Special Topics in Natural Language Processing

CS698O

Ashutosh Modi
CSE Department, IIT Kanpur

Lecture 6: Language Models 3
Jan 13, 2020

LM Review

$$p(X_n = w_n, X_{n-1} = w_{n-1}, \ldots, X_1 = w_1) = \prod_{i=1}^{n} p(X_i = w_i \mid X_{i-1} = w_{i-1}, X_{i-2} = w_{i-2})$$

$$where,$$

$$w_0 = w_{-1} = START; \quad w_n = STOP; \quad w_i \in \mathcal{V} \;\; \forall \, 1 \leq i \leq n-1$$

$$p(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} \theta(w_i \mid w_{i-1}, w_{i-2})$$

$$\theta(q \mid r, s) = \frac{C(q, r, s)}{C(r, s)}$$

$C(q, r, s) =$ Number of times trigram {s,r,q} occurs in the corpus

$C(r, s) =$ Number of times bigram {s,r} occurs in the corpus

LM Review

TRI-GRAM

$$\theta(q \mid r, s) = \frac{C(q, r, s)}{C(r, s)}$$

BI-GRAM

$$\theta(q \mid r) = \frac{C(q, r)}{C(r)}$$

UNI-GRAM

$$\theta(q) = \frac{C(q)}{N}$$

Problems with MLE

$$\theta(q \mid r, s) = \frac{C(q, r, s)}{C(r, s)}$$

Overfitting (Sparsity):

$\text{Many trigram estimates } (\theta(q \mid r, s)) \text{ are } 0$ Hapax Legomenon

Indeterminate Estimates:

$C(r, s) \text{ can be } 0$

Linear Interpolation

- The key idea is to rely on lower order statistics.
- Each n-gram has strengths and weaknesses
- For trigram estimates, also rely on bigram and unigram.

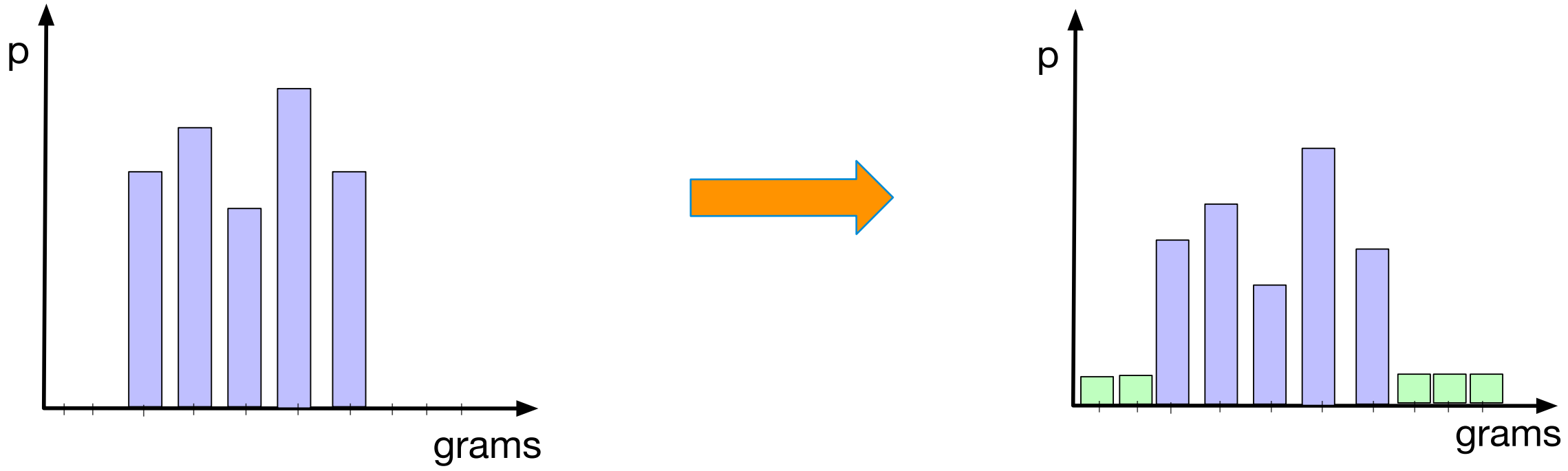
$$\theta(q \mid r, s) = \lambda_1 \times \theta(q \mid r, s) + \lambda_2 \times \theta(q \mid r) + \lambda_3 \times \theta(q)$$

$$\lambda_i \geq 0 \quad \forall i \in \{1, 2, 3\}$$

$$\sum_{i=1,2,3} \lambda_i = 1$$

Discounting and Smoothing

The key idea is to take away some probability mass from non-zero counts and distribute to counts of grams never seen before.



Discounting: Trigrams

$$\theta_D(q|r,s) = \begin{cases} \frac{C^*(q,r,s)}{C(r,s)} & \text{If } q \in \mathcal{A}(r,s) \\ \alpha(r,s) \times \frac{\theta(q|r)}{\sum_{q \in \mathcal{B}(r,s)} \theta(q|r)} & \text{If } q \in \mathcal{B}(r,s) \end{cases}$$

$$\mathcal{A}(r,s) = \{q : C(q,r,s) > 0\}$$

$$\mathcal{B}(r,s) = \{q : C(q,r,s) = 0\}$$

$$C^*(q,r,s) := C(q,r,s) - \beta$$

Laplace Smoothing (Add-one Smoothing)

The key idea is to add one to all counts and normalize

UNI-GRAM

$$\theta(q) = \frac{C(q) + 1}{N + \mid \mathcal{V} \mid}$$

BI-GRAM

$$\theta(q \mid r) = \frac{C(q, r) + 1}{C(r) + \mid \mathcal{V} \mid}$$

TRI-GRAM

$$\theta(q \mid r, s) = \frac{C(q, r, s) + 1}{C(r, s) + \mid \mathcal{V} \mid}$$

NOTE: In all cases, $\mid \mathcal{V} \mid$ is the vocabulary size

Evaluating LMs

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2 \ p(S_i)$$

$$\text{Perplexity} \ := 2^{-l}$$

Evaluating LMs

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2 \ p(S_i)$$

$$\text{Perplexity} \ := 2^{-l}$$

The *smaller* the value of perplexity,
better the language model is at modeling unseen data.

Perplexity

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i) \qquad \text{Perplexity} := 2^{-l}$$

$$\text{Suppose, } N = |\ \{\mathcal{V} \cup STOP\}\ |$$

$$\theta(q \mid r, s) = \frac{1}{N}$$

Uniform Distribution

Perplexity

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i) \qquad \text{Perplexity} := 2^{-l}$$

$$\text{Suppose, } N = |\ \{\mathcal{V} \cup STOP\}\ |$$

$$\theta(q \mid r, s) = \frac{1}{N}$$

Uniform Distribution

What is the perplexity of this LM?

Perplexity

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2 \; p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

$$\text{Suppose, } N = \mid \{\mathcal{V} \cup STOP\} \mid$$

Uniform Distribution

$$\theta(q \mid r, s) = \frac{1}{N}$$

$$\text{Perplexity} = N$$

Under uniform LM, perplexity is equal to vocabulary size

Perplexity

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

Perplexity is a measure of effective vocabulary size under the LM

Perplexity: alternative formulation

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

Perplexity is a measure of effective vocabulary size under the LM

$$\text{Perplexity} = \frac{1}{t}$$

$$\text{where, } t = \sqrt[M]{\prod_{i=1}^{m} p(S_i)}$$

Perplexity: alternative formulation

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

Perplexity is a measure of effective vocabulary size under the LM

$$\text{Perplexity} = \frac{1}{t}$$

$$\text{where,}\quad t = \sqrt[M]{\prod_{i=1}^{m} p(S_i)}$$

t is the geometric mean of the trigram probability terms in a Trigram LM

Perplexity: alternative formulation

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

$$\text{Perplexity} = \frac{1}{t}$$

$$\text{where,}\quad t = \sqrt[M]{\prod_{i=1}^{m} p(S_i)}$$

t is the geometric mean of the trigram probability terms in a Trigram LM

Are the two formulations equal?

Perplexity

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

$$\text{Perplexity} = \frac{1}{t}$$

$$\text{where, } t = \sqrt[M]{\prod_{i=1}^{m} p(S_i)}$$

What is the perplexity

if any of trigram probability $\theta(q \mid r, s)$ in the test set is zero?

Perplexity

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

$$\text{Perplexity} = \frac{1}{t}$$

$$\text{where, } t = \sqrt[M]{\prod_{i=1}^{m} p(S_i)}$$

What is the perplexity

if any of trigram probability $\theta(q \mid r, s)$ in the test set is zero?

$$\infty$$

Perplexity: practical values

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

$$\text{Perplexity} = \frac{1}{t}$$

$$\text{where, } t = \sqrt[M]{\prod_{i=1}^{m} p(S_i)}$$

$$\mathcal{V} = 50000$$

Model	Perplexity
Unigram	955
Bigram	137
Trigram	74

Perplexity: practical values

$$l = \frac{1}{M} \sum_{i=1}^{m} log_2\ p(S_i)$$

$$\text{Perplexity} := 2^{-l}$$

$$\text{Perplexity} = \frac{1}{t}$$

$$\text{where, } t = \sqrt[M]{\prod_{i=1}^{m} p(S_i)}$$

$$\mathcal{V} = 50000$$

Model	Perplexity
Unigram	955
Bigram	137
Trigram	74

SOTA using deep learning models : perplexity of 21.8 with vocabulary size of 800K

Katz Back-Off Model

- Idea similar to interpolation
- In practice, works better than linear interpolation

Katz Back-Off Model

- Idea similar to interpolation
- In practice, works better than linear interpolation

$$\theta_{katz}(q \mid r, s) = \begin{cases} \theta^*(q \mid r, s) & \text{if } \; C(q, r, s) > 0 \\ \alpha(r, s)\ \theta_{katz}(q \mid r) & \text{else if } \; C(q, r) > 0 \\ \theta^*(q) & \text{otherwise} \end{cases}$$

Katz Back-Off Model

- Idea similar to interpolation
- In practice, works better than linear interpolation

$$\theta_{katz}(q \mid r,s) = \begin{cases} \theta^*(q \mid r,s) & \text{if } C(q,r,s) > 0 \\ \alpha(r,s)\ \theta_{katz}(q \mid r) & \text{else if } C(q,r) > 0 \\ \theta^*(q) & \text{otherwise} \end{cases}$$

$$\theta_{katz}(q \mid r) = \begin{cases} \theta^*(q \mid r) & \text{if } C(q,r) > 0 \\ \alpha(r)\ \theta^*(q) & \text{otherwise} \end{cases}$$

Katz Back-Off Model

- How do we determine alpha?

$$\theta_{katz}(q \mid r) = \begin{cases} \theta^*(q \mid r) & \text{if } C(q,r) > 0 \\ \alpha(r)\ \theta^*(q) & \text{otherwise} \end{cases}$$

Katz Back-Off Model

- How do we determine alpha?

$$\theta_{katz}(q \mid r) = \begin{cases} \theta^*(q \mid r) & \text{if } C(q,r) > 0 \\ \alpha(r)\ \theta^*(q) & \text{otherwise} \end{cases}$$

$$\sum_{q \in \mathcal{V}} \theta_{katz}(q \mid r) = 1$$

$$\sum_{q:C(q,r)>0} \theta^*(q \mid r) + \alpha(r) \sum_{q:C(q,r)=0} \theta^*(q) = 1$$

Katz Back-Off Model

- How do we determine alpha?

$$\theta_{katz}(q \mid r) = \begin{cases} \theta^*(q \mid r) & \text{if } C(q,r) > 0 \\ \alpha(r)\ \theta^*(q) & \text{otherwise} \end{cases}$$

$$\sum_{q \in \mathcal{V}} \theta_{katz}(q \mid r) = 1 \quad \Longrightarrow \quad \sum_{q:C(q,r)>0} \theta^*(q \mid r) + \alpha(r) \sum_{q:C(q,r)=0} \theta^*(q) = 1$$

$$\alpha(r) = \frac{1 - \sum_{q:C(q,r)>0} \theta^*(q \mid r)}{\sum_{q:C(q,r)=0} \theta^*(q)}$$

Kneser-Ney Smoothing

I enjoy drinking Mango _____

Kneser-Ney Smoothing

I enjoy drinking Mango __Juice__

Kneser-Ney Smoothing

I enjoy drinking Mango _____

What if I used a back-off language model ?

Kneser-Ney Smoothing

I enjoy drinking Mango _____

BUT

$$\underbrace{C(paper)}_{=134,939,426} >> \underbrace{C(juice)}_{=10,973,425}$$

Source: Google Web Trillion Word Corpus: https://www.kaggle.com/rtatman/english-word-frequency

Kneser-Ney Smoothing

- The idea is to base the estimate on number of different contexts a word has appeared in.
- Words which have appeared in more context are likely to appear in some new context as well.

Kneser-Ney Smoothing

- The idea is to base the estimate on number of different contexts a word has appeared in.
- Words which have appeared in more context are likely to appear in some new context as well.

$$\theta_{KN}(q \mid r) = \begin{cases} \frac{C(q,r)-\beta}{C(r)} & \text{if } C(q,r) > 0 \\ \alpha(q) \frac{|\{r:C(q,r)>0\}|}{\sum_q |\{r:C(q,r)>0\}|} & \text{otherwise} \end{cases}$$

Kneser-Ney Smoothing

- One can have an interpolated version as well
- In practice, it works better than back-off version

$$\theta_{KN}(q \mid r) = \frac{C(q,r) - \beta}{C(r)} + \alpha^{'}(q) \frac{\mid \{r : C(q,r) > 0\} \mid}{\sum_{q} \mid \{r : C(q,r) > 0\} \mid}$$

Class Based Language Models

- Words can be categorized into different classes
- Make use of word class information
- Helps to alleviate sparsity issues

Class Based Language Models

- Words can be categorized into different classes
- Make use of word class information
- Helps to alleviate sparsity issues

……..flew from London to Delhi
……..flew from Mumbai to Kolkata
……..travelled from Bucharest to Paris
:
……worked in Shanghai
:

Class Based Language Models

- Words can be categorized into different classes
- Make use of word class information
- Helps to alleviate sparsity issues

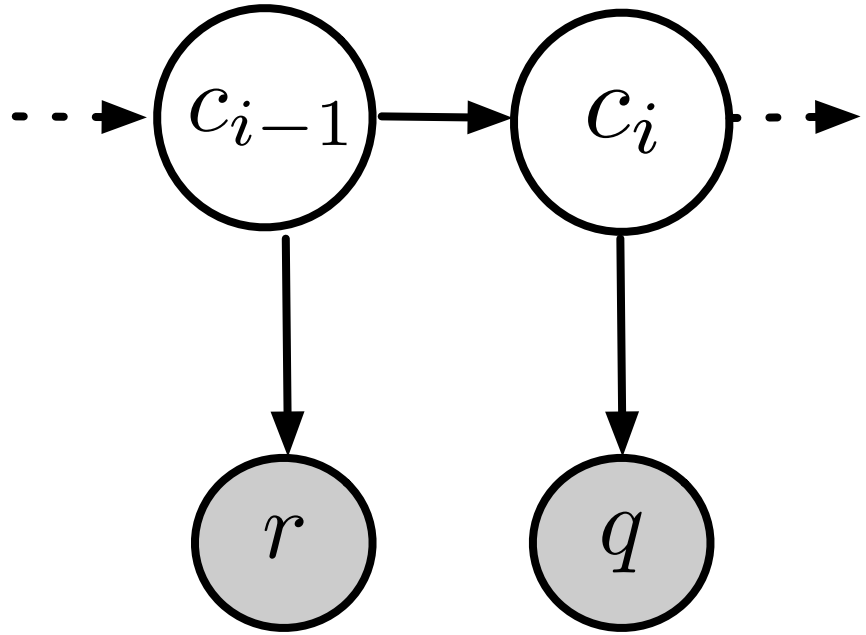
……..flew from London to Delhi
……..flew from Mumbai to Kolkata
……..travelled from Bucharest to Paris
:
……..worked in Shanghai
:

P (from Mumbai to Shanghai) = ?

Class Based Language Models

- Cluster N-Grams

$$\theta(q \mid r) \approx \theta(c_i|c_{i-1}) \times \theta(q \mid c_i)$$



Cluster N-Gram Estimation

$$\theta(q \mid r) \approx \theta(c_i|c_{i-1}) \times \theta(q \mid c_i)$$

Cluster N-Gram Estimation

$$\theta(q \mid r) \approx \theta(c_i|c_{i-1}) \times \theta(q \mid c_i)$$

MLE

Cluster N-Gram Estimation

$$\theta(q \mid r) \approx \theta(c_i|c_{i-1}) \times \theta(q \mid c_i)$$

MLE

$$\theta(q \mid c_i) = \ ?$$

Cluster N-Gram Estimation

$$\theta(q \mid r) \approx \theta(c_i|c_{i-1}) \times \theta(q \mid c_i)$$

MLE

$$\theta(q \mid c_i) = \frac{C(q)}{C(c_i)}$$

$C(c_i)$ is the number of words for which the class is c_i

Cluster N-Gram Estimation

$$\theta(q \mid r) \approx \theta(c_i|c_{i-1}) \times \theta(q \mid c_i)$$

MLE

$$\theta(c_i \mid c_{i-1}) = \ ?$$

Cluster N-Gram Estimation

$$\theta(q \mid r) \approx \theta(c_i|c_{i-1}) \times \theta(q \mid c_i)$$

MLE

$$\theta(c_i \mid c_{i-1}) = \frac{C(c_{i-1}, c_i)}{\sum_c C(c_{i-1}, c)}$$

$C(c_{i-1}, c_i)$ are total number of word bigrams for which classes are c_{i-1} and c_i, *respectively.*

Cluster N-Gram Estimation

$$\theta(q \mid r) \approx \theta(c_i|c_{i-1}) \times \theta(q \mid c_i)$$

MLE

$$\theta(q \mid c_i) = \frac{C(q)}{C(c_i)}$$

$$\theta(c_i \mid c_{i-1}) = \frac{C(c_{i-1}, c_i)}{\sum_c C(c_{i-1}, c)}$$

Language Models: Extending Context

- Larger the context the better

Language Models: Extending Context

- Larger the context the better
- In principle we could have larger window in n-gram, but this is computationally problematic

Language Models: Extending Context

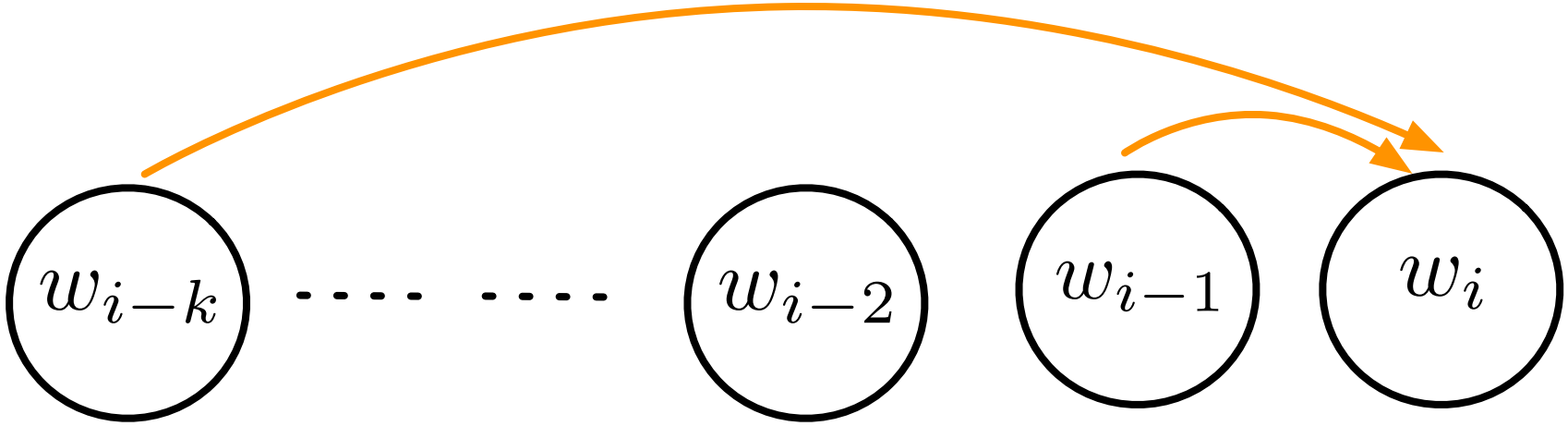
- Larger the context the better
- In principle we could have larger window in n-gram, but this is computationally problematic
- One alternative: **Skip-k-Grams Language Model**

$$P(w_i|w_{i-1}, w_{i-k})$$

Language Models: Extending Context

- **Skip-k-Grams LM**

$$P(w_i|w_{i-1}, w_{i-k})$$



Caching Based LMs

- If a speaker uses a word, it is likely that he will use the same word again in the near future.
- This is the basis of Caching
- In a unigram cache, we form a unigram model from the most recently used words (e.g. words within last paragraph or article)
- Cache models are linearly interpolated with conventional n-gram models

Caching Based LMs

- If a speaker uses a word, it is likely that he will use the same word again in the near future.
- This is the basis of Caching
- In a unigram cache, we form a unigram model from the most recently used words (e.g. words within last paragraph or article)
- Cache models are linearly interpolated with conventional n-gram models

$$\theta_{trigram-cache}(w_i \mid w_{i-1}, w_{i-2}) =$$
$$\lambda\theta_{trigram}(w_i \mid w_{i-1}, w_{i-2}) + (1-\lambda)\theta_{unigram-cache}(w_i, w_{i-1}, \ldots, w_1)$$

Practical Issue: Out of Vocabulary (OOV)

- What about words in test set which are never seen in training?

Practical Issue: Out of Vocabulary (OOV)

- What about words in test set which are never seen in training?
- These are Out Of Vocabulary (OOV) words

Practical Issue: Out of Vocabulary (OOV)

- What about words in test set which are never seen in training?
- These are Out Of Vocabulary (OOV) words
- One strategy:
  - Convert words in the training set that are less than certain pre-decided frequency count to a special symbol: ***<UKN>***

Practical Issue: Out of Vocabulary (OOV)

- What about words in test set which are never seen in training?
- These are Out Of Vocabulary (OOV) words
- One strategy:
  - Convert words in the training set that are less than certain pre-decided frequency count to a special symbol: ***<UKN>***
  - Estimate the language model with the ***<UKN>*** word in the vocabulary

Practical Issue: Out of Vocabulary (OOV)

- What about words in test set which are never seen in training?
- These are Out Of Vocabulary (OOV) words
- One strategy:
  - Convert words in the training set that are less than certain pre-decided frequency count to a special symbol: ***<UKN>***
  - Estimate the language model with the ***<UKN>*** word in the vocabulary
  - For the test set, any unseen word is tagged as ***<UKN>***

Practical Issue: Out of Vocabulary (OOV)

- What about words in test set which are never seen in training?
- These are Out Of Vocabulary (OOV) words
- Another similar strategy:
  - Decide the vocabulary in advance and tag words not part of the vocabulary as ***<UKN>***
  - Estimate the language model with the ***<UKN>*** word in the vocabulary
  - For the test set, any unseen word is tagged as ***<UKN>***

Summary

- Perplexity is a measure of LM's ability to model language
- We looked at Katz Back-Off model
- We looked at Kneser-Ney smoothing technique
- Sometimes class information about words can be useful for LMs
- LM context can be extended by using Skip-k-Gram model
- OOV words can be tagged as <UKN>

References

1. Chapter 4, Speech and Language Processing, Dan Jurafsky and James Martin
2. Michael Collin's NLP Lecture Notes: http://www.cs.columbia.edu/~mcollins/lm-spring2013.pdf
3. A Bit of Progress in Language Modeling: https://arxiv.org/pdf/cs/0108005.pdf
4. Class based LM, the first paper to propose the idea: https://www.cs.cmu.edu/~roni/11761/PreviousYearsHandouts/classlm.pdf

- Next class
  - Maximum Entropy Language Models
  - Neural Language Models