statistical classifier #1

Vectorization process:

Using k=5 sliding window to create a single vector / instance, from a single utterance of the corpus.

Shorter vectors, i.e.: the beginning of each utterance is padded with 0, see below.

**Features**: pos tag, ipa length, token length, is cognate (binary)

Vector example:

The utterance is: “porque it 's a lot cheaper”

The vectors created:

porque it (2 tokens)

lng: ['spa', 'eng']

len vec: [0, 0, 0, 6, 2]

postag vec: [0, 0, 0, 6, 8]

binary cognate vec: [0, 0, 0, 0, 0]

label: 1

porque it 's (3 tokens)

lng: ['spa', 'eng', 'eng']

len vec: [0, 0, 6, 2, 1]

postag vec: [0, 0, 6, 8, 7]

binary cognate vec: [0, 0, 0, 0, 0]

label: 0

porque it 's a (4 tokens)

lng: ['spa', 'eng', 'eng', 'eng']

len vec: [0, 6, 2, 1, 1]

postag vec: [0, 6, 8, 7, 12]

binary cognate vec: [0, 0, 0, 0, 0]

label: 0

porque it 's a lot

lng: ['spa', 'eng', 'eng', 'eng', 'eng']

len vec: [6, 2, 1, 1, 3]

postag vec: [6, 8, 7, 12, 11]

binary cognate vec: [0, 0, 0, 0, 0]

label: 0

it 's a lot cheaper

lng: ['eng', 'eng', 'eng', 'eng', 'eng']

len vec: [2, 1, 1, 3, 7]

postag vec: [8, 7, 12, 11, 1]

binary cognate vec: [0, 0, 0, 0, 0]

label: 0

data:

sampled 2799 utterances from the whole corpus (which contains 42910), that are candidates for code-switching, by **filtering** those with at least 2 languages (es, en, shared other, shared eng, shared es)

cs labeling:

a vector / instance is labeled 1 (cs) when the 5th token and the 4th token have opposite language, Spanish and English. Ignoring the shared.

Or the 4th token is tagged with “shared other” and 5th and 3rd tokens are again opposite, English or Spanish.

Instances for classification:

2799 utterances were converted to **28059** vectors (using k=5):

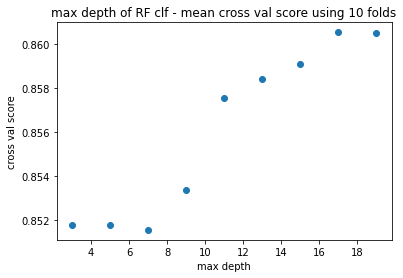
4159 with cs, and 23900 w/o.

Majority classifier:

A majority cls would vote 0 all the time, giving accuracy of 23900/28059 = 0.851

Random forest:

Fitting a random forest (with entropy), using cross validation score with 10 folds:



The best clf has **0.86** cross validation score**. 0.009** better then the majority cls.

Fitting a random forest on a stratified splitting of 70-30 train-test results in:

Chart, treemap chart

Description automatically generated Chart, bar chart

Description automatically generated

1. We can see that some samples are predicted with 1, in contrast to the majority cls.
2. No surprise that pos tag 3 and 4 (the 3rd and 4th tokens) are the most important features.
3. Binary cognates are less important.

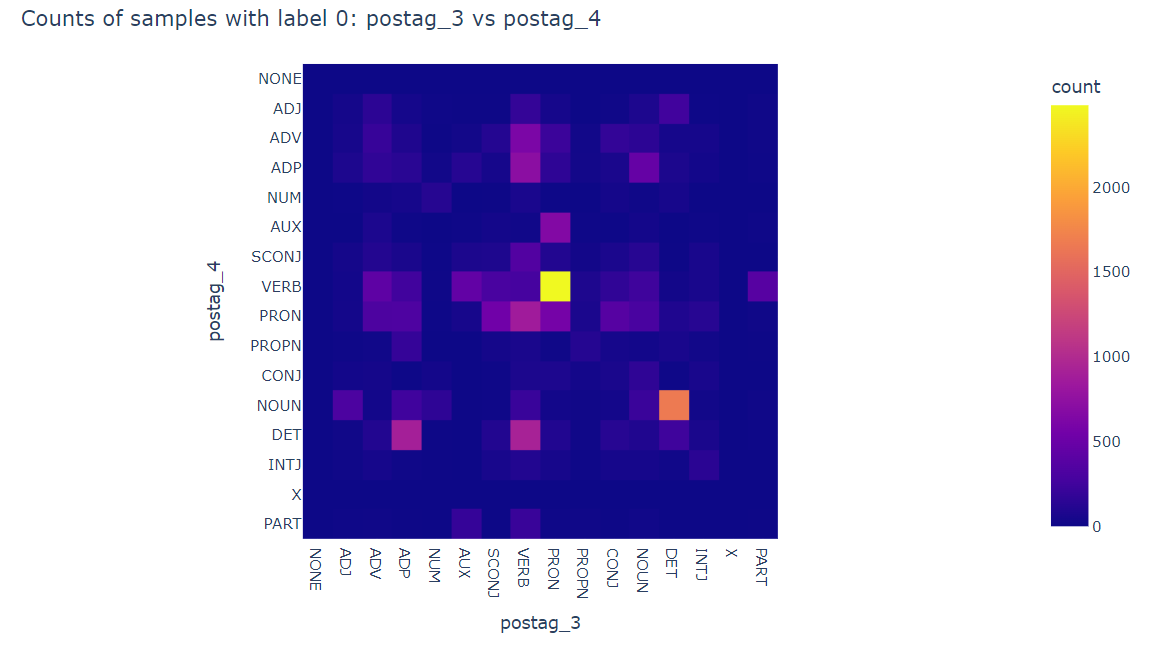
Data analysis:

Setting a side, the classifier for a moment, and looking at the 28k vectors we have.

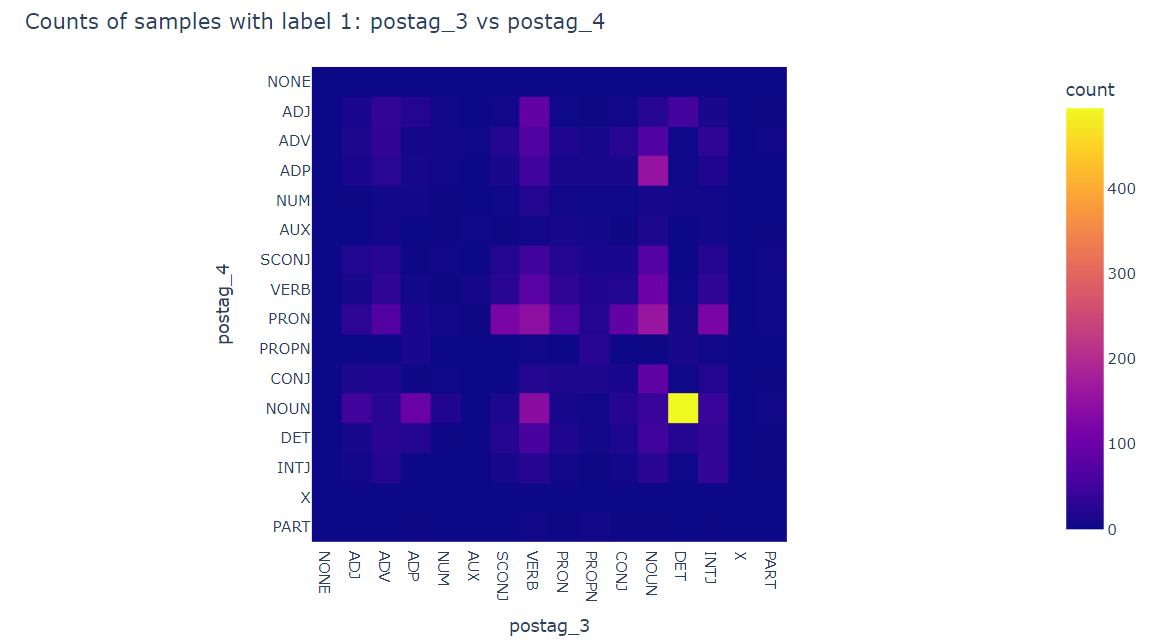
(Reminder: only 14% are CS).

The figures below will show the distribution of samples (vectors) based on their pos tag 3 and 4, which are the most important features for the classification we performed above.

For example, looking only at the samples with label 0, the yellow square has count value of 2487 – meaning, 2487 samples that are labeled 0, has pos tag 3 = PRON, and pos tag 4 = VERB.



Same map for samples with label 1:

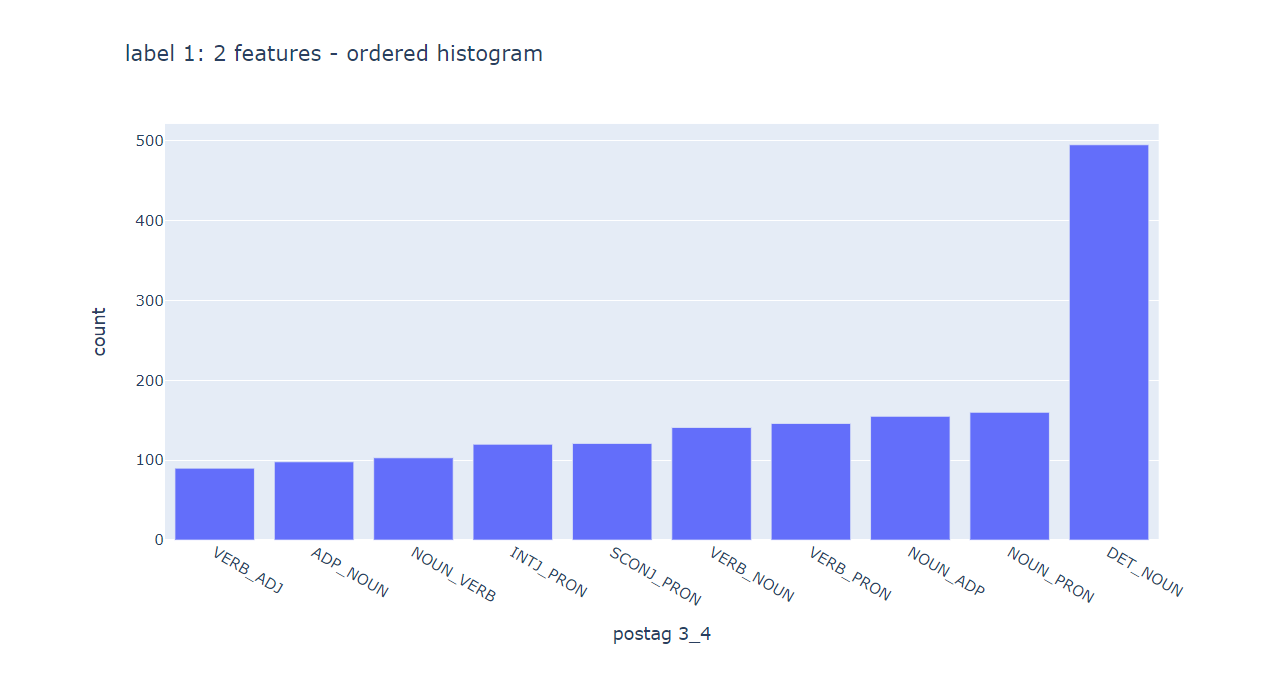


Interesting: the squares where “label 1 > label 0” even though there are only 14% 1’s:

We have a binary map, 1 means that there are more samples labeled with 1:

Chart, histogram

Description automatically generated



Chart, bar chart

Description automatically generated

Let’s explore those yellow squares above where label 1 > label 0, and highlight the examples where the count value of is high:

pos4: ADJ pos3: SCONJ

pos count: 4.0

neg count: 3.0

pos4: ADJ pos3: INTJ

pos count: 12.0

neg count: 6.0

pos4: ADV pos3: NUM

pos count: 4.0

neg count: 3.0

pos4: NUM pos3: INTJ

pos count: 3.0

neg count: 1.0

pos4: SCONJ pos3: PART

pos count: 3.0

neg count: 2.0

pos4: CONJ pos3: PART

pos count: 1.0

neg count: 0.0

pos4: NOUN pos3: SCONJ

pos count: 15.0

neg count: 10.0

pos4: NOUN pos3: INTJ

pos count: 42.0

neg count: 23.0

pos4: PART pos3: INTJ

pos count: 1.0

neg count: 0.0

for example INTJ as pos tag 3, then NOUN as pos tag 4, is counted almost twice with samples that have CS.

statistical classifier #2

Vectorization process:

This time, we look only on the pos tag 3 and 4 and we want to see which tags are important more to classification. We will use 1 hot encoding technique:

Still using K=5 as a sliding window size.

There are 14 Tags:

{0: 'NONE', 1: 'X', 2: 'ADV', 3: 'ADJ', 4: 'NUM', 5: 'PRON', 6: 'AUX', 7: 'CONJ', 8: 'INTJ', 9: 'NOUN', 10: 'PART', 11: 'SCONJ', 12: 'ADP', 13: 'DET', 14: 'VERB', 15: 'PROPN'}

The ‘NONE’ tag is just the padding we add to vectors smaller than k.

Example:

Text: “and your”:

('and', 'eng', 'CONJ')

('your', 'eng', 'DET')

postag vec: [0, 0, 0, 7, 13]

label: 0

so, for pos tag 4, only 13 (DET) is on, and for pos tag 3, only 7 (CONJ) is on.

Graphical user interface, application

Description automatically generated

Classification results:

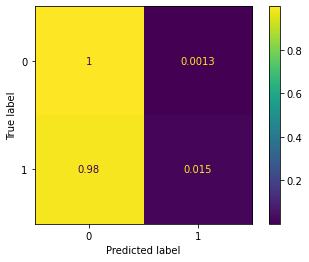
Again, fitting random forest

**The unbalanced case**: label 1: 4159, label 0: 23900

Majority cls accuracy: 0.851

Pos tag cls: cross val mean score: 0.852

70-30 splitting instance:

 Chart, funnel chart

Description automatically generated

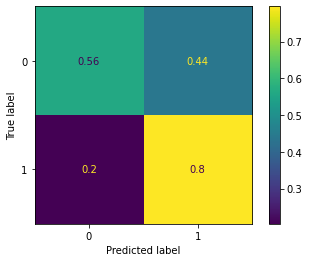
**The balanced case**: label 1: 4159, label 0: 4159

(I think this this the interesting case because it manages to predict 1’s – i.e. learn CS, so the feature important will tell us something on how that was done)

Majority cls accuracy: 0.5

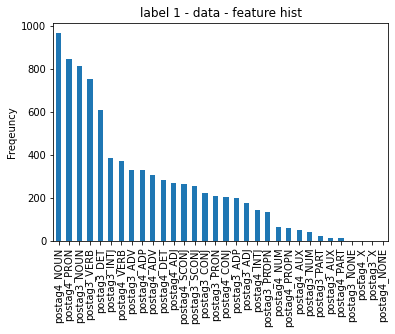
Pos tag cls: cross val mean score: 0.66

70-30 splitting instance:

 Chart, funnel chart

Description automatically generated

**Data**:

Chart, histogram

Description automatically generated

Chart, histogram

Description automatically generated

Validation check –most important feature are not the most common.

Adding NER:

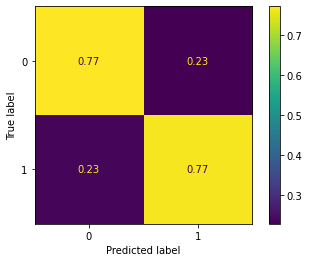
Used spacy models: <https://spacy.io/models/en> and <https://spacy.io/models/es>

Used the medium models (same scores – f1, prec, rec) as the large models.

Back to classifier / vectorizer #1 :

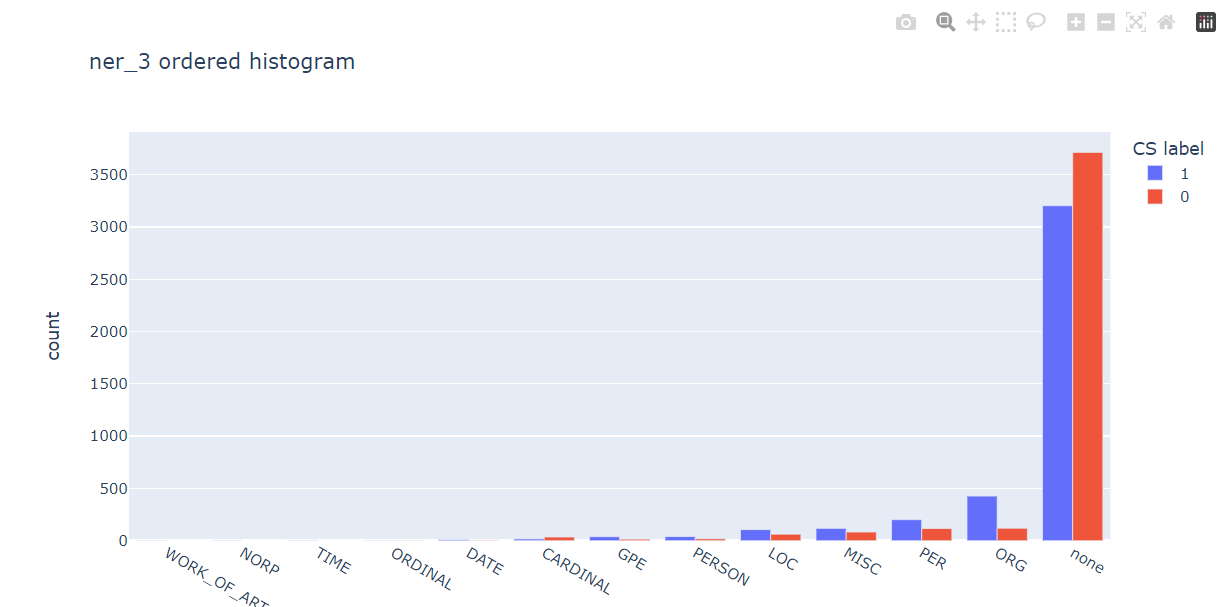
Balanced data:

Cross val score: 0.75

 Chart, bar chart

Description automatically generated

Ner 3 and 4 joins the party. + full plots in notebook.



Un balanced:

Cross val score: **0.868** (Majority cls accuracy: 0.851) – best so far

Chart, treemap chart

Description automatically generated

Using all corpus: (side experiment)

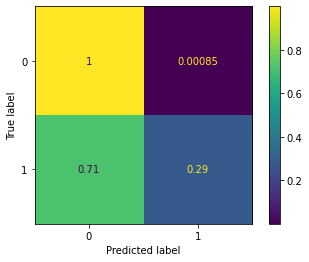
using all 42k utterances not just the 2799 candidates of CS (with more than 1 lang code).

Unbalanced case:

all: 247443, label: 0 241708, label: 1 5735

majority\_cls\_acc: 0.9768229450823018

cls (ner + postag + len + ipa + cogntaes) cross val score: = 0.9803550644312041

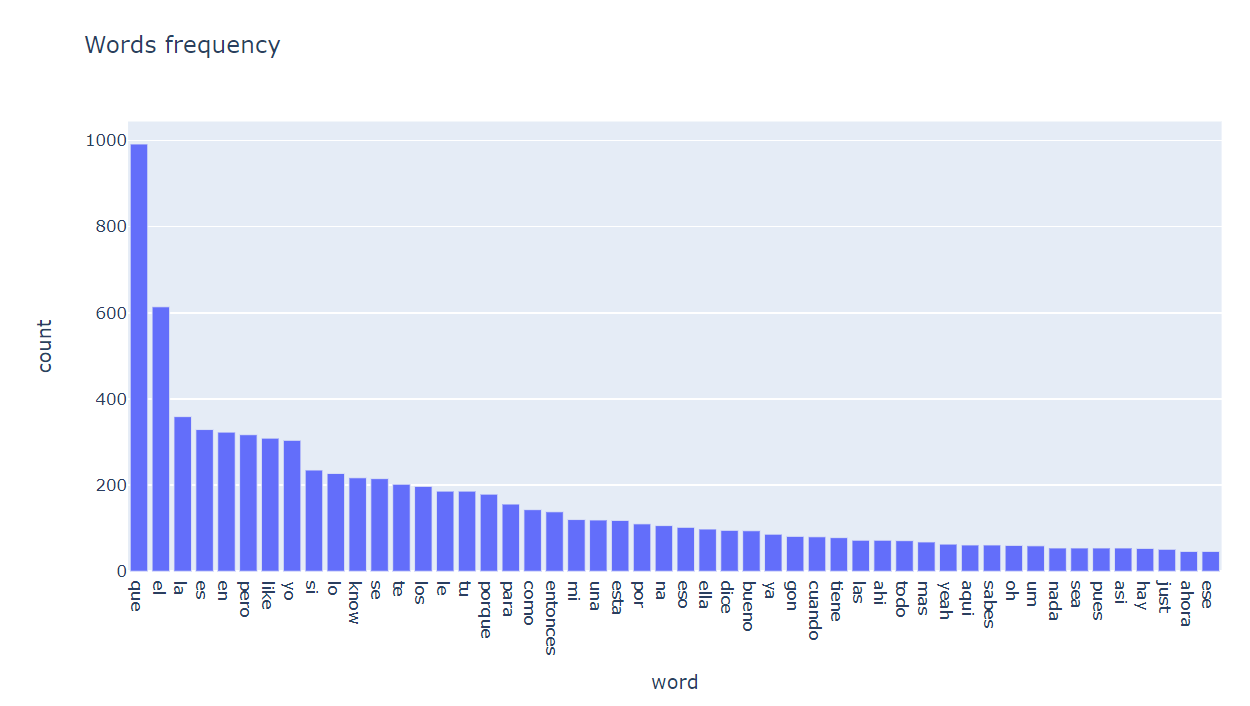


Not a big difference in results between taking all data (5k 1’s to 241k 0’s ) to the sub set previously used – i.e. I think we can use the small data set for experiments (hyper param tuning, other classification, feature combination etc… ) - much faster. And at the end, run those on the full dataset.

Adding bag of words:

adding this feature on top of all others (pos tag, ner etc)

vocabulary size (all features - 4313)



small experiment setup:

majority\_cls\_acc: 0.8517766135642753 in the unbalanced case

|  |  |  |
| --- | --- | --- |
| Max features | Un balanced cv res | Balanced cv res |
| 500 | 0.866 (top words: Un, el, harina, los) | 0.764 (top words: Un, los,na,gon, to) |
| All 4313 | To slow skipping |  |
| 30 | 0.870 (best so far) (words: un, es, los) | 0.762 |