Text Classification

Really Minimalistic Guide

Examples

- See first few slides of Stanford PDF
- Reuters21578, slide 54 there
- The question:
  - How do we feed that into a classifier ?

Pipeline

1. Load from CSV (Pandas, Matplotlib, Seaborn)
2. Explore – how many docs / words / classes
3. Turn text to vectors (how???)

   (scikit-learn, NLTK, gensim, spaCy, hand-crafted rules (!!!!!))
4. Pass it on to a few classifiers (scikit-learn)
5. Tune hyper-parameters and (scikit-learn)
6. cross-validate on different train/test splits

Bag of Words

1. I love dogs.

2. I hate dogs and knitting.

3. Knitting is my hobby and my passion.

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

example from [this post]

Bag of Words with TF-IDF

1. I love dogs.

2. I hate dogs and knitting.

3. Knitting is my hobby and my passion.

TF-IDF = TF * IDF

$$TF = \frac{\text{\# this term in this doc}}{\text{\# total terms in this doc}}$$

$$IDF = \ln \frac{\text{\# docs total}}{\text{\# docs with this term}}$$

TF-IDF is big if the term is frequent in this doc and rare overall

	A	B	C	D	E	F	G	H	I	J	K
1		I	love	dogs	hate	and	knitting	is	my	hobby	passion
2	1	1	1	1							
3	2	1		1	1	1	1	1			
4	3					1	1	1	1	1	1
5											
6		I	love	dogs	hate	and	knitting	is	my	hobby	passion
7	1	0,14	0,37	0,14	0,00		=D2/SUM($B2:$K2)*LN(3/SUM(D$2:D$4))				
8	2	0,07			0,18	0,07	0,07	0,07			
9	3					0,07	0,07	0,07	0,18	0,18	0,18

TF = 1 / 3

IDF = ln(3 / 2) = 0.4

TF-IDF = 0.135

Pre-processing

- Too many terms
  - 130 000 distinct in Reuters21578 if split on space

- Remove stop words
- Distinct numbers, dates and other entities
- Stem (Stanford says it does not help much, generally, but depends on task)
- Drop the long tail of very rare words
- Use n-grams to capture part of word order info

  bigrams: “I love big dogs” -> “I love”, “love big”, “big dogs”

Jupyter

- Demo of Reuters21578 dataset
  - All topics – only stats
  - Five topics – full flow and model comparison