.R.) Inc.	18.48%	444	Boston Scientific	81.36%
21344	COCA COLA CO	210	Coca Cola Company	10.00%	64	Berkshire Hathaway	89.88%
30554	DUPONT E I DE NEMOURS & CO	227	Du Pont (E.I.)	0.00%	449	Morgan Stanley	91.16%
51143	INTERNATIONAL BUSINESS MACHINES CORP	287	International Business Machines	6.07%	64	Berkshire Hathaway	34.74%
72971	NORWEST CORP	342	Wells Fargo	0.14%	64	Berkshire Hathaway	99.61%
316709	SCHWAB CHARLES CORP	238	Charles Schwab Corporation	3.68%	324	Bank of America Corp	96.32%
320187	NIKE INC	242	Nike	0.00%	137	Under Armour	44.75%
788784	PUBLIC SERVICE ENTERPRISE GROUP INC	375	Public Serv. Enterprise Inc.	4.24%	194	Hewlett Packard Enterprise	93.55%

Looking through both confusion matrix results, there was only one entity that appeared in both the Baseline Model and the EntityPlus model. This indicates that even though the metrics gathered demonstrate that our hypothesis that accuracy score is improved by including associated unstructured text data appears valid, there is a precedent to be concerned that adding more data will cause the model to incorrectly assign a label that it would have assigned correctly if only trained on entity names. Put another way, including the unstructured text data allowed the EntityPlus model to correctly match 12 entities that it was not able to correctly match in the

Baseline model, but also caused 7 entities that were matched correctly in the Baseline model to become incorrect matches.

The PredProb column shows the model's confidence in the choice it made. Given that the model was roughly 99.5% sure of an incorrect choice for an example in both the Baseline and the Entity plus models, there is no indication that setting a lower limit where the model does not make a choice if it is below a confidence threshold would produce significant results. A real-world implementation of this model would at this time still require a human to validate the accuracy after each round of predictions to eliminate the risk of false matches having adverse impacts in billing or reporting systems. We will discuss in the next section some further actions that can be taken to try to accurately predict these last few entities that are showing incorrect results.

Conclusion and Further Opportunities

The stark improvement in accuracy for the unstructured text model over the baseline model leads us to believe that there is likely value in an organization investing in the process of associating unstructured data sources to their respective entities to fill in the gaps that exist when only entity names are used to train a corpus of knowledge for the purpose of entity resolution. Particularly encouraging was that the confusion matrix analysis revealed no correct entity matches made by the baseline model were lost after incorporating the unstructured text data. The next step in our journey should involve attempting to apply the logic to an organization's data landscape that has a need to match up ungrounded entity names with their match in a grounded entity source.

Before doing so, there are model adjustments and subsequent analysis that the team would like to conduct on the current dataset in hopes of understanding the impact that these adjustments may have on the relative accuracy scores. First and foremost, we would like to propose changing the architecture of the model to add additional layers and analyze the impact to accuracy in the baseline model and the unstructured text model. Based on a recommendation from our class instructor, we implemented a third architecture that we are calling our 'Many Layers Model'. This model includes 3 additional iterations of the Dropout and Dense Layer pairing that was introduced in the 'Advanced Model'. The results of this architecture were not promising. It took much longer to train each Epoch, and the maximum accuracy did not appear to be approaching the same accuracy score of 97-98% seen using the 'Advanced Model' architecture. The inclusion of several additional Dropout layers appears to add too much randomization, which hurts the models ability to learn and apply the information in the training model to perform the matching process with a high rate of accuracy. Our takeaway here is that more layers does not necessarily give a higher accuracy score. Given more time, we would have liked to be able to try out additional layer types, iterations of more dense layers compared to Dropout layers, and even doing our own learning rate optimization rather than relying on 'Adam' to optimize the learning rate.

Another proposed model change that may produce interesting results involves attempting to extract and put more emphasis on entity names in the unstructured text data to allow the model to give greater weights to text that may be more important in matching the ungrounded entity names to the grounded entity names. This operation may be more subjective than other model adjustments, but could be particularly useful in speeding up the training process for larger unstructured text sources and improving the accuracy for smaller unstructured text sources.

A third adjustment involves increasing the size of the grounded dataset to include entities that do not exist in the evaluation set to see the extent of degradation in accuracy when the algorithm has more categories to choose from. This will involve additional manual work in collecting the wikipedia text data for additional entities, but will be important knowledge to have in case the target organization has a need to match a smaller subset of ungrounded entity data with a large subset of grounded entity data. This model adjustment will be analyzed against the original results by analyzing the difference in baseline accuracy, unstructured text model accuracy, and gap between baseline and unstructured text model accuracy before and after the model adjustment.

Another potential future step would be to deploy the use of embeddings as a replacement to the countVectorization approach we utilized. Before this approach could be utilized, we would need to specifically understand some of the risks associated with creating an embedding value from whole sentences in unstructured text, and comparing it to the embedding value produced only by a company title. Having said this, we would like to look at tools such as Google's Universal Sentence Encoder to better understand if it's use is applicable in this context.

Finally, we would like to see the impact of treating our algorithm as a progressive learning model, where new evaluation data is found or created and the existing evaluation data becomes part of the training corpus of knowledge. This adjustment in data sources would allow us to simulate the change in model performance as corpus of knowledge grows in complexity. If possible, we would like to propose introducing additional unstructured text data as well, with the understanding that training time will eventually become a limiting factor and the ability to explain incorrect matches will become more difficult as additional data is incorporated.

References

Data Sources

1. EDGAR. (2009, December 9). Retrieved April 20, 2020, from https://www.sec.gov/edgar/searchedgar/cik.htm

2. List of S&P 500 companies. (2020, April 19). Retrieved from https://en.wikipedia.org/wiki/List_of_S&P_500_companies

Methods

3. Real Python. (2018, October 23). Practical Text Classification With Python and Keras.

   Retrieved from https://realpython.com/python-keras-text-classification/

4. Chris, & Yang. (2020, February 11). How to use sparse categorical crossentropy in Keras?

   Retrieved from

https://www.machinecurve.com/index.php/2019/10/06/how-to-use-sparse-categorical-crossentropy-in-keras/

Background and Domain

5. Anastácio, I., Martins, B., & Calado, P. (1970, January 1). [PDF] Supervised Learning for Linking Named Entities to Knowledge Base Entries: Semantic Scholar. Retrieved from https://www.semanticscholar.org/paper/Supervised-Learning-for-Linking-Named-Entities-to-Anastácio-Martins/10a8d3ec5fff000e356b80207e4acd74f1d8a309

6. Bachman, J., Gyori, B., & Sorger, P. (n.d.). FamPlex: a resource for entity recognition and relationship resolution of human protein families and complexes in biomedical text mining. Retrieved from https://www.biorxiv.org/content/10.1101/225698v2.full.pdf

7. Leidner, J., Sinclair, G., & Weber, B. (n.d.). Grounding spatial named entities for information extraction and question answering. Retrieved from https://www.aclweb.org/anthology/W03-0105.pdf

8. Volk, M. (n.d.). Named Entity Recognition in Digitized Historical Texts. Retrieved from https://www.mlta.uzh.ch/dam/jcr:cda50f3f-88a3-4e88-a6d9-73deef9df12c/Masterarbeit_YGwerder_FS17.pdf