Feature Engineering

Natural Language Processing

University of Maryland

Discovery
In the data directory for nlp-hw repo, there are files called inclassFeateng_train.csv (and likewise for dev, test).
Load Packages

```python
import pandas as pd
import numpy as np
import sklearn
from sklearn.utils.validation import check_is_fitted
from sklearn.exceptions import NotFittedError
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction import DictVectorizer
```
Create function to generate features

```python
def extract_features(sentence_list):
    d = {}
    d["length"] = len(sentence_list)
    return d
```
vec = DictVectorizer()
X = {}
Y = {}
df = {}
for fold in ["train", "dev", "test"]:
    df[fold] = pd.read_csv("%s.csv" % fold)
    Y[fold] = df[fold]["label"]
    try:
        check_is_fitted(vec)
        X[fold] = vec.transform(extract_features(x.split())
        for x in df[fold]["text"])
    except NotFittedError as exc:
        X[fold] = vec.fit_transform(extract_features(x.split())
        for x in df[fold]["text"])

Load Data
Train Classifier

classifier = LogisticRegression()
classifier.fit(X["train"], Y["train"])
coef = pd.DataFrame(zip(vectorizer.feature_names_,
    np.transpose(classifier.coef_)),
    columns=['features', 'coef'])
coef.loc[len(coef.index)] = ['Intercept',
    classifier.intercept_]
print(coef.head())

print("Accuracy: \%f" %
    classifier.score(X["dev"], Y["dev"]))
### Results

<table>
<thead>
<tr>
<th>features</th>
<th>coef</th>
</tr>
</thead>
<tbody>
<tr>
<td>length</td>
<td>0.009380652536927882</td>
</tr>
<tr>
<td>Intercept</td>
<td>-0.11619382668652307</td>
</tr>
</tbody>
</table>

Accuracy: 0.485398
Results

<table>
<thead>
<tr>
<th>features</th>
<th>coef</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 length</td>
<td>[0.009380652536927882]</td>
</tr>
<tr>
<td>1 Intercept</td>
<td>[-0.11619382668652307]</td>
</tr>
</tbody>
</table>

Accuracy: 0.485398

Not great! (Roughly balanced dataset)
Look at Errors

df["dev"]["error"] = (df["dev"]["label"].astype(int) -
    classifier.predict_proba(X["dev"])[:,1])
df["dev"].sort_values(by='error', key=abs,
    ascending=False, inplace=True)
print(df["dev"].head())
num_rows = 0
for index, row in df["dev"].iterrows():
    num_rows += 1
    print(row["error"], len(row["text"].split()),
    row["text"])
    if num_rows > 10:
        break
I'm sending you a couple of customers — yeah — just get them out of my hair and keep them out — I don't give a damn what you tell them — only don't believe a word they say — they're out to make trouble for me and it is up to you to stop them — I don't care how — and one more thing — Cate's Cafe closed at eleven like always last night and Rose and Clarence Corsi left for Quebec yesterday — some shrine or other — I think it was called Saint Simon's — yeah, yesterday.

Two millions were added to what had been set aside for it in Mrs. Meeker's lifetime, and the proviso made that as long as Brian Thayer continued to discharge his duties as administrator of the ... his present capacity at a salary commensurate with the increased responsibilities enlargement of the fund would entail.

Like Eliot, in my fantasies, I had a proud bearing and, with a skill that was vaguely continental, I would lead Jessica through an evening of dancing and handsome descriptions of my newest exploits, would guide her gently to the night's climax which, in my dreams, was always represented by our almost suffocating one another to death with deep, moist kisses burning with love.

So, for happy years, Helva scooted around in her shell with her classmates, playing such games as Stall, Power-Seek, studying her lessons in trajectory, propulsion techniques, computation, logistics, mental hygiene, basic alien psychology, philology, ... , traffic, codes: all the et ceteras that eventually became compounded into a reasoning, logical, informed citizen.

When he was bent over behind the wheel of the station wagon, feeling in his trouser cuffs for the ignition key which he had dropped a moment before, she came out of the house with an enormous Rumanian shawl over her head, which she had bought in that country during one of their trips abroad, and handed him a clean handkerchief through the window.
Notice anything?

- The examples are long, sure, but is there anything suspicious about their exact length?
- Add in new feature and see if it helps...
- Remember that you need to redo `fit_transform`
More hints

- Can perfectly reconstruct with four features
- Two features you can figure out by using all words as features and seeing patterns on the top words
- Pay attention to how words are written
- Pay attention to what words mean
- Pay attention to the order of words
Even more hints

1. Feature based on length has positive weight, but isn’t monotonic... but is monotonic on the length of something related to the length. Do this feature first.

2. Feature based on how words are written has a positive weight and is easy to compute.

3. Feature based on what words mean has a negative weight and will require using a dictionary to get it perfectly right. Consider using wordnet from nltk

4. The value of the feature based on the order of words is around 0.5 for most sentences and ranges between 0.0 and 0.1, and the weight of the feature is negative. I’d recommend looking for this feature last.
You have a logistic regression classifier that takes in words and then classifies them. What three features would explain the following inputs and probabilities?

| Word  | \( p(y|x) \) |
|-------|---------------|
| short | 0.11          |
| hot   | 0.26          |
| saving| 0.26          |
| taking| 0.5           |
| surely| 0.73          |
| bigly | 0.88          |
Solution

- Intercept: -1
- Starts with “s”: -1
- Ends with “ing”: +1
- Ends with “ly”: +3