BITS Pilani

Pilani Campus

Information Retrieval

Abhishek

April 2020

innovate achieve lead

BITS Pilani
Pilani Campus

CS F469, Information Retrieval

Lecture topics: Cross-Lingual IR, Machine Translation, Language Modeling

Recap: Summary of CLIR

- CLIR = Query Translation + IR
    - Integrate QT with IR
    - QT is one step in the global IR process

Recap: Multilingual IR

- MLIR = CLIR + merging
  - Translate the query into different languages
  - Retrieve doc. in each language
  - Merge the results into a single list.

Noisy Channel Model

- Goal:
  - translation system from **Source to Target** language.
  - Have a model **p(t | s)** which estimates conditional probability of any target language sentence **t** given the source language sentence **s**. Use the training corpus to set the parameters.
- A Noisy Channel Model has two components:
  - **p(t)** the language model
  - **p(s | t)** the translation model
- Using the above two, we can estimate:
  - Learn a distribution $\mathbf{p(t \mid s)} = \boldsymbol{arg\ max_t}\ \mathbf{p(t)p(s \mid t)}$

More about Noisy Channel Model

- The language model **p(t)** could be a trigram model, estimated from any data (parallel corpus not needed to estimate the parameters)
- The translation model **p(s | t)** is trained from a parallel corpus of Source/Target pairs.
- Note:
  - The translation model is backwards!
  - The language model can make up for deficiencies of the translation model.
  - Later we'll talk about how to build **p(s | t)**
  - Decoding, i.e., finding

$$\mathrm{argmax}_t\, p(t)p(s \mid t)$$

  - is also a challenging problem.

Language Modeling Applications

- By accurately assigning probability to a natural sequence (words or characters), you can improve
    - **Machine Translation:** p(strong tea) > p(powerful tea)
    - **Speech Recognition:** p(speech recognition) > p(speech wreck ignition)
    - **Question Answering / Summarization:** p(President X attended ...) is higher for X=Obama
    - **Query Completion:** p(Michael Jordan Berkeley) > p(Michael Jordan sports)

Examples from Nitish Shirish Keskar and Stephen Merity slides.

Trigram Language Models

- A trigram language model consists of:
  a. A finite set V.
  b. A parameter $q(w \mid u, v)$ for each trigram (u, v, w) such that $w \in V \cup \{STOP\}$, and $u, v \in V \cup \{*\}$

- For any sentence $x_1 \ldots x_n$, where $x_i \in V$ for $i = 1 \ldots (n-1)$, and $x_n = STOP$, the probability of the sentence under the trigram language model is

$$p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i \mid x_{i-2}, x_{i-1})$$

  where we define $x_0 = x_{-1} = *$

An Example

- For a sentence:

  The dog barks STOP

We would have:

$$
\begin{aligned}
p(\text{the dog barks STOP}) = \; & q(\text{the} \mid *, *) \\
& \times q(\text{dog} \mid *, \text{the}) \\
& \times q(\text{barks} \mid \text{the}, \text{dog}) \\
& \times q(\text{STOP} \mid \text{dog}, \text{barks})
\end{aligned}
$$

The Trigram Estimation Problem

$q(w_i | w_{i-2}, w_{i-1})$

For example: q(barks | the, dog)

A natural estimate (the “maximal likelihood estimate”)

$$q(w_i | w_{i-2}, w_{i-1}) = \text{Count}(w_{i-2}, w_{i-1}, w_i) / \text{Count}(w_{i-2}, w_{i-1})$$

q(barks | the, dog) = Count(the dog barks) / Count(the dog)

Sparse Data Problems

$$q(w_i | w_{i-2}, w_{i-1}) = \text{Count}(w_{i-2}, w_{i-1}, w_i) / \text{Count}(w_{i-2}, w_{i-1})$$

q(barks | the, dog) = Count(the dog barks) / Count(the dog)

Say our vocabulary size is $N = |V|$, then there are N^3 parameters in the model.

E.g. $N = 20{,}000 \rightarrow 20000^3 = 8 \times 10^{12}$ parameters

Linear Interpolation

- Take our estimate $q(w_i| w_{i-2}, w_{i-1})$ to be

$$q(w_i| w_{i-2}, w_{i-1}) = \lambda_1 \times q_{ML}(w_i| w_{i-2}, w_{i-1}) + \lambda_2 \times q_{ML}(w_i| w_{i-1}) + \lambda_3 \times q_{ML}(w_i)$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$ and $\lambda_i \geq 0$ for all i.

Example:

q(barks| the, dog) = ⅓ q_{ML}(barks| the, dog) + ⅓ q_{ML}(barks | dog) + ⅓ q_{ML}(barks)

Assuming all lambdas values are equal.

Discounting methods

- Say we've seen the following counts:

x	Count(x)	$q_{ML}(w_i \mid w_{i-1})$
the	48	
the, dog	15	15/48
the, woman	11	11/48
the, man	10	10/48
the, park	5	5/48
the, job	2	2/48
the, telescope	1	1/48
the, manual	1	1/48
the, afternoon	1	1/48
the, country	1	1/48
the, street	1	1/48

The maximum-likelihood estimates are high (particularly for low count items)

Discounting methods

- Now define "discounted" counts,
  Count* (x) = Count(x) − 0.5
- New estimates:

x	$c(x)$	$c^*(x)$	$\frac{c^*(x)}{c(the)}$
the	48		
the, dog	15	14.5	14.5/48
the, woman	11	10.5	10.5/48
the, man	10	9.5	9.5/48
the, park	5	4.5	4.5/48
the, job	2	1.5	1.5/48
the, telescope	1	0.5	0.5/48
the, manual	1	0.5	0.5/48
the, afternoon	1	0.5	0.5/48
the, country	1	0.5	0.5/48
the, street	1	0.5	0.5/48

Discounting methods

- We now have some “missing probability mass”:

$$\alpha(w_{i-1}) = 1 - \sum_{w} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

e.g., in our example, α(the) = 10 × 0.5/48 = 5/48

Katz Back-Off Models (Bigrams)

- For a bigram model, define two sets

$$\mathcal{A}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) > 0\}$$
$$\mathcal{B}(w_{i-1}) = \{w : \text{Count}(w_{i-1}, w) = 0\}$$

- A bigram model

$$q_{BO}(w_i \mid w_{i-1}) = \begin{cases} \frac{\text{Count}^*(w_{i-1}, w_i)}{\text{Count}(w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-1}) \\ \alpha(w_{i-1}) \frac{q_{\text{ML}}(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} q_{\text{ML}}(w)} & \text{If } w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-1})} \frac{\text{Count}^*(w_{i-1}, w)}{\text{Count}(w_{i-1})}$$

Katz Back-Off Models (Trigrams)

- For a trigram model, first define two sets

$$\mathcal{A}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) > 0\}$$
$$\mathcal{B}(w_{i-2}, w_{i-1}) = \{w : \text{Count}(w_{i-2}, w_{i-1}, w) = 0\}$$

- A trigram model

$$q_{BO}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} \dfrac{\text{Count}^*(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})} & \text{If } w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\[2ex] \dfrac{\alpha(w_{i-2}, w_{i-1}) q_{BO}(w_i \mid w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} q_{BO}(w \mid w_{i-1})} & \text{If } w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \end{cases}$$

where

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-2}, w_{i-1})} \frac{\text{Count}^*(w_{i-2}, w_{i-1}, w)}{\text{Count}(w_{i-2}, w_{i-1})}$$

LM: Summary

- Three steps in deriving the language model probabilities:
  - Expand $p(w_1, w_2 \ldots w_n)$ using Chain rule.
  - Make Markov Independence Assumptions

    $$p(w_i \mid w_1, w_2 \ldots w_{i-2}, w_{i-1}) = p(w_i \mid w_{i-2}, w_{i-1})$$

  - Smooth the estimates using low order counts innovate achieve lead

Evaluation of Language Model

Evaluating a Language Model: Perplexity

- We have some test data, m sentences

$$s_1, s_2, s_3, \ldots, s_m$$

- We could look at the probability under our model $\prod_{i=1}^{m} p(s_i)$. Or more conveniently, the *log probability*

$$\log \prod_{i=1}^{m} p(s_i) = \sum_{i=1}^{m} \log p(s_i)$$

- In fact the usual evaluation measure is *perplexity*

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \frac{1}{M} \sum_{i=1}^{m} \log p(s_i)$$

and M is the total number of words in the test data.

Some Intuition about Perplexity

- Say we have a vocabulary $\mathcal{V}$, and $N = |\mathcal{V}| + 1$ and model that predicts

$$q(w|u, v) = \frac{1}{N}$$

for all $w \in \mathcal{V} \cup \{\text{STOP}\}$, for all $u, v \in \mathcal{V} \cup \{*\}$.

- Easy to calculate the perplexity in this case:

$$\text{Perplexity} = 2^{-l} \quad \text{where} \quad l = \log \frac{1}{N}$$

$\Rightarrow$

$$\text{Perplexity} = N$$

Perplexity is a measure of effective "branching factor"

Typical Values of Perplexity

- Results from Goodman ("A bit of progress in language modeling"), where $|\mathcal{V}| = 50,000$

- A trigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i | x_{i-2}, x_{i-1})$.
  Perplexity $= 74$

- A bigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i | x_{i-1})$.
  Perplexity $= 137$

- A unigram model: $p(x_1 \ldots x_n) = \prod_{i=1}^{n} q(x_i)$.
  Perplexity $= 955$

Language Modeling Variants

State-of-the-art in Language Modeling

LM: State of the art

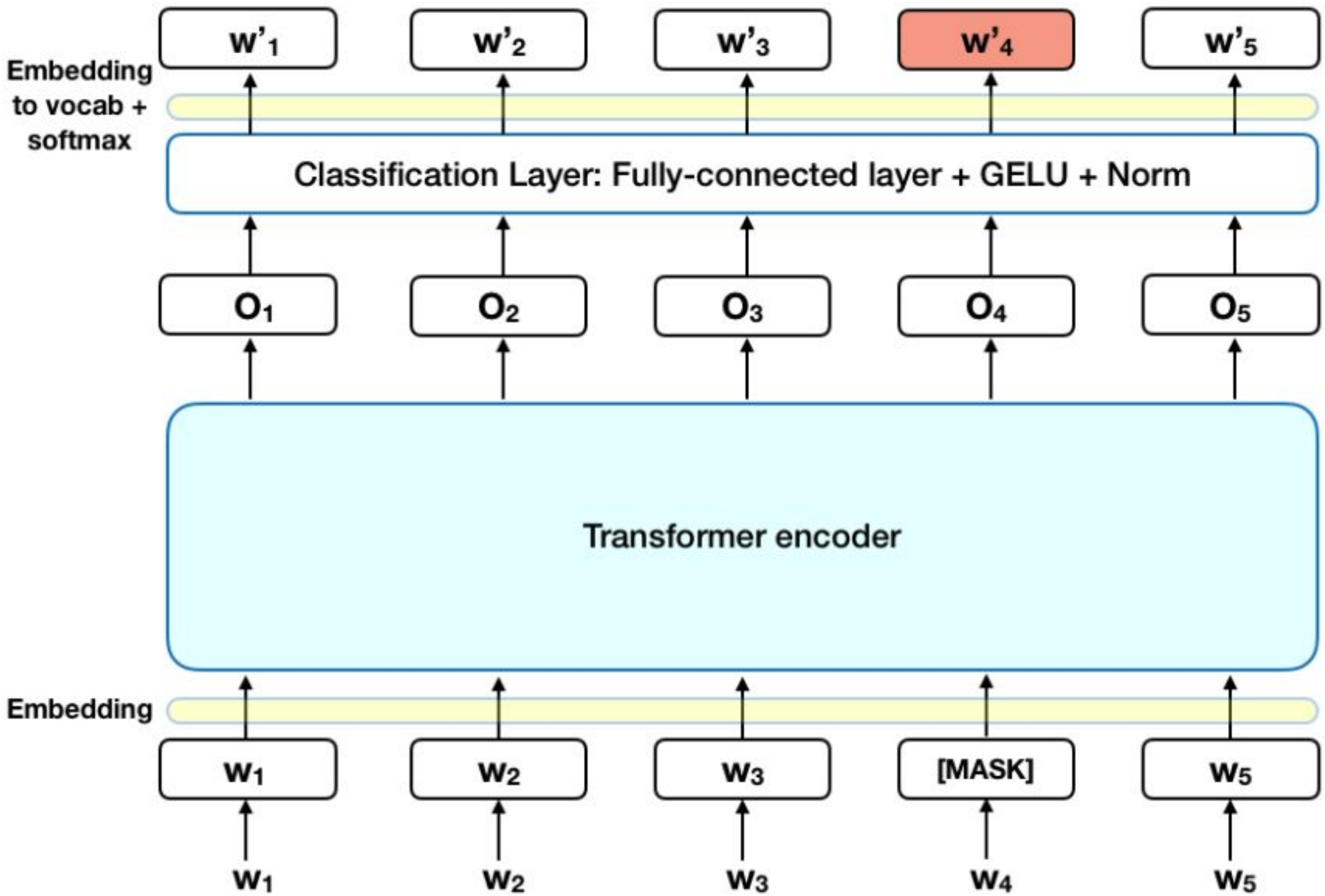
- Models the whole sequence of words, rather than the tri-gram or n-gram.
- Trained on huge datasets. (approx 1 billion tokens)
- Around half a million parameters

In past two years, use of language modeling based pre-training resulted in significant performance improvements in lots of NLP applications, including question answering and sentence classification

How use of language modeling has impacted NLP applications

BERT, XLNET, Transformer XL, ELMO, ULMFit

BERT: Bidirectional Encoder Representations from Transformers



Lots of Parameters and Expensive to Train

- BERT largest model has 340 Million parameters.
- Transformer XL has 277 Million parameters
- Expensive to train from scratch:

From the Google research paper: “training of BERT – Large was performed on 16 Cloud TPUs (64 TPU chips total). Each pretraining took 4 days to complete.” Assuming the training device was Cloud TPU v2, the total price of one-time pretraining should be 16 (devices) * 4 (days) * 24 (hours) * 4.5 (US$ per hour) = **US$6,912.**

Training Cost of XLNET: More than US$ 30,000

https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/

Back to Machine Translation

Noisy Channel Model

- Goal:
  - translation system from **Source to Target** language.
  - Have a model **p(t | s)** which estimates conditional probability of any target language sentence **t** given the source language sentence **s**. Use the training corpus to set the parameters.
- A Noisy Channel Model has two components:
  - **p(t)** the language model
  - **p(s | t)** the translation model
- Using the above two, we can estimate:
  - Learn a distribution $\mathbf{p(t \mid s)} = \boldsymbol{arg\ max_t}\ \mathbf{p(t)p(s \mid t)}$

IBM Model 1: Alignments

- How do we model **p(s | t)**?
- The target sentence **t** has **l** words $t_1, t_2, \ldots, t_l$.
- The source sentence **s** has **m** words $s_1, s_2, \ldots, s_m$.
- The alignment **a** identifies which target word each source word originated from.
- Formally, an alignment a is $\{a_1, \ldots, a_m\}$, where each $a_j \in \{0, \ldots, l\}$
- There are $(l + 1)^m$ possible alignments.

IBM Model 1: Alignments: Example

- E.g., l = 6, m = 7
- t = And the program has been implemented
- s = Le programme a ete mis en application
- One alignment is {2, 3, 4, 5, 6, 6, 6}
- Another (bad!) alignment is

  {1, 1, 1, 1, 1, 1, 1}

References

Language Modeling:
http://www.cs.columbia.edu/~mcollins/lm-spring2013.pdf

Machine Translation:
http://www.cs.columbia.edu/~mcollins/ibm12.pdf