Search Engine Architecture

4. Modeling and Evaluation
Today’s Agenda

• Language models
  • Application to statistical translation
• Retrieval models
  • Preprocessing
  • Scoring
• Model evaluation
Language Models
Language Models

\[ P(w_1, w_2, \ldots, w_T) \]

\[ = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2) \ldots P(w_T|w_1, \ldots, w_{T-1}) \]

[chain rule]

Is this tractable?

When estimating distributions...

• Two important rules
  • Probabilities must sum to one
  • Smooth

Approximating Probabilities

Basic idea: limit history to fixed number of words $N$ (Markov Assumption)

$$P(w_k | w_1, \ldots, w_{k-1}) \approx P(w_k | w_{k-N+1}, \ldots, w_{k-1})$$

**N=1:** Unigram Language Model

$$P(w_k | w_1, \ldots, w_{k-1}) \approx P(w_k)$$

$$\Rightarrow P(w_1, w_2, \ldots, w_T) \approx P(w_1)P(w_2)\ldots P(w_T)$$

Approximating Probabilities

Basic idea: limit history to fixed number of words $N$ (Markov Assumption)

$$P(w_k|w_1, \ldots, w_{k-1}) \approx P(w_k|w_{k-N+1}, \ldots, w_{k-1})$$

**N=2**: Bigram Language Model

$$P(w_k|w_1, \ldots, w_{k-1}) \approx P(w_k|w_{k-1})$$

$$\Rightarrow P(w_1, w_2, \ldots, w_T) \approx P(w_1|<S>)P(w_2|w_1)\ldots P(w_T|w_T-1)$$

Approximating Probabilities

Basic idea: limit history to fixed number of words $N$ (Markov Assumption)

$$P(w_k | w_1, \ldots, w_{k-1}) \approx P(w_k | w_{k-N+1}, \ldots, w_{k-1})$$

**N=3**: Trigram Language Model

$$P(w_k | w_1, \ldots, w_{k-1}) \approx P(w_k | w_{k-2}, w_{k-1})$$

$$\Rightarrow P(w_1, w_2, \ldots, w_T) \approx P(w_1 | < S >> S >) \cdots P(w_T | w_{T-2}w_{T-1})$$

Building N-Gram Language Models

- Compute maximum likelihood estimates (MLE) for individual n-gram probabilities

- Unigram: \( P(w_i) = \frac{C(w_i)}{N} \)

- Bigram: \( P(w_i, w_j) = \frac{C(w_i, w_j)}{N} \)

\[
P(w_j | w_i) = \frac{P(w_i, w_j)}{P(w_i)} = \frac{C(w_i, w_j)}{\sum_w C(w_i, w)} = \frac{C(w_i, w_j)}{C(w_i)}
\]

- Generalizes to higher-order n-grams

Thou shalt smooth!

- Zeros are bad for any statistical estimator
  - Need better estimators because MLEs give us a lot of zeros
  - A distribution without zeros is “smoother”
- The Robin Hood Philosophy: Take from the rich (seen $n$-grams) and give to the poor (unseen $n$-grams)
  - And thus also called discounting
  - Make sure you still have a valid probability distribution!
- Lots of techniques:
  - Laplace, Good-Turing, Katz backoff, Jelinek-Mercer
  - Kneser-Ney represents best practice

Stupid Backoff

- Let’s break all the rules:

\[
S(w_i | w_{i-k+1}^{i-1}) = \begin{cases} 
\frac{f(w_{i-k+1}^i)}{f(w_{i-k+1}^{i-1})} & \text{if } f(w_{i-k+1}^i) > 0 \\
\alpha S(w_i | w_{i-k+2}^{i-1}) & \text{otherwise}
\end{cases}
\]

\[
S(w_i) = \frac{f(w_i)}{N}
\]

- But throw *lots* of data at the problem!

Statistical Machine Translation

Statistical Machine Translation

Training Data
- i saw the small table
- vi la mesa pequeña
- Parallel Sentences
- he sat at the table
- the service was good
- Target-Language Text

Word Alignment
- (vi, i saw)
- (la mesa pequeña, the small table)
- ...

Phrase Extraction
- is saw the small table
- vi la mesa pequeña

Language Model

Translation Model

Decoder

maria no daba una bofetada a la bruja verde
- Foreign Input Sentence

mary did not slap the green witch
- English Output Sentence

\[
\hat{e}^l_i = \arg\max_{e^l_i} \left[ P(e^l_i \mid f^J_i) \right] = \arg\max_{e^l_i} \left[ P(e^l_i) P(f^J_i \mid e^l_i) \right]
\]

Translation as a Tiling Problem

\[ \hat{e}'_i = \arg \max_{e'_i} \left[ P(e'_i | f'_i) \right] = \arg \max_{e'_i} \left[ P(e'_i)P(f'_i | e'_i) \right] \]

# Results: Running Time

<table>
<thead>
<tr>
<th></th>
<th>target</th>
<th>webnews</th>
<th>web</th>
</tr>
</thead>
<tbody>
<tr>
<td># tokens</td>
<td>237M</td>
<td>31G</td>
<td>1.8T</td>
</tr>
<tr>
<td>vocab size</td>
<td>200k</td>
<td>5M</td>
<td>16M</td>
</tr>
<tr>
<td># (n)-grams</td>
<td>257M</td>
<td>21G</td>
<td>300G</td>
</tr>
<tr>
<td>LM size (SB)</td>
<td>2G</td>
<td>89G</td>
<td>1.8T</td>
</tr>
<tr>
<td>time (SB)</td>
<td>20 min</td>
<td>8 hours</td>
<td>1 day</td>
</tr>
<tr>
<td>time (KN)</td>
<td>2.5 hours</td>
<td>2 days</td>
<td>–</td>
</tr>
<tr>
<td># machines</td>
<td>100</td>
<td>400</td>
<td>1500</td>
</tr>
</tbody>
</table>

Results: Translation Quality

Source: Brants et al. (EMNLP 2007) via Data Intensive Computing with MapReduce 2013
Retrieval Models
Today’s Agenda

• Language models
  • Application to statistical translation
• Retrieval models
  • Preprocessing
  • Scoring
• Model evaluation
How do we represent text?

• Remember: computers don’t “understand” anything!
• “Bag of words”
  • Treat all the words in a document as index terms
  • Assign a “weight” to each term based on “importance” (or, in simplest case, presence/absence of word)
  • Disregard order, structure, meaning, etc. of the words
  • Simple, yet effective!
• Assumptions
  • Term occurrence is independent
  • Document relevance is independent
  • “Words” are well-defined

What's a word?

What's a word?

What's a word?

What's a word?

What's a word?

What's a word?

What's a word?

What's a word?

What's a word?

What's a word?

What's a word?
Preprocessing

- Case folding
- Tokenization
- Stop words
- Stemming
- Collocations
Case Folding

- Convert all terms to lower case
  - Users will use lower case in queries anyway
- What about proper nouns? Acronyms?
  - Exception: upper case in middle of sentence?
- Retaining case information might be useful for other features
  - E.g. recognizing named entities
Tokenization

- Input: “To be, or not to be”
- Output: to, be, or, not, to, be
- Issues:
  - “New York University”
  - “Shakespeare’s play”
  - “state-of-the-art”
Stopwords

- A, an, to, of, ...
- Issues:
  - What if you query for a phrase?
  - Other ways to reduce importance of common terms...
Stemming

- Heuristic-based removal of prefixes and suffixes
- E.g. Porter stemmer
  - “sses” -> “ss” (caresses -> caress)
  - “ies” -> “l” (ponies -> poni)
  - “s” -> “” (cats -> cat)
- Porter, Snowball, Lancaster stemmers
  - In increasing likelihood of overstemming
- (Cf. lemmatization)
Collocations

- Find phrases for use downstream
- Student’s t
- Chi square
- Pointwise mutual information
- Likelihood ratio
- Variations of $p(x, y) / (p(x) + p(y))$
Retrieval

• TF-IDF
• Variants
• Norms
• Coordination
• Boosting
Vector Space Model

Assumption: Documents that are “close together” in vector space “talk about” the same things

Therefore, retrieve documents based on how close the document is to the query (i.e., similarity ~ “closeness”)

Similarity Metric

- Use “angle” between the vectors:

\[ d_j = [w_{j,1}, w_{j,2}, w_{j,3}, \ldots w_{j,n}] \]
\[ d_k = [w_{k,1}, w_{k,2}, w_{k,3}, \ldots w_{k,n}] \]

\[ \cos \theta = \frac{d_j \cdot d_k}{|d_j||d_k|} \]

\[ \text{sim}(d_j, d_k) = \frac{d_j \cdot d_k}{|d_j||d_k|} = \frac{\sum_{i=0}^{n} w_{j,i} w_{k,i}}{\sqrt{\sum_{i=0}^{n} w_{j,i}^2} \sqrt{\sum_{i=0}^{n} w_{k,i}^2}} \]

- Or, more generally, inner products:

\[ \text{sim}(d_j, d_k) = d_j \cdot d_k = \sum_{i=0}^{n} w_{j,i} w_{k,i} \]

Term Weighting

- Term weights consist of two components
  - Local: how important is the term in this document?
  - Global: how important is the term in the collection?
- Here’s the intuition:
  - Terms that appear often in a document should get high weights
  - Terms that appear in many documents should get low weights
- How do we capture this mathematically?
  - Term frequency (local)
  - Inverse document frequency (global)

TF-IDF Term Weighting

\[ w_{i,j} = tf_{i,j} \cdot \log \frac{N}{n_i} \]

- \( w_{i,j} \): weight assigned to term \( i \) in document \( j \)
- \( tf_{i,j} \): number of occurrence of term \( i \) in document \( j \)
- \( N \): number of documents in entire collection
- \( n_i \): number of documents with term \( i \)

Variant – use sublinear tf (e.g. \( \log tf \))

Normalization

• Divide vectors by some value
• L2 – square root of sum of squares
• L1 – sum of absolute values
• Tradeoffs but L2 more common
Coordination Factor

- Consider query “quick brown fox”
- One document contains “quick brown” many times
- Another contains “quick brown fox” only a few times
- Coord factor rewards occurrence of all three terms
Boosting

- Consider query “quick brown fox”
- One document contains “quick brown fox” in the title
- Another contains “quick brown fox” twice in the body
- Boosting reflects that first document is likely more relevant
Today’s Agenda

• Language models
  • Application to statistical translation
• Retrieval models
  • Preprocessing
  • Scoring
• Model evaluation
Model Evaluation
Evaluation

• Which features are effective?
• Stop lists, stemming, IDF...
• Requires gold standard or ground truth
  • Standard test collections: TREC, Reuters, etc.
  • Click logs
• UI concerns as well
Why do we care?

- *Lean Startup* by Eric Ries:
  - Build
  - *Measure*
  - Learn
- Biggest risk to a new venture isn’t falling behind schedule or buggy software
  - It’s building the wrong thing entirely!
Unranked Evaluation

• Simple case: search engine returns a set of results
• This is the case for Boolean retrieval
• Examples:
  • Precision
  • Recall
  • Accuracy?
  • F-Measure
Precision

- \( P(\text{relevant} \mid \text{retrieved}) \)

\[
\text{precision} = \frac{\# \text{ relevant items retrieved}}{\# \text{ retrieved items}}
\]
Recall

- $P(\text{retrieved} \mid \text{relevant})$

\[
\text{recall} = \frac{\# \text{ relevant items retrieved}}{\# \text{ relevant items}}
\]
Accuracy

- Almost never a good idea
- If 10 in 10,000 documents are relevant, then never returning anything gives 99.9% accuracy
  - But recall is 0%
F Measure

- Harmonic mean of precision and recall

\[ F = \frac{2PR}{P + R} \]

- Why harmonic mean?
  - If we use arithmetic mean, returning everything gives 100% recall, and 50% arithmetic mean
  - But if 1 in 10,000 documents is relevant then F=0.02%
Ranked Evaluation

• How do we evaluate effectiveness when top results should be more relevant?
  • Set measures on top K
  • Mean Average Precision
  • Cumulative Gain
  • NDCG
Precision/Recall Graph

- **Blue line**: P/R for increasing k
- **Red line**: highest P for a given R
Normalized Discounted Cumulative Gain

- CG is the sum of graded relevances:
  \[ CG_p = \sum_{i=1}^{p} rel_i \]

- DCG penalizes relevance by position:
  \[ DCG_p = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2(i)} \]

- nDCG normalizes to interval \([0, 1]\)
  \[ nDCG_p = \frac{DCG_p}{IDCG_p} \]
Questions?