Language and Computation

week 4, Thursday, February 6, 2014

Tamás Biró
Yale University
`tamas.biro@yale.edu`
`http://www.birot.hu/courses/2014-LC/`

Practical matters

- **Post-reading:** JM 23.1.1, 4.1-4.3
- **Pre-reading:** JM 5.1-5.4 (eventually: chapter 7)
- **Python:** this week H 3 and 4; next week H 5.
- Previous Problem Set
- **Next Problem Set:** published tomorrow, due Tu 02/18.
- Session: pseudo-codes.

Today

- Text classification
- Machine learning (evaluation metrics)
- N-grams
- Smoothing (basics)
- Probability (refresher)

Next time: Markov-models

Comparing documents with n-grams

Task: document categorization/classification

Many documents entering a news agency, to be classified by

- language
- topic
- author
- genre
- political preference

  etc.

Machine learning: the basic idea

- Given set X (e.g., of [possible] documents)
  Given set Y of tags (e.g., of languages, of topics, of authors, etc.)

- Unknown correct mapping $M^* : X \to Y$

- **Training set** of learning data $\{(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)\}$, where $(x_i, y_i) \in X \times Y$, $y_i = M^*(x_i)$,

- is employed to identify some mapping $M : X \to Y \in \mathcal{M}$

- such that M approximates M^* on the **test set** X':
  maximize performance on $|\{x \in X' | M(x) = M^*(x)\}|$.

Excursus: Evaluating a document classifier

a_{ij}: number of documents categorized by M as i, and being in reality (categorized by M^*) as j.

In reality:	English	French	Spanish
Categorized as English	a_{ee}	a_{ef}	a_{es}
Categorized as French	a_{fe}	a_{ff}	a_{fs}
Categorized as Spanish	a_{se}	a_{sf}	a_{ss}

$$\text{Accuracy} = \frac{a_{ee} + a_{ff} + a_{ss}}{\sum_{i,j \in \{e,f,s\}} a_{ij}}$$

Excursus: Evaluating a binary classifier

In reality:	positive	negative
Categorized as positive	true positives	false positives
Categorized as negative	false negatives	true negatives

$$\text{Accuracy} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fp} + \#\text{fn} + \#\text{tn}}$$

Excursus: Evaluating a binary classifier

In reality:	positive	negative
Categorized as positive	true positives	false positives
Categorized as negative	false negatives	true negatives

$$\text{Precision} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fp}}$$

$$\text{Recall} = \frac{\#\text{tp}}{\#\text{tp} + \#\text{fn}}$$

$$F\text{-measure} = \frac{2PR}{P + R}$$

The practice of doing Machine Learning

- Define task: text classification, disambiguation, parse selection, part-of-speech tagging, information retrieval, etc.
- Define your goal: which **evaluation metric** most important?
- Choose a training set/corpus and a test set/corpus.
- Choose a machine learning technique (entails $\mathcal{M}$)
- Go!

Back to text classification. A text as. . .

- a meaning, a message
- as a series of sentences
- a string of words
- a bag of words
- a series of n-grams:
  - a string of n characters / letters / words / etc.
  - overlapping or non-overlapping

Vector Space Models and the Cosine Metric

- $f(w_i, D)$: frequency of word / n-gram w_i in document D.
- Given document D, create vector $(f(w_1, D), f(w_2, D), \ldots f(w_n, D))$
- Distance of two vectors: use their **cosine distance** (normalized dot product):
$$d(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}}$$
- For each $y \in Y$, create reference vector D_y.
To categorize document D, find closest reference vector.

Vector Space Models

Document D and references D_y characterized by a vector of

- word frequencies, including / excluding **stopwords**
- character frequencies aka *unigrams* of words, letters. . .
- character *bigrams* frequencies
- word *bigram* frequencies
- trigrams, . . . n-grams (aka n-tuples)
- which do / do not overlap

Vector Space Models

How to create the reference vector D_y for each $y \in Y$?

- Best guess: from the training set.
- Optimal if training set $=$ test set.
- But what about **generalizability**?
- Goal: optimize on yet-unknown test set.
- Training set $\rightarrow$ held-out sets (and devset) $\rightarrow$ test-set.

Vector Space Models

How to create the reference vector D_y for each $y \in Y$?

- Best guess: from the training set.
- Word/letter/n-gram frequencies estimated from training set.
- The **sparse data problem**, as well as
- room for **unknown words** (aka **out-of-vocabulary words**)?
- Therefore, obtain a better approximation of the ideal D_y (the one "used" by M^*) by introducing **smoothing**.

Smoothing (overview only)

- N: corpus size (# of tokens)
  V: vocabulary size (# of types)
  c_i: count of word (type) w_i.

- Unsmoothed **Maximum Likelihood Estimate:**

$$P(w_i) = \frac{c_i}{N}$$

Smoothing (overview only)

- N: corpus size (# of tokens)
  V: vocabulary size (# of types, including those with 0 frequency!)
  c_i: count of word (type) w_i.

- **Laplace Smoothing** or **add-one smoothing:**

$$P_L(w_i) = \frac{c_i + 1}{N + V}$$

- as if we used $c_i^* = (c_i + 1)\frac{N}{N+V}$ in MLE,
  discounting c_i and reallocating probability mass to unseen words.

Smoothing (overview only)

- N: corpus size (# of tokens), c_i: count of word (type) w_i
  N_c: # of types that occur c times (frequency of frequency)
- **Good-Turing Smoothing/discounting**:

$$P_{GT}(w_i) = \frac{(c_i + 1)N_{c_i+1}}{NN_{c_i}}$$

- as if we used $c_i^* = (c_i + 1)\frac{N_{c_i+1}}{N_{c_i}}$ in MLE,
  discounting c_i and reallocating probability mass to unseen words.
- Count the hapaxes $\rightarrow$ estimate the count of types unseen in training: $P_{GT}(\text{unseen}) = N_1/N$.

From frequency to probability to scores

What is “probability” P?

- Observed frequency in the training set/corpus?
- Expected frequency in the test set/corpus?
- Expected frequency in the “entire” set/corpus X?
- A technicality that sums up to 1?

Basics of probability

- **Sample space**: possible outcomes of an experiment.
- **Event**: a subset of the sample space.
- Given set X of *events* (a *random variable*)),
- **probability** $P : X \to [0,1]$.
- $P(X) = 1$, $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$
- Conditional probability: $P(A|B) = P(A \wedge B)/P(B)$.

See you next week!