# A Short Overview of Statistical Language Models

Jon Dehdari

Invited Talk at
the Workshop on Data Mining
and its Use and Usability for Linguistic Analysis

March 2015

# Overview

## What is a Statistical Language Model?

At the broadest level, it is a probability distribution.

# 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

# Incremental Language Models

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

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

# Incremental Language Models

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

$$P(w_i \mid \underbrace{w_1, \ldots, w_{i-1}}_{h})$$

## For Example:

- It's raining cats and ____

# Incremental Language Models

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

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

## For Example:

- It's raining cats and ____
- They went on a shopping ____

# Incremental Language Models

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

$$P(w_i \mid \underbrace{w_1, \ldots, w_{i-1}}_{h})$$

**For Example:**

- It's raining cats and ____
- They went on a shopping ____
- I cooked the fish in a ____

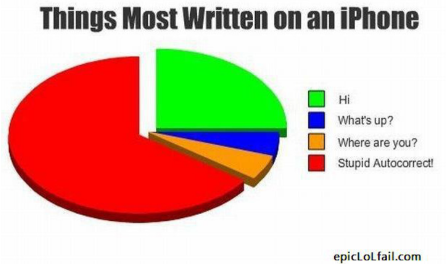
# A Few Uses for LMs

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.

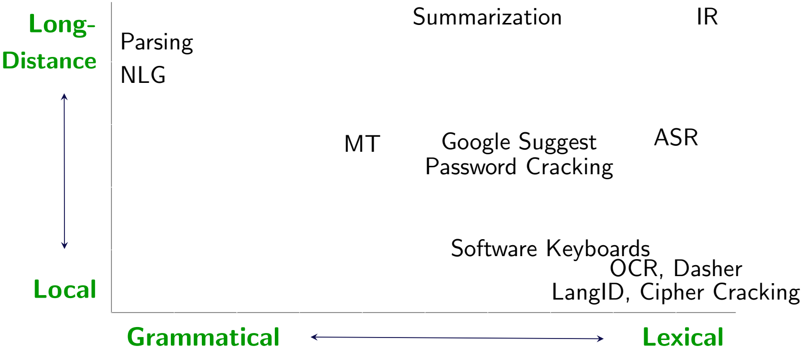
Sometimes they don't work so well...



# 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



# LM Usage

## Typical LM Queries in ...

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

**Cipher cracking** : p(attack at dawn) vs. p(uebvmkdvkdbsqk)

**Google Suggest** : p(how to cook french fries) vs. p(how to cook french dictionary)

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

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

**OCR** : p(today is your day) vs. p(+qdav ls y0ur d4ij)

A good cipher should obey the principle of diffusion (Shannon, 1949).

# Language Modeling is Interesting!

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

From Li & Hovy (2015)

# *n*-gram Language Models

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

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

History Sux!

# *n*-gram Language Models

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

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

History Sux!

## For Example:

- [redacted] and ____

# *n*-gram Language Models

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

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

History Sux!

## For Example:

- [redacted] and ____
- [redacted] cats and ____

# *n*-gram Language Models

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

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

History Sux!

## For Example:

- [redacted] and ____
- [redacted] cats and ____
- [redacted] shopping ____

# *n*-gram Language Models

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

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

History Sux!

## For Example:

- [redacted] and ____
- [redacted] cats and ____
- [redacted] shopping ____
- [redacted] 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*)

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

History Sux!

## For Example:

- [redacted] and ____
- [redacted] cats and ____
- [redacted] shopping ____
- [redacted] a shopping ____
- [redacted] the ____

# *n*-gram Language Models

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

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

History Sux!

## For Example:

- [redacted] and ____
- [redacted] cats and ____
- [redacted] shopping ____
- [redacted] a shopping ____
- [redacted] the ____
- [redacted] in the ____

# *n*-gram LMs


- In spite of their many, many shortcomings, *n*-gram LMs are still widely used
  1. They train quickly
  2. They require no manual annotation
  3. They are incremental

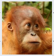
# Uniform Distribution (Zero-gram)

## 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*":

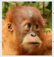
p(  ) $= \mathrm{p}(\lambda P \in D_{\langle e,t \rangle}.\iota x[P(x) \wedge C(x)])$

# 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(  ) $\ll$ $\mathrm{p}(\lambda P \in D_{\langle e,t \rangle}.\iota x[P(x) \wedge C(x)])$

# 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*"

# 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:

$$p(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

# 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:

$$p(w_i | w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

$$w_0 \rightarrow w_1 \rightarrow w_2 \rightarrow w_3 \rightarrow w_4 \rightarrow w_k$$

# *n*-gram LMs

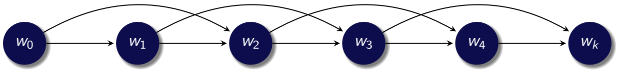
Trigram and other *n*-gram LMs use a longer *contiguous* history

$$p(w_i | w_{i-2}, w_{i-1}) = \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})}$$

# *n*-gram LMs

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

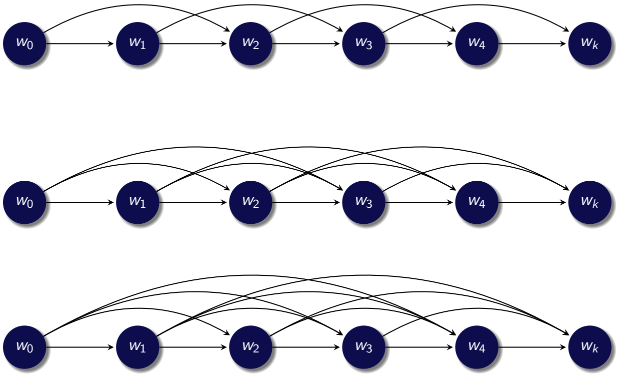
$$p(w_i | w_{i-2}, w_{i-1}) = \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})}$$



# $n$-gram LMs

Trigram and other $n$-gram LMs use a longer *contiguous* history

$$p(w_i | w_{i-2}, w_{i-1}) \quad = \quad \frac{\text{count}(w_{i-2}, w_{i-1}, w_i)}{\text{count}(w_{i-2}, w_{i-1})}$$



# 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
- See: `http://statmt.org/book/slides/07-language-models.pdf`

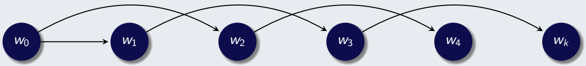
# Skip LMs

- 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

- 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:

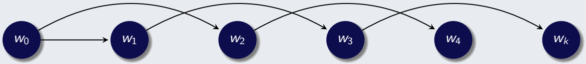
$$p(w_i | w_{i-2}) = \frac{\text{count}(w_{i-2}, w_i)}{\text{count}(w_{i-2})}$$



# Skip LMs

- 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:

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



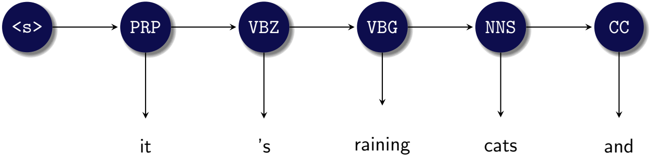

+ They capture basic word order variation, and are still (more) useful with large corpora (Goodman, 2001, § 4)

± There's many possible combinations of histories to use

− They unnecessarily fragment the training data instead of generalizing it (Rosenfeld, 1994, pg. 16).

# Class LMs

- 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:

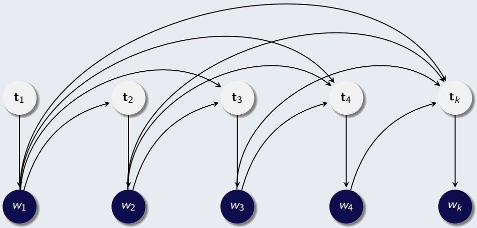


# Topic LMs

- 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

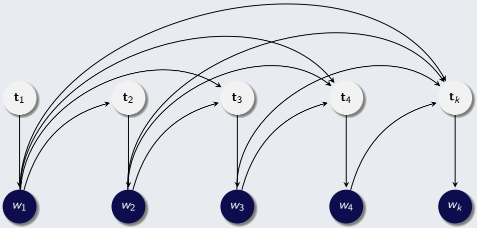
# Topic LMs

- 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



# 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

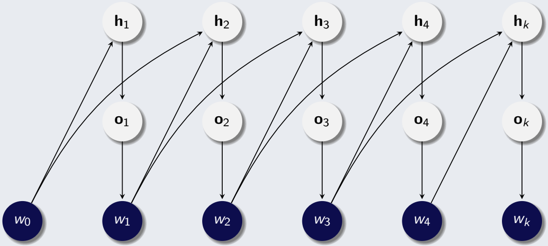



+ Useful for domain adaptation. Widely used in information retrieval

– 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

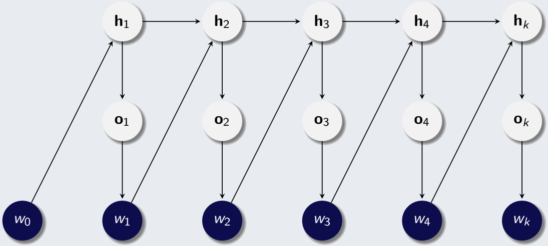
- Like topic-based LMs, neural net LMs reduce high-dimensional discrete probability distributions to low-dimensional continuous distributions
- Original idea inspired by biological neurons, but architecture has diverged from biology
- Has (multiple) hidden layers, to allow multiple levels of generalization



# Recurrent Neural Net LMs

## Elman Networks

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



# RNNLM's Continued

When training Elman networks the cycle gets unwrapped (called BPTT)

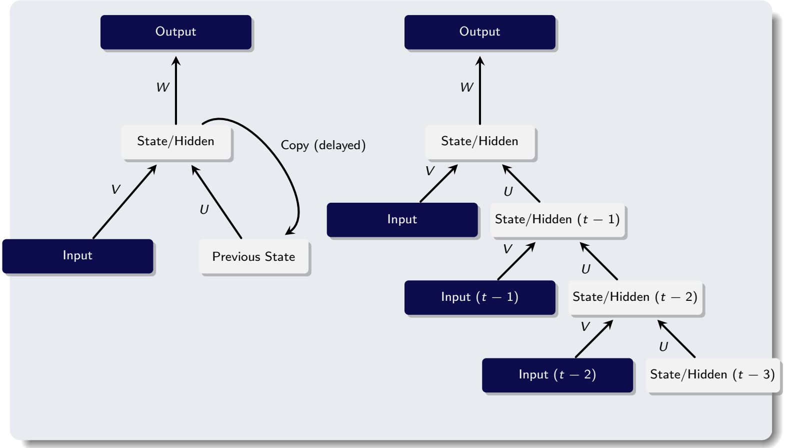



Image derived from Bodén (2002)

# Comparison

| Language Model | Incremental | Lexical | Distance | Speed |
|---|---|---|---|---|
| *n*-gram | Y | Y | Short | Fast |
| Class | Y | N | Medium | Fast |
| Cache | Y | Y | Long | Fast |
| Skip | Y | Y | Medium | Fast |
| PCFG | N | N | Long | Slow |
| Topic | Y | N | Long | Slow |
| FF-NN | Y | Y | Medium | Slow |
| RNN | Y | Y | Medium | Slow |

Fine print goes here. No purchase necessary. Void where prohibited. Medium distance here is longer than a typical *n*-gram history length (eg. 5-gram), but isn't the full sentence history. Elman network-based LMs aren't trained using the full sentence history due to BPTT, and a word's effect decays over time. While advances have been made in improving softmax normalization, models that use it are still orders of magnitude slower at training than a typical generative model. Induced PCFG (eg. Baker (1979), Lari & Young (1990,1991), Hänig et al (2008), Hänig (2010)) are costly to train from unannotated texts.

# References I

Baker, J. K. (1979).
Trainable grammars for speech recognition.
In Klatt, D. H. and Wolf, J. J., editors, *Speech Communication Papers for the 97th Meeting of the Acoustical Society of America*, pages 547–550, Cambridge, MA, USA.

Bodén, M. (2002).
A guide to recurrent neural networks and backpropagation.
Technical report, Halmstad University, School of Information Science, Computer and Electrical Engineering.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992).
Class-based $n$-gram models of natural language.
*Computational Linguistics*, 18(4):467–479.

Elman, J. L. (1990).
Finding structure in time.
*Cognitive Science*, 14(2):179–211.

Gildea, D. and Hofmann, T. (1999).
Topic-based language models using EM.
In *Proceedings of EUROSPEECH*, pages 2167–2170.

Goodman, J. T. (2001).
A bit of progress in language modeling, extended version.
Technical Report MSR-TR-2001-72, Microsoft Research.

Hänig, C. (2010).
Improvements in unsupervised co-occurrence based parsing.
In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 1–8, Uppsala, Sweden. Association for Computational Linguistics.

# References II

Hänig, C., Bordag, S., and Quasthoff, U. (2008).
UnsuParse: Unsupervised parsing with unsupervised part of speech tagging.
In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odjik, J., Piperidis, S., and Tapias, D., editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco.

Huang, X., Alleva, F., Hon, H.-W., Hwang, M.-Y., Lee, K.-F., and Rosenfeld, R. (1993).
The SPHINX-II speech recognition system: an overview.
*Computer Speech and Language*, 2:137–148.

Lari, K. and Young, S. J. (1990).
The estimation of stochastic context-free grammars using the inside-outside algorithm.
*Computer Speech and Language*, 4:35–56.

Li, J. and Hovy, E. (2015).
The NLP engine: A universal Turing machine for NLP.
*Arxiv.org Preprint*.

Mikolov, T., Karafiát, M., Burget, L., Černocký, J., and Khudanpur, S. (2010).
Recurrent neural network based language model.
In *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010)*, pages 1045–1048.

Rosenfeld, R. (1994).
*Adaptive Statistical Language Modeling: A Maximum Entropy Approach*.
PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986).
Learning representations by back-propagating errors.
*Nature*, 323(6088):533–536.

# References III

Werbos, P. J. (1988).
Generalization of backpropagation with application to a recurrent gas market model.
*Neural Networks*, 1(4):339–356.

Werbos, P. J. (1990).
Backpropagation through time: What it does and how to do it.
*Proceedings of the IEEE*, 78(10):1550–1560.