Name Entity Recognition Project Report

Yuxiang (Alvin) Chen

For the 3 tasks of this project, I researched and found that there exist multiple ways to approach it. Thus, to practise using different methods to extract the three different types of entities. I applied 3 different approaches to them. Here I presented my approaches in detail.

There are 3 separate jupyter notebook files for the 3 different tasks attached in the github repo.

Company Name extraction

Firstly, I preprocess the provided company labels by observing and removing the labels that do not exist in any of the articles.

Sample Preparation

Then, I started with finding all appearances of the provided company name labels and took 50 characters on the left and 50 characters on the right as the snippet to generate features.

For the negative samples, to make them smart negative samples instead of just random ones. I applied two filters.

1. I used the Spacy package of python to do entity extraction and if the entity is not part of the true label, then I mark the entity as a negative sample
2. During exploring different feature candidates, I find two features that always has same value in my positive samples - a. Positive entity's text is never all lower case; b. Positive entity's text never has number in it. Thus, I also only use entities that meet these two characteristics as negative samples.

After the 2-step filtering process, I have more negative samples than positive ones. At first I used downsampling negative samples to make my two labels balanced. However, Prof. Klabjan said that oversampling is a better solution than downsampling in most cases so I tried to oversampling my positive samples and it indeed works a little better especially in this case where some of my negative samples are not correctly labelled (they may be positive).

Model training

After preparing an equal number of positive and negative samples, I did a 80% 20% train test split on the data. Then only use the training set for model training.

Model used: Logistic Regression (2-class classification)

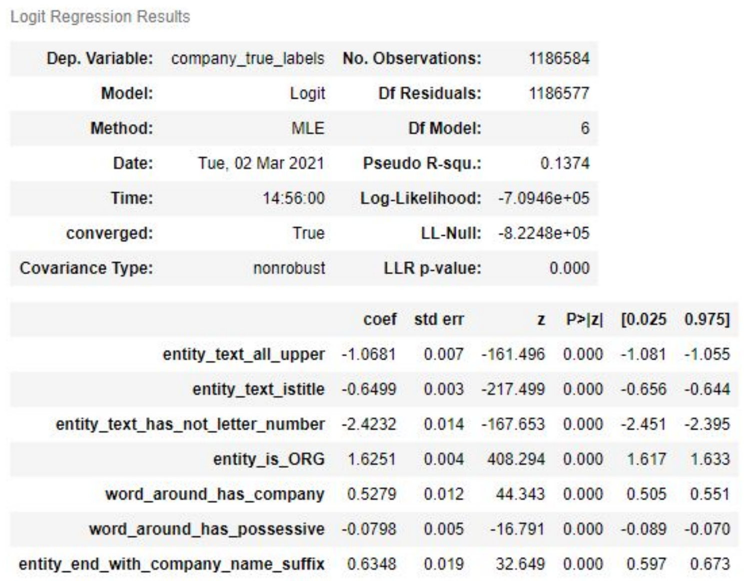
I tried around 15 candidate features based on either just the entity text or the text around the entity. Through feature selection methods and trial and errors, I finally selected 7 features to use. (I also found the two perfect separating features and added the adjustion of my negative sample selection method as mentioned above during this feature exploration process)

Features such as position of entity in the sentence, entity length were dropped during feature exploration process.

Features used:

Features (all binary)	Explanation
entity_text_all_upper	The entity text has all upper case letters
entity_text_istitle	The entity text has every word starting with a capitalized letter
entity_text _has_not_letter_number	The entity text has character that is not among [a-zA-Z0-9.\s\']
entity_is_ORG	The entity text is identified as an Organization name by Spacy NER tagger
word_around_has_company	The selected text snippet around the entity has ‘C(c)ompany’
word_around_has_possessive	The selected text snippet around the entity has ‘’s’
entity_end_with_company_na me_suffix	The entity text ends with one of the company name suffix {ltd, llc, inc, corp, corporation, co, group}

Model Result:



Logit Regression Results

Dep. Variable:	company_true_labels	**No. Observations:**	1186584
Model:	Logit	**Df Residuals:**	1186577
Method:	MLE	**Df Model:**	6
Date:	Tue, 02 Mar 2021	**Pseudo R-squ.:**	0.1374
Time:	14:56:00	**Log-Likelihood:**	-7.0946e+05
converged:	True	**LL-Null:**	-8.2248e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
entity_text_all_upper	-1.0681	0.007	-161.496	0.000	-1.081	-1.055
entity_text_istitle	-0.6499	0.003	-217.499	0.000	-0.656	-0.644
entity_text_has_not_letter_number	-2.4232	0.014	-167.653	0.000	-2.451	-2.395
entity_is_ORG	1.6251	0.004	408.294	0.000	1.617	1.633
word_around_has_company	0.5279	0.012	44.343	0.000	0.505	0.551
word_around_has_possessive	-0.0798	0.005	-16.791	0.000	-0.089	-0.070
entity_end_with_company_name_suffix	0.6348	0.019	32.649	0.000	0.597	0.673

Model Performance Evaluation

1. Confusion Matrix on the test set:

	Actual Positive	Actual Negative
Predict Positive	112678	42274
Predict Negative	35602	106092

2. On the test set, the model has a F1 score of 0.743 on the testing set for categorizing positive labels.

3. I checked what proportion of the provided labels are covered by the extracted names from the model (inference on the whole dataset), about 84.7% is covered. Another model I developed but decided not to use reaches above 90% in this metric but the negative samples are much less smart so it includes a large number of False negative predictions.

4. Following Prof. Klabjan's suggestion, I selected 10 random extracted company names to evaluate. I found 7 to be company names, 1 that could be company name and person's name, 1 person name and 1 organization name.

In summary, the performance of the model is not excellent but it gives a balanced result of company extractions.

All the extractor company names are in 'extracted companies.xlsx'

CEO Names Extraction

For the CEO extraction, instead of looking at all entities from the raw text, I made an assumption that a CEO's name will be tagged as 'PERSON' by the python spacy NER tagger and focused on these 'PERSON' entities. Thus, I focused on the task of distinguishing whether a 'PERSON' entity is a CEO name or not a CEO name. I think this is a task that is crucial for extracting CEO names since there exists well-trained NER taggers for tagging people's names with high accuracy but not every name belongs to a CEO.

Before building the model, I also processed and cleaned the labels by combining first and last name if both exist and get rid of labels that do not exist in the articles.

I believe that to identify whether a person is CEO, the information around in the sentence the name is in is important. Thus, I did sentence segmentation and generated my features based on sentence context information. I also originally tried some features that focus on the name words themselves but I excluded them after feature selection. This matches with real life experience because people's names should not be different in some way consistently for CEOs and non-CEOs.

Sample Preparation

I used the whole sentence of the person's name entity and the person's name entity itself to generate features.

Positive samples are the entities that are marked as PERSON and also exist in the provided labels. Negative samples are all the other PERSON entities. Since there are more negative samples, I also used oversampling positive samples as the method to balance the two categories.

Model used: Logistic Regression (2-class classification)

I also selected the final 6 features through feature exploration and trial and errors. Here are the final 6 features used (it turns out that information from name itself is not useful for the model):

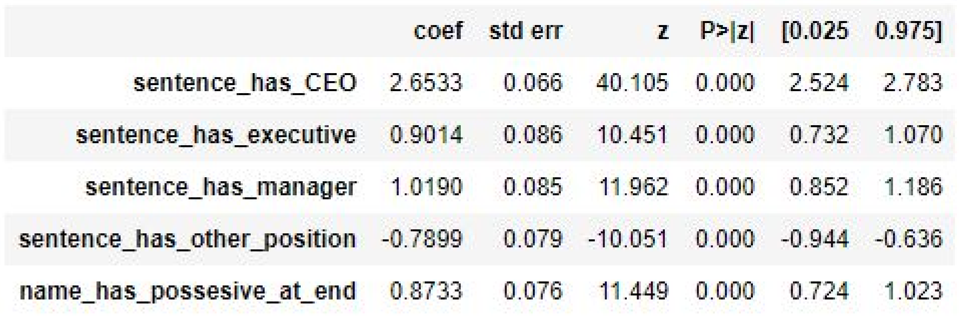
Feature Used:

Features (all binary)	Explanation
sentence_has_CEO	The sentence that the person's name entity is in has a string 'CEO' in the sentence.
sentence_has_executive	The sentence that the person's name entity is in has a string 'E(e)xecutive(s)' in the sentence.
sentence_has_manager	The sentence that the person's name entity is in has a string 'M(m)anager(s)' in the sentence.

sentence_has_other_position	The sentence that the person's name entity is in has another position string ('analyst', 'strategist', capital and plural forms considered) in the sentence.
name_has_possesive_at_end	The person's name entity has the possessive expression "'s" right at the end of the name.

These features are used through both observing some number of positive and negative samples, real life experience and result of the logistic regression model.

The trained regression model parameters are presented below and the positive/negative characteristics of the coefficients do match with the expectation based on real-life experience. The coefficient is negative only for sentence_has_other_position and that is expected. Also, sentence_has_CEO has much larger positive coefficient than others.



	coef	std err	z	P>|z|	[0.025	0.975]
sentence_has_CEO	2.6533	0.066	40.105	0.000	2.524	2.783
sentence_has_executive	0.9014	0.086	10.451	0.000	0.732	1.070
sentence_has_manager	1.0190	0.085	11.962	0.000	0.852	1.186
sentence_has_other_position	-0.7899	0.079	-10.051	0.000	-0.944	-0.636
name_has_possesive_at_end	0.8733	0.076	11.449	0.000	0.724	1.023

Another important evaluation is that I found around 76% of the records have the value false for all five features.

Thus, the logistic regression model will give 0.5 as predicted value for these records. I decide to classify these records as non-CEO record because of the following reason:

1. Many names appear multiple times among all the articles. Thus, if a CEO's name is captured just once through the whole process, this name will be extracted and marked as a CEO name and I can go back to find all the sentences with the name. Thus, I would like to avoid identifying a name as CEO's wrong (prefer very low False Negative rate).

Model Performance Evaluation

1. Confusion Matrix on the test set (using > 0.5 as criteria for positive predict label):

	Actual Positive	Actual Negative
Predict Positive	12662	2171
Predict Negative	22052	32710

   The false negative rate is about 0.14, which is much smaller than 0.49 from using >= 0.5 as criteria for positive predict label.

2. On the test set, the model has a F1 score of 0.51 on the testing set for categorizing positive labels. This is low but it meets our need better as mentioned (the criteria >= 0.5 gives F1 score 0.66 but it is less appropriate for our need)

3. I checked what proportion of the provided labels are covered by the extracted names from the model (inference on the whole dataset), about 81.5% is covered.

4. Following Prof. Klabjan's suggestion, I selected 10 random extracted company names to evaluate. I found 7 to be possible CEO, 3 that are not CEOs. 1 of the 7 possible CEOs has some other parts not related to the name included in the name. The 3 non-CEOs are in fact some managers or senior people but not CEO.

In summary, the performance of the model is not excellent. It does extract most CEOs but still has inaccurate extractions.

All the extractor CEO names are in 'extracted CEOs.xlsx'

Percent Extraction

I started from looking at the provided percent labels and found three types of wrong labels so I excluded them in the later part. I also excluded some labels that do not exist in the whole text.

1. Pure numbers in digit format (eg. 50, 3.34)
2. Pure numbers in english words format (eg. thirty one)
3. Label has quote signs in it (‘ or “) (eg. “50%”)

To extract all percentages, I took another approach different from the above two. Since percentages have a limited number of formats to catch, Instead of building a logistic regression with generated features, I implemented two methods to capture different needs and make sure I extract as much percentage out as possible.

Method 1

Firstly, I built a method using regular expressions checking. I go over the whole text and look for the following four patterns:

1. Any connected part of characters that ends with % sign, take the sign and every character before the % sign that is among 0-9, a-z, A-Z or ‘+’, ‘-’, ‘.’, ‘,’, ‘/’ signs.
   re.findall('[0-9+-.,/a-zA-Z]*%', article_str)

2. Any connected part of characters that has ‘percent’ in it, take the whole word with ‘percent’ and the word in front of it that is among 0-9, a-z, A-Z or ‘+’, ‘-’, ‘.’, ‘,’, ‘/’ signs.
   re.findall("[0-9+-.,/a-zA-Z]* ?percent+[a-zA-Z]*", article_str)

3. Any connected part with structure ‘...percent… point(s)’, take the whole word with ‘percent’ and the next word ‘point(s) and the word in front with chars among 0-9, a-z, A-Z or ‘+’, ‘-’, ‘.’, ‘,’, ‘/’ signs.
   re.findall('[0-9+-.,/a-zA-Z]*\s?percent+[a-zA-Z]*\s?(?:point|Point)s?', article_str)

4. Any connected part with structure ‘of a ...percent… point(s)’ take the whole part and one word in front of ‘of’ with achras among 0-9, a-z, A-Z or ‘+’, ‘-’, ‘.’, ‘,’, ‘/’ signs.
   re.findall('[0-9+-.,/a-zA-Z]*\sof\sa\s?percent+[a-zA-Z]*\s?(?:point|Point)s?', article_str)

The extracted list from this method is in 'all_extract_percent_re.xlsx'

The final extracted percent covered about 99.7% of the provided percent labels. Only several special cases or wrong cases are not captured.

Method 2

If the accurate information about the percentage is important, for example, if originally, the text is ‘between 80% and 85%’, Method 1 will extract the two numbers separately. I would like to have a method that keeps all important information.

Thus, method 2 uses, spacy NER tagger. Firstly it takes all the parts that is marked as ‘PERCENT’ by the tagger. Secondly, it also complements by looking at each part is tagged as ‘CARDINAL’ by the NER tagger and look around to see whether there is ‘percent’ or ‘%’ sign in the word on the left or right. If so, it will also extract the whole part.

Thus, method 2 could extract parts like ‘around 45%’, ‘about Thirty one percents’, which are not possible through method 1.

The extracted list from this method 2 is in 'all_extract_percent.xlsx' as a complement for the result from method 1 when information around percentage is important.