Small and Medium-sized language models

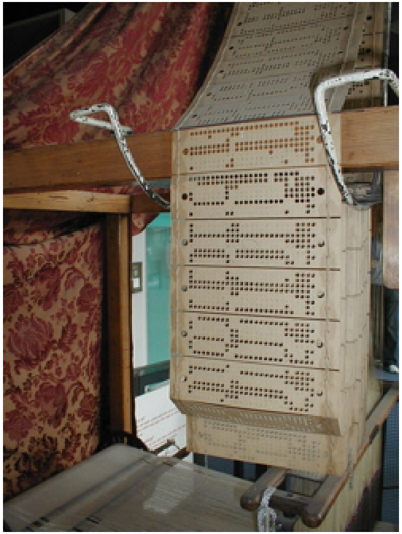
A lecture about Computers and Language for Pavel Kral's UIR class.

Stephen Taylor

9 April 2025

Overview

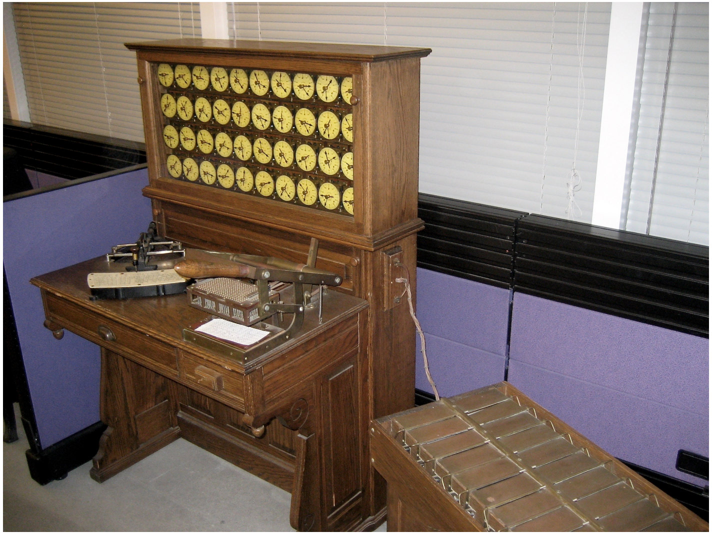
The Jacquard loom

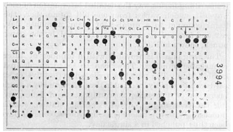


Patented in 1804

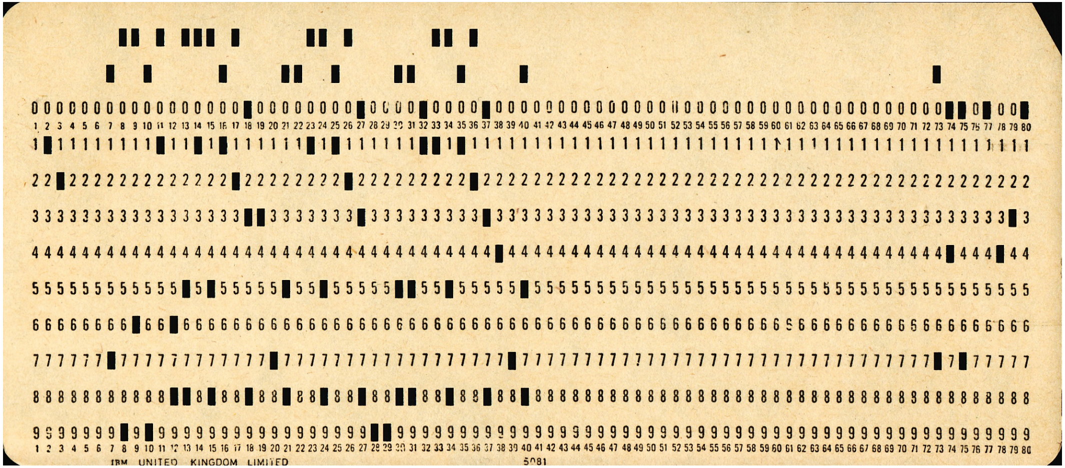
Mechanical process lifts specific threads for each hole.

Hollerith's census tabulator

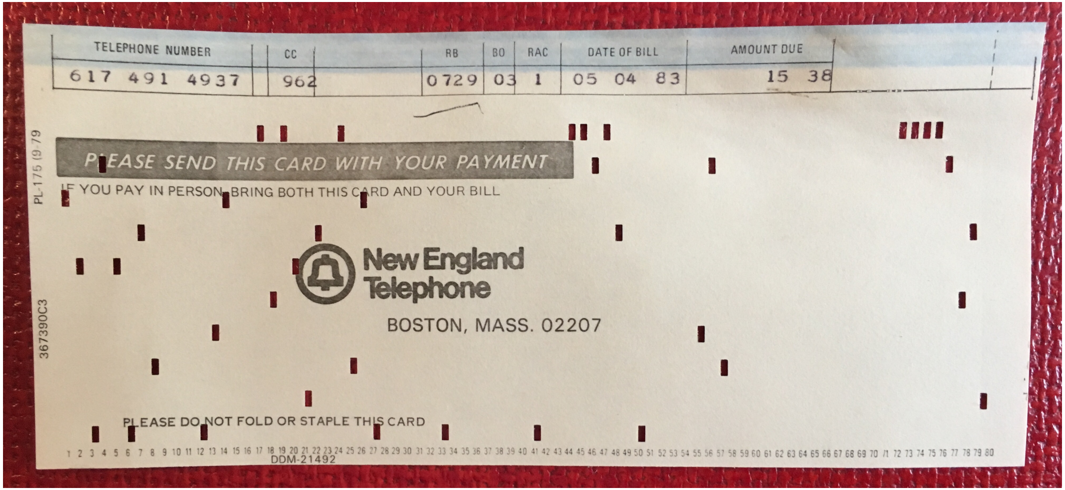




IBM card



a phone bill with card punches



TELEPHONE NUMBER	CC	RB	BO	RAC	DATE OF BILL	AMOUNT DUE
617 491 4937	962	0729	03	1	05 04 83	15 38

PL-175 (9-79)

PLEASE SEND THIS CARD WITH YOUR PAYMENT

IF YOU PAY IN PERSON, BRING BOTH THIS CARD AND YOUR BILL

New England Telephone

BOSTON, MASS. 02207

367390C3

PLEASE DO NOT FOLD OR STAPLE THIS CARD

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80

DDM-21492

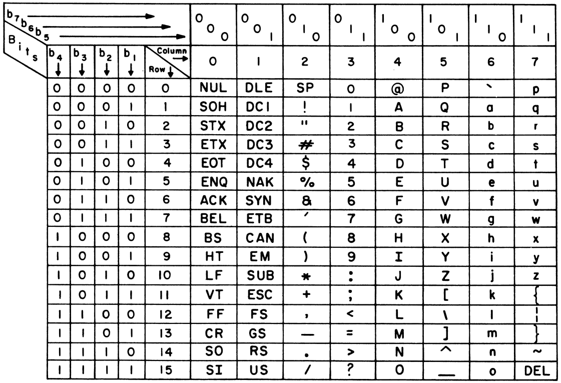
"Do not fold, spindle, or mutilate"

IBM card code

```
    _______________________________________________
   / &-0123456789ABCDEFGHIJKLMNOPQR/STUVWXYZ
12|  x           xxxxxxxxx
11|   x                   xxxxxxxxx
 0|    x                           xxxxxxxxx
 1|     x        x        x        x
 2|      x        x        x        x
 3|       x        x        x        x
 4|        x        x        x        x
 5|         x        x        x        x
 6|          x        x        x        x
 7|           x        x        x        x
 8|            x        x        x        x
 9|             x        x        x        x
  |________________________________________________
```

ASCII

In 1967, the American Standard Code for Information Interchange was published to solve the problem of transferring information between different brands of computers. It took a few more years before all the companies supported it; IBM dragged their feet the longest.



b_7 b_6 b_5 → Bits					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 ↓	b_3 ↓	b_2 ↓	b_1 ↓	Column → Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	¦
1	1	0	1	13	CR	GS	—	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

Unicode rationale

- The advent of the internet makes it necessary to mix languages.
- Even as recently as 30 years ago, Arabic newspapers used to post their pages to the web as images.
- But images of text take up much more space in memory than character codes. The dot-matrix printer I had back then had a matrix of 35 dots for printing a single character; that's 35 bits/character instead of 7. And the readability of those dotty characters is nothing special.
- A PDF image might store a block of 64 characters in a 90000 bit image – that's over 1500 bits / character.

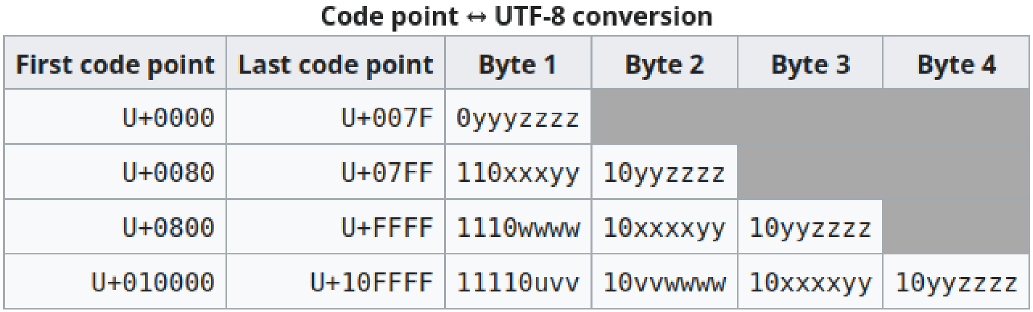
Unicode

```
Basic Latin (ASCII)      0000    007f
Latin-1 Supplement       0080    00ff
Latin Extended-A         0100    017f
Latin Extended-B         0180    024f

Greek                    0370    03ff
Cyrillic                 0400    04ff
Hebrew                   0590    05ff
Arabic                   0600    06ff
Syriac                   0700    074f
Arabic Supplement        0750    077f
Samaritan                0800    083f
.
.
.
Latin Extended-C         2c60    2c7f
.
.
.
CJK Ideographs           4e00    9FFF
```

utf8

- A problem with Unicode is that the codes are wider; the largest code value on the previous slide is 9FFF, which is 16-bit hexadecimal number.
- For awhile it seemed like that might be enough. Last time I checked, Java stored a character as sixteen bits,
- but there are new releases of Unicode every year, and as of last fall, the highest code was something like 10 FFFF, which is 21 bits.
- We call that Unicode32, because it's clear that it will keep growing.

utf8



Code point ↔ UTF-8 conversion

First code point	Last code point	Byte 1	Byte 2	Byte 3	Byte 4
U+0000	U+007F	0yyyzzzz			
U+0080	U+07FF	110xxxyy	10yyzzzz		
U+0800	U+FFFF	1110wwww	10xxxxyy	10yyzzzz	
U+010000	U+10FFFF	11110uvv	10vvwwww	10xxxxyy	10yyzzzz

Single character frequency language modeling

So here's the scheme for a 1-character frequency model:

- Select a representative corpus.
- count the occurrences of each letter in the corpus. A python dictionary is a handy data structure for this.
- Compute the probabilities of each character by dividing each count by the total number of characters in the corpus.

What can you do with a 1-character frequency model?

- Although you have only saved frequencies, they are good approximation of probabilities – just as the individual character counts add up to the total number of characters, the frequencies add up to 1.
- Although the individual character probability distributions are not really independent, (for example, in English, a 'Q' is almost always followed by a 'U' or a 'u') it's a usable approximation that the counts for text samples will follow the binomial distribution. So the expected number of character 's' with probability p_s in a text with n characters will be

$$\mu_s = np_s \tag{1}$$

  the variance of the character count will be

$$\sigma_s^2 = np_s(1 - p_s) \tag{2}$$

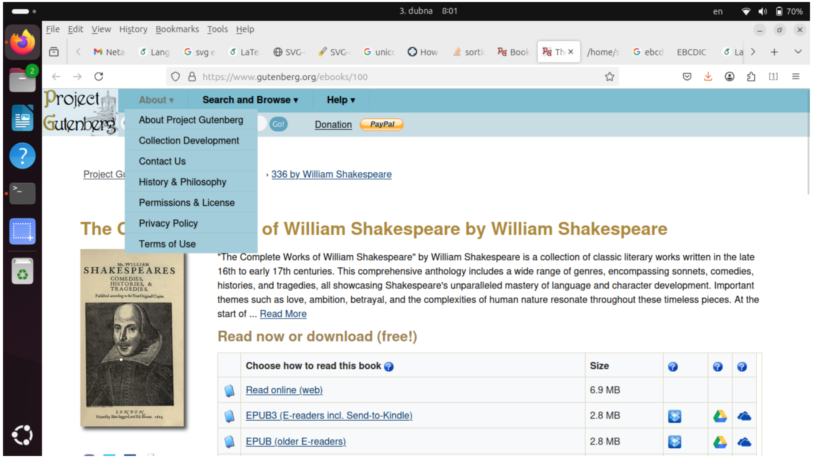
- The binomial distribution approximates the Gaussian bell curve.

What can you do with a 1-character frequency model?

1. We can estimate how well text samples are represented by the model.
2. We can generate samples of text which are represented by the model by using a random number generator.
3. If we have several different models, we can see which is the best fit for a particular sample.

Estimating model fit

- First we need a model. I built one based on the *Complete works of Shakespeare*, which I downloaded from Project Gutenberg.



The C... of William Shakespeare by William Shakespeare

"The Complete Works of William Shakespeare" by William Shakespeare is a collection of classic literary works written in the late 16th to early 17th centuries. This comprehensive anthology includes a wide range of genres, encompassing sonnets, comedies, histories, and tragedies, all showcasing Shakespeare's unparalleled mastery of language and character development. Important themes such as love, ambition, betrayal, and the complexities of human nature resonate throughout these timeless pieces. At the start of ... Read More

Read now or download (free!)

Choose how to read this book	Size			
Read online (web)	6.9 MB			
EPUB3 (E-readers incl. Send-to-Kindle)	2.8 MB			
EPUB (older E-readers)	2.8 MB			

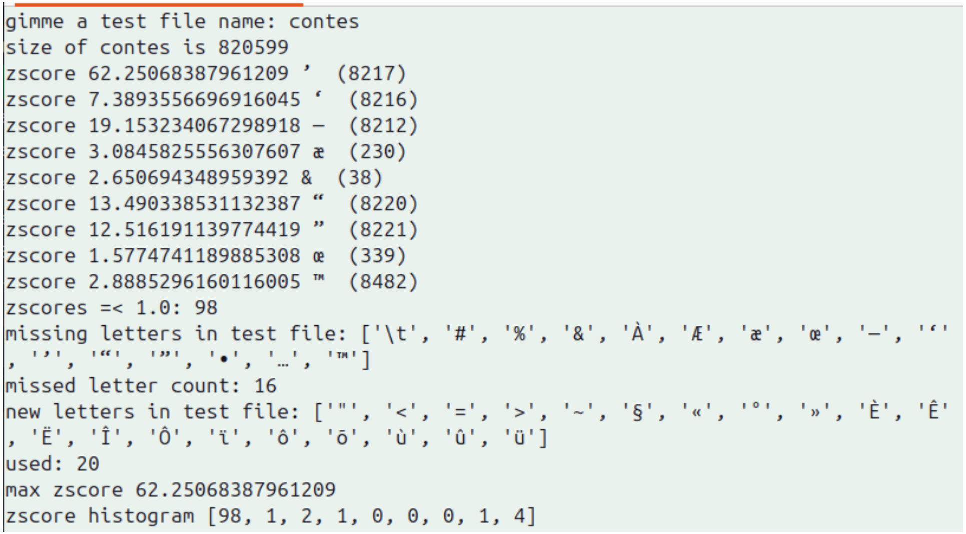
1-character language model

'T'	0.007118869503443513
'h'	0.044708508415567214
'e'	0.08434568218904959
' '	0.1547401277975586
.	
.	
.	

number of characters 5378663

this model has 107 keys, including some funny characters which are part of the Gutenberg preamble. The preamble and postscript are in English, so I left them in, but it was probably a mistake...

language model run



```
gimme a test file name: contes
size of contes is 820599
zscore 62.25068387961209 ’  (8217)
zscore 7.3893556696916045 ‘  (8216)
zscore 19.153234067298918 –  (8212)
zscore 3.0845825556307607 æ  (230)
zscore 2.650694348959392 &  (38)
zscore 13.490338531132387 “  (8220)
zscore 12.516191139774419 ”  (8221)
zscore 1.5774741189885308 œ  (339)
zscore 2.8885296160116005 ™  (8482)
zscores =< 1.0: 98
missing letters in test file: ['\t', '#', '%', '&', 'À', 'Æ', 'æ', 'œ', '–', '‘'
, '’', '“', '”', '•', '…', '™']
missed letter count: 16
new letters in test file: ['"', '<', '=', '>', '~', '§', '«', '°', '»', 'È', 'Ê'
, 'Ë', 'Î', 'Ô', 'ï', 'ô', 'ō', 'ù', 'û', 'ü']
used: 20
max zscore 62.25068387961209
zscore histogram [98, 1, 2, 1, 0, 0, 0, 1, 4]
```

Some comments

- zscores are the number of standard deviations that measurements are from the expected value. (Here mine are absolute values, so they could be too high, or too low.) A zscore of 1 might happen 32% of trials. A zscore of 2 might happen 5% of the time.

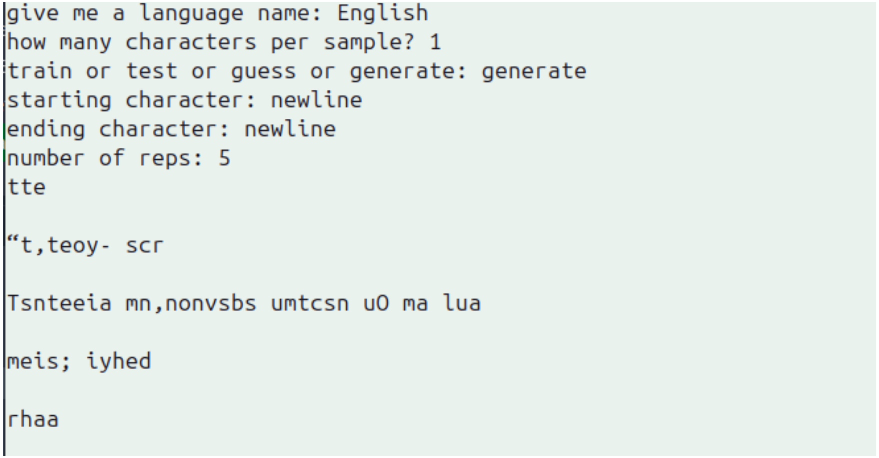
zscore	**How likely?**
1	$1/3$
2	$1/20$
3	$3/1000$
4	$1/16000$
62	$2/10^{837}$

- My code doesn't estimate the zscore for letters which don't occur in the original model.
- It doesn't combine the zscores to estimate the total likelihood of the test document matching the training corpus.

But we can see a route to those elaborations.

Generating some text from a model

For this program I used a novel *A tale of two cities* by Charles Dickens. (Less white-space than Shakespeare.)



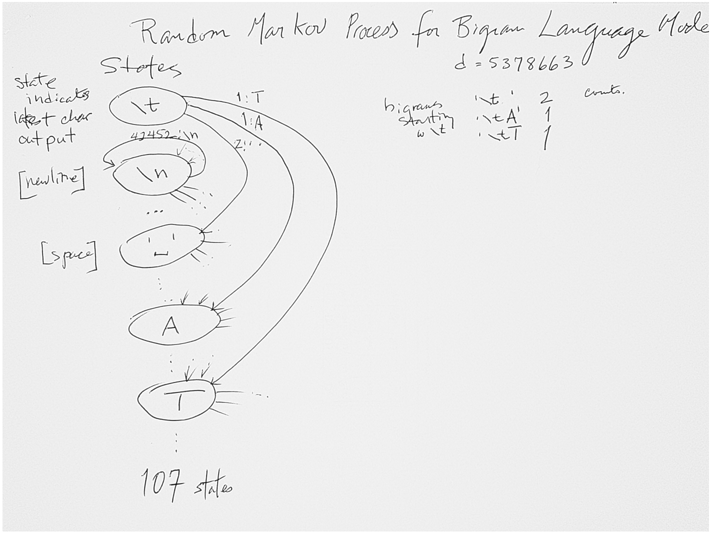
```
give me a language name: English
how many characters per sample? 1
train or test or guess or generate: generate
starting character: newline
ending character: newline
number of reps: 5
tte

“t,teoy- scr

Tsnteeia mn,nonvsbs umtcsn u0 ma lua

meis; iyhed

rhaa
```

- This model can only generate what it knows about – and all it knows is character frequency.
- So the words are about the right length, because it knows how often it should output a space.
- And there are the right number of vowels, they just aren't properly distributed.
- Punctuation is randomly scattered among the letters.
- So this is not a great model for generating English text. We need something more exciting!

Character n-gram models

- What we want is a model that pays attention to what it has already typed.
- We could imagine a model that looked all the way back to the beginning of its output, but what I'm going to talk about is models which keep track of the frequency of n-grams.
- A character **n-gram** is a series of *n* characters. (We'll talk about word n-grams in a few minutes.)
- A corpus with 100000 characters has 99999 2-grams in it ... they overlap. And of course they repeat. So we can build a model of bigram frequencies.
- a 2-gram (or bigram) character model has quite a bit better idea of language structure than a 1-gram model, which is what we have already seen.

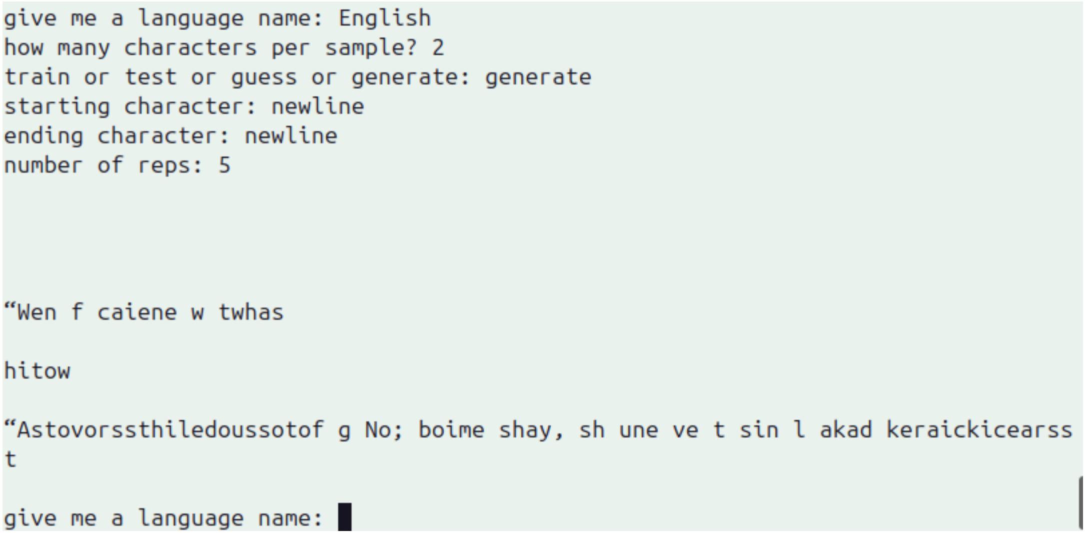
A character bigram model



What about those little probabilities?

In the previous slide, there were some small probabilities. (once or twice in the corpus)
Should we trust those numbers?
Well, if they are not bogus, they are also not accurate:
Suppose that we had a bigram which really had a probability of occurring once in 5378663 trials, that is, once in my Shakespeare corpus. What would we expect to observe?
For starters, we might not see any occurrence at all. For the binomial distribution with probability p and n trials, the probability of k events is the term in the binomial expansion: $\binom{n}{k} p^k (1-p)^{n-k}$
For $p = 1/n$ and $k = 0$, we get: $(1 - \frac{1}{n})^n$, which happens to be a well-known limit: $\lim_{n\to\infty}(1 + \frac{a}{n})^n = e^a$
So it turns out that the probability of seeing no instances of that item with probability 1/n is: $e^{-1} \approx 0.3679$; of 1 instance: $n\frac{1}{n}(1 - frac1n)^n/(1 - 1/n) \approx 0.3679$; of 2 instances: $\frac{n(n-1)}{2n}(1 - \frac{1}{n})^{n-2} \approx \frac{e^{-1}}{2}$, etc.
If the bigram probability were $p = \frac{1}{2n}$ the chance of seeing a single occurrence would be $\frac{1}{2}e^{-\frac{1}{2}} \approx 0.3033$; of 2, 0.1516, etc.

Generating English with bigram character model



```
give me a language name: English
how many characters per sample? 2
train or test or guess or generate: generate
starting character: newline
ending character: newline
number of reps: 5





“Wen f caiene w twhas

hitow

“Astovorssthiledoussotof g No; boime shay, sh une ve t sin l akad keraickicearss
t

give me a language name: 
```

character bigram commentary

- First off, we can see that the five lines I asked for included two blank ones; `'\n\n'` turns out to be one of the most common bigrams.
- Similarly, two of our three non-blank lines begin with a quotation mark. (The training corpus has a lot of dialog.)
- The punctuation looks pretty good – quotation mark is followed with an upper-case letter,
- Comma and semicolon are followed by a space.
- the syllables of the words sort of make sense; the vowels are well-placed, and there are only a couple of run-on consonant sequences, like **rssthh**.
- It looks like we might complain about comparing files that have the right proportions of letters, but different structure.
- But overall, we need a little more context to do nice generation.

Character n-grams

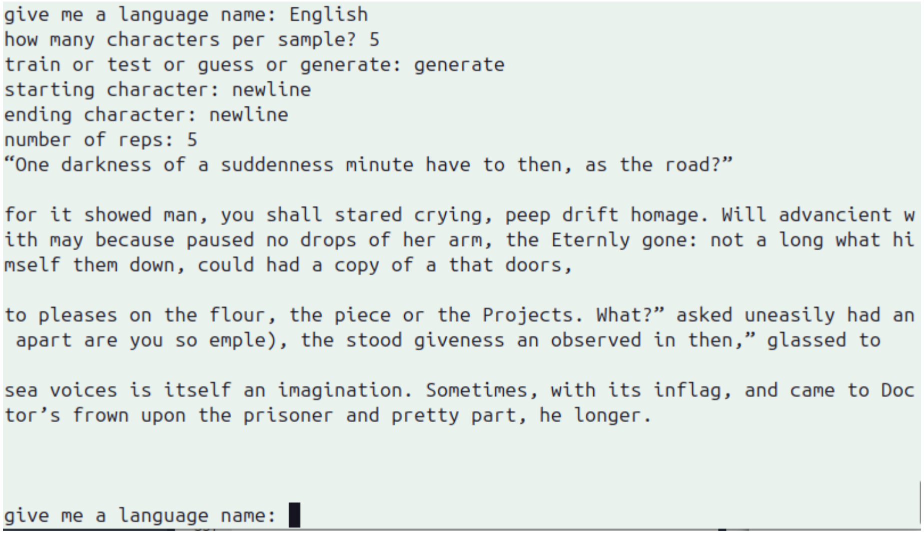
we could get that context with character n-grams.

- The algorithms are pretty much like the bigram ones; the resulting model is bigger, but the cost of finding the next character is about the same as in the bigram case.
- Of course there are more n-grams, and we have to store them all, but they are a small fraction of the total possibilities.
- One issue we have to think (more) about is backoff. What do you do if there isn't any (n-1)-gram matching prefix in your table? (This always comes up at the beginning of generation, because you don't have n-1 characters of context yet.)

Growth of n-gram model files

```
-rw-rw-r-- 1 staylor staylor    1221 dub 13  2024 English.1
-rw-rw-r-- 1 staylor staylor   24611 dub 13  2024 English.2
-rw-rw-r-- 1 staylor staylor  161331 dub 13  2024 English.3
-rw-rw-r-- 1 staylor staylor  617646 dub 13  2024 English.4
-rw-rw-r-- 1 staylor staylor 1618797 dub  2 22:19 English.5
-rw-rw-r-- 1 staylor staylor 3183189 dub 13  2024 English.6
-rw-rw-r-- 1 staylor staylor 5071400 dub 13  2024 English.7
-rw-rw-r-- 1 staylor staylor 7068310 dub 13  2024 English.8
-rw-rw-r-- 1 staylor staylor 8977779 dub 13  2024 English.9
-rw-rw-r-- 1 staylor staylor 10658908 dub 13  2024 English.10
-rw-rw-r-- 1 staylor staylor 12068210 dub 13  2024 English.11
-rw-rw-r-- 1 staylor staylor 13195099 dub 13  2024 English.12
```

5-gram character model generation

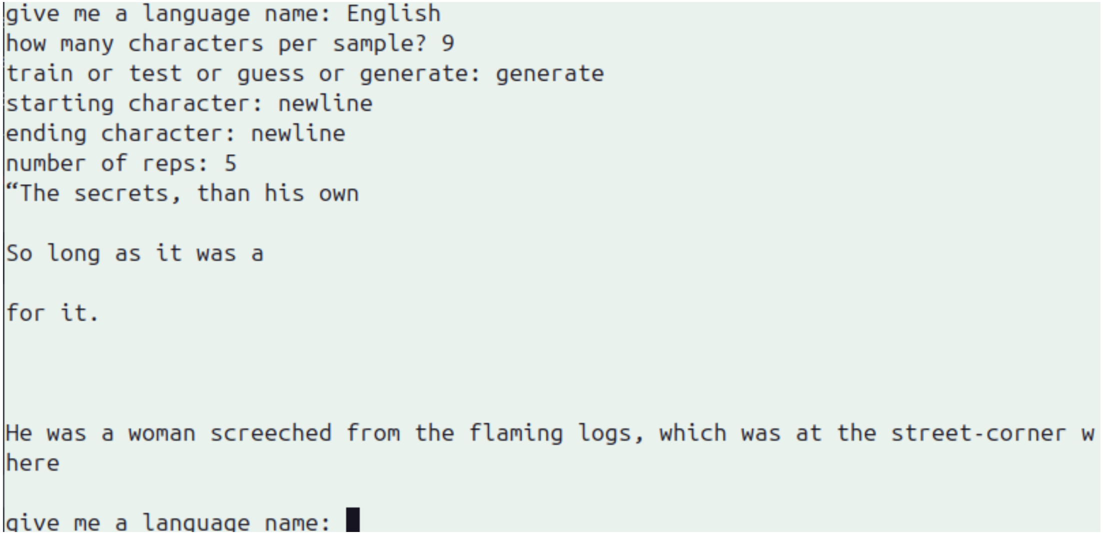


```
give me a language name: English
how many characters per sample? 5
train or test or guess or generate: generate
starting character: newline
ending character: newline
number of reps: 5
“One darkness of a suddenness minute have to then, as the road?”

for it showed man, you shall stared crying, peep drift homage. Will advancient w
ith may because paused no drops of her arm, the Eternly gone: not a long what hi
mself them down, could had a copy of a that doors,

to pleases on the flour, the piece or the Projects. What?” asked uneasily had an
 apart are you so emple), the stood giveness an observed in then,” glassed to

sea voices is itself an imagination. Sometimes, with its inflag, and came to Doc
tor’s frown upon the prisoner and pretty part, he longer.



give me a language name: 
```

Commentary

- Wow! It's nonsense, but it's words!
- Some words longer than 5 letters are spelled correctly, but many are obviously 'hallucinated'.
- Five characters seems to be enough that successive words often go together.
- We can strengthen the connection between words with a little more context.

A 9-gram character

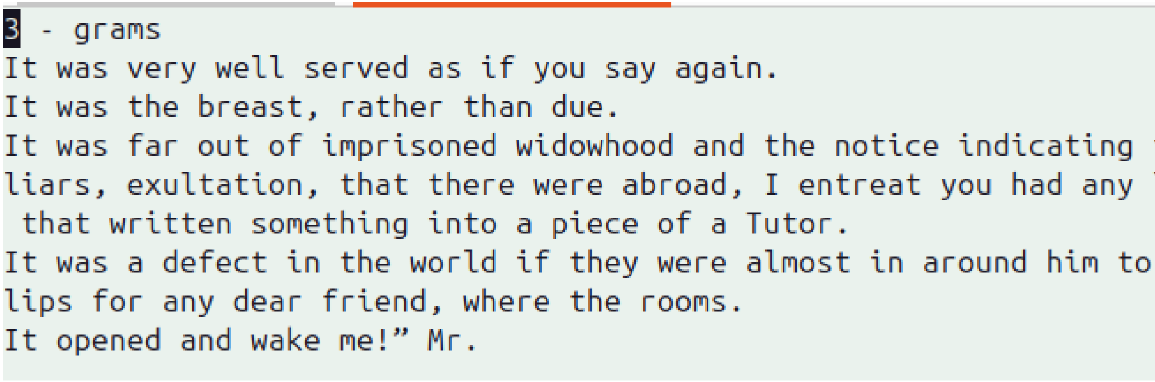


```
give me a language name: English
how many characters per sample? 9
train or test or guess or generate: generate
starting character: newline
ending character: newline
number of reps: 5
“The secrets, than his own

So long as it was a

for it.



He was a woman screeched from the flaming logs, which was at the street-corner w
here

give me a language name: 
```

n-gram word models

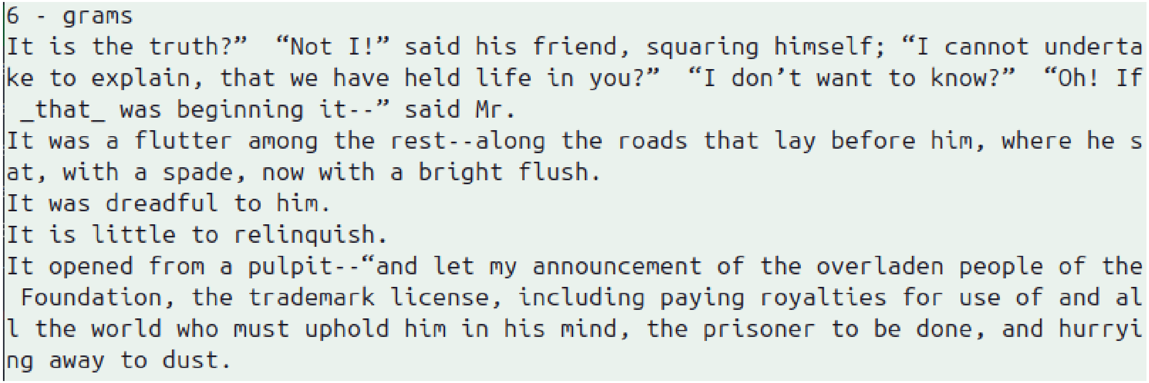
1. Word models are a little more inconvenient to store.
2. It isn't obvious whether we should include punctuation as complete words.
3. If we do imagine punctuation as complete words, we have to build spacing models for punctuation into both our training scan and our generation code.
4. Although the word sentences get a little more coherent as we add context, the improvement feels more gradual than with character models.
5. As the context gets longer, we are more nearly quoting the training data.

word 3-grams



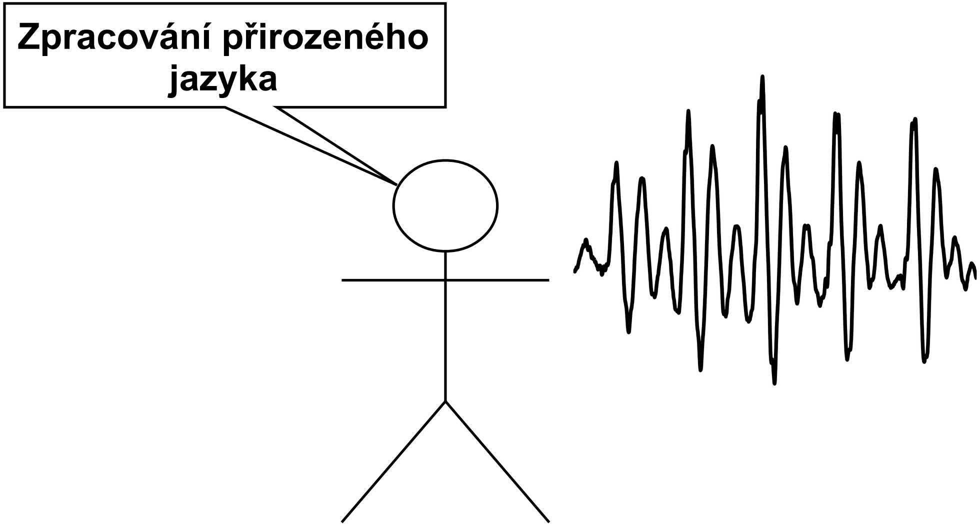
```
3 - grams
It was very well served as if you say again.
It was the breast, rather than due.
It was far out of imprisoned widowhood and the notice indicating
liars, exultation, that there were abroad, I entreat you had any
 that written something into a piece of a Tutor.
It was a defect in the world if they were almost in around him to
lips for any dear friend, where the rooms.
It opened and wake me!” Mr.
```

word 6-gram Model generation



```
6 - grams
It is the truth?”  “Not I!” said his friend, squaring himself; “I cannot underta
ke to explain, that we have held life in you?”  “I don’t want to know?”  “Oh! If
 _that_ was beginning it--” said Mr.
It was a flutter among the rest--along the roads that lay before him, where he s
at, with a spade, now with a bright flush.
It was dreadful to him.
It is little to relinquish.
It opened from a pulpit--“and let my announcement of the overladen people of the
 Foundation, the trademark license, including paying royalties for use of and al
l the world who must uphold him in his mind, the prisoner to be done, and hurryi
ng away to dust.
```

Speech Recognition



Speech Recognition

- I - possible utterance (the speech 'input' is the output of ASR)
- O - acoustic signal – the sounds our program detects
- $P(I)$ - estimated by the language model
- $P(O|I)$ - estimated by the acoustic model
- $P(O)$ - It happened so it must be possible. But we don't need P(O) to find I.

Calculation to find best I:

$$I^* = \arg\max_I P(I|O) = \arg\max_I \frac{P(O|I)P(I)}{P(O)} = \arg\max_I P(O|I)P(I)$$

Context-free Grammars

You'll probably recognize these as common descriptors for programming languages. But [Chomsky, 1956] developed them to describe natural languages. They need crutches to handle Czech, but for less inflected languages like English and Chinese, they can be useful.

```
    <sentence> := <declaration> | <question> | <imperative>
    <declaration := <noun-phrase> <verb-phrase>
    <noun-phrase> := <pronoun> | <determiner>? <adjective>* <noun> | <prope
    <proper-noun> := <human name> | <organization name> | <place name>
```

I didn't complete or debug these rules; it's a frustrating task. But you can use this kind of formalism for an outline of a text generation algorithm for English.

Word Embedding

- Word embeddings are a scheme for representing the meaning of words as vectors of floating point numbers.
- They were developed as a technique to simplify word inputs to neural networks.
- One scheme in use before word-embeddings was called '1-of-n', in which only one of *n* inputs would be turned on to indicate that a particular word was being used as input. This worked, but it made neural networks which dealt with words very wide.
- At the time, there were already descriptions of words as vectors, a scheme called Latent Semantic Analysis, which was developed for information retrieval.
- It turns out that these two origins produce *Semantic Spaces* with very similar properties.
- Transformers and LLMs use a sort of syllable/word-chunk embedding, so that they have complete coverage of anything typed as input. Each word typed may be several word-chunks, but they are already dealing with sequences, so the sequence just gets longer.

Word Embeddings

- [Deerwester et al., 1990] introduced LSA for Information Retrieval.
- Over the following decade, NLP folks noticed new uses for the vectors, including that words with similar vectors often have related meanings.
- in 2000 [Bengio et al., 2000] introduced a new technique for training vectors from corpora which was less computationally expensive – he produced a 30,000 word embedding with 50-dimensional vectors in only a few weeks of computer time.
- [Collobert and Weston, 2008] published a paper in 2008 in which he made state-of-the-art claims for several tasks using word-embeddings.
- In 2013, [Mikolov et al., 2013] et al, then working at Google, published a paper and released code for an algorithm which could build a word embedding from a billion word corpus overnight.
- Mikolov's paper included the observation that the semantics seems to be additive: The embedding for Volga is the sum of the embeddings for Russian and River.

Word Embedding

- Today you can download embeddings for more than a hundred languages from several different sites. I often use `vectors.nlpl.eu/repository/`
- In python, you can load an embedding into your program with `mx=gensim.models.KeyedVectors.load_Word2Vec_format(filename)`
- Then you can retrieve a vector with `V=mx['mother']`
- Mikolov published a list of *word analogies*, that could be approximately solved with vector arithmetic. The most famous is
  $\text{mx['man']} - \text{mx['woman']} = \text{mx['king']} - \text{mx['queen']}$
  or: man : woman :: king : queen
  or: man : woman :: king : ?

Vector arithmetic

- Most freestanding word embeddings today have a 100-300 dimensional vector for each word.
- You can think of the m word vectors of n elements each as a $m \times n$ matrix, and in fact we sometimes do matrix arithmetic on the whole embedding.
- Consider vectors $A = (a_1, a_2, a_3, \ldots)$ and $B = (b_1, b_2, b_3, \ldots)$
  vector addition $X = A + B$ is defined as: $\forall x_i : x_i = a_i + b_i$
  vector subtraction $X = A - B$ as: $\forall x_i : x_i = a_i - b_i$
  scalar multiplication $X = cA$ as: $\forall x_i : x_i = ca_i$
  scalar division $X = A/c$ as: $\forall x_i : x_i = a_i/c$
  dot product $x = A \cdot B$ as: $x = \sum_1^n a_i b_i$
  (similar arithmetic: $x = A \times B^T$ since A and B are both $1 \times n$ matrices, but this invites the quibble, would x be a scalar, or a 1×1 matrix.)
  matrix multiplication $X = A \times F$ where F is a $n \times k$ matrix, produces $1 \times k$ matrix, that is a vector, as a result. Frequently used when $k = n$ to rotate or scale A; when $k < n$ to **project** A, that is, provide a version with fewer elements, perhaps to graph on paper.

Word Embedding Normalization

- Some embeddings you download are already *normalized*, and others are not. There are several kinds of normalization:

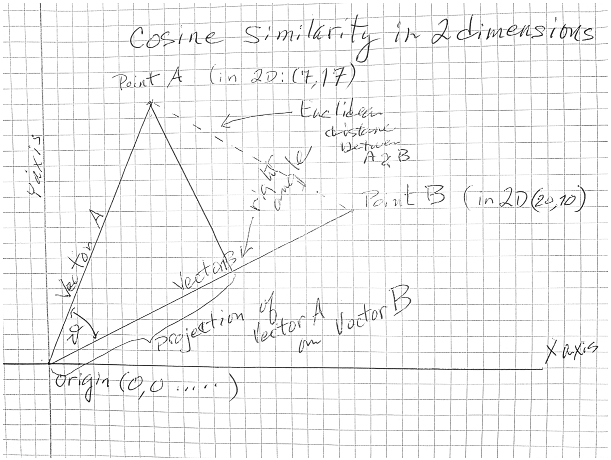
Unit normalization	This is dividing each embedding vector by its own norm, which is defined as $\text{norm}(X) = \lvert X \rvert = \sqrt{X \cdot X}$
Zero centering	This is adding the mean of all the vectors to each vector, so that the average of the embedding is zero
Gaussian normalization	(Which I've not done) Adjusts the size of the elements so that each column of the embedding matrix has a Gaussian distribution.

- normalization seems to compensate for word frequency, and gives cosine distance and Euclidean distance clearer relationship to each other.

distance in n-space

- There are two common measures for comparing word vectors: Euclidean distance, and cosine similarity.
- Euclidean distance between vectors A and B is $\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$, which looks about like the 2-dimensional distance we learned in Geometry class. Equivalently, you can write it as $|A - B|$
- Cosine similarity is defined to be the cosine of the angle between two vectors, or equivalently, the projection of either vector on the other one, divided by each of their norms, and written $\frac{A \cdot B}{|A||B|}$
- Because cosine similarity has a maximum value of 1 when two vectors are identical, and gets smaller as they diverge, we sometimes define cosine distance as $d(A, B) = 1 - \frac{A \cdot B}{|A||B|}$

distances diagram

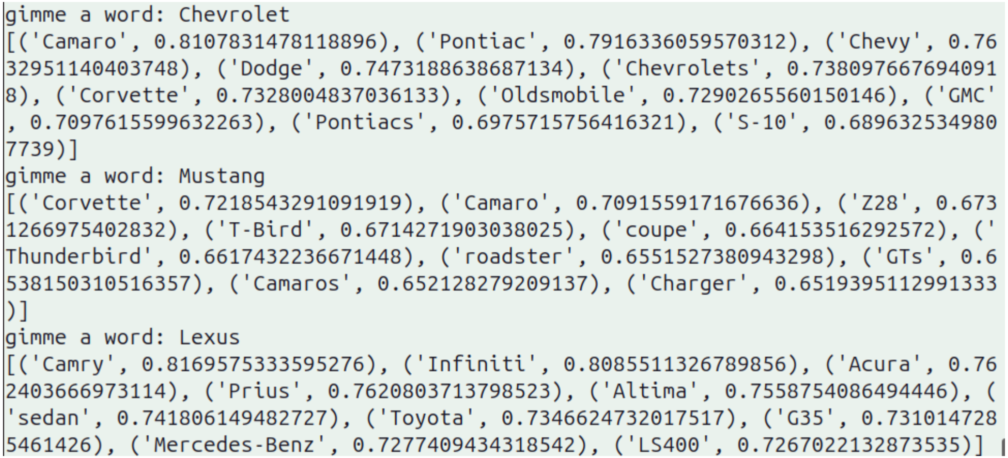


Word Embedding Applications

- Because the process of training embeddings is stochastic (it involves lots of randomness) no two embeddings are the same.
- But, often two embeddings can be transformed into a common space with matrix multiplication.
- This involves selecting a number of anchor pairs S',T' from embeddings S and T that you want to end up identical, and
- Then training or computing a d $d \times d$ matrix M such that $S' \times M = T'$
- Applying M to S, we have $S \times M = U$, where U is aligned with T so that not only do the S' embeddings have values similar the their counterparts in T', but so do most other vectors in S.
  1. If S and T are embeddings for different languages, U can enable finding translations in T for words in S, and vice-versa. (One problem with this approach is that many common words have several different meanings. The sets of meaning in one language overlap with the set in the other for the usual translation, but are seldom identical.)
  2. if S and T are based on corpora in the same language, but from different historical periods, we can surmise that words in U which do not match the same words in T have different meanings.

Word Embedding Applications

Finding words with similar usages:



```
gimme a word: Chevrolet
[('Camaro', 0.8107831478118896), ('Pontiac', 0.7916336059570312), ('Chevy', 0.76
32951140403748), ('Dodge', 0.7473188638687134), ('Chevrolets', 0.73809766769409l
8), ('Corvette', 0.7328004837036133), ('Oldsmobile', 0.7290265560150146), ('GMC'
, 0.7097615599632263), ('Pontiacs', 0.6975715756416321), ('S-10', 0.68963253498O
7739)]
gimme a word: Mustang
[('Corvette', 0.7218543291091919), ('Camaro', 0.7091559171676636), ('Z28', 0.673
1266975402832), ('T-Bird', 0.6714271903038025), ('coupe', 0.664153516292572), ('
Thunderbird', 0.6617432236671448), ('roadster', 0.6551527380943298), ('GTs', 0.6
538150310516357), ('Camaros', 0.652128279209137), ('Charger', 0.6519395112991333
)]
gimme a word: Lexus
[('Camry', 0.8169575333595276), ('Infiniti', 0.8085511326789856), ('Acura', 0.76
240366973114), ('Prius', 0.7620803713798523), ('Altima', 0.7558754086494446), (
'sedan', 0.741806149482727), ('Toyota', 0.7346624732017517), ('G35', 0.731014728
5461426), ('Mercedes-Benz', 0.7277409434318542), ('LS400', 0.7267022132873535)]
```

References

Bengio, Y., Ducharme, R., and Vincent, P. (2000).
A neural probabilistic language model.
In *Advances in Neural Information Processing Systems 13 (NIPS 2000)*.

Chomsky, N. (1956).
Three models for the description of language.
IRE Transactions on information theory.

Collobert, R. and Weston, J. (2008).
A unified architecture for natural language processing: Deepneural networks with multitask learning.
In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990).
Indexing by latent semantic analysis.
Journal of the American Society for Information Science.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013).
Efficient estimation of word representations in vector space.
In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

The End