Can You Polish Your Dutch?

Jacek Pardyak

November 8th, 2017

Outline

- Business understanding
- Data understanding
- Data preparation
- Model construction
- Model evaluation
- Insights from the data

Business understanding

To communicate people use words. Words are composed of letters over an alphabet.

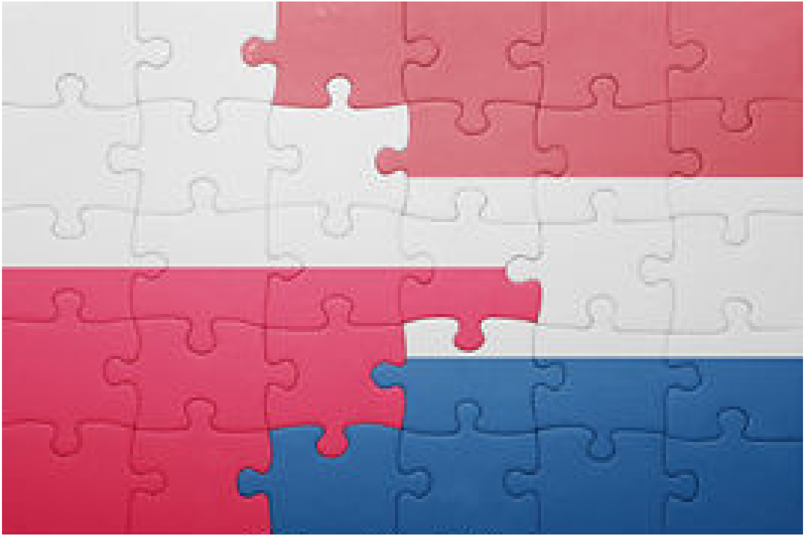
Letters common for the two languages: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, r, s, t, u, w, y, z.

Letters exclusively used in Polish: ą, ć, ę, ł, ń, ó, ś, ź, ż

Letters exclusively used in Dutch: q, v, x

Problem understanding

Build a model which distinguishes Polish words from Dutch.



Eventually find (dis)similarities of the two languages.

Data understanding

To train and test models we use **Aspell** dictionaries.

Sample of the Polish data:

nieprzegadywanie/UV
snowboard/NQsT
najkompletniej
Komiaczka/MmN
synonimowy/bXxY
ziębł

Polish is inflected language. Symbols after **/** are used to mark affixes.

Data understanding cont.

Sample of the Dutch data:

aanbrei
lichtwaterreactor
ouderhuis
inhoudstafels
selectiekamer
linkerhelft

We use 341461 Dutch and 289840 Polish words. After applying 7k inflection rules number of words in Polish grows to 3.8M.

Data preparation

1. Remove everything after "/" sign

```
words$Word <- gsub("\\/.*","",words$Word)
```

2. Split words into list of letters

```
words$Word_split <- lapply(words$Word, function(x) {
  paste(unlist(strsplit(x, "")), collapse = " ")})
```

3. Coerce list of words into "document-term-matrix"

```
dtm <- DocumentTermMatrix(Corpus(VectorSource(
  words$Word_split)), control = list(
    tokenize = UnicodeTokenizer, wordLengths = c(1,2)))
```

Data preparation cont.

For example words:

```
## [1] "Adrianna" "Rea"
```

are represented as:

```
## <<DocumentTermMatrix (documents: 2, terms: 40)>>
## Non-/sparse entries: 8/72
## Sparsity           : 90%
## Maximal term length: 2
## Weighting          : term frequency (tf)
## Sample             :
##        Terms
## Docs    a b c d e i n p r s
##   311   3 0 0 1 0 1 2 0 1 0
##   31197 1 0 0 0 1 0 0 0 1 0
```

Data preparation cont.

4. Create output variable of two classes (1 for Dutch word, 0 otherwise)
5. Split the data into training and test data sets (70/30)

Finally:

- `X_train` - matrix of 441910 rows and 40 columns stores input variables of training data set
- `Y_train` - vector of 441910 elements stores output variable of training data set
- `X_test` - matrix of 189391 rows and 40 columns stores input variables of test data set
- `Y_test` - vector of 189391 elements stores output variable of test data set

Model construction

To construct our first Deep Neural Network model we need to perform following steps:

- initialize the model,
- add layers to the model,
- compile and fit our model.

```r
# Load 'keras' - API to 'TensorFlow' engine
require(keras)
# Apply one-hot-bit encoding
Y_train <- to_categorical(Y_train)
# Construct an empty sequential model
# composed of a linear stack of layers
model <- keras_model_sequential()
```

Model construction cont.

```
model %>%
  # add a dense layer
  layer_dense(units = 500, input_shape = 40,
              kernel_initializer="glorot_uniform",
              activation="sigmoid") %>%
  # add dropout to prevent overfitting
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 300,
              kernel_initializer="glorot_uniform",
              activation="sigmoid")  %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 100,
              kernel_initializer="glorot_uniform",
              activation="sigmoid")  %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 2,
              kernel_initializer="glorot_uniform",
              activation="softmax")
```

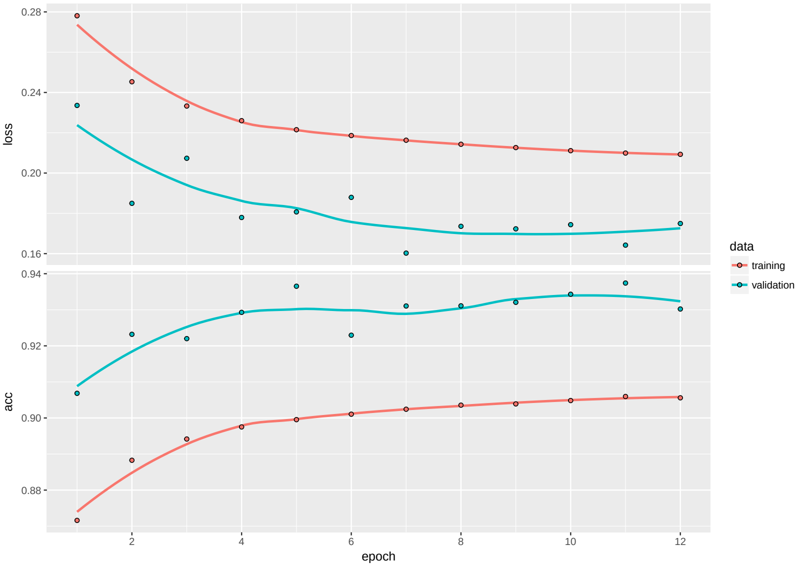
Model construction cont.

```
# Compile the model
model %>%
  compile(loss = 'categorical_crossentropy',
          optimizer = optimizer_adam(lr=0.001,
                                     beta_1=0.9,
                                     beta_2=0.999,
                                     epsilon=1e-08,
                                     decay=0.0),
          metrics = 'accuracy')
```

```
# Fit the model from the training data
training_history <- model %>%
  # batch_size - number of samples per gradient update
  # epochs - number of times to iterate on a dataset
  fit(X_train, Y_train, batch_size = 64,
      epochs = 12, verbose = 1,
      validation_split = 0.1)
```

Model construction cont.

Training history:



Model evaluation

```
# Make predictions on the test dataset
Y_test_hat <- model %>%
  predict_classes(X_test)

Y_test_hat <- as.integer(Y_test_hat)
```

```
##        Y_test_hat
## Y_test     0     1
##      0 78309  8592
##      1  8164 94326
```

```
## accuracy:  91.15269%
## precision:  91.65161%
## recall:     92.03434%
## f-measure:  91.84258%
```

Results

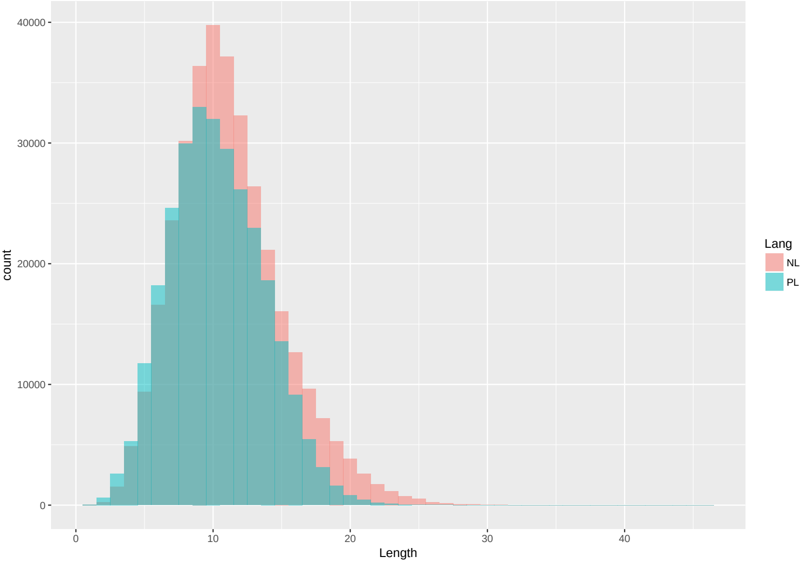
We applied Neural Networks to identify language of a word just using letter frequencies.

Performance of the model is very good - accuracy, precision and recall above 90%.

Further we try to understand this behavior.

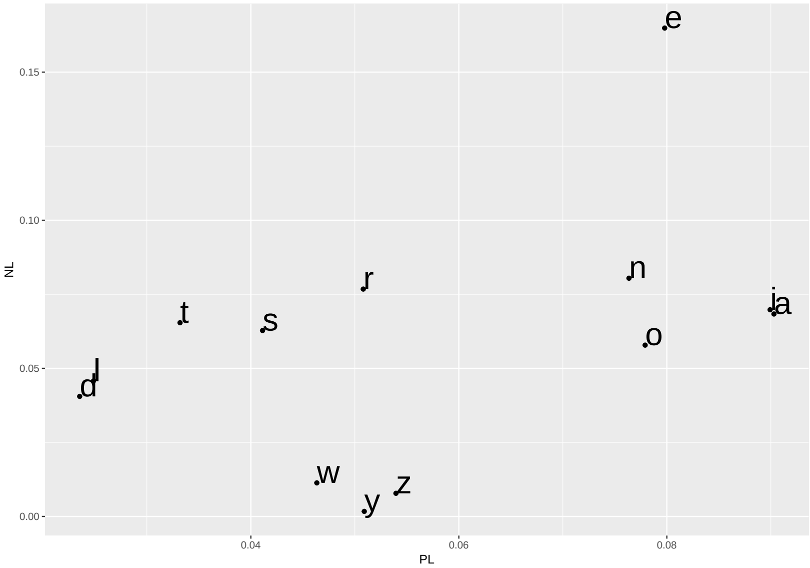
Insigths from data

Word length comparison:



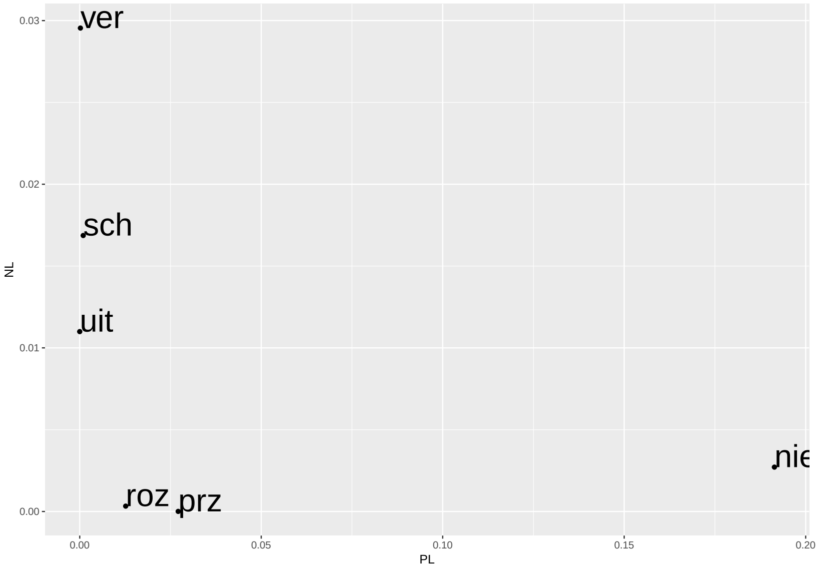
Insigths from data cont.

Relative letter frequency:



Insigths from data cont.

Relative frequency of initial trigrams:



Insigths from data cont.

Polish words perfectly and approximately matching Dutch words:

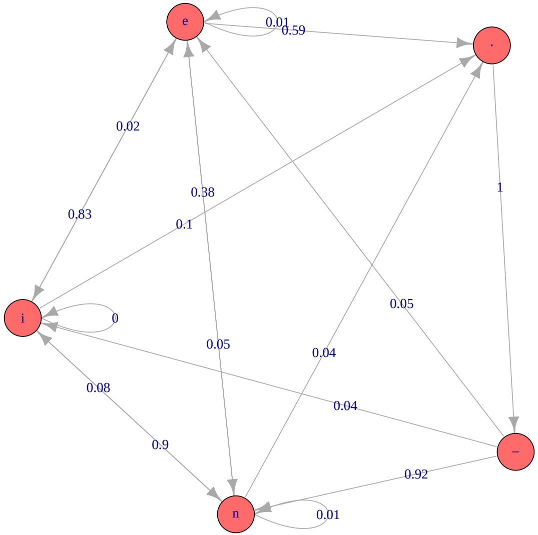
- ananas, balkon, chaos, duet, echo, filet, gratis, handel, impotent, jacht, kapsel, legenda, wiek and 3.3k more
- abiturient $\sim$ abituriënt, banan $\sim$ banaan, bestseler $\sim$ bestseller, dermatolog $\sim$ dermatoloog, fortepian $\sim$ fortepiano, wachta $\sim$ wacht and 2.6k more

Our DNN model was trained on words assigned to two different classes.

Watch 'false friends' - words spelled the same but meaning something different.

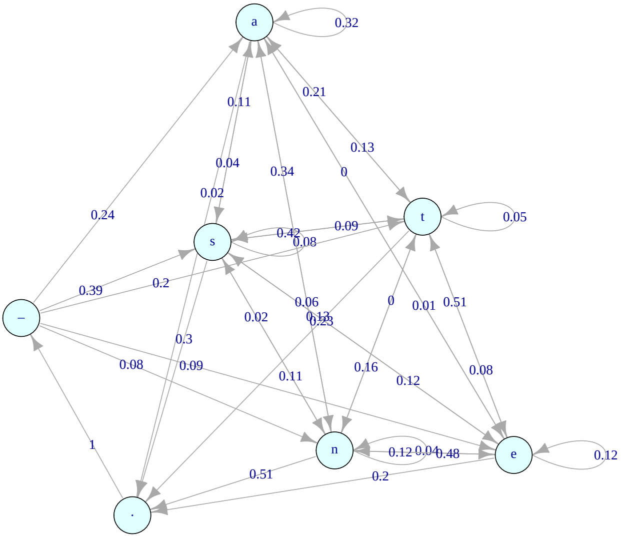
Insigths from data cont.

We can use Markov chains to build probabilistic model of a language. Excerpt of the Polish model:



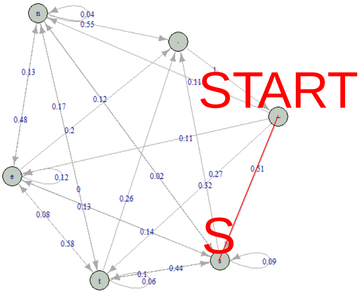
Insigths from data cont.

Excerpt of the Dutch language probabilistic model:



Insigths from data cont.

Animated most likely “random” walk through the Dutch graph:



Insigths from data cont.

Probabilistic language models can be used to generate 'synthetic' words:

- wypcy, ośm, donie, bonijny, tać, nionwry, szero, zberemy
- vevon, orin, veden, gaaauk, ilin, ommouin, pamoe, parle

Our model accurately recognized language of these synthetic words.

Summary

- Deep Learning is a very powerful technique
- Use of bi- and trigrams will lead to even better performance
- Dutch and Polish are dissimilar languages
- About 3% of words is commonly used in Polish and Dutch