Introduction to Natural Language Processing I [Statistické metody zpracování přirozených jazyků I] (NPFL067)

http://ufal.mff.cuni.cz/courses/npfl067

prof. RNDr. Jan Hajič, Dr. / doc. RNDr. Pavel Pecina, Ph.D.
ÚFAL MFF UK
{hajic,pecina}@ufal.mff.cuni.cz
http://ufal.mff.cuni.cz/jan-hajic
http://ufal.mff.cuni.cz/~pecina/index.html

Mutual Information and Word Classes

The Problem

- Not enough data
  - **Language Modeling: we do not see "correct" n-grams**
    - solution so far: smoothing
  - **suppose we see:**
    - short homework, short assignment, simple homework
  - **but not:**
    - simple assigment
  - **What happens to our (bigram) LM?**
    - p(homework | simple) = high probability
    - p(assigment | simple) = low probability (smoothed with p(assigment))
  - They should be much closer!

Word Classes

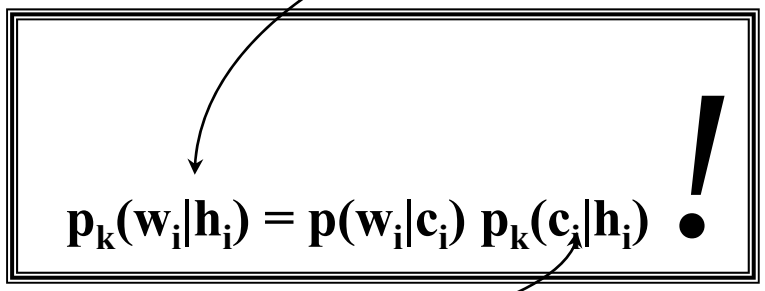
- Observation: similar words behave in a similar way
  - trigram LM:
  - trigram LM, conditioning:
    - a ... homework (any atribute of homework: short, simple, late, difficult),
    - ... the woods (any verb that has the woods as an object: walk, cut, save)
  - trigram LM: both:
    - a (short,long,difficult,...) (homework,assignment,task,job,...)

Solution

- Use the Word Classes as the "reliability" measure
- Example: we see
    - **short homework, short assignment, simple homework**
  - but not:
    - **simple assigment**
  - Cluster into classes:
    - **(short, simple) (homework, assignment)**
      - covers "simple assignment", too
- Gaining: realistic estimates for unseen n-grams
- Loosing: accuracy (level of detail) within classes

The New Model

- Rewrite the n-gram LM using classes:
  - Was: [k = 1..n]
    - **$p_k(w_i|h_i) = c(h_i,w_i) / c(h_i)$ [history: (k-1) words]**
  - Introduce classes:



    - **history: classes, too: [for trigram: $h_i = c_{i-2},c_{i-1}$, bigram: $h_i = c_{i-1}$]**
  - Smoothing as usual
    - **over $p_k(w_i|h_i)$, where each is defined as above (except uniform which stays at 1/|V|)**

Training Data

- Suppose we already have a mapping:
  - r: V $\rightarrow$ C assigning each word its class ($c_i = r(w_i)$)
- Expand the training data:
  - $T = (w_1, w_2, ..., w_{|T|})$ into
  - $T_C = (<w_1,r(w_1)>, <w_2,r(w_2)>, ..., <w_{|T|},r(w_{|T|})>)$
- Effectively, we have two streams of data:
  - word stream: $w_1, w_2, ..., w_{|T|}$
  - class stream: $c_1, c_2, ..., c_{|T|}$ (def. as $c_i = r(w_i)$)
- Expand Heldout, Test data too

Training the New Model

- As expected, using ML estimates:
  - $p(w_i|c_i) = p(w_i|r(w_i)) = c(w_i) / c(r(w_i)) = c(w_i) / c(c_i)$
    - **!!! $c(w_i,c_i) = c(w_i)$ [since c_i determined by w_i]**
  - $p_k(c_i|h_i)$:
    - **$p_3(c_i|h_i) = p_3(c_i|c_{i-2},c_{i-1}) = c(c_{i-2},c_{i-1},c_i) / c(c_{i-2},c_{i-1})$**
    - **$p_2(c_i|h_i) = p_2(c_i|c_{i-1}) = c(c_{i-1},c_i) / c(c_{i-1})$**
    - **$p_1(c_i|h_i) = p_1(c_i) = c(c_i) / |T|$**
- Then smooth as usual
  - not the $p(w_i|c_i)$ nor $p_k(c_i|h_i)$ individually, but the **$p_k(w_i|h_i)$**

Classes: How To Get Them

- We supposed the classes are given
- Maybe there are in [human] dictionaries, but...
  - dictionaries are incomplete
  - dictionaries are unreliable
  - do not define classes as equivalence relation (overlap)
  - do not define classes suitable for LM
    - **small, short... maybe; small and difficult?**
- $\rightarrow$ we have to construct them from data (again...)

Creating the Word-to-Class Map

- We will talk about bigrams from now
- Bigram estimate:
  - $\mathbf{p_2(c_i|h_i) = p_2(c_i|c_{i-1}) = c(c_{i-1},c_i) / c(c_{i-1}) = c(r(w_{i-1}),r(w_i)) / c(r(w_{i-1}))}$
- Form of the model:
  - just raw bigram for now:
    - $\mathbf{P(T) = \Pi_{i=1..|T|} p(w_i|r(w_i))\ p_2(r(w_i)|r(w_{i-1}))}$ $\quad \mathbf{(p_2(c_1|c_0) =_{df} p(c_1))}$
- Maximize over r (given r $\rightarrow$ fixed p, p_2):
  - define objective $L(r) = 1/|T| \Sigma_{i=1..|T|} \log(p(w_i|r(w_i))\ p_2(r(w_i))|r(w_{i-1})))$
  - $r_{best} = \text{argmax}_r L(r)$ $\quad$ (L(r) = norm. logprob of training data... as usual)

Simplifying the Objective Function

- Start from $L(r) = 1/|T| \Sigma_{i=1..|T|} \log(p(w_i|r(w_i)) p_2(r(w_i)|r(w_{i-1})))$:

$1/|T| \Sigma_{i=1..|T|} \log(p(w_i|r(w_i)) \underline{p(r(w_i))} p_2(r(w_i)|r(w_{i-1})) / \underline{p(r(w_i))}) =$

$1/|T| \Sigma_{i=1..|T|} \log(\underline{p(w_i,r(w_i))} p_2(r(w_i)|r(w_{i-1})) / p(r(w_i))) =$

$1/|T| \Sigma_{i=1..|T|} \log(\underline{p(w_i)}) + 1/|T| \Sigma_{i=1..|T|} \log(p_2(r(w_i)|r(w_{i-1})) / p(r(w_i))) =$

$-H(W) + 1/|T| \Sigma_{i=1..|T|} \log(p_2(r(w_i)|r(w_{i-1})) \underline{p(r(w_{i-1})) / (p(r(w_{i-1}))} p(r(w_i)))) =$

$-H(W) + 1/|T| \Sigma_{i=1..|T|} \log(\underline{p(r(w_i),r(w_{i-1}))} / (p(r(w_{i-1})) p(r(w_i)))) =$

$-H(W) + \Sigma_{d,e \in C} p(d,e) \log(p(d,e) / (p(d) p(e))) =$

$-H(W) + I(D,E)$

(event E picks class adjacent (to the right) to the one picked by D)

- Since W does not depend on r, we ended up with I(D,E).

the need to maximize

Maximizing Mutual Information
(dependent on the mapping r)

- Result from previous foil:
  - Maximizing the probability of data amounts to maximizing I(D,E), the mutual information of the adjacent classes.
- Good:
  - We know what a MI is, and we know how to maximize.
- Bad:
  - There is no way how to maximize over so many possible partitionings: $|V|^{|V|}$ - no way to test them all.

Training or Heldout?

- Training:
  - best I(D,E): all words in a class of its own
    → **will not give us anything new.**
- Heldout: ok, but:
  - must smooth to test any possible partitioning (unfeasible):
    → **using raw model: 0 probability of heldout (almost) guaranteed**
      → will not be able to compare anything
  - some smoothing estimates? (to be explored...)
- Solution:
  - use training anyway, but only keep I(D,E) as large as possible

The Greedy Algorithm

- Define merging operation on the mapping r: $V \rightarrow C$:
  - merge: $R \times C \times C \rightarrow R' \times C^{-1}$: $(r,k,l) \rightarrow r',C'$ such that
  - $C^{-1} = \{C - \{k,l\} \cup \{m\}\}$ (throw out k and l, add new $m \notin C$)
  - $r'(w) =$ ..... m for $w \in r_{INV}(\{k,l\})$,
    ..... r(w) otherwise.
- 1. Start with each word in its own class (C = V), r = id.
- 2. Merge two classes k,l into one, m, such that

$$(k,l) = \text{argmax}_{k,l} I_{merge(r,k,l)}(D,E).$$

- 3. Set new (r,C) = merge(r,k,l).
- 4. Repeat 2 and 3 until |C| reaches predetermined size.

Word Classes in Applications

- Word Sense Disambiguation: context not seen [enough(-times)]
- Parsing: verb-subject, verb-object relations
- Speech recognition (acoustic model): need more instances of [rare(r)] sequences of phonemes
- Machine Translation: translation equivalent selection [for rare(r) words]