INFO 2950
Intro to Data Science

Lecture 18: Big Data

Paul Ginsparg

Cornell University, Ithaca, NY

31 Oct 2013

More Statistical Methods

Peter Norvig, “How to Write a Spelling Corrector”
`http://norvig.com/spell-correct.html`

(See video:
`http://www.youtube.com/watch?v=yvDCzhbjYWs`
“The Unreasonable Effectiveness of Data”, given 23 Sep 2010.)

Additional related references:
`http://doi.ieeecomputersociety.org/10.1109/MIS.2009.36`
A. Halevy, P. Norvig, F. Pereira,
The Unreasonable Effectiveness of Data,
Intelligent Systems Mar/Apr 2009 (copy at resources/unrealdata.pdf)

`http://norvig.com/ngrams/ch14.pdf`
P. Norvig, “Natural Language Corpus Data”

A little theory

Find the correction c that maximizes the probability of c given the original word w:

$$\mathrm{argmax}_c \; P(c|w)$$

By Bayes' Theorem, equivalent to $\mathrm{argmax}_c \; P(w|c)P(c)/P(w)$. $P(w)$ the same for every possible c, so ignore, and consider:

$$\mathrm{argmax}_c \; P(w|c)P(c) \; .$$

Three parts :

- $P(c)$, the probability that a proposed correction c stands on its own. The language model: "how likely is c to appear in an English text?" ($P($"the"$)$ high, $P($"zxzxzxzyyy"$)$ near zero)
- $P(w|c)$, the probability that w would be typed when author meant c. The error model: "how likely is author to type w by mistake instead of c?"
- argmax$_c$, the control mechanism: choose c that gives the best combined probability score.

Example

$w=$"thew"

- two candidate corrections $c=$"the" and $c=$"thaw".
- which has higher $P(c|w)$?
- "thaw" has only small change "a" to "e"
- "the" is a very common word, and perhaps the typist's finger slipped off the "e" onto the "w".

To estimate $P(c|w)$, have to consider both the probability of c and the probability of the change from c to w

[Recall the joint probability "p of A given B", written $P(A|B)$, for events A and B, can be estimated by counting the number of times that A and B both occur, and dividing by the total number of times B occurs. Intuitively it is the fraction of times A occurs out of the total times that B occurs.]

Complete Spelling Corrector

```
import re, collections

def words(text): return re.findall('[a-z]+', text.lower())

def train(features):
    model = collections.defaultdict(lambda: 1)
    for f in features:
        model[f] += 1
    return model

NWORDS = train(words(file('big.txt').read()))

alphabet = 'abcdefghijklmnopqrstuvwxyz'
```

⟹

```
def edits1(word):
    s = [(word[:i], word[i:]) for i in range(len(word) + 1)]
    deletes = [a + b[1:] for a, b in s if b]
    transposes = [a + b[1] + b[0] + b[2:] for a, b in s if len(b)>1]
    replaces = [a + c + b[1:] for a, b in s for c in alphabet if b]
    inserts = [a + c + b for a, b in s for c in alphabet]
    return set(deletes + transposes + replaces + inserts)
def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)
def known(words): return set(w for w in words if w in NWORDS)
def correct(word):
    candidates = known([word]) or known(edits1(word))
                                or known_edits2(word) or [word]
    return max(candidates, key=NWORDS.get)
```

(For word of length n: n deletions, n-1 transpositions, 26n alterations, and 26(n+1) insertions, for a total of 54n+25 at edit distance 1)

Improvements

language model $P(c)$: need more words. add -ed to verb or -s to noun, -ly for adverbs

bad probabilities: wrong word appears more frequently? (didn't happen)

error model $P(w|c)$: sometimes edit distance 2 is better
('adres' to 'address', not 'acres')
or wrong word of many at edit distance 1
(in addition better error model permits adding more obscure words)
allow edit distance 3?

best improvement:
look for context ('they where going', 'There's no there thear')
⇒ Use n-grams
(See Whitelaw et al. (2009), "Using the Web for Language Independent Spellchecking and Autocorrection": Precision, recall, F1, classification accuracy)

Outline

1. More Statistical Learning

More Data

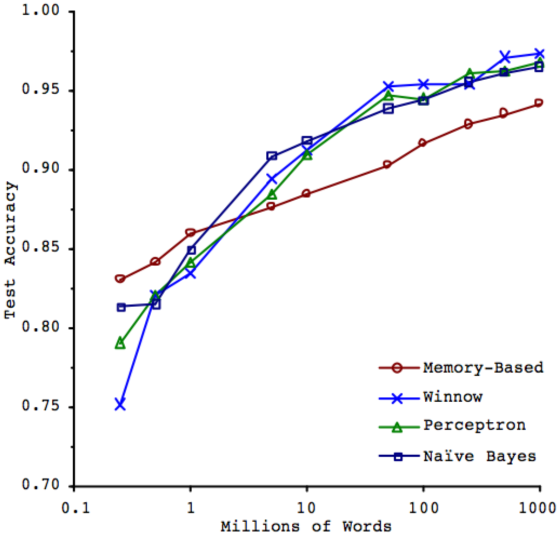

Figure 1. Learning Curves for Confusion Set Disambiguation

http://acl.ldc.upenn.edu/P/P01/P01-1005.pdf

Scaling to Very Very Large Corpora for Natural Language Disambiguation

M. Banko and E. Brill (2001)

More Data for this Task

http://acl.ldc.upenn.edu/P/P01/P01-1005.pdf
Scaling to Very Very Large Corpora for Natural Language Disambiguation
M. Banko and E. Brill (2001)

The amount of readily available on-line text has reached hundreds of billions of words and continues to grow. Yet for most core natural language tasks, algorithms continue to be optimized, tested and compared after training on corpora consisting of only one million words or less. In this paper, we evaluate the performance of different learning methods on a prototypical natural language disambiguation task, confusion set disambiguation, when trained on orders of magnitude more labeled data than has previously been used. We are fortunate that for this particular application, correctly labeled training data is free. Since this will often not be the case, we examine methods for effectively exploiting very large corpora when labeled data comes at a cost.

(Confusion set disambiguation is the problem of choosing the correct use of a word, given a set of words with which it is commonly confused. Example confusion sets include: {principle , principal}, {then , than}, {to , two , too} , and {weather,whether}.)

Segmentation

- nowisthetimeforallgoodmentocometothe
- Probability of a segmentation = P(first word) × P(rest)
- Best segmentation = one with highest probability
- P(word) = estimated by counting

Trained on 1.7B words English, 98% word accuracy

Spelling with Statistical Learning

- Probability of a spelling correction, c = P(c as a word) × P(original is a typo for c)
- Best correction = one with highest probability
- P(c as a word) = estimated by counting
- P(original is a typo for c) = proportional to number of changes

Similarly for speech recognition, using language model $p(c)$ and acoustic model $p(s|c)$
(Russel & Norvig, “Artificial Intelligence”, section 24.7)

And others

- Statistical Machine Translation
  - Collect parallel texts ("Rosetta stones"), Align (Brants, Popat, Xu, Och, Dean (2007), "Large Language Models in Machine Translation")
- fill in occluded portions of photos (Hayes and Efros, 2007)

General "Big Data" Procedure

- Define a probabilistic model
  (i.e., use data to create language model, a probability distribution over all strings in the language, learned from corpus, and use model to determine probability of candidates)
- Enumerate candidates
  (e.g., segmentations, corrected spellings)
- Choose the most probable candidate:

$$\mathrm{best} = \mathrm{argmax}_{c} \in \mathrm{candidates}\, P(c)$$

Python: best = max(candidates, key=P)

Big Data = Simple Algorithm

back to segmentation

e.g., unigram model for segmentation:

$$P(w_1 \dots w_n) = P(w_1) \dots P(w_n)$$

To segment ‘wheninrome’, consider candidates such as “when in rome”, and compute P(when) × P(in) × P(rome).
Gives best answer If product is larger than any other candidate’s.

’wheninthecourseofhumaneventsitbecomesnecessary’ has 35 trillion segmentations, but can be read by finding probable words in sequence (not by considering all 2^{n-1} segmentations)

So use the largest product recursively: P(first) × P(remaining)

Other Tasks

- Secret codes
- Language Identification
- Spam Detection and Other Classification Tasks
- Author Identification (Stylometry)

Statistical Machine Translation

Google *n*-gram corpus created by researchers in the machine translation group (released 2006).

Translating from foreign language (f) into English (e) similar to correcting misspelled words.

The best English translation is modeled as:

$$\text{best} = \text{argmax}_e P(e|f) = \text{argmax}_e P(f|e)P(e)$$

where $P(e)$ is the language model for English, which is estimated by the word *n*-gram data, and $P(f|e)$ is the translation model, learned from a bilingual corpus (where pairs of documents are marked as translations of each other). Although top systems make use of many linguistic features, including parts of speech and syntactic parses of the sentences, seems that majority of knowledge necessary for translation resides in the *n*-gram data.

Further details in Brants,Popat,Xu,Och,Dean (2007)
"Large Language Models in Machine Translation",
http://acl.ldc.upenn.edu/D/D07/D07-1090.pdf