Northwestern University

Named Entity Recognition on Business Insider Articles

Aarij Rehman

March 9th, 2020

Introduction

Business Insider is a portal for business news. Articles are posted daily about different financial, economic, and global topics. For that reason, Business Insider is an ideal candidate for natural language processing.

In this project, we analyze an article from every day in 2013-2014. From these articles, we're interested in extracting three particular pieces of information:

1) Names of CEOs
2) Names of Companies
3) Percentages

We use a combination of regular expressions and logistic regressions to build NERs, named entity recognizers. These NERs will be able to then parse through the data in the articles and extract what the relevant information. Each NER is built with the same framework, but different implementations.

Structure

The corpus is the collection of the two years of Business Insider articles. The articles are all tokenized by sentence and combined in a single list. When extracting entities, we use a combination of features regarding the entity itself and the sentence it is in. Some features were created through domain knowledge of the problem, and many others were created through a trial and error process.

The dissection of the features and the regular expressions used to capture candidates are explained in further detail below. Afterwards, there is a discussion of shortcomings of each model, and where error is likely introduced.

CEO Names

Regular Expression

When extracting CEO names, an NER based on logistic regression was built. To extract candidates, the following regex pattern was used:

```
[A-Z][a-z]+ [A-Z][a-z]+
```

This regular expression aims to capture full names. It matches a first name and last name with traditional English grammar – the first letter of the first and last name will be capitalized, and the remaining letters will be lowercase.

Features

To determine if a candidate was actually the name of a CEO, the following features were used.

- length_sentence
- ceo_in_sentence
- length_candidate
- ceo_close
- stop_in_name

length_sentence

This feature determines the length of a sentence which the candidate is in. In a broader sense, this feature does more to remove wrong candidates than highlight correct ones. We can expect that if the sentence is very short, then it's unlikely an entire CEO is mentioned. This allows our NER to exclude many incorrect extractions.

ceo_in_sentence

This feature determines if the word 'CEO' or the phrase 'Chief Executive Officer' is in a sentence. Many times, when a CEO is introduced, he's mentioned with his title as well. This allows us to look for those instances to better train our model for correct classification.

length_candidate

This feature determines the length of the candidate from the extraction. Similar to the other length feature, this function does more to remove incorrect extractions than highlight correct ones. Namely, many candidates are extracted as short, meaningless phrases, and this feature helps flag those as unlikely CEO names.

ceo_close

This feature determines if the word 'CEO' is within 20 characters of the extracted candidate. The 20-character limit was chosen with trial and error. Using smaller numbers found the feature to be too restrictive and using larger numbers would flag too many candidates. This feature is very similar to *ceo_in_sentence*, but because it imposes a strict distance measure, it correlates much better with correct candidates.

stop_in_name

This feature determines if either the ‘first’ or ‘last’ name of a candidate are stop words. This feature excludes many meaningless candidates, primarily ones that occur at the beginning of a sentence. Many false candidates are extracted when there are two capitalized words after one another, often containing stop words. By checking if the candidate has a stop word, we can almost always determine if the candidate is not a CEO name. This feature has a strong negative log-likelihood in the model, and it helps trim much of the data.

Performance

The model was trained on a randomly selected partition of half the data. It correctly predicted the label of the output 97% of the time. Although this figure seems convincing, it’s important to consider that most of the candidates have label 0. For this reason, it’s not very meaningful to look at the success rate. Instead, we can see that there are 6139 candidates that are marked as being CEOs, and the model classified 993 candidates as being CEOs. Clearly, only a small portion of CEOs were successfully captured. Furthermore, many of the classified CEOs are likely false positives. As expected, building an NER through logistic regression and sentence specific feature creation didn’t lend to promising results, but this model serves as a potential foundation for further development.

Shortcomings

Building a simple NER model like this involved making assumptions and ignoring potential issues. Firstly, the regular expression used to capture names only considers full names. That means if a CEO is ever mentioned by just the first or last name, the expression wouldn’t capture that candidate. Ignoring single names allowed the candidate sample space to be reduced dramatically.

Another issue is the presence of the multicollinear features – *ceo_in_sentence* and *ceo_close*. Specifically, if we know that *ceo_close* is 1, then *ceo_in_sentence* will necessarily be 1 as well. The logistic regression would perform best normally perform best with independent variables, but after testing, it was found that although counterintuitive, the best results were obtained when including both features as opposed to just one of the two. This may have to do with the fact that sometimes the word 'CEO' is mentioned in a sentence, but it's far from the actual candidate, so *ceo_close* will be 0 but *ceo_in_sentence* will be 1, which will help draw more candidates. Ultimately, many of the features are making strong assumptions about what surrounds a CEO name. Most of these assumptions are not very robust, but they're specific enough that they still perform adequately well.

Future Suggestions

For a better NER, I would suggest building an extension to an existing trained NER. For example, using spaCy or NLTK to screen candidates that are actually names will greatly reduce the number of samples. From there, building features will be more impactful because there will be a closer split between candidates that are actually CEOs and not.

Companies

Regular Expression

When extracting company names, an NER based on logistic regression was built. To extract candidates, the following regex pattern was used:

```
(([A-Z][a-z]+ ?)+)
```

This regular expression aims to capture repeated words that all start with an upper-case letter. The logic behind it is company names will be any number of words long, and they each word will start with an uppercase letter.

Features

To determine if a candidate was actually a company, the following features were used.

1) company_in_sentence
2) stock_in_sentence
3) shares_in_sentence
4) trade_in_sentence
5) length_of_company
6) plural_word
7) number_of_words
8) location_in_sentence
9) stop_word
10) company_word

company_in_sentence

This feature determines if the word ‘company’ is in the same sentence as a candidate. It’s fairly simple and has a marginal impact on the model, but the results were slightly better when keeping the feature in.

stock_in_sentence

This feature determines if the word ‘stock’ is in the same sentence as a candidate. Using domain knowledge of the Business Insider articles, we can expect that many times a company is mentioned, it will be in the context of its financial performance. For that reason, we may be interested if a candidate has the word stock in the sentence.

shares_in_sentence

Similar to *stock_in_sentence*, this feature looks to see if any mention of a ‘share’ is in a sentence.

trade_in_sentence

Similar to the above two features, this looks for any mentions of trading in a sentence.

length_of_company

This feature determines the length of a candidate. This is very important because the generous regular expression captures many arbitrary words that are capitalized. Generally, we find company names to be longer than average words, so it’s helpful to consider this feature.

plural_word

This feature determines if a candidate is plural. It does this by checking the final letter of the word and seeing if it is an ‘s’. This feature was introduced because a strong majority of the companies in the label data were not plural nor did they have an ‘s’ at the end. Furthermore, many captured candidates were plural, so by flagging them as such, we’re able to penalize them in the logistic regression.

number_words

This feature determines how many words a candidate has. Although company names aren’t necessarily more than one word, they often are. This also helps us penalize many of the candidates that are random, single capitalized words which may otherwise get classified as a company.

location_in_sentence

This feature determines where in the sentence our candidate is. Because we’re looking for any number of capitalized words, we’re always going to be capturing the start of every sentence because English grammar dictates sentences start with capital letters. Because of this, we have thousands of extra candidates which are likely just noise. This feature tends to penalize those candidates which start at the beginning of a sentence.

stop_word

This feature determines if a candidate has a stop word in it. Similar to *location_in_sentence* it penalize many of the random capitalized words that show up in the corpus.

company_words

This feature determines if a candidate has any of the popular words affiliated with a company. For example: inc, group, corp, etc. This feature deserves extra discussion because it has a very strong positive log likelihood on the model. The labeling data has many companies with these 'company words,' and generally, any candidate that has a company word tends to also be a company. This is good for feature extraction, but it is also a bit problematic because it often dominates the other features. We sometimes find a company name with moderately strong performance on the other features, but because it lacks a 'company word' it fails to get properly classified. That being said, because this feature introduces a very high accuracy rate into the model, I've decided to keep it. Most extracted entities involve a company word, but these entities are also very likely to be actual companies.

Performance

The model was trained on a randomly selected partition of half the data. It correctly predicted the label of the output 93% of the time. Similar to before, it's important not to get swayed by the accuracy percentage because most labels are 0. Instead, we again can see that in the testing data there are 52,000 candidates that are marked as being companies, and the model classified 628 candidates as being companies. Once again, only a small portion of companies were successfully captured. However, in this case, we can see that most of those capture companies actually are true companies. This is largely due to the fact that *company_words* feature has such a strong impact on the classifier. Most of our companies have some sort of company word in the name, which tends to mark it as a company. Ultimately, the model only captured a small portion of the company names, but it was very accurate in the names that it did capture. Without knowing what application these extractions may be used for, it’s tough to say whether accuracy or breadth is more important.

Shortcomings

Once again, building a simple NER model like this involves assumptions and ignoring potential issues. First off, the regular expression used to capture companies only considers companies that have all upper-case words. For example, ‘Bank of America’ would never get captured by the regular expression. In order to introduce this, the regular expression would have to be even more liberal with what it accepts, and there were already far too many candidates to begin, so this case was left out.

The other glaring issue with the model is the strength of the *company_words* feature. It often outweighed other features, which led to many false negatives. I originally tried recreating the features without *company_words,* but the results would always be dramatically better when including it, even if it was at risk of overfitting. Ideally, there would be a way to include this feature without having it simultaneously dominate the feature space.

Future Suggestions

This NER has a high accuracy, but it relies too heavily on the *company_words* feature. There should be more features which can cleverly include *company_words* without having it be dominating. This may involve introducing other features that take some of the predictive power. Alternatively, we can introduce more training data to help classify company names which don’t have a ‘company word’ correctly.

Percentages

Regular Expression

When extracting percentages, an NER based on logistic regression was built. To extract candidates, a regex pattern was built that would extract all numbers. This included integers, numbers with decimals, and numbers that are spelled out in words. Because of the complexity of the pattern, it's not included here in the report.

The idea is that we can consider all numbers as candidates, and using just a few features, determine if those candidates are actually percentages or not.

Features

To determine if a candidate was actually a percentage, the following features were used.

- word_after
- symbol_after

word_after

This feature determines if the word after the candidate is 'percent'. Although simple, this feature would almost certainly indicate if a given number is a percentage.

symbol_after

This feature determines if a candidate is followed by a % symbol. Similar to the other feature, it's simple yet effective.

Performance

The model was trained on a randomly selected partition of half the data. It correctly predicted the label of the output 77% of the time. In this example, we found a significantly lower accuracy score. This is because there are many more labels that are 1 compared to the previous two cases. Furthermore, the model also is predicting many of the results to be 1, even though they do not exist in the labels provided. This isn’t necessarily because the model is incorrect but because the labels provided are nowhere near exhaustive, so this is a good sign that the model is predicting percentages that were not previously given. Furthermore, there are 96,000 labels that are marked as 1, and the model classified 14,000 candidates as percentages. In this case, this can be a bit misleading because in many cases a random number is positively labeled because it exists in the labeling set, but it actually won’t be a percentage. For that reason, I’m more confident of my model than of the provided labels. Of the three models, I believe this one performs the best, likely because of how simple it is to extract percentages.

Shortcomings

Because of the simplicity of this model, it’s hard to point towards any shortcomings, but perhaps the most obvious is the issue with the training data. Ideally, there is enough positive data that the model understands the general trend, but because there are so many different ways to express a percentage, it’s likely that many actual percentages get labeled as 0 just because they don’t exist in the provided labeling set. That being said, by having such a simple model, it seems that this issue is mostly avoided.

Future Suggestions

Since I believe this NER performed well, I would suggest an efficiency improvement. Specifically, I think that the use feature creation with logistic regression is unnecessary for an NER this simple. I think in the future it would be equally appropriate to just use a through regular expression to capture all percentages. In fact, this was something I originally did, but changed in the end for the sake of consistency among the NERs.

Output

The outputs of the three different NERs are given as text files. For both the CEO and company names, only *unique* values were given, but for percentages all values were given. This is because the problem statement requires all values, but in the context of a company or CEO, it would be meaningless to report it multiple times; however, in the context of a percent, the same percent can be used for different things, so I decided it would make the most sense to report them all.