EC999: N-Grams

Thiemo Fetzer

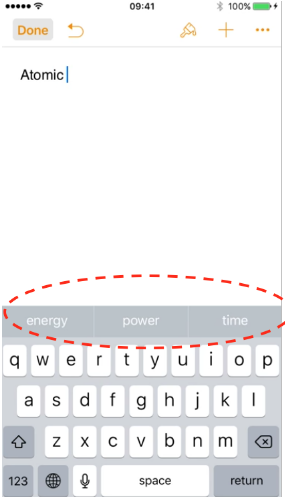
University of Chicago & University of Warwick

January 16, 2017

Plan

N-Gram Language Models

Autocomplete Function



Behind the scense works an n-gram language model.

Autocomplete Function

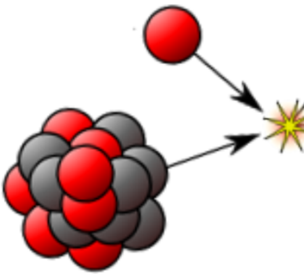
Atomic Energy will have been made availabl
source

Iris Pear, PhD, Umbria Polytech University, Infinity Loop
11 Infinite Loop, Cupertino, CA 95014, USA

Abstract

Atomic Physics and I shall not have the same problem with a separate section for a very long long way. Nuclear weapons will not have to come out the same day after a long time of the year he added the two sides will have the two leaders to take the same way to bring up to their long ways of the same as they will have been a good place for a good time at home the united front and she is a great place for a good time. The atoms of a better universe will have the right for the same as you are the way we shall have to be a great place for a great time to enjoy the day you are a wonderful person to your great time to take the fun and take a great time and enjoy the great day you will be a wonderful time for your parents and kids. Molecular diagnostics will have been available for the rest by a single day and a good day to the rest have a wonderful time and aggravation for the rest day at home time for the two of us will have a great place for the rest to be great for you tomorrow and tomorrow after all and I am a very happy boy to the great day and I hope he is wonderful.

Image



Uses of Probabilistic Language Models

- Spelling correction
- Auto complete
- Language detection (classification)
- Other classification tasks

Probabilistic Language Models

We begin by introducing the idea of a language model. In this course, we will work with two dominant language models

1. Probabilistic N-Gram language model
2. Bag of Words model

We start with a simple N-Gram language model and then look at statistical methods to detect collocations and present an application from research.

Probabilistic Language Models

What is the likely next word?

```
Make America ...
```

N-gram language models see sentences as sequences of words, the occurrence of each word is a function of the likelihood of the sequence of words.

```
Make America Great Again
```

The predicted next word naturally depends on the corpus on which a language model was trained on and a range of other factors. Lets formalize things a bit.

Probabilistic Language Models

What is the probability of:

$$P(\text{again}|\text{Make America great})$$

We can express this probability as:

$$P(\text{again}|\text{Make America great}) = \frac{P(\text{Make America great again})}{P(\text{Make America great})}$$

which we may be inclined to estimate as

$$\hat{P}(\text{again}|\text{Make America great}) = \frac{C(\text{Make America great again})}{C(\text{Make America great})}$$

where the $C(.)$ indicates the raw counts of the text fragments.

Make America ...

```r
library(ngram)
TRUMP <- readLines(con = "../../Data/Trump-Speeches.txt")
# concatenate into one massive string, remove empty lines
TRUMP <- TRUMP[-grep("^SPEECH|^$", TRUMP)]
TRUMP <- gsub(" +", " ", TRUMP)
TRUMP <- paste(TRUMP, collapse = " ")
TRUMP <- preprocess(TRUMP, remove.punct = TRUE)

p1 <- length(strsplit(TRUMP, "make america great again")[[1]])
p1
```

```
## [1] 45
```

```r
p2 <- length(strsplit(TRUMP, "make america great")[[1]])
p2
```

```
## [1] 48
```

```r
p1/p2
```

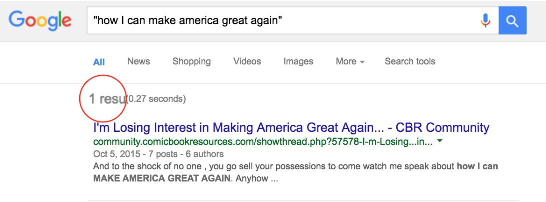
```
## [1] 0.938
```

So here estimating the conditional probability is possible as the sentence is rather short. However, for longer sentences this becomes much more difficult.

Longer Sentences

For longer sentences, it becomes much less likely that we will observe sufficient number of raw counts.

```
how I can make America Great again
```



Meaning the estimate

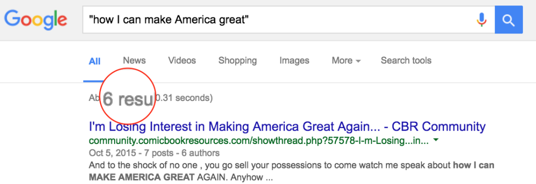
$$\hat{P}(\text{again}|\text{how I can make America great}) = \frac{C(\text{how I can make America great again})}{C(\text{how I can make America great})}$$

is very imprecise. For longer sentence, the counts in numerator and denominator would be exactly zero.

Longer Sentences

For longer sentences, it becomes much less likely that we will observe sufficient number of raw counts.

`how I can make America Great again`



Meaning the estimate

$$\hat{P}(\text{again}|\text{how I can make America great}) = \frac{C(\text{how I can make America great again})}{C(\text{how I can make America great})}$$

is very imprecise. For longer sentence, the counts in numerator and denominator would be exactly zero.

A bit of notation...

What is the joint probability of observing a sequence of words $w_1, ..., w_n$?

$$\begin{aligned}
P(w_1, .., w_n) &= P(w_1)P(w_2, ..., w_n|w_1) \\
&= P(w_1)P(w_2|w_1)P(w_3, ..., w_n|(w_1, w_2)) \\
&= P(w_1)P(w_2|w_1)P(w_3|(w_1, w_2))P(w_4, ..., w_n|(w_1, w_2, w_3)) \\
\dots \\
&= \prod_{k=1}^{n} P(w_k|(w_1, ..., w_{k-1})
\end{aligned}$$

iteratively applying the **Chain Rule of Probability**.

Curse of Dimensionality

We can compute probability of a sentence

$$P(w_1, .., w_n) \quad = \quad \prod_{k=1}^{n} P(w_k | (w_1, ..., w_{k-1})$$

by multiplying a sequence of conditional probabilities.

- We can not estimate each individual conditional probability because its highly unlikely that a stable estimate does exist.
- Similarly, it would be computationally infeasible.
- Parameter space (number of conditional probabilities that need to be estimated) grows exponentially in n.

N-gram language model

Given the computational issues, the `n-gram` statistical language model assumes that we can approximate the probability of a word w given a history h by looking back just at the last N words in the history.
The simplest case is "not to look back", i.e. approximate the probability of a word w with an empty history, $h = \emptyset$. This yields the **unigram** language model.
So we would approximate $P(w_k|(w_1, ..., w_{k-1})) \approx P(w_k)$, which yields:

$$P(w_1, .., w_n) = \prod_{k=1}^{n} P(w_k)$$

This assumes that a sequence of words can be best approximated by the unconditional probabilities of an individual word appearing or not appearing.
The factorization implies that we assume that words are *stochastically independently drawn* from one another.

An unigram babbler

Some random sequences created with a unigram Trump babbler

```
## [1] "prosperity not talking about the game and more you mean but ill tell you k
## [2] "o terminate obamacare with a lot its deadly totally destabilized the worst
## [3] "ated me so im "
## [1] "statements theyve never ever see what that stupid your life obviously you 
## [2] "aid i talked about results yesterday i wrote the way i order televisions s
## [3] "somebody we order "
## [1] "jeep holding a lot of affection and the hell wants i dont think its really
## [2] "ing to even talk about the way if i dont have said recently he sent out th
```

Unigrams create poor results, because natural linguistic dependencies encoded in word collocations (e.g. verb follows an object, prepositions preceede locations, etc.) are ignored.
Adding more **context**, by looking and history of preceeding words.

N-gram models

A bigram model defined as

$$P(w_1, ..., w_n) = \prod_{k=1}^{n} P(w_k | w_{k-1})$$

This assumption that the probability of a word depends only on the previous word is called the **Markov** assumption. Here we approximate $P(w_k | (w_1, ..., w_{k-1})) \approx P(w_k | w_{k-1})$
Markov models are a class of probabilistic models that assume that the future can be predicted without looking *too far* into the past.
We can generalize to N-gram models

$$P(w_1, ..., w_n) = \prod_{k=1}^{n} P(w_k | w_{k-1}, ..., w_{k-N+1})$$

where $P(w_k | (w_1, ..., w_{k-1})) \approx P(w_k | w_{k-1}, w_{k-2}, ...w_{k-N+1})$

A Mini Example

Suppose you have a small corpus

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

the $<s>$ indicate start and end tags, they are to be treated just as words. It is important to include them to ensure that the probability estimates that we are extracting are well behaved.

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

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

Bigram illustration

`I want to make America great again`

$$P(\text{I}|{<}\text{start}>)P(\text{want}|\text{I})P(\text{to}|\text{want})P(\text{make}|\text{to})P(\text{America}|\text{make})$$
$$P(\text{great}|\text{America})P(\text{again}|\text{great})P({<}\text{end}>|\text{again})$$

Assume $P(\text{I}|{<}\text{start}>) = 25\%$ and $P({<}\text{end}>|\text{again}) = 25\%$.

Bigram illustration

```
##       ngrams estprop          ngrams  estprop
##  1:   to be   0.1108  make america  0.186782
##  2: want to   0.6382        make a  0.146552
##  3:  i dont   0.0797      make our  0.140805
##  4:   to do   0.0696       make it  0.137931
##  5: i think   0.0723 america great  0.269663
##  6:  i mean   0.0614 america first  0.117978
##  7:  i said   0.0611    america is  0.044944
##  8:  i have   0.0609 america shower 0.005618
##  9: to have   0.0403   great again  0.107715
## 10:  to get   0.0386  great people  0.045124
## 11:   i was   0.0420    great with  0.034934
## 12:  i want   0.0404     great and  0.032023
## 13: to make   0.0299      a shower  0.000279
## 14:  to the   0.0271  shower again  1.000000
```

$$P(\text{I}|<\text{start}>)P(\text{want}|\text{I})P(\text{to}|\text{want})P(\text{make}|\text{to})P(\text{America}|\text{make})$$
$$P(\text{great}|\text{America})P(\text{again}|\text{great})P(<\text{end}>|\text{again})$$
$$= 0.25x0.04x0.64x0.03x0.19x0.27x0.11x0.25 = 0.0000002708$$

$$P(\text{I}|<\text{start}>)P(\text{want}|\text{I})P(\text{to}|\text{want})P(\text{make}|\text{to})P(\text{America}|\text{make})$$
$$P(\textbf{shower}|\text{America})P(\text{again}|\textbf{shower})P(<\text{end}>|\text{again})$$
$$= 0.25x0.04x0.64x0.03x0.19x0.0056179775x1x0.25 = 0.00000005123$$

What do we learn?

- N-gram models capture syntatic features and general knowledge (here, knowledge about the underlying speeker)
- It turns out that linguistic features such as word sequences "I want" are reasonably frequent and they capture linguistic features: verbs tend to follow subject as indicated by "I".
- "america shower" is much more rare in Trump's speeches compared to "america great".
- N-gram models can be trained by counting and normalization

Illustration of estimating N-Gram probabilities.

We will use Maximum Likelihood estimation, illustrate and proof what is the maximum likelihood estimator using a unigram model

$$P(w_1, ..., w_n) = \prod_{k=1}^{n} P(w_k)$$

This can think of this as a sequence of independent *Bernoulli* trials with success probability p_k for word k.
Suppose you observe a sample of size N of word sequences of length n, $\{W^1, ..., W^N\}$. So each $W^i = (w_{i1}, ..., w_{in})$, where $w_{ij} = 1$ in case word w_j is present in sequence i.
What is the likelihood of observing a specific sequence?

$$P(W^i) = \prod_{k=1}^{n} p_k^{w_{ik}} (1 - p_k)^{1 - w_{ik}}$$

What is the likelihood of observing the whole sample?

$$\prod_{i=1}^{N} P(W^i) = \prod_{i=1}^{N} \prod_{k=1}^{n} p_k^{w_{ik}} (1 - p_k)^{1 - w_{ik}}$$

Log Likelihood

Taking logs

$$\sum_{i=1}^{N}\sum_{k=1}^{n} w_{ik} log(p_k) + (1 - w_{ik}) log(1 - p_k)$$

We want to find optimal $p_1, ..., p_n$, so take FOC. Notice that everything is additive and there are no interactions between individual p_k. Take FOC with respect to p_k.

$$\sum_{i=1}^{N} \frac{1}{p_k} w_{ik} - \sum_{i=1}^{N} (1 - w_{ik}) \frac{1}{1 - p_k} = 0$$

This yields

$$p_k = \frac{\sum_{i=1}^{N} w_{ik}}{N} \quad \forall k$$

where the numerator is just the number of word sequences that contain the word and N is just the sample size.

In case of Bigram model

For Bigram model, the intuitive way to estimate the probability $P(w_k|w_{k-1})$ is to get the counts of word sequences $C(w_{k-1}, w_k)$ from a corpus and normalize this by the counts of word that share the same first word, i.e. we estimate

$$P(w_k|w_{k-1}) = \frac{C(w_{k-1}, w_k)}{\sum_w C(w_{k-1}, w)} = \frac{C(w_{k-1}, w_k)}{C(w_{k-1})}$$

Note that $\sum_w C(w_{k-1}, w) = C(w_{k-1})$
I.e. the number of word pairs that share the starting word w_{k-1} should simply add up to the number of times the word w_{k-1} appears.

A Bigram Example

`I want to make America great again`

```
##             i want  to make america great again
## i           2  196   0    3       0     0     0
## want        5    0 485    0       0     0     0
## to          5    4   0  163       0     1     0
## make        0    0   0    0      65    17     0
## america     1    0   4    0       0    48     1
## great      16    0   7    0       0    14    74
## again      19    0   3    0       1     0     0
```

Normalization: divide each row's counts by appropriate unigram counts for w_{n-1}

```
##    [,1]   [,2]   [,3]   [,4]    [,5]   [,6]       [,7]
## X1 "to"   "want" "i"    "great" "make" "america"  "again"
## V1 "5460" " 760" "4857" " 687"  " 348" " 178"     " 246"
```

A Bigram Example

```
##                 i     want     to     make america    great   again
## i        0.000412 0.040354 0.0000 0.000618 0.00000 0.000000 0.00000
## want     0.006579 0.000000 0.6382 0.000000 0.00000 0.000000 0.00000
## to       0.000916 0.000733 0.0000 0.029853 0.00000 0.000183 0.00000
## make     0.000000 0.000000 0.0000 0.000000 0.18678 0.048851 0.00000
## america  0.005618 0.000000 0.0225 0.000000 0.00000 0.269663 0.00562
## great    0.023290 0.000000 0.0102 0.000000 0.00000 0.020378 0.10771
## again    0.077236 0.000000 0.0122 0.000000 0.00407 0.000000 0.00000
```

- Ratio of $\frac{C(w_n, w_{n-1})}{C(w_{n-1})}$ is a maximum Likelihood estimate for $P(w_n|w_{n-1})$
- We observe that many raw counts are zero: the matrix is sparse.
- The larger N, the more sparse will these matrices get.
- However, larger N generally results in better performance as more history is incorporated.

Trade-off: Higher order N-gram versus lower order N-grams

Unigram Trump Babbler

```
## [1] "prosperity not talking about the game and more you mean but ill tell you know shes going t"
## [2] "o terminate obamacare with a lot its deadly totally destabilized the worst human being tre"
## [3] "ated me so im not smart and tell you get on trade agreement you look we have to end "
```

Bigram Trump Babbler

```
## [1] "from pakistan and he was talking to some place it costs 3 billion and i said ok then i hav"
## [2] "e to happen with these two nations and must regard them with their families we mourn as on"
## [3] "e united people with force purpose and determination but the muslims living in this "
```

Trigram Trump Babbler

```
## [1] "permanently admits more than 100000 immigrants from the middle east our government has bee"
## [2] "n admitting ever growing numbers year after year without any effective plan for our own se"
## [3] "curity in fact clintons state department was in charge of admissions and the admissions pr"
## [4] "ocess for people applying to enter from overseas "
```

Quadrigram Trump Babbler

```
## [1] "tough as nails hes going to be your champion im going to be good for womens health issues "
## [2] "its very important to me it was instructional when i did the art of the deal pac after the"
## [3] " book they have all these pacs and the money comes in and its "
```

as N increases, the number of parameters to be estimated explodes. In addition, as we have seen, the matrices are very sparse - many zeroes!

Curse of Dimensionality of N-gram model

Suppose you have a vocabulary of size $|V|$. Assuming no constraints imposed by language structure. How many different conditional probabilites are there to estimate?

- There are $|V|$ sentences, containing exactly 1 words.
- There are $|V| \times |V|$ sentences containing 2 words
- There are $|V| \times |V| \times |V|$ sentences containing 3 words

In total there are $|V|^N$ parameters in an n-gram for vocabulary size $|V|$.

Uses of N-gram models for language categorization

https://www.let.rug.nl/~vannoord/TextCat/textcat.pdf

Babbling Donald Trump

```
library(ngram)
TRUMP <- readLines(con = "../../Data/Trump-Speeches.txt")
# concatenate into one massive string, remove empty lines
TRUMP <- TRUMP[-grep("^SPEECH|^$", TRUMP)]
TRUMP <- gsub(" +", " ", TRUMP)
TRUMP <- paste(TRUMP, collapse = " ")
TRUMP <- ngram(TRUMP, n = 3)
str_break(babble(TRUMP, genlen = 35, seed = 130))
```

```
## [1] "I have as big a heart as anybody. We want to win Iowa, folks. Because look, I love the poo"
## [2] "rly educated. Were the smartest people, were the most loyal people by far. Everybody say"
## [3] "s it. "
```