Roger Mei

Numbers involving percentages:

Regular expressions:

```
percentage_pat1 = r'((?:\(?-?\w*[,\d\/]*(\.\d+)? (to |- ))?\(?-?\w*[,\d\/]*(\.\d+)?
?percent(?:age)?(?:ile)?(?: points?)?\)?)'
percentage_pat2 = r'((?:\(?-?\w*[,\d\/%]*(\.\d+)?%? (to |- ))?\(?-?\w*[,\d\/]*(\.\d+)? ?%\)?)'
```

-Captures examples such as: one percent/1.0 percent/one.0 percent/1,000,000.00 percent/ (-5,000,000/2) percentage/ 1 - 2 percentile/one %/1.0 %/-5,000,000.0% / (50.0%)/ (5,000,000%)/ one to two %/ (-1/2 to 2/232,23013.2 %)/ 1% to 2%/ one percentage point/ two – three percentile points

```
percentage_pat3 = r'\w+[,\w\/\.]*-\w+[,\w\/\.]* ?%'
percentage_pat4 = r'\w+[,\w\/\.]*-\w+[,\w\/\.]* ?percent(?:age)?(?:ile)? (?: points?)?\)?)'
```

- Captures percentage_pat1 and percentage_pat2 without spaces such as 0.00-0.25%/ 5,000,000-6,000,000%/ 6-1/2 %/ forty-one percent/single-digit percent/50-80 percent/0.00-0.25 percentile

```
percentage_pat5 = r'point \w+ percent(?:age)?(?:ile)?'
```

-Captures examples such as: point five percentage/ point six percentile/ point seven percent

```
percentage_pat6 = r'\w+[\.\/\w,]* of a percent(?:age)?(?:ile)? ?(?:point)?'
```

-Captures examples such as: half of a percentage point/ three-quarters of a percentage point

Filtering:

Out of all matches that are two words in length, remove any matches that begin with a stop word such as “the percent”. Out of all two-word length matches, keep only the matches that begin with a cardinal number ('CD').

For matches that are 3 words in length, remove any matches that begin with a stop word or don’t begin with a cardinal number (‘CD’). Some matches begin with an adjective ('JJ') such as same, fixed, small, and big. However, adjectives with '-' in it such as 'fifty-five' are kept.

All other matches longer than 3 words in length such as “half of a percentage point” are kept.

Names of CEO's:

Regular expression:

capitalized_regex = r'([A-Z][A-Za-z\.]+(?=\s[A-Z])(?:\s[A-Z][A-Za-z\.]+)+)'

-Captures consecutive words that begin with a capital letter such as: Mark Zuckerberg/ Steve Jobs/ Elon Musk

Pre-processing:

196 negative examples were manually extracted from the corpus and saved into the text file, "neg_ceo.txt."

For positive examples, names of CEO's from the training data file "ceo.csv" is extracted from the corpus with the following regular expressions:

re.escape(full_name) where full_name is the first name followed by a space and the last name such as Elon Musk

re.escape(last_first) where last_first is the last name followed by a comma, space, and the last name such as Musk, Elon

re.escape(first_name) where first_name is only the text in the first column of "ceo.csv" since some examples have both first and last name in the same entry.

196 random samples are then chosen as positive examples for training the model.

Filtering:

To make classification easier with more separable data, only the examples with two or more words are kept such as "Warren Buffet" as opposed to just "Warren" in the positive training data. Initially, 255 samples were randomly selected but after filtering, conveniently, 196 samples were kept.

Normalization:

For the training set, 145 positive and 149 negative examples were used. This training set is normalized first (subtract mean and divide by standard deviation) before fitting the model. The test set is also normalized with the mean and standard deviation of the training set.

Model: Logistic Regression

Examples are in the tuple form (sentence, match), where sentence is the sentence in which the match is found in and the match is the match returned from the regular expressions.

The training set contains 294 total examples with 145 positive and 149 negative (49/51 split).

Features:

'is_NNP' – binary; whether all words in match are proper nouns ('NNP')

'num_NNP' – numeric; count of number of proper nouns in sentence ('NNP')

- In sentences with CEO names, there are also names of their companies.

'contains_CEO' – binary; whether the sentence contains "CEO"

'is_stop' – binary; whether there is a stop word in the match

'match_len' – numeric; number of letters in the match

- CEO names tend to be shorter than proper nouns of organizations as in Elon Musk vs Price Waterhouse Coopers

'num_words' – numeric; number of words in the match

- CEO names usually have only 2 words as opposed to organizations such as Bayerische Motoren Werke AG

'followed_by_verb' – binary; whether the match is followed by a verb in the sentence
- CEOs are people that can perform actions as opposed to other proper nouns like organizations and locations that can't

'contains_digit' – binary; whether or not the sentence contains a cardinal digit ('CD') tag
- Sentences with CEOs usually also talk about their company and the numbers associated with their business such as revenue and stock price

'contains_company' – binary; whether or not the sentence contains 'compan' (stem of company)
- Sentence with CEO names may also talk about their companies.

'contains_head' – binary; whether the sentence contains 'head' (CEO is head of company)

'contains_illion' – binary; whether the sentence contains 'illion' as in 'million' or 'billion'

'contains_pay' – binary; whether the sentence contains 'pay' as in 'pay cut'

'contains_investor' – binary; whether the sentence contains 'investor'

'contains_quarter' – binary; whether the sentence contains 'quarter'

'contains_boss' – binary; whether the sentence contains 'boss'

'contains_executive' – binary; whether the sentence contains 'executive'

Performance of model:

The test set contains 98 total examples with 51 positive and 47 negative (52/48 split).

Confusion Matrix:

```
[[28 19]
 [ 1 50]]
Accuracy of logistic regression classifier on test set: 0.80
Precision of logistic regression classifier on test set: 0.72
Recall of logistic regression classifier on test set: 0.98
```

Company names:

Regular expressions:

capitalized_regex = r'([A-Z][A-Za-z\.]+(?=\s[A-Z])(?:\s[A-Z][A-Za-z\.]+)+)'

-Captures consecutive words that begin with a capital letter such as: Google.com Inc./ Boston Consulting Group / The Boring Company / Amazon.com Inc.

single_capitalized = r'([A-Z][A-Za-z\.]+)(?:(?:\s[a-z]+)|\.)'

-Captures single words that begin with a capitalized letter and followed by another word that begins with a lower case letter such as in: (Apple) is a big company. / (Overstock.com) sells furniture. / (Samsung) and (Huawei) make phones.

year_regex = r'\d{4}'

-Captures 4 consecutive digits as in '2014'

Pre-processing:

151 negative examples were manually extracted from the corpus and saved into the text file "neg_company.txt"

For positive examples, company names from the training data file "companies.csv" is extracted from the corpus with the following regular expression:

re.escape(company_name) where company_name is the name of the company give in the first column of the "companies.csv" file

151 random samples were then chosen as positive examples for the training data.

Filtering:

For all captured single capitalized words, only words that are not pronouns ('PRP' tag) and not stop words are kept.

Normalization:

For the training set, 113 positive and 113 negative examples were used. This training set is normalized first (subtract mean and divide by standard deviation) before fitting the model. The test set is also normalized with the mean and standard deviation of the training set.

Model: Logistic Regression

Examples are in the tuple form (sentence, match), where sentence is the sentence in which the match is found in and the match is the match returned from the regular expressions.

The training set contains 226 total examples with 113 positive and 113 negative (50/50 split).

Features:

'inc_co_sent' – binary; whether 'inc' or 'co' is in the sentence

'inc_co_match' – binary; whether 'inc' or 'co' is in the match

'dollar_sign_sent' – binary; whether '$' is in the sentence

'percent_sent' – binary; whether '%' or 'percent' is in the sentence

'year_sent' – binary; whether a year is in the sentenced based on the 'year_regex' regular expression

'num_CD' – numeric; number of cardinal digit tags ('CD') in the sentence

'compan_sent' – binary; whether 'compan' is in the sentence

'num_NNP_match' – numeric; number of proper noun tags ('NNP') in the match

'stock_sent' – binary; whether 'stock' is in the sentence

'quarter_sent' – binary; whether 'quarter' is in the sentence

‘revenue_sent – binary; whether ‘revenue’ is in the sentence
‘share_sent’ – binary; whether ‘share’ is in the sentence
‘illion_sent’ – binary; whether ‘illion’ is in the sentence such as in ‘million’ or ‘billion’

Performance of model:

The test set contains 76 total examples with 38 positive and 38 negative (50/50 split).

Confusion matrix:

```
[[33  5]
 [ 5 33]]
Accuracy of logistic regression classifier on test set: 0.87
Precision of logistic regression classifier on test set: 0.87
Recall of logistic regression classifier on test set: 0.87
```