Language Modeling

Natural Language Processing

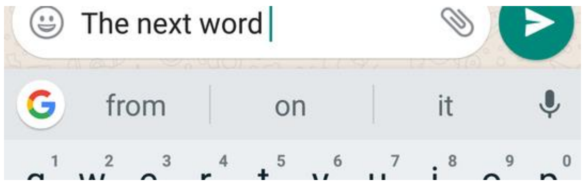
Master in Business Analytics and Big Data

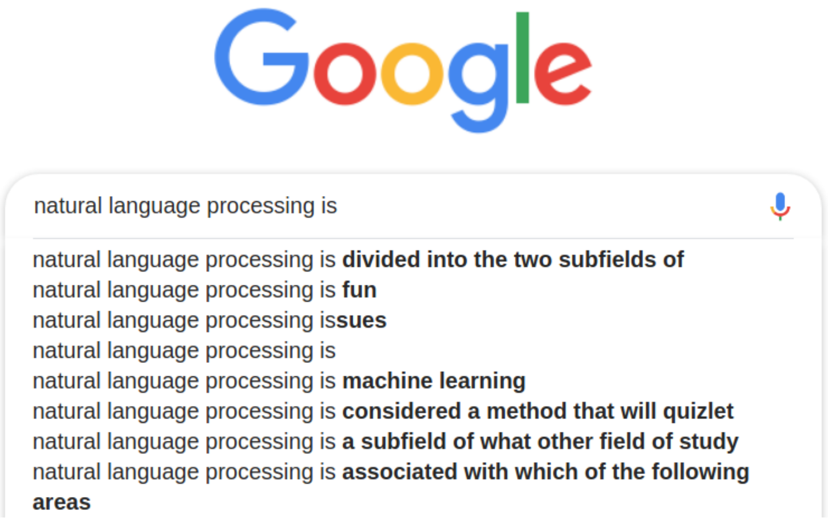
acastellanos@faculty.ie.edu

Language Modeling

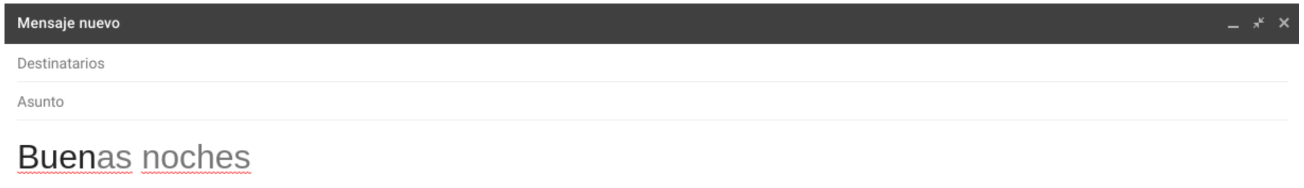
- A **language model** is a model of **what words are more or less likely** to be generated in some language, domain or class.
- Predicts **what the next word will be**, given the words so far.
- Instead of on words, language models can also be defined on **characters or signs, or symbols**.

Language Modeling

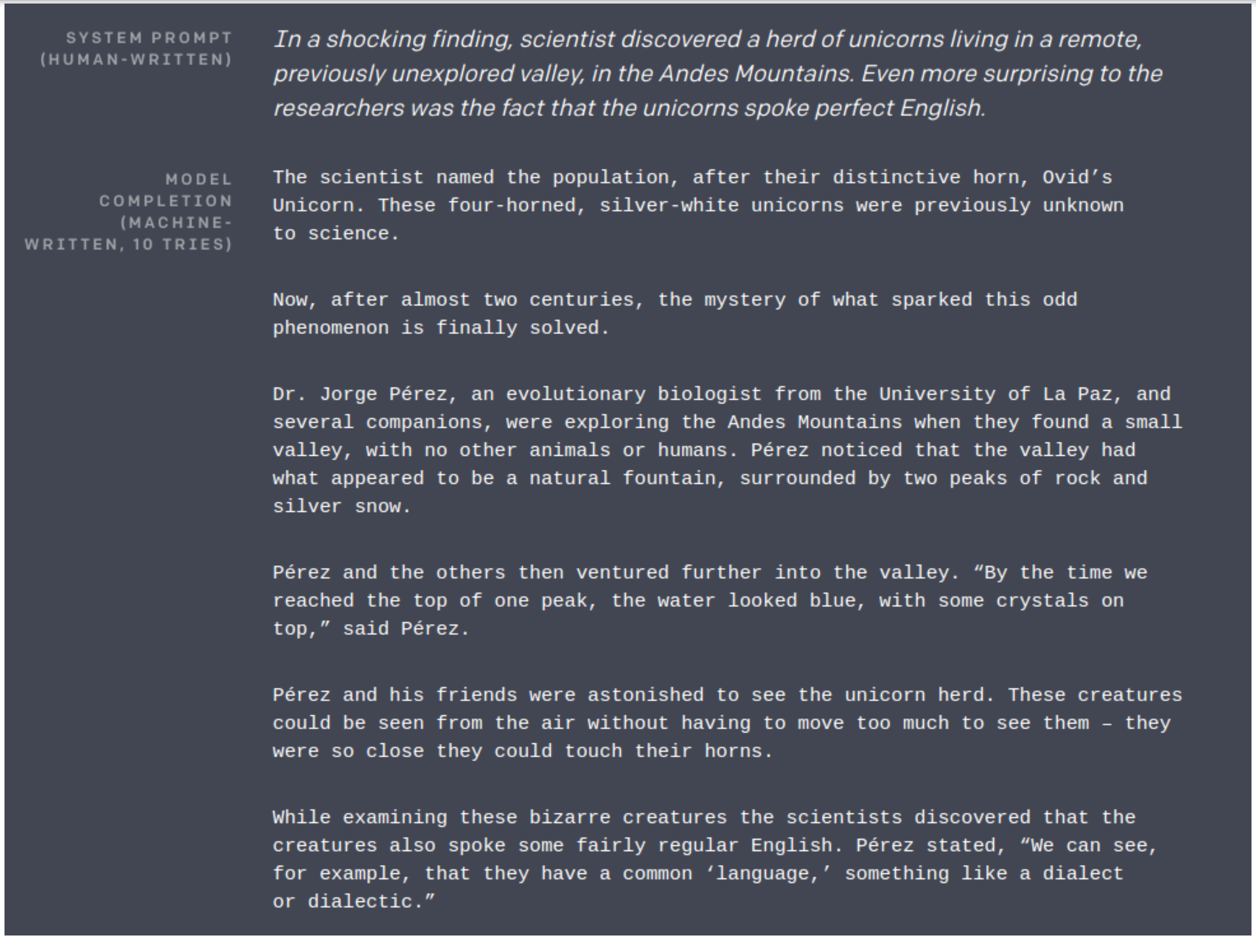

The next word


natural language processing is

natural language processing is **divided into the two subfields of**
natural language processing is **fun**
natural language processing is**sues**
natural language processing is
natural language processing is **machine learning**
natural language processing is **considered a method that will quizlet**
natural language processing is **a subfield of what other field of study**
natural language processing is **associated with which of the following areas**


Mensaje nuevo
Destinatarios
Asunto
Buenas noches

Language Modeling



SYSTEM PROMPT (HUMAN-WRITTEN)

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

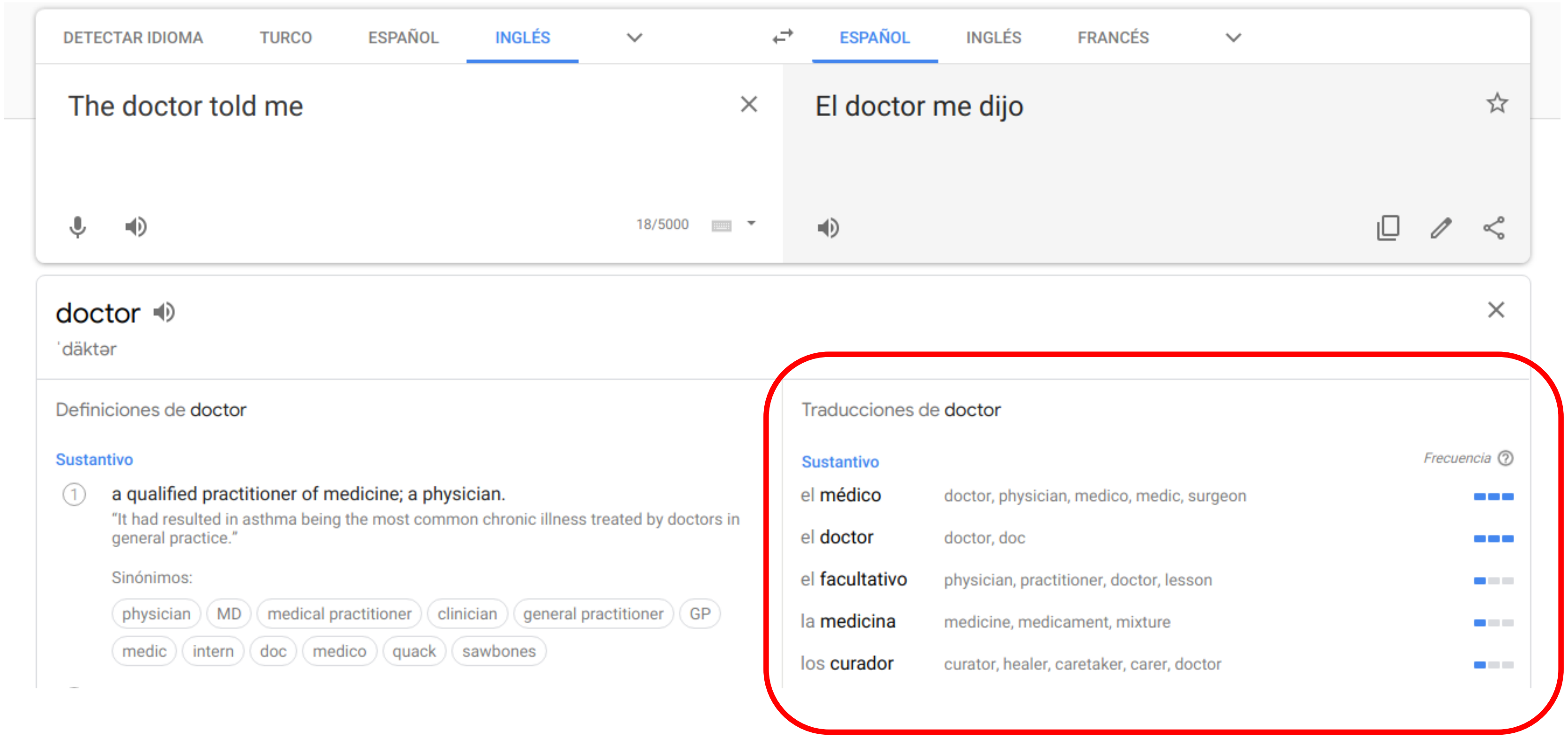
Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

While examining these bizarre creatures the scientists discovered that the creatures also spoke some fairly regular English. Pérez stated, "We can see, for example, that they have a common 'language,' something like a dialect or dialectic."

OpenAI GPT-2. Source: https://openai.com/blog/better-language-models/

Language Modeling



DETECTAR IDIOMA | TURCO | ESPAÑOL | INGLÉS — ESPAÑOL | INGLÉS | FRANCÉS

The doctor told me

18/5000

El doctor me dijo

doctor
ˈdäktər

Definiciones de doctor

Sustantivo

1. a qualified practitioner of medicine; a physician.
   "It had resulted in asthma being the most common chronic illness treated by doctors in general practice."

   Sinónimos: physician, MD, medical practitioner, clinician, general practitioner, GP, medic, intern, doc, medico, quack, sawbones

Traducciones de doctor

Sustantivo		Frecuencia
el médico	doctor, physician, medico, medic, surgeon	▬▬▬
el doctor	doctor, doc	▬▬▬
el facultativo	physician, practitioner, doctor, lesson	▬
la medicina	medicine, medicament, mixture	▬
los curador	curator, healer, caretaker, carer, doctor	▬

Probabilistic Language Modeling

- **How?:**
  - compute the probability of a **sequence of words**:

$$P(W) = P(w_1, w_2, w_3, w_4, w_5 \dots w_n)$$

  - and the probability of an **upcoming word**:

$$P(w_5 | w_1, w_2, w_3, w_4)$$

- **Language Model:** A model that computes them

$$P(W) \quad \text{or} \quad P(w_n \mid w_1, w_2 \dots w_{n-1})$$

Probabilistic Language Models

- **Assign a probability to a sentence**
  - Machine Translation:
    - $P(\textbf{high}\text{ winds tonight}) > P(\textbf{large}\text{ winds tonight})$
  - Spell Correction
    - The office is about fifteen **minuets** from my house
      - $P(\text{about fifteen }\textbf{minutes}\text{ from}) > P(\text{about fifteen }\textbf{minuets}\text{ from})$
  - Speech Recognition
    - $P(\text{I saw a van}) >> P(\text{eyes awe of an})$
  - Summarization, question-answering, Text Classification...

How to compute P(W)

- How to compute **this joint probability**:

$$P(\text{its, water, is, so, transparent, that})$$

- **Chain Rule of Probability**

$$P(x_1, x_2, x_3, \dots, x_n) =$$
$$P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \dots P(x_n|x_1, \dots, x_{n-1})$$

$$P(\text{“its water is so transparent”}) =$$
$$P(\text{its}) \times P(\text{water} \mid \text{its}) \times P(\text{is} \mid \text{its water})$$
$$\times P(\text{so} \mid \text{its water is}) \times P(\text{transparent} \mid \text{its water is so})$$

How to estimate these probabilities

- Now we want to predict the next word: "the"

$$P(\text{the} \mid \text{its water is so transparent that})$$

- We know how to do it, just count

$$\frac{count(\text{its water is so transparent that the})}{count(\text{its water is so transparent that})}$$

- Right?
  - No! Too many possible sentences!
  - Well never see enough data for estimating these

How to estimate these probabilities

- **We'll never see enough data to estimate these**

Number of unique n-grams in Europarl corpus

Order	Unique n-grams	Singletons
unigram	86,700	33,447 (38.6%)
bigram	1,948,935	1,132,844 (58.1%)
trigram	8,092,798	6,022,286 (74.4%)
4-gram	15,303,847	13,081,621 (85.5%)
5-gram	19,882,175	18,324,577 (92.2%)

Markov Assumption

- **Simplifying** assumption
  - The probability of seeing a word is based on **only the previous word**

$$P(\text{the}|\text{its water is so transparent that}) \approx P(\text{the}|\text{that})$$

  - Or **on the two previous ones**

$$P(\text{the}|\text{its water is so transparent that}) \approx P(\text{the}|\text{ transparent that})$$

Markov Assumption: Unigram model

$$P(w_1, w_2, \dots, w_n) \approx \prod_i P(w_i)$$

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

Markov Assumption: Bigram model

- Condition on the previous word:

$$P(w_1, w_2, \ldots, w_n) \approx \prod_i P(w_i | w_{i-j})$$

```
texaco, rose, one, in, this, issue, is, pursuing, growth, in,
a, boiler, house, said, mr., gurria, mexico, 's, motion,
control, proposal, without, permission, from, five, hundred,
fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november
```

Estimating bigram probabilities

- The **Maximum Likelihood Estimate**

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

Practical Example

$$P(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

$P(\texttt{I} \mid \texttt{<s>}) = \frac{2}{3} = .67$ $P(\texttt{Sam} \mid \texttt{<s>}) = \frac{1}{3} = .33$ $P(\texttt{am} \mid \texttt{I}) = \frac{2}{3} = .67$

$P(\texttt{</s>} \mid \texttt{Sam}) = \frac{1}{2} = 0.5$ $P(\texttt{Sam} \mid \texttt{am}) = \frac{1}{2} = .5$ $P(\texttt{do} \mid \texttt{I}) = \frac{1}{3} = .33$

Practical example: Berkeley Restaurant Project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced thai food is what i’m looking for
- tell me about chez panisse
- can you give me a listing of the kinds of food that are available
- i’m looking for a good place to eat breakfast
- when is caffe venezia open during the day

Bigram Counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Bigram probabilities

- Normalize by unigrams:

i	want	to	eat	chinese	food	lunch	spend
2533	927	2417	746	158	1093	341	278

- Result:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

What kinds of knowledge?

- $P(\text{english}|\text{want}) = .0011$
- $P(\text{chinese}|\text{want}) = .0065$
- $P(\text{to}|\text{want}) = .66$
- $P(\text{eat} \mid \text{to}) = .28$
- $P(\text{food} \mid \text{to}) = 0$
- $P(\text{want} \mid \text{spend}) = 0$
- $P(\text{i} \mid \text{<s>}) = .25$

Bigram estimates of sentence probabilities

$$\boldsymbol{P}(\textbf{<s>I want english food </s>}) =$$

$$P(\mathrm{I}|\text{<s>}) \times P(\text{want}|\mathrm{I}) \times P(\text{english}|\text{want}) \times P(\text{food}|\text{english}) \times P(\text{</s>}|\text{food})$$

$$= \mathbf{.000031}$$

Practical Issues

- We do everything in **log space**
  - Avoid **underflow**
  - (also adding is faster than multiplying)

$$\log(p_1 \acute{} \ p_2 \acute{} \ p_3 \acute{} \ p_4) = \log p_1 + \log p_2 + \log p_3 + \log p_4$$

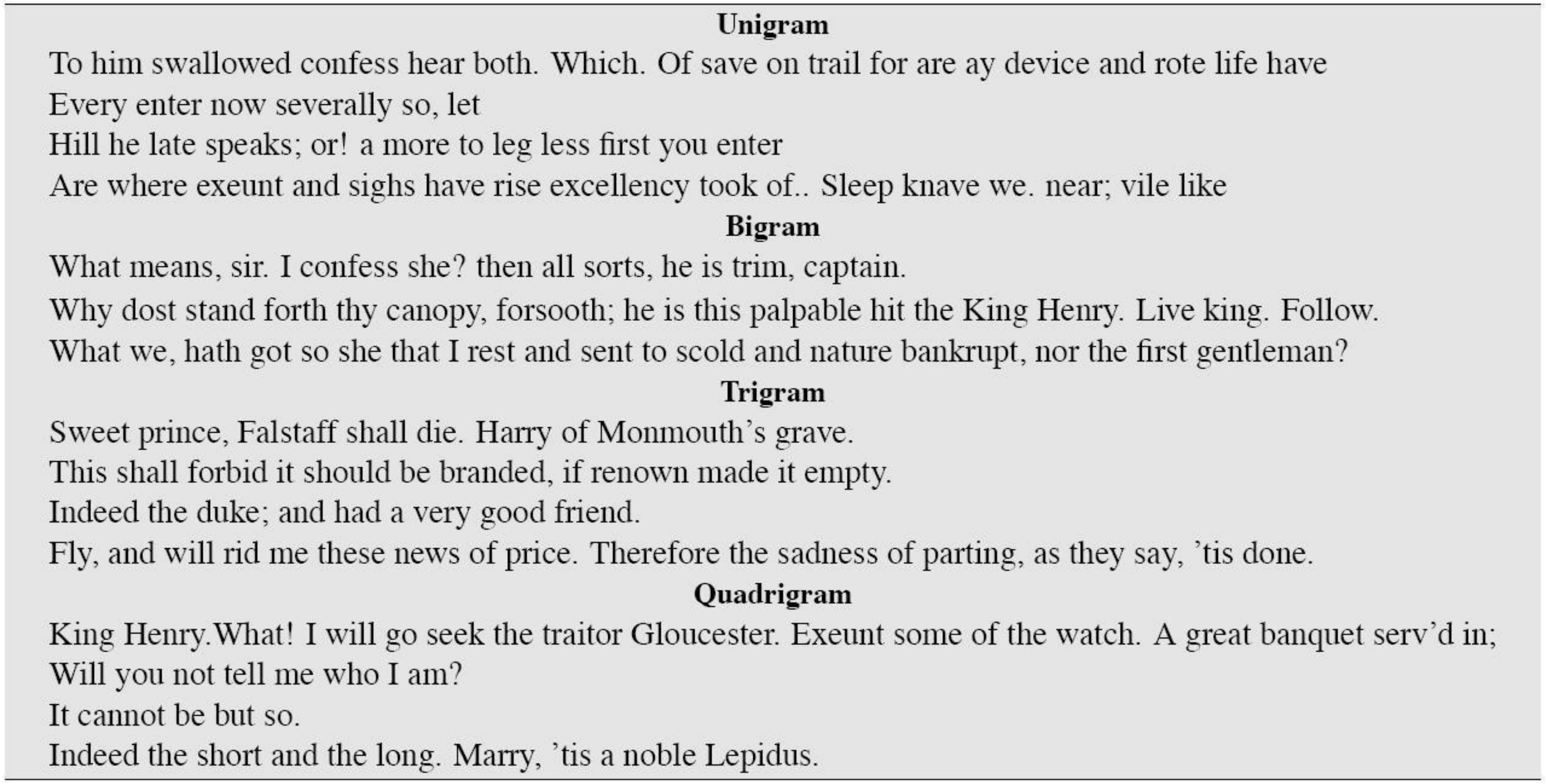
N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an **insufficient model of language**
  - Language has **long-distance dependencies**:

“The computer which I had just put into the machine room on the fifth floor crashed.”

- But **we can often get away** with N-gram models

Approximating Shakespeare



Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

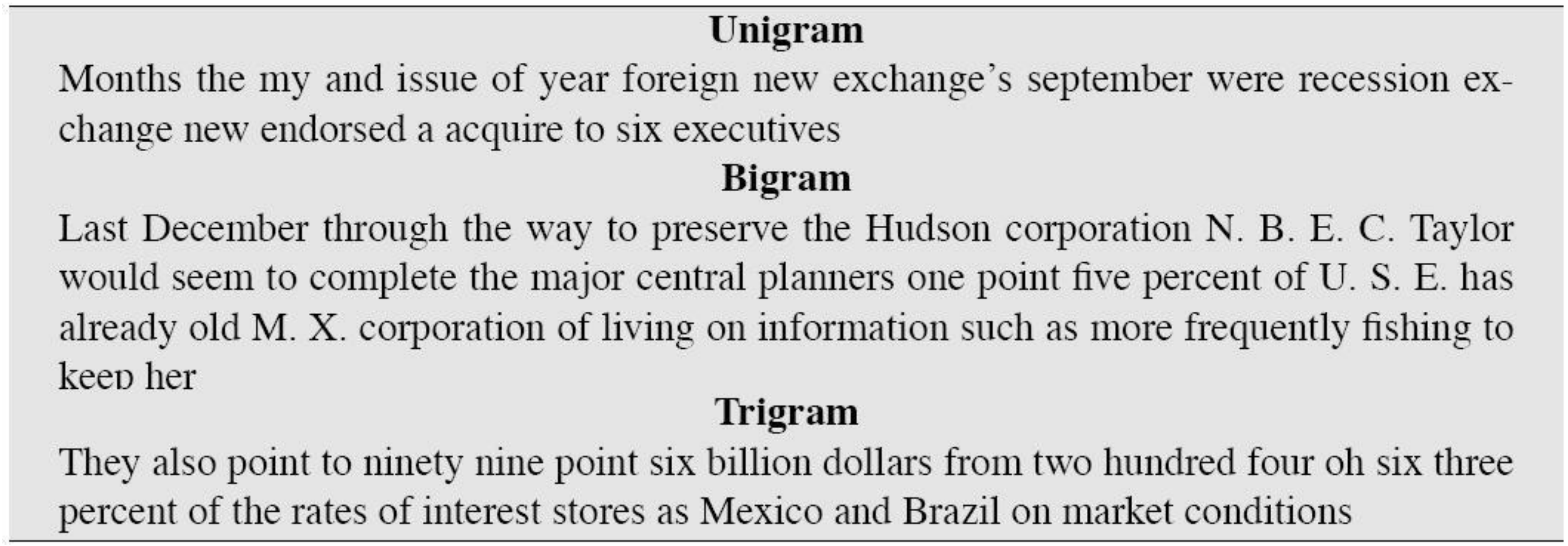
Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.

The wall street journal corpus



Unigram

Months the my and issue of year foreign new exchange's september were recession exchange new endorsed a acquire to six executives

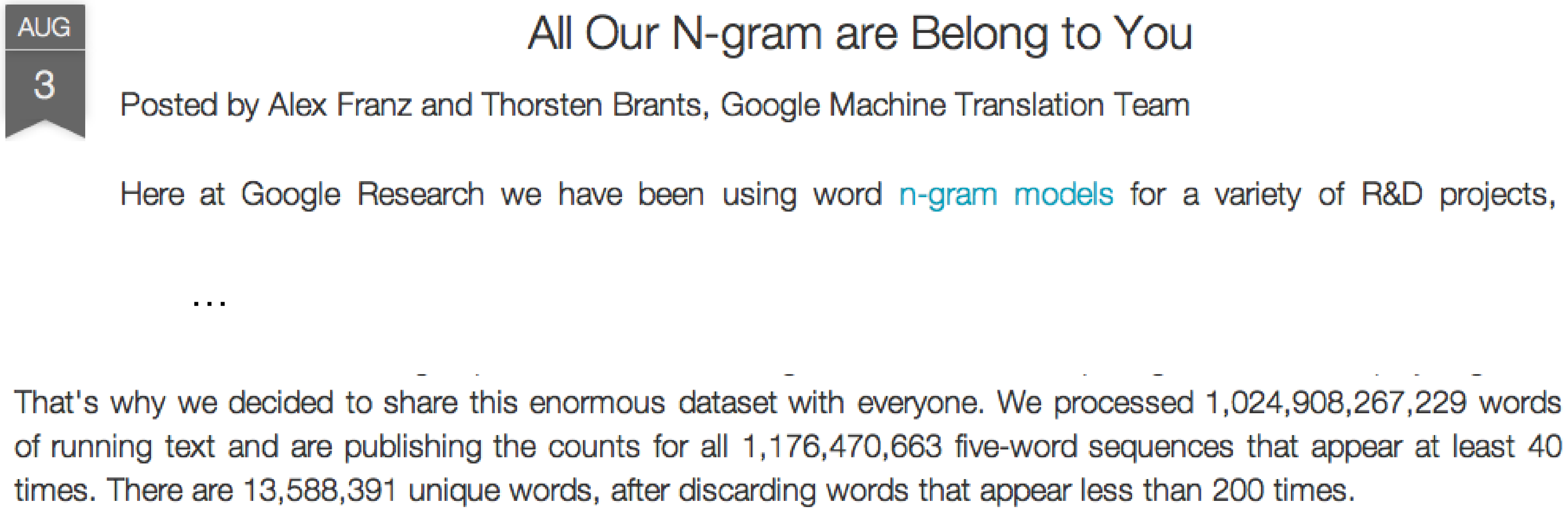
Bigram

Last December through the way to preserve the Hudson corporation N. B. E. C. Taylor would seem to complete the major central planners one point five percent of U. S. E. has already old M. X. corporation of living on information such as more frequently fishing to keep her

Trigram

They also point to ninety nine point six billion dollars from two hundred four oh six three percent of the rates of interest stores as Mexico and Brazil on market conditions

Google N-Gram Release



AUG
3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

...

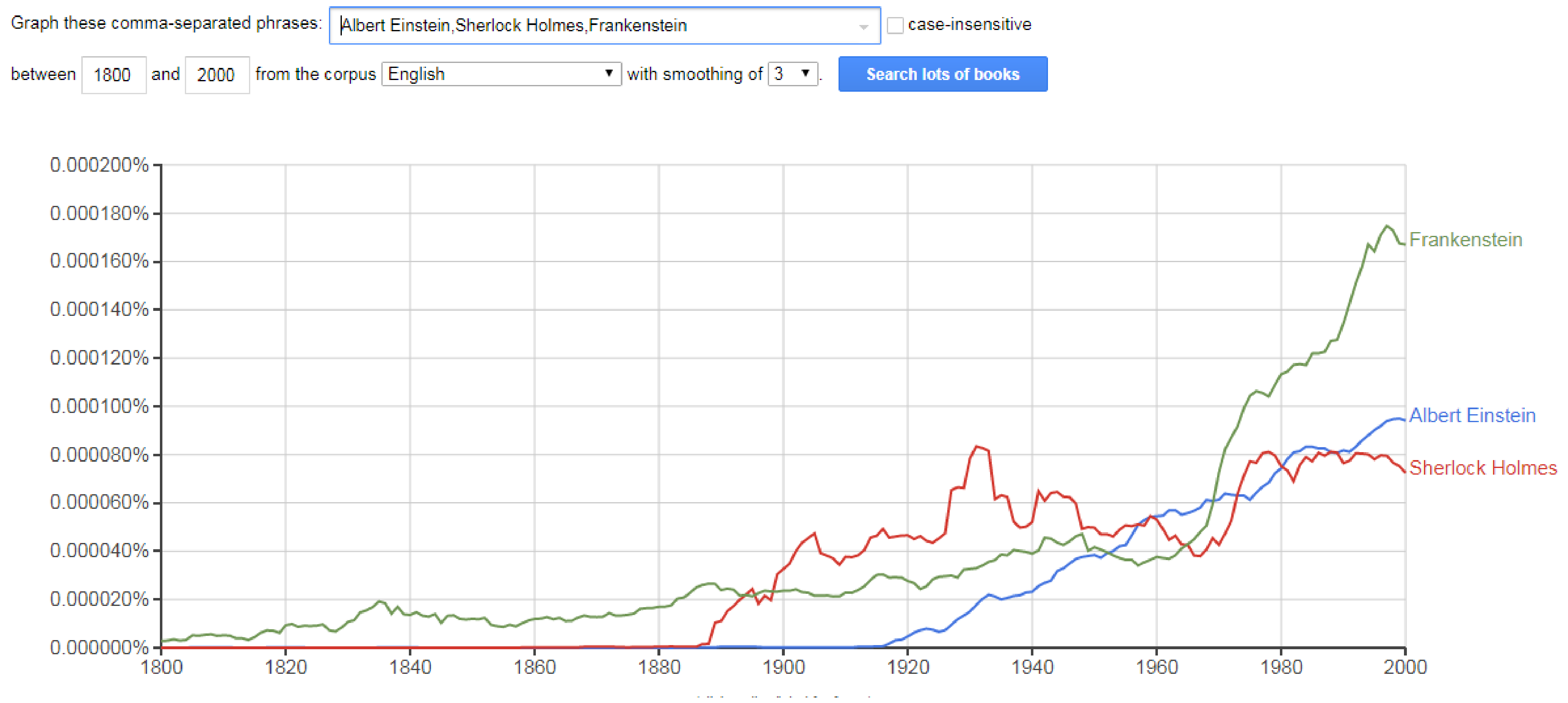
That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- `serve as the incoming 92`
- `serve as the incubator 99`
- `serve as the independent 794`
- `serve as the index 223`
- `serve as the indication 72`
- `serve as the indicator 120`
- `serve as the indicators 45`
- `serve as the indispensable 111`
- `serve as the indispensible 40`
- `serve as the individual 234`

http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Google Book N-grams


Graph these comma-separated phrases: Albert Einstein,Sherlock Holmes,Frankenstein ☐ case-insensitive

between 1800 and 2000 from the corpus English with smoothing of 3 .

Evaluation: How good is our model?

- **Higher probability to "real"** or "frequently observed" sentences **than "ungrammatical**" or "rarely observed" sentences

- We train parameters of our model on a **training set**.

- We test the model's performance on data we haven't seen: a **test set**

Extrinsic evaluation of N-gram models

- **Best evaluation for comparing models A and B**
  - Put **each model in a task**
    - spelling corrector, speech recognizer, MT system
  - **Run the task**, get an accuracy for A and for B
    - How many misspelled words corrected properly
    - How many words translated correctly
  - **Compare accuracy for A and B**

Difficulty of extrinsic evaluation

- **Extrinsic evaluation**
  - Time-consuming; can take days or weeks
- **Intrinsic evaluation**
  - Based on **perplexity**
  - **Bad approximation**
    - **test data has to look just like the training data**
    - So generally only **useful in pilot experiments**
  - But is helpful to think about.

Intuition of Perplexity

- **The Shannon Game:**
  - How well can we **predict the next word**?

I always order pizza with cheese and ____

The 33rd President of the US was ____

I saw a ____

mushrooms	0.1
pepperoni	0.1
anchovies	0.01
….	
fried rice	0.0001
….	
and	1e-100

  - Unigrams are terrible at this game. (Why?)
- A better model assigns a higher probability to the word that actually occurs

Perplexity

The best language model is one that best predicts an unseen test set

- Gives the highest *P(sentence)*

Perplexity is the inverse probability of the test set, normalized by the number of words

$$PP(W) = P(w_1 w_2 \dots w_N)^{-\frac{1}{N}}$$

$$= \sqrt[N]{\frac{1}{P(w_1 w_2 \dots w_N)}}$$

Chain rule:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \dots w_{i-1})}}$$

For bigrams:

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Minimizing perplexity is the same as maximizing probability

The Shannon Game intuition for perplexity

- How hard is the task of recognizing digits ‘0,1,2,3,4,5,6,7,8,9’
  - Perplexity 10
- How hard is recognizing (30,000) names at Microsoft.
  - Perplexity = 30,000
- If a system has to recognize
  - Operator (1 in 4)
  - Sales (1 in 4)
  - Technical Support (1 in 4)
  - 30,000 names (1 in 120,000 each)
  - Perplexity is 53
- Perplexity is weighted equivalent branching factor

Perplexity as branching factor

- Let’s suppose a sentence consisting of random digits
- What is the perplexity of this sentence according to a model that assign P=1/10 to each digit?

$$\begin{aligned} \mathrm{PP}(W) &= P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} \\ &= (\frac{1}{10}^N)^{-\frac{1}{N}} \\ &= \frac{1}{10}^{-1} \\ &= 10 \end{aligned}$$

Lower perplexity = better model

- Training 38 million words, test 1.5 million words, WSJ

N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

The Shannon Visualization Method

- Choose a random bigram (<s>, w) according to its probability
- Now choose a random bigram (w, x) according to its probability
- And so on until we choose </s>
- Then string the words together

```
<s> I
    I want
      want to
           to eat
              eat Chinese
                  Chinese food
                          food  </s>
I want to eat Chinese food
```

Shakespeare as corpus

- Included in nltk.corpus
- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.
  - So 99.96% of the possible bigrams were never seen (have zero entries in the table)
- Quadrigrams worse: What's coming out looks like Shakespeare because it ***is*** Shakespeare

Overfitting

- N-grams only **work well if test and training looks alike**
  - In real life, **it often doesn’t**
  - We need to models that **generalize**
  - One kind of generalization: **Zeros**
    - Things that don’t ever occur in the training set but occur in the test set

Zeros

- Training set:
  … denied the allegations
  … denied the reports
  … denied the claims
  … denied the request

- Test set
  … denied the offer
  … denied the loan

$$P(\text{“offer”} \mid \text{denied the}) = 0$$

- We cannot compute perplexity (can’t divide by 0)!
- Smoothing

Add-one estimation

- **Laplace smoothing**
  - Add one to all the counts

- MLE estimate:

$$P_{MLE}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

- Add-1 estimate:

$$P_{Add\text{-}1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)+1}{c(w_{i-1})+V}$$

Maximum Likelihood Estimates

- The maximum likelihood estimate
  - of some parameter of a model M from a training set T
  - maximizes the likelihood of the training set T given the model M
- Suppose the word “bagel” occurs 400 times in a corpus of a million words
- What is the probability that a random word from some other text will be “bagel”?
- MLE estimate is 400/1,000,000 = .0004
- This may be a bad estimate for some other corpus
  - But it is the **estimate** that makes it **most likely** that “bagel” will occur 400 times in a million word corpus.

Laplace smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Laplace-smoothed bigrams

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Reconstituted counts

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n)+1] \times C(w_{n-1})}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

Compare with raw bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

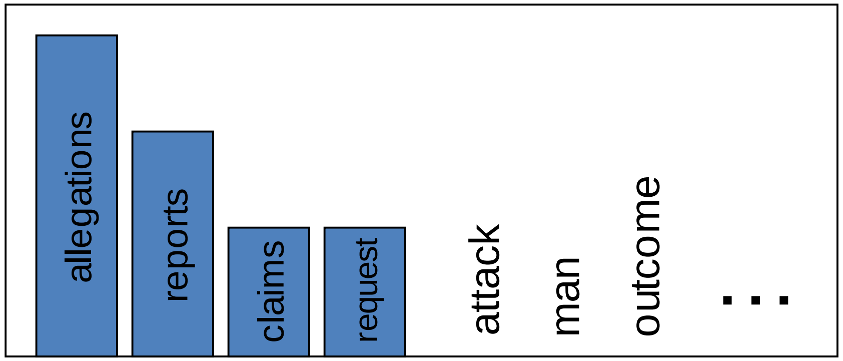
	i	want	to	eat	chinese	food	lunch	spend
i	3.8	527	0.64	6.4	0.64	0.64	0.64	1.9
want	1.2	0.39	238	0.78	2.7	2.7	2.3	0.78
to	1.9	0.63	3.1	430	1.9	0.63	4.4	133
eat	0.34	0.34	1	0.34	5.8	1	15	0.34
chinese	0.2	0.098	0.098	0.098	0.098	8.2	0.2	0.098
food	6.9	0.43	6.9	0.43	0.86	2.2	0.43	0.43
lunch	0.57	0.19	0.19	0.19	0.19	0.38	0.19	0.19
spend	0.32	0.16	0.32	0.16	0.16	0.16	0.16	0.16

The intuition of smoothing

- When we have sparse statistics:

  P(w | denied the)
  3 allegations
  2 reports
  1 claims
  1 request

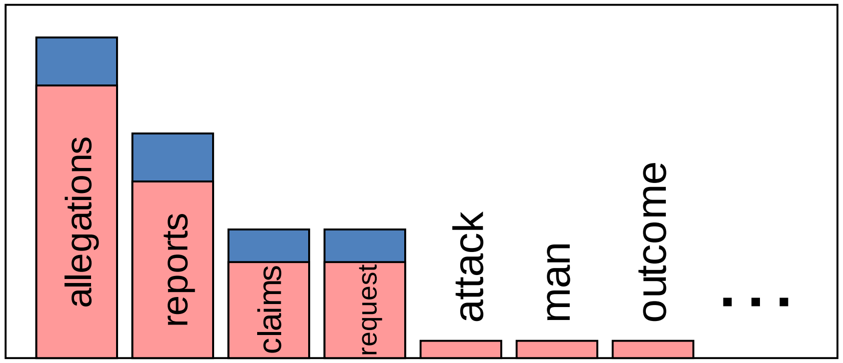
  7 total



- Steal probability mass to generalize better

  P(w | denied the)
  2.5 allegations
  1.5 reports
  0.5 claims
  0.5 request
  2 other

  7 total



Add-1 estimation is a blunt instrument

- So **add-1 isn't used for N-grams**:
  - We'll see better methods

- But add-1 is **used to smooth other NLP models**
  - For text classification (Naïve Bayes)
  - **In domains where the number of zeros isn't so huge.**

Backoff and Interpolation

- Sometimes it helps to use **less context**
    - Condition on less context for contexts you haven’t learned much about
- **Back off:**
    - use trigram if you have good evidence,
    - otherwise bigram, otherwise unigram
- **Interpolation:**
    - mix unigram, bigram, trigram
- **Interpolation works better**

Back off

- In given corpus, **we may never observe:**
  - Scottish beer drinkers
  - Scottish beer eaters
- Both have **count = 0**
  - LM will assign them **same probability**
- **Better --> back off to bigrams:**
  - beer drinkers
  - beer eaters

Linear Interpolation

- Simple interpolation

$$\begin{aligned}\hat{P}(w_n|w_{n-1}w_{n-2}) &= \lambda_1 P(w_n|w_{n-1}w_{n-2}) \\ &+\lambda_2 P(w_n|w_{n-1}) \\ &+\lambda_3 P(w_n)\end{aligned}$$

$$\sum_i \lambda_i = 1$$

- Lambdas conditional on context:

$$\begin{aligned}\hat{P}(w_n|w_{n-2}w_{n-1}) &= \lambda_1(w_{n-2}^{n-1})P(w_n|w_{n-2}w_{n-1}) \\ &+\lambda_2(w_{n-2}^{n-1})P(w_n|w_{n-1}) \\ &+\lambda_3(w_{n-2}^{n-1})P(w_n)\end{aligned}$$

Unknown words

- If **we know** all the words in advanced
  - Vocabulary V is fixed
- If **we don't know** this
  - **Out Of Vocabulary** = OOV words
  - Create an unknown word token **<UNK>**
  - **Training of <UNK>** probabilities
    - Create a fixed lexicon L of size V
    - Any **training word not in L** changed to **<UNK>**
    - Now we train its probabilities like a normal word
  - **At decoding time**
    - Use **UNK probabilities** for any word not in training

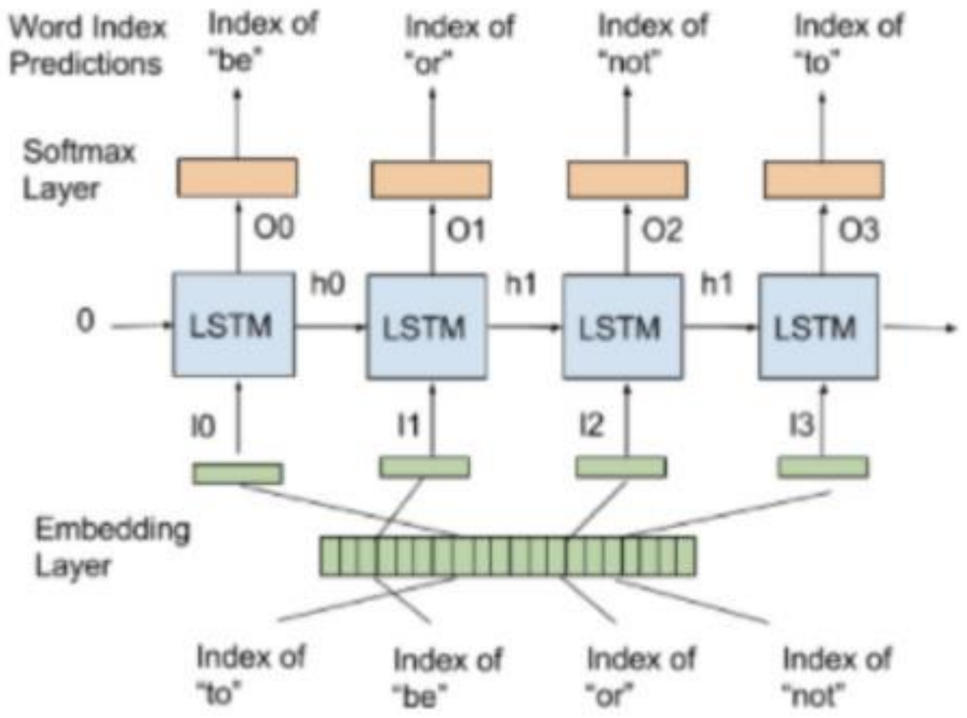
Huge web-scale n-grams

e.g., Google N-gram corpus

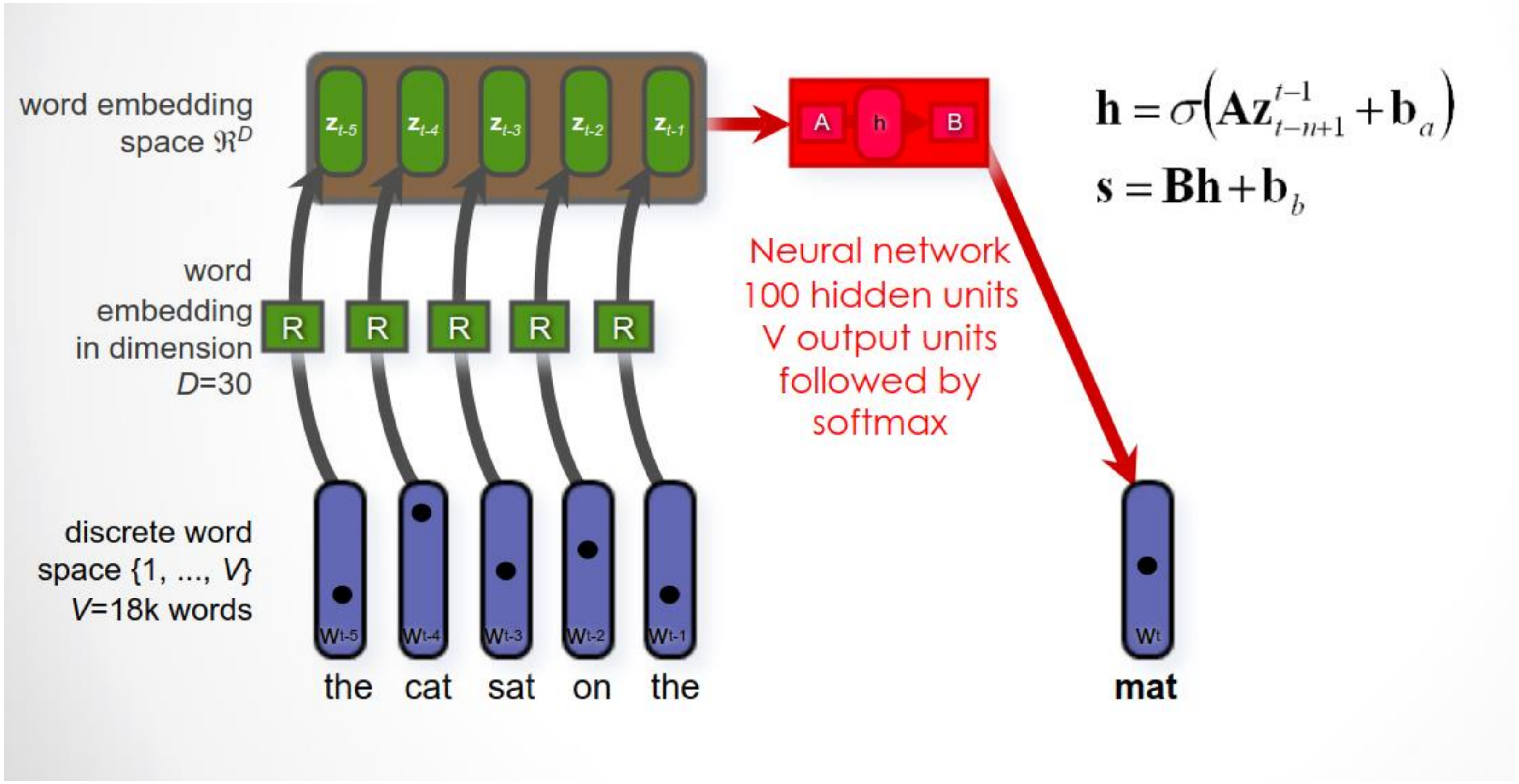
- **Pruning**
  - Only store N-grams with **count > threshold.**
    - Remove singletons of higher-order n-grams
  - **Entropy-based** pruning
- **Efficiency**
  - Efficient data structures like trees
  - Store words as indexes, not strings
    - Use Huffman coding to fit large numbers of words into two bytes
  - Quantize probabilities (4-8 bits instead of 8-byte float)

Recurrent Neural Networks

- Potentially learn to exploit **long-range interactions**



Neural Probabilistic Language Model

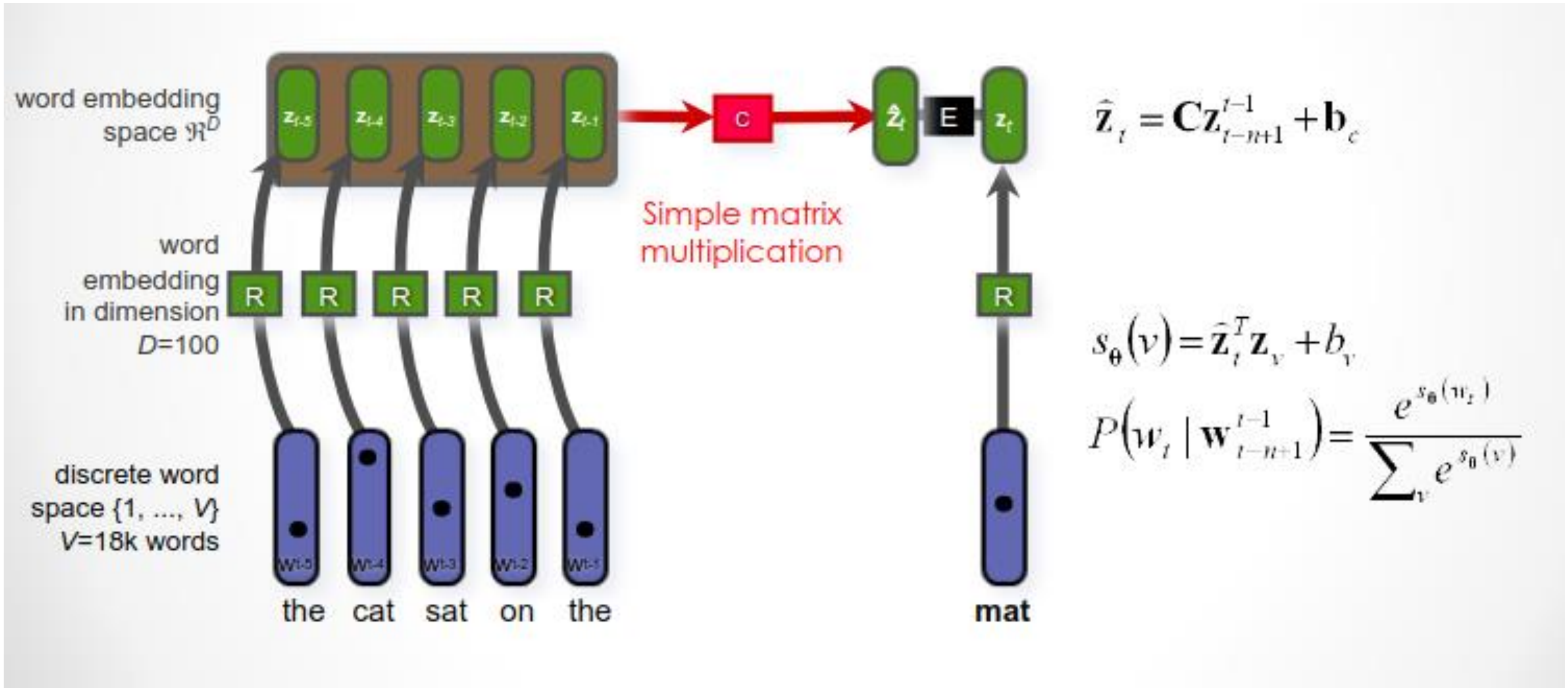

word embedding space $\Re^D$
z_{t-5}
z_{t-4}
z_{t-3}
z_{t-2}
z_{t-1}
A
h
B
word embedding in dimension D=30
R
R
R
R
R
Neural network 100 hidden units V output units followed by softmax
discrete word space {1, ..., V} V=18k words
Wt-5
Wt-4
Wt-3
Wt-2
Wt-1
Wt
the cat sat on the
mat

$$\mathbf{h} = \sigma\left(\mathbf{A}\mathbf{z}_{t-n+1}^{t-1} + \mathbf{b}_a\right)$$

$$\mathbf{s} = \mathbf{B}\mathbf{h} + \mathbf{b}_b$$

Connectionist language modeling for large vocabulary continuous speech recognition

Log-Bilinear Language Model


word embedding space $\Re^D$
word embedding in dimension D=100
discrete word space {1, ..., V} V=18k words
z_{t-5}
z_{t-4}
z_{t-3}
z_{t-2}
z_{t-1}
C
$\hat{z}_t$
E
z_t
Simple matrix multiplication
R
w_{t-5}
w_{t-4}
w_{t-3}
w_{t-2}
w_{t-1}
the
cat
sat
on
the
mat

$$\hat{\mathbf{z}}_t = \mathbf{C}\mathbf{z}_{t-n+1}^{t-1} + \mathbf{b}_c$$

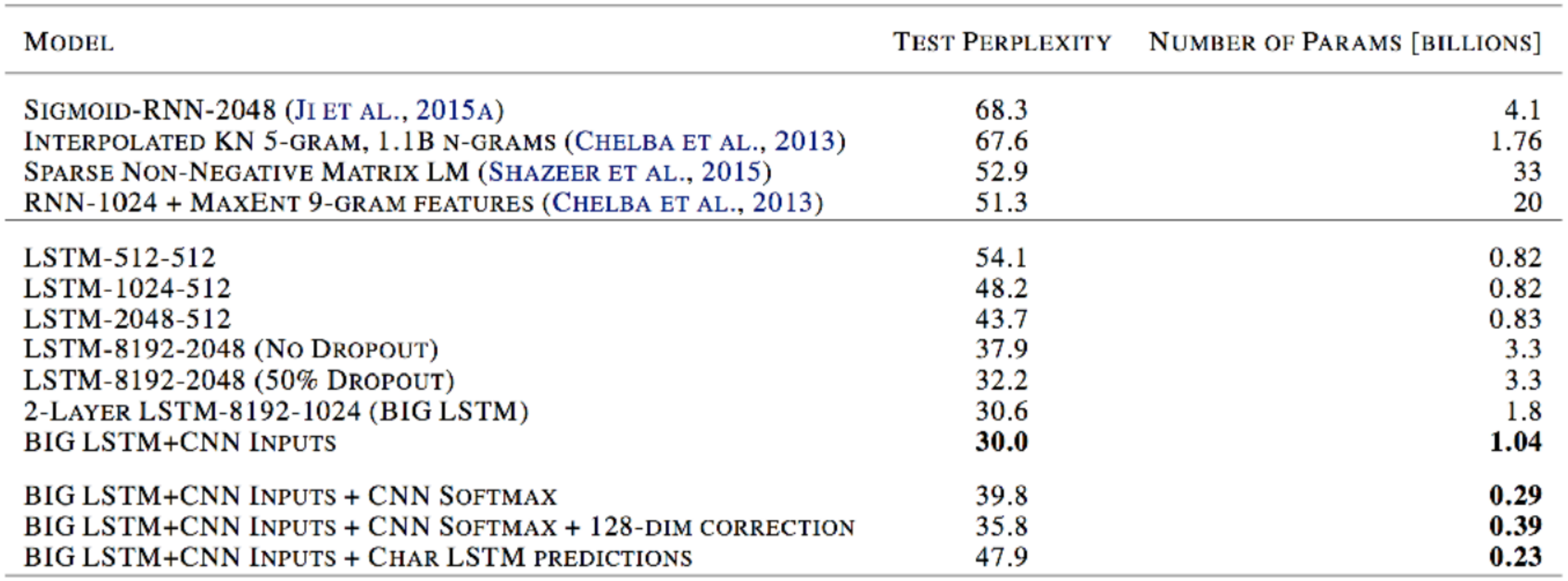
$$s_\theta(v) = \hat{\mathbf{z}}_t^T \mathbf{z}_v + b_v$$

$$P\left(w_t \mid \mathbf{w}_{t-n+1}^{t-1}\right) = \frac{e^{s_\theta(w_t)}}{\sum_v e^{s_\theta(v)}}$$

Three new graphical models for statistical language modelling

Recurrent Neural Networks

- Potentially learn to exploit **long-range interactions**
- Dramatically **reduced perplexity** at the cost of **much more complex models**



Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

Recurrent Neural Networks

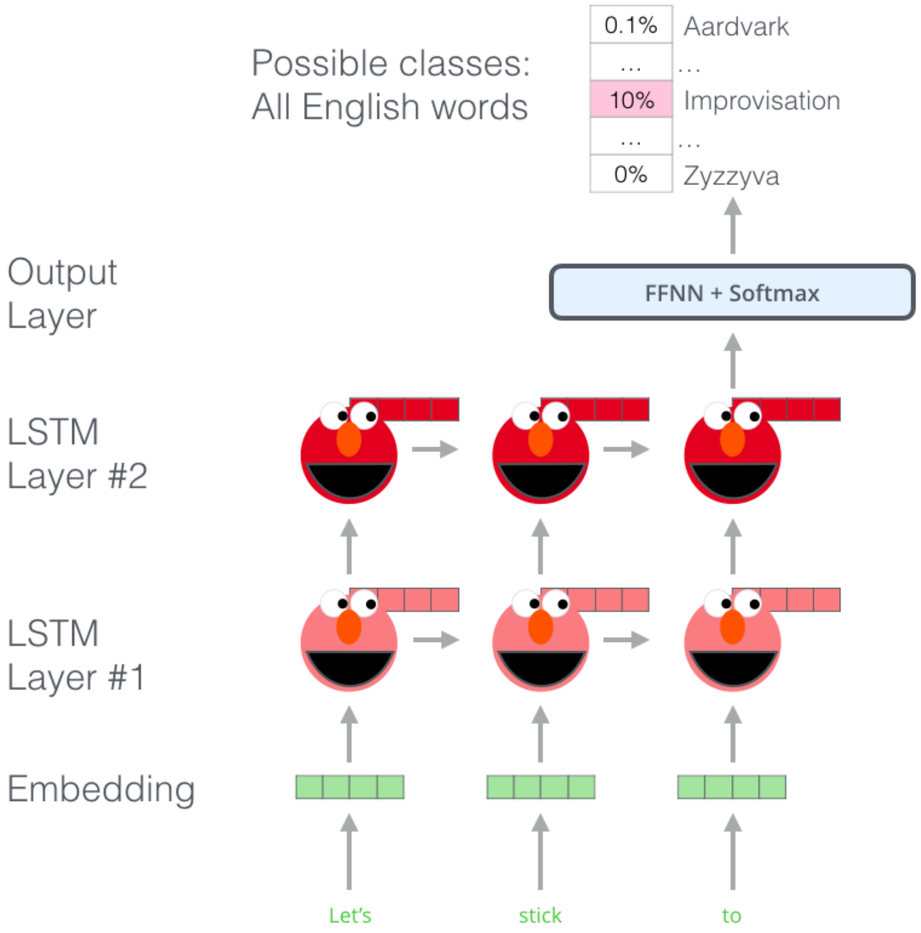
- Potentially learn to exploit **long-range interactions**
- Dramatically **reduced perplexity** at the cost of **much more complex models**
- Different levels
  - **Word-level**
    - Tensorflow tutorial:
      https://www.tensorflow.org/tutorials/recurrent#language_modeling
      https://github.com/tensorflow/models/tree/master/tutorials/rnn/ptb
  - **Character-level**
    - http://gluon.mxnet.io/chapter05_recurrent-neural-networks/simple-rnn.html
    - http://alpha-epsilon.de/python/2018/04/04/an-lstm-based-startup-name-generator/

Recurrent Neural Networks

- **To learn more**
  - M. Sundermeyer, H. Ney and R. Schlüter, **From Feedforward to Recurrent LSTM Neural Networks for Language Modeling**, in IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 23, no. 3, pp. 517-529, 2015.
  - Kazuki Irie, Zoltan Tuske, Tamer Alkhouli, Ralf Schluter, Hermann Ney, **LSTM, GRU, Highway and a Bit of Attention: An Empirical Overview for Language Modeling in Speech Recognition**, Interspeech, 2016
  - Ke Tran, Arianna Bisazza, Christof Monz, **Recurrent Memory Networks for Language Modeling**, NAACL, 2016
  - Jianpeng Cheng, Li Dong and Mirella Lapata, **Long Short-Term Memory-Networks for Machine Reading**, arXiv preprint, 2016

- **Tutorial on neural probabilistic language model**
  - http://slideplayer.com/slide/4559559/

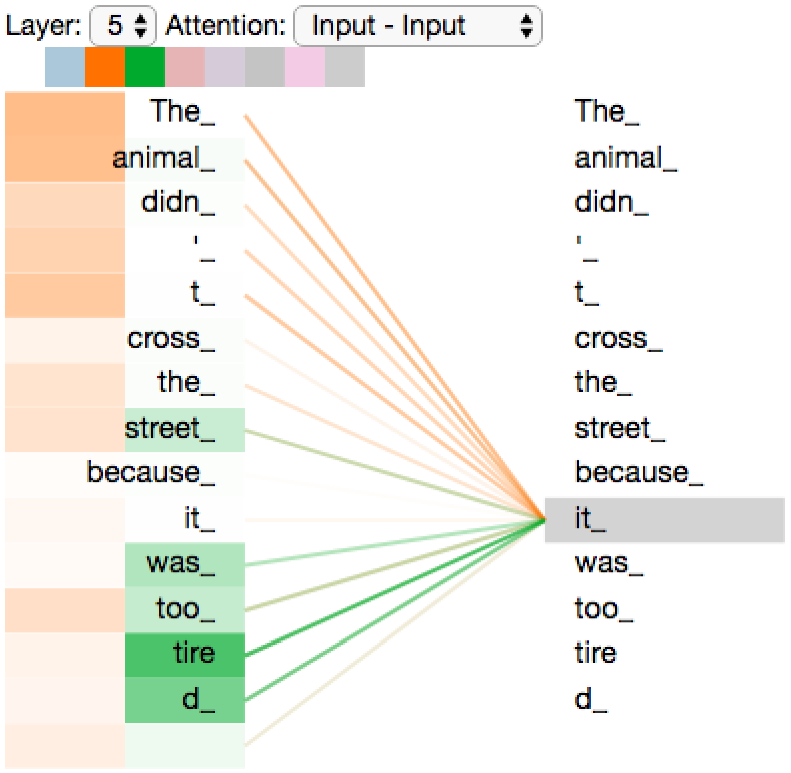
ELMo: throwing context into RNN



- Embedding **based on the context**
- ELMo **looks at the entire sentence** before assigning each word in it an embedding
- Trained on **classical LM objective**: predict the next word in a sequence of words

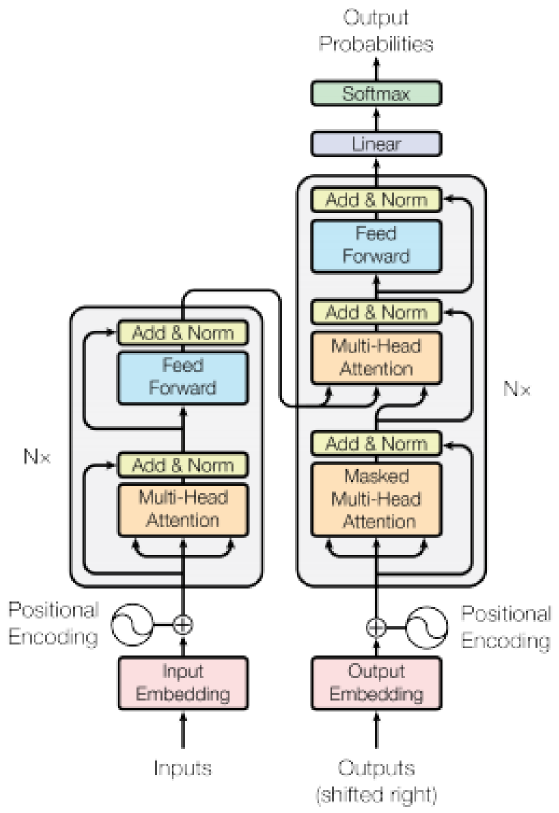
Source: http://jalammar.github.io/illustrated-bert/

Make way for The Transformer (GPT)



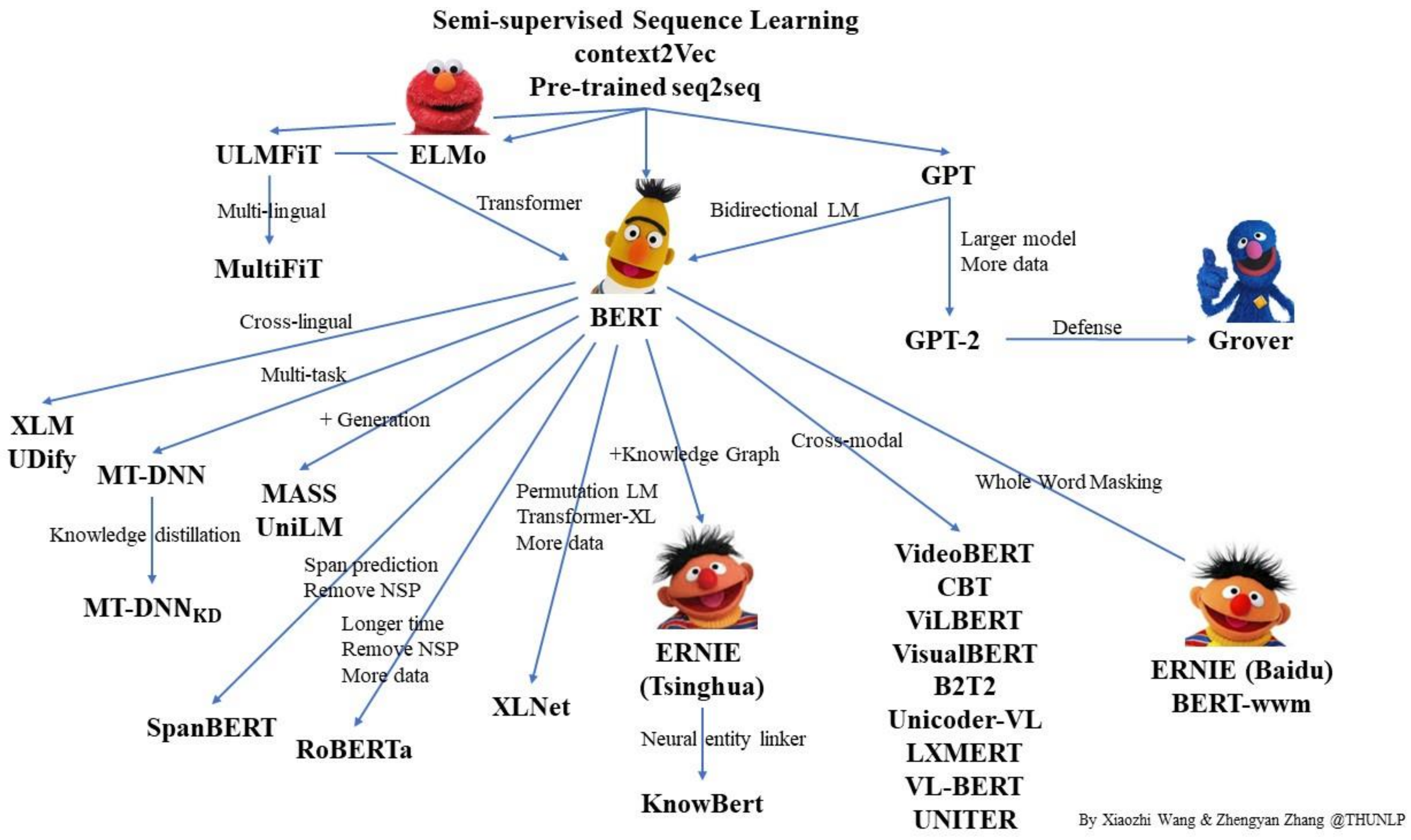
- Based on the idea of **Self-Attention (Attention is all you need)**
- For each word in the input sentence -> create an **attention vector**
- Attention vector **focus on the sentence itself (hence the self) to find words related** to the target word

BERT



- Masked Language Modeling
- Stack a bunch of these self-attention layers (plus other DL stuff)
  - Each of them focus on a particular piece of information in the original sentence
- Gather billions of documents
- Throw some dollars to rent GPUs

The entire Sesame Street Family


Semi-supervised Sequence Learning
context2Vec
Pre-trained seq2seq
ULMFiT
ELMo
GPT
Multi-lingual
MultiFiT
Transformer
Bidirectional LM
Larger model
More data
BERT
GPT-2
Defense
Grover
Cross-lingual
Multi-task
XLM
UDify
MT-DNN
+ Generation
MASS
UniLM
Knowledge distillation
MT-DNNKD
Span prediction
Remove NSP
Longer time
Remove NSP
More data
SpanBERT
RoBERTa
Permutation LM
Transformer-XL
More data
XLNet
+Knowledge Graph
ERNIE
(Tsinghua)
Neural entity linker
KnowBert
Cross-modal
VideoBERT
CBT
ViLBERT
VisualBERT
B2T2
Unicoder-VL
LXMERT
VL-BERT
UNITER
Whole Word Masking
ERNIE (Baidu)
BERT-wwm
By Xiaozhi Wang & Zhengyan Zhang @THUNLP