Implementing a Turkish-English Spam Filter

Spam Mail Problem

We need to distinguish mails as spam ot non-spam(ham).

Lots of of techniques

- Hand crafted: Check phrases, domains, senders, whitelisting-blacklisting
- Automated: Machine learning methods

No technique is a complete solution.

Machine Learning Solution

A supervised machine learning method can do the work.

- Provide a dataset of spam and ham mails.
- Process mails with common and language-specific text processing methods.
- Train a Naive-Bayes classifier with spam/ham labels and tokens from processed texts.
- Bayesian Classifier will correlate the use of words in mails with spam/ham labels.

English Language Processing

NLTK has built-in support for stopwords, stemmer, tokenization, lemmatization.

Data source for English mails is **SpamAssassin** dataset, which stores mails in .eml format.

.eml format contains all data and metadata of a mail as well as attachments, signatures.

Python has an **email** module, which correctly handles .eml files.

Turkish Language Processing

Stopwords

https://github.com/ahmetax/trstop

Stemming

Snowball Stemmer has Turkish support.

Lemmatizer? Or a multi-purpose nlp library for Turkish?

trnltk: https://github.com/aliok/trnltk

Turkish Language Processing

trnltk is for **Python 2.x** and the project was integrated to **Zemberek**, which is a Java library.

How to use Zemberek jars in Python?

There are modules for calling .jar from Python: **jpy, jpype**

None of them works with Java8 code.

Final solution: Create a separate service in Java that stems and lemmatizes using Zemberek.

Turkish Language Processing

Turkish spam mail dataset: http://ceng.anadolu.edu.tr/par/

400 ham and 400 spam mails. Not sufficient. Poorly organized.

- Lots of incorrect spellings (“istermisin”)
- Same words with different spellings (“fırsat”, “firsat”)
- Wrong use of punctuations (e.g. endings without space: “...oldu.Sonra...”)
- No utf-8 encoding.

Causes lots of “unknown” response coming from lemmatizers.

Training

After preprocessing, extract features like [{{‘word1’: count1, ‘word2’:count2 }, spam},..]

Divide the data as train and test datas.

Train with **NLTK Naive Bayes Classifier.**

Classifier provides accuracy and most informative results.

Running the Filter

SpamFilter.py is the main script. There are two optional parameters.

>> **python SpamFilter.py [‘zemberek’] [‘bow’]**

If ‘zemberek’ parameter is applied, Turkish features will be extracted with Zemberek.

If ‘bow’ is applied, features will be extracted with word counts(bag-of-words) instead of boolean mode.

Input

User input must contain the body part of the mail.

To provide a multi-line input, a stopword is used to stop the input and start processing. (-q).

Since preprocessing text are specific to each language, language of the input must be known before classifying.

langId module for Python successfully guesses the language of a string.

Accuracy Results

Dataset	Preprocessor	Bow Accuracy (train / test)	Boolean Accuracy (train / test)
Turkish	Snowball	0.9984 / 0.8509	0.9984 / 0.8509
	Zemberek	0.9968 / 0.9068	0.9906 / 0.9068
English	nltk	0.9890 / 0.8267	0.9853 / 0.8299
Turkish + English	nltk + Snowball	**0.9893 / 0.8574**	0.9845 / 0.8555
	nltk + Zemberek	0.9863 / 0.8564	0.9802 / 0.8528

Further Improvement

- Extract HTML data properly.
- Classify mail sender, subject.
- Find a faster way to recognize input language.
- Find a faster way to process Turkish.
- Create features with n-grams.

https://github.com/cyurtoz/turkish_english_spammail_filter