Overview of Statistical Language Models

Jon Dehdari

Introduction

n-gram LMs
Skip LMs
Class LMs
Topic LMs
Neural Net LMs
Conclusion
References

Workshop on Data Mining
and its Use and Usability for Linguistic Analysis
Overview

What is a Statistical Language Model?
At the broadest level, it is a probability distribution.
Overview of Statistical Language Models

Jon Dehdari

Introduction
- n-gram LMs
- Skip LMs
- Class LMs
- Topic LMs
- Neural Net LMs
- Conclusion
- References

Overview

What is a Statistical Language Model?
At the broadest level, it is a probability distribution.

Input
Natural Language. Usually entire or prefix of:
- Words in a sentence (eg. for speech recognition, machine translation)
- Characters (eg. for OCR, Dasher)
- Paragraph/Document (eg. for information retrieval)
Overview

What is a Statistical Language Model?
At the broadest level, it is a probability distribution.

Input
Natural Language. Usually entire or prefix of:
- Words in a sentence (eg. for speech recognition, machine translation)
- Characters (eg. for OCR, Dasher)
- Paragraph/Document (eg. for information retrieval)

Output
- Probability [0, 1] – all possible outcomes sum to 1
- An unnormalized score, for ranking

References
Incremental statistical language models provide the probability that a given word will occur next, based on the preceding words:

\[ P(w_i|w_1, \ldots, w_{i-1}) \]
Incremental statistical language models provide the probability that a given word will occur next, based on the preceding words:

$$P(w_i | w_1, \ldots, w_{i-1})$$

For Example:

- It’s raining cats and ____
Incremental statistical language models provide the probability that a given word will occur next, based on the preceding words:

\[ P(w_i | w_1, \ldots, w_{i-1}) \]

For Example:

- It’s raining cats and ____
- They went on a shopping ____
Incremental statistical language models provide the probability that a given word will occur next, based on the preceding words:

$$P(w_i | w_1, \ldots, w_{i-1})$$

For Example:
- It’s raining cats and ____
- They went on a shopping ____
- I cooked the fish in a ____
A Few Uses for LMs

Statistical language models ensure fluency in speech recognition (like Siri), machine translation (like Google Translate), on-screen keyboards (smartphones), etc.
A Few Uses for LMs

Statistical language models ensure fluency in speech recognition (like Siri), machine translation (like Google Translate), on-screen keyboards (smartphones), etc.

Sometimes they don’t work so well...
Actually, There’s a Lot of Uses!

- Google suggest
- Machine translation
- Assisting people with motor disabilities. For example, Dasher
- Speech Recognition (ASR)
- Optical character recognition (OCR) and handwriting recognition
- Information retrieval / search engines
- Data compression
- Language identification, as well as genre, dialect, and idiolect identification (authorship identification)
- Software keyboards
- Surface realization in natural language generation
- Password cracking
- Cipher cracking
Differences in LM Uses

- **Long-Distance**
  - Grammar
  - Parsing
  - NLG
  - MT
  - Google Suggest
  - Password Cracking
- **Local**
  - Summarization
  - IR
  - ASR
  - Software Keyboards
  - OCR, Dasher
  - LangID, Cipher Cracking

- **Grammatical**
- **Lexical**
Overview of Statistical Language Models

Jon Dehdari

Introduction

n-gram LMs

Skip LMs

Class LMs

Topic LMs

Neural Net LMs

Conclusion

References

LM Usage

Typical LM Queries in ...

**ASR** : \( p(\text{recognize speech}) \) vs. \( p(\text{wreck a nice beach}) \) vs. \( p(\text{wreck an ice peach}) \), ...  

**Cipher cracking** : \( p(\text{attack at dawn}) \) vs. \( p(\text{uebvmkdvkdbsqk}) \)  

**Google Suggest** : \( p(\text{how to cook french fries}) \) vs. \( p(\text{how to cook french dictionary}) \)  

**IR** : query(cats and the cradle): \( \text{doc1(i like cats)} \) vs. \( \text{doc2(i like dogs)} \)  

**MT & NLG** : lex: \( p(\text{use the force}) \) vs. \( p(\text{use the power}) \); ordering: \( p(\text{ready are you}) \) vs. \( p(\text{are you ready}) \)  

**OCR** : \( p(\text{today is your day}) \) vs. \( p(\text{+qdav ls y0ur d4ij}) \)
## Overview of Statistical Language Models

Jon Dehdari

### Introduction

- **n-gram LMs**
- **Skip LMs**
- **Class LMs**
- **Topic LMs**
- **Neural Net LMs**

### Conclusion

### References

---

**Language Modeling is Interesting!**

<table>
<thead>
<tr>
<th>NLP Task</th>
<th>Avg. Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language Modeling (=Word Prediction)</td>
<td>7.12</td>
</tr>
<tr>
<td>English-Chinese Translation</td>
<td>5.17</td>
</tr>
<tr>
<td>English-French Translation</td>
<td>3.92</td>
</tr>
<tr>
<td>QA (Open Domain)</td>
<td>3.87</td>
</tr>
<tr>
<td>Syntactic Parsing</td>
<td>1.18</td>
</tr>
<tr>
<td>QA (Multi-class Classification)</td>
<td>1.08</td>
</tr>
<tr>
<td>Text Classification (20 News)</td>
<td>0.70</td>
</tr>
<tr>
<td>Sentiment Analysis</td>
<td>0.58</td>
</tr>
<tr>
<td>Part-of-Speech Tagging</td>
<td>0.42</td>
</tr>
<tr>
<td>Named Entity Recognition</td>
<td>0.31</td>
</tr>
</tbody>
</table>

From Li & Hovy (2015)
Overview of Statistical Language Models

Introduction

- n-gram LMs
- Skip LMs
- Class LMs
- Topic LMs
- Neural Net LMs

Conclusion

References

$n$-gram Language Models

The simplest statistical language models, $n$-gram LMs, base their prediction on the previous word or two (Markov assumption)

$$P(w_i|w_1 \ldots w_{i-1}) \approx P(w_i|w_{i-n+1} \ldots w_{i-1})$$
**n-gram Language Models**

The simplest statistical language models, *n*-gram LMs, base their prediction on the previous word or two (*Markov assumption*)

\[ P(w_i | w_1 \ldots w_{i-1}) \approx P(w_i | w_{i-n+1} \ldots w_{i-1}) \]

**For Example:**

- ____ and ____
The simplest statistical language models, \( n \)-gram LMs, base their prediction on the previous word or two (Markov assumption):

\[
P(w_i | w_1 \ldots w_{i-1}) \approx P(w_i | w_{i-n+1} \ldots w_{i-1})
\]

For Example:

- [ ] and ____
- [ ] cats and ____
The simplest statistical language models, \( n \)-gram LMs, base their prediction on the previous word or two (Markov assumption):

\[
P(w_i | w_1 \ldots w_{i-1}) \approx P(w_i | w_{i-n+1} \ldots w_{i-1})
\]

For Example:

- and ____
- cats and ____
- shopping ____
The simplest statistical language models, \(n\)-gram LMs, base their prediction on the previous word or two (Markov assumption)

\[
P(w_i|w_1 \ldots w_{i-1}) \approx P(w_i|w_{i-n+1} \ldots w_{i-1})
\]

For Example:

- [ ] and ____
- [ ] cats and ____
- [ ] shopping ____
- [ ] a shopping ____
$n$-gram Language Models

The simplest statistical language models, $n$-gram LMs, base their prediction on the previous word or two (Markov assumption)

$$P(w_i | w_1 \ldots w_{i-1}) \approx P(w_i | w_{i-n+1} \ldots w_{i-1})$$

For Example:

- and ____
- cats and ____
- shopping ____
- a shopping ____
- the ____
**n-gram Language Models**

The simplest statistical language models, 
*n*-gram LMs, base their prediction on the previous word or two (*Markov assumption*)

\[
P(w_i|w_1 \ldots w_{i-1}) \approx P(w_i|w_{i-n+1} \ldots w_{i-1})
\]

For Example:

- and ____
- cats and ____
- shopping ____
- a shopping ____
- the ____
- in the ____
In spite of their many, many shortcomings, \textit{n}-gram LMs are still widely used

1. They train quickly
2. They require no manual annotation
3. They are incremental
Uniform Distribution (Zero-gram)

**Zero-gram Model**

- In a zero-gram model, all words from the vocabulary \( V \) are equally likely:

\[
p(w_i) = \frac{1}{|V|} = |V|^{-1}
\]

- For example, if we were to open a dictionary and randomly point to a word, then “orangutan” would have the same probability as “the”:

\[
p(\text{orangutan}) = p(\lambda P \in D_{(e,t)}: \mathcal{I}_x[P(x) \land C(x)])
\]
Unigram Model

- In a unigram model, using maximum likelihood estimation, probabilities are based on word counts:

\[ p(w_i) = \frac{\text{count}(w_i)}{\text{count}(w)} \]

- For example, if we were to open a novel and randomly point to a word, then “orangutan” would have much less probability than “the”:

\[ p(\text{orangutan}) \ll p(\lambda P \in D_{\langle e,t \rangle} \cdot \mathcal{I}[P(x) \land C(x)]) \]
But what about:

“I gave a banana to a furry orange ______”

Here, a unigram model would give too much probability to “the” and not enough to “orangutan”
Bigram Model

- But what about:
  “I gave a banana to a furry orange _____”

- Here, a unigram model would give too much probability to “the” and not enough to “orangutan”
Bigram Model

- But what about:
  
  “I gave a banana to a furry orange ______”

- Here, a unigram model would give too much probability to “the” and not enough to “orangutan”

- In a bigram model, using maximum likelihood estimation, probabilities are based on bigram and word counts:

\[
p(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}
\]
But what about: 

“I gave a banana to a furry orange ______”

Here, a unigram model would give too much probability to “the” and not enough to “orangutan”

In a bigram model, using maximum likelihood estimation, probabilities are based on bigram and word counts:

\[
p(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}
\]
Trigram and other $n$-gram LMs use a longer *contiguous* history

$$p(w_i | w_{i-2}, w_{i-1}) = \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})}$$
Trigram and other *n*-gram LMs use a longer *contiguous* history

\[ p(w_i|w_{i-2}, w_{i-1}) = \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})} \]
**n-gram LMs**

Trigram and other \(n\)-gram LMs use a longer *contiguous* history

\[
p(w_i|w_{i-2}, w_{i-1}) = \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})}
\]
Using $n$-gram LMs

Using Multiple $n$-gram Models

**Backoff** – Use the highest-order $n$-gram model that has enough occurrences in the training set

**Interpolation** – Use all $n$-gram models, weighting them differently
Using $n$-gram LMs

Using Multiple $n$-gram Models

- **Backoff** – Use the highest-order $n$-gram model that has enough occurrences in the training set
- **Interpolation** – Use all $n$-gram models, weighting them differently

Smoothing $n$-grams

- *Smoothing* allows us to deal with unseen histories
- Usually steals some probability mass from seen events and gives some to unseen events
Skip LMs

- Skip LMs like $n$-gram LMs, but allow intervening words between the predicted word and its conditioning history. These are combined (interpolated) with $n$-gram models.
Skip LMs like $n$-gram LMs, but allow intervening words between the predicted word and its conditioning history. These are combined (interpolated) with $n$-gram models.

Example skip bigram:

$$p(w_i | w_{i-2}) = \frac{\text{count}(w_{i-2}, w_i)}{\text{count}(w_{i-2})}$$
Skip LMs like $n$-gram LMs, but allow intervening words between the predicted word and its conditioning history. These are combined (interpolated) with $n$-gram models.

Example skip bigram:

$$p(w_i | w_{i-2}) = \frac{\text{count}(w_{i-2}, w_i)}{\text{count}(w_{i-2})}$$

- They capture basic word order variation, and are still (more) useful with large corpora (Goodman, 2001, § 4)
- There’s many possible combinations of histories to use
- They unnecessarily fragment the training data instead of generalizing it (Rosenfeld, 1994, pg. 16).
Class-based LMs abstract beyond specific words, so that, eg. ‘Thursday’ and ‘Friday’ are grouped together to function similarly. 

+ They’re useful for small- and medium-sized corpora (up to a billion tokens), and easy to use. Words can be automatically clustered.

± They have advantages and disadvantages for morphologically-rich & freer word order languages:

– They’re poor at handling fixed phrases and multi-word expressions:

<s> <s> → PRP it → VBZ ’s → VBG raining → NNS cats → CC and
Topic LMs

- Both class-based and topic-based LMs use a *bottleneck variable* to generalize the history
- Class-based LMs generalize the short-term grammatical history
- Topic-based LMs generalize the long-term lexical history

Documents are (soft) clustered into a set of topics automatically.
Both class-based and topic-based LMs use a *bottleneck variable* to generalize the history

- Class-based LMs generalize the short-term grammatical history
- Topic-based LMs generalize the long-term lexical history
- Documents are (soft) clustered into a set of topics automatically
Both class-based and topic-based LMs use a **bottleneck variable** to generalize the history.

- Class-based LMs generalize the short-term grammatical history.
- Topic-based LMs generalize the long-term lexical history.
- Documents are (soft) clustered into a set of topics automatically.

+ Useful for domain adaptation. Widely used in information retrieval.

- They’re slow and don’t scale up well. They don’t capture local grammatical info, so they’re combined with other LMs.
Neural Net LMs

- Like topic-based LMs, neural net LMs reduce high-dimensional discrete probability distributions to low-dimensional continuous distributions.
- Original idea inspired by biological neurons, but architecture has diverged from biology.
- Has (multiple) hidden layers, to allow multiple levels of generalization.
Elman Networks

- Like previous feedforward layout, but also has the previous hidden state feed into current hidden state
- In principle can capture longer dependencies

![Elman Network Diagram](image-url)
When training Elman networks the cycle gets unwrapped (called BPTT)

Image derived from Bodén (2002)
### Comparison

<table>
<thead>
<tr>
<th>Language Model</th>
<th>Incremental</th>
<th>Lexical</th>
<th>Distance</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>n-gram</td>
<td>Y</td>
<td>Y</td>
<td>Short</td>
<td>Fast</td>
</tr>
<tr>
<td>Class</td>
<td>Y</td>
<td>N</td>
<td>Medium</td>
<td>Fast</td>
</tr>
<tr>
<td>Cache</td>
<td>Y</td>
<td>Y</td>
<td>Long</td>
<td>Fast</td>
</tr>
<tr>
<td>Skip</td>
<td>Y</td>
<td>Y</td>
<td>Medium</td>
<td>Fast</td>
</tr>
<tr>
<td>PCFG</td>
<td>N</td>
<td>N</td>
<td>Long</td>
<td>Slow</td>
</tr>
<tr>
<td>Topic</td>
<td>Y</td>
<td>N</td>
<td>Long</td>
<td>Slow</td>
</tr>
<tr>
<td>FF-NN</td>
<td>Y</td>
<td>Y</td>
<td>Medium</td>
<td>Slow</td>
</tr>
<tr>
<td>RNN</td>
<td>Y</td>
<td>Y</td>
<td>Medium</td>
<td>Slow</td>
</tr>
</tbody>
</table>
References I

Trainable grammars for speech recognition.

A guide to recurrent neural networks and backpropagation.
Technical report, Halmstad University, School of Information Science, Computer and Electrical Engineering.

Class-based *n*-gram models of natural language.

Finding structure in time.

Topic-based language models using EM.

A bit of progress in language modeling, extended version.

Improvements in unsupervised co-occurrence based parsing.
References II

UnsuParse: Unsupervised parsing with unsupervised part of speech tagging.

The SPHINX-II speech recognition system: an overview.

The estimation of stochastic context-free grammars using the inside-outside algorithm.
*Computer Speech and Language*, 4:35–56.

The NLP engine: A universal Turing machine for NLP.
*Arxiv.org Preprint*.

Recurrent neural network based language model.

Rosenfeld, R. (1994).
PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

Learning representations by back-propagating errors.
Generalization of backpropagation with application to a recurrent gas market model. 

Backpropagation through time: What it does and how to do it. 