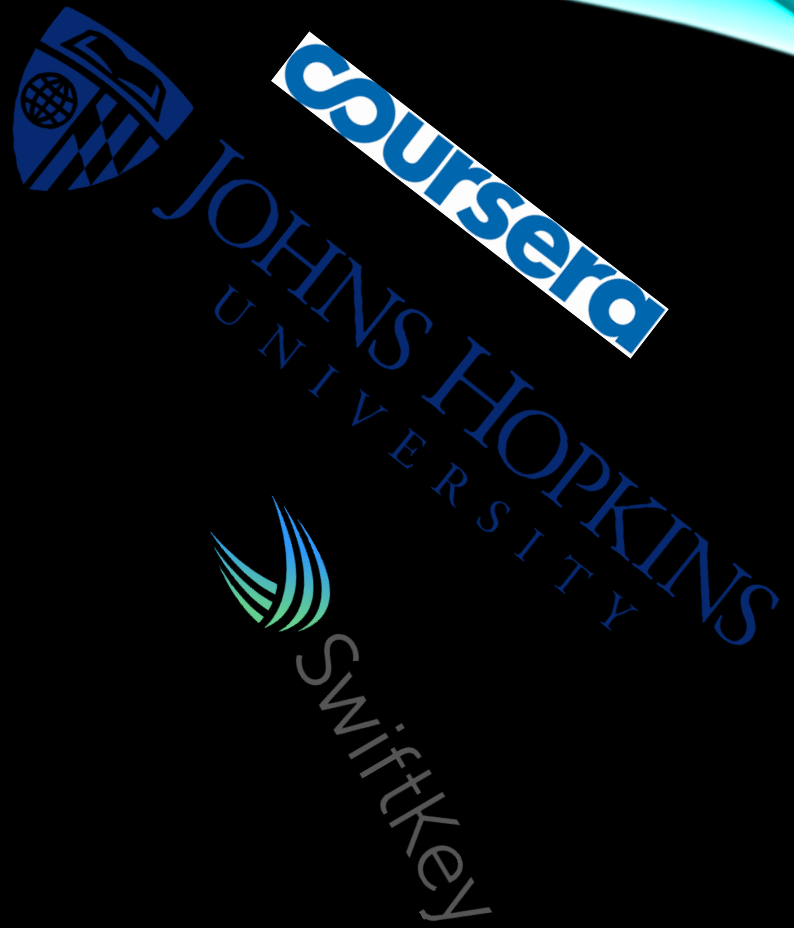
# LESSON LEARNED "DEVELOPING A **PREDICTIVE MODEL** FOR TEXT PREDICTION"

Paracchini Pier Lorenzo




"Around the world, people are spending an increasing amount of time on their mobile devices for email, social networking, banking and a whole range of other activities. But typing on mobile devices can be a serious pain. "

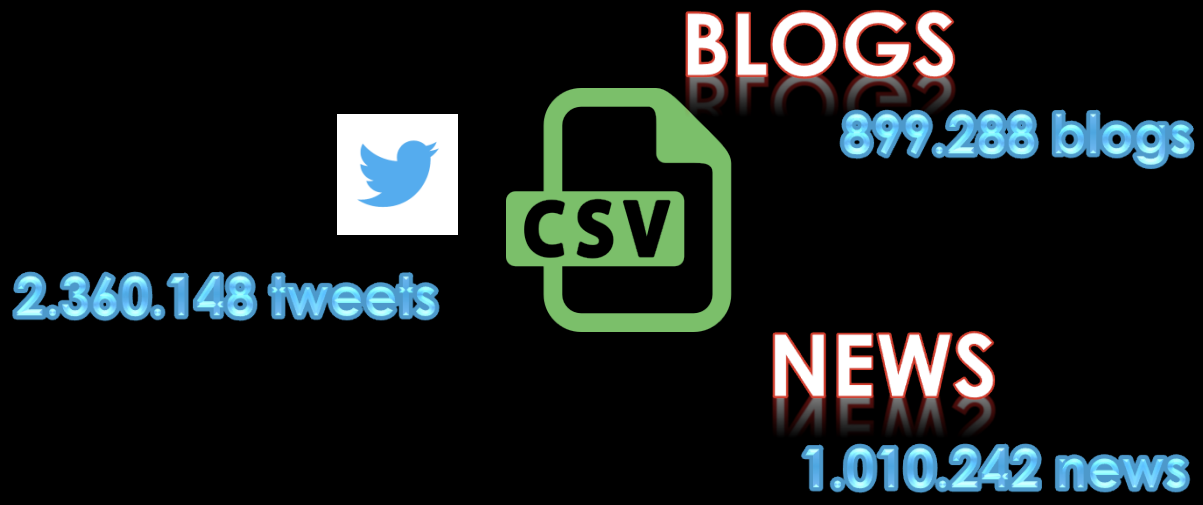
**When someone types:**

"I went to the …"

the application **should presents three options for what the next word might be.** For example, the three words might be "gym", "store", "restaurant".

# THE CHALLENGE:
## BUILD AN APP FOR «NEXT WORD» PREDICTION

"Large databases comprising of text in a target language are commonly used when **generating language models** for various purposes."


# THE SUPPORTING DATA (CORPORA):

3 **CORPUS** – TWITTER (TWEETS), NEWS AND BLOGS
ENGLISH LANGUAGE

# THE PROCESS

**Week#1:**
Overview, Understanding the Problem & Getting the data


**Week#2:**
Exploratory Data Analysis & Modeling


Milestone Report

**Week#3:**
Prediction Model


Prediction Model Evaluation

**Week#4:**
Creative Exploration


Prediction Model Evaluation

**Week#5:**
Product Design & Development

**Week#6:**
Product Presentation Deck


Slide Deck

**Week#7 & #8:**
Submission & Evaluation


Product (shinyApp)

**What do I need to do to solve the challenge at hand?**

**What type of data structure do I need to create language models?**

Basic Models

(MLE) n-grams model

Uni-grams

Bi-grams

N-grams

Model Evaluation

Perplexity

Practical Issues

Overfitting

Zeros

Improved Models

Smoothing

Backoff

**What type of issues do I need to be aware of? And how can I avoid them?**

# UNDERSTANDING THE CHALLENGE

FOUNDATION OF STATISTICAL NLP

# UNDERSTANDING THE CHALLENGE

FOUNDATION OF STATISTICAL NLP

(Probabilistic)
**Language Model**
based on the **provided**
**Corpora**

P(S) = P(w1, w2, .., wn) or P(wn|w1,w2,..,wn-1)

(Probabilistic)
**N-gram Model**

Dan Jurafsky

**The Chain Rule applied to compute joint probability of words in sentence**

$$P(w_1 w_2 \ldots w_n) = \prod_i P(w_i \mid w_1 w_2 \ldots w_{i-1})$$

P("its water is so transparent") =
P(its) × P(water|its) × P(is|its water)
× P(so|its water is) × P(transparent|its water is so)

Dan Jurafsky

**Markov Assumption**

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

# An example: Bi-gram Model (Markov Assumption)

P(S) = P(w1, w2, .., wi) = P(w1 | <s>) P(w2 | w1) P(w3 | w2) ...

## MLE

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

## Corpus

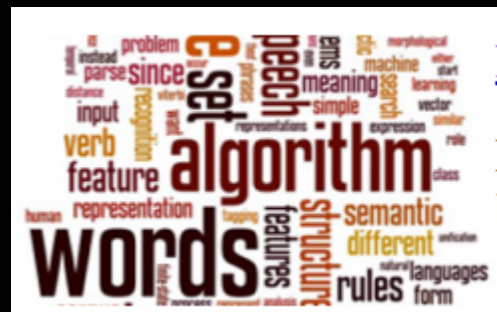
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

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

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

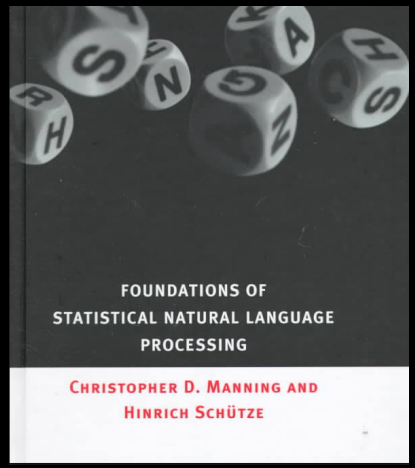
# UNDERSTANDING THE CHALLENGE

FOUNDATION OF STATISTICAL NLP




[Natural Language Processing]

**Lecture Slides from the Stanford Coursera course by [Dan Jurafsky] and [Christopher Manning]**




# UNDERSTANDING THE CHALLENGE

## FOUNDATION OF STATISTICAL NLP - REFERENCES

# UNDERSTANDING THE CHALLENGE

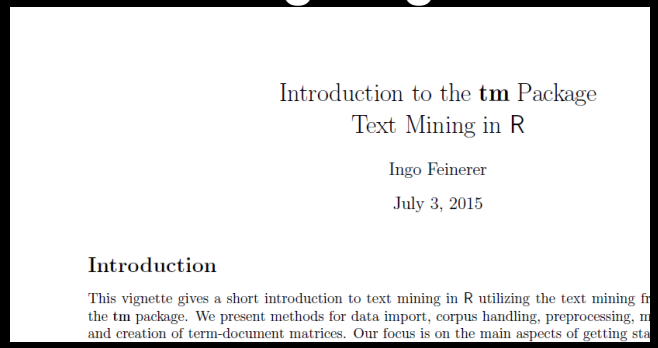
## TECHNOLOGY SUPPORT

## [Technology] R packages

For

## Natural Language Processing

**Package Vignettes**



Introduction to the **tm** Package
Text Mining in R

Ingo Feinerer

July 3, 2015

**Introduction**

This vignette gives a short introduction to text mining in R utilizing the text mining fr
the **tm** package. We present methods for data import, corpus handling, preprocessing, m
and creation of term-document matrices. Our focus is on the main aspects of getting sta

# DATA UNDERSTANDING & PREPARATION

## How to read the data?

The Corpora is quite big and
R uses in-memory data structure ….

## Splitting the data & Exploring the data…

Is there anything peculiar in the available data? What should I look for?

## Data preparation [Feature Engineering]

What do we need to do on the data in order to get required features to be used to create the models? Uni-grams, bi-grams, tri-grams and n-grams?

## Identify preliminary Ingestion Process….

Identify steps to clean and prepare the required data (iterative development)….

## How to increase the % of corpora?

Used to create the "processed" data
From 5% to 60% - a learning experience ….

**See Data Exploration Report**

# The Ingestion Process: the basic flow

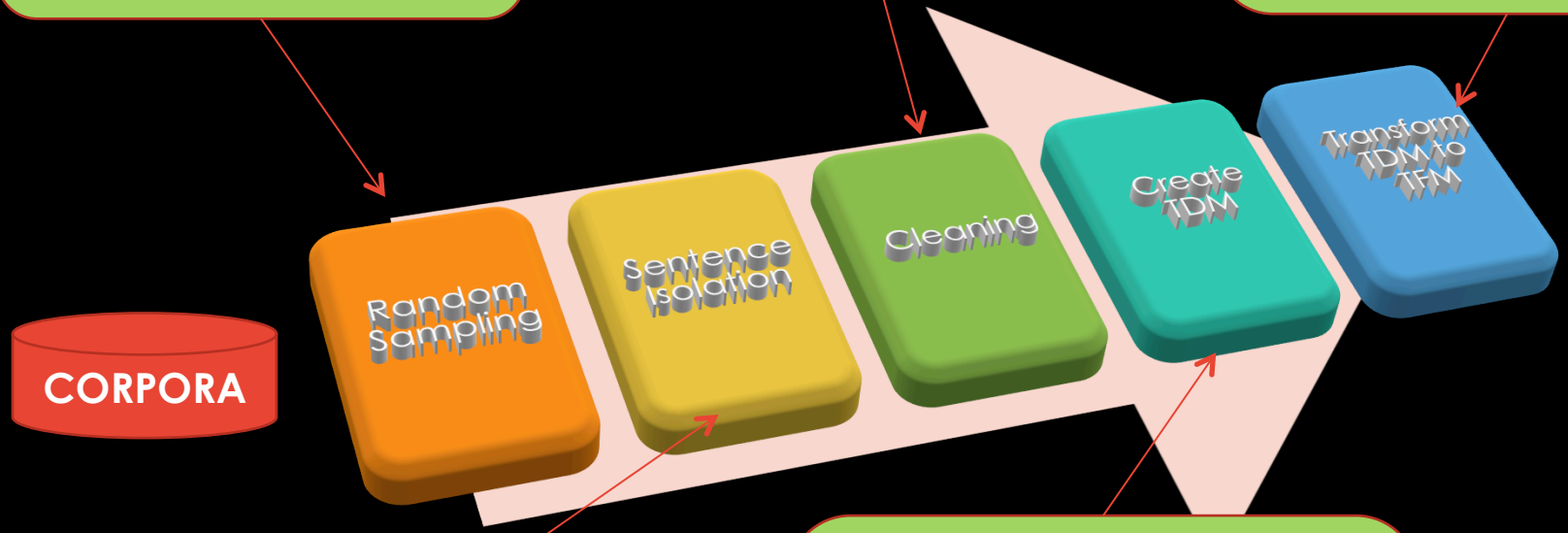
10% of the original corpora for creating the language model (random sampling).

Removing short entries – less than 50 chars.

Adding start/ end sentence markers
Removing gremling (encoding issue)
Removing some contractions (u, r, u.s, ...)

Reduce Term Document Matrices (**TDM**) to Term Frequency Matrices (**TFM**) for unigrams, bigrams and trigrams.

Example for unigrams:
**Term, Frequency**
the, 477030

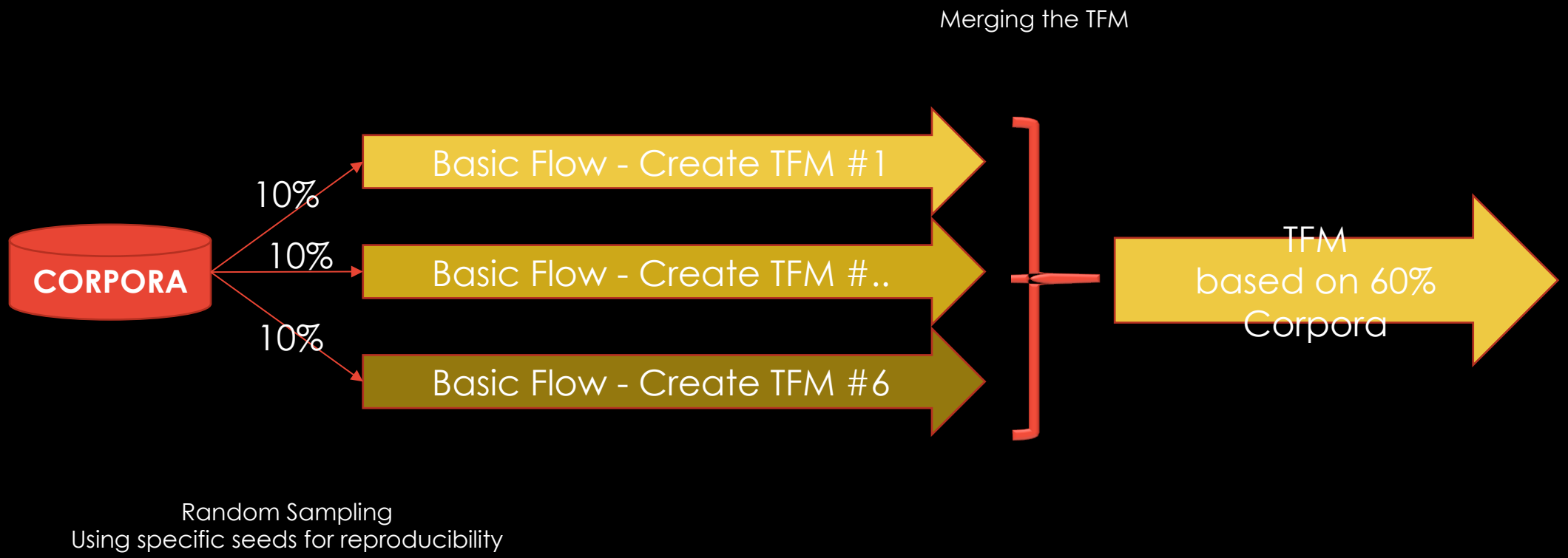



Splitting up each entry (tweet, news, blog) into separate sentences

Create the Term Document Matrices (TDM)

*tolower
*remove profanity words
*remove numbers
* remove punctation (except apostrophe)
* tokenize in n-grams

# The Ingestion Process: increasing the % of corpora used




Random Sampling
Using specific seeds for reproducibility

See Ingestion Report

# MODELLING & EVALUATION

## Test dataset for evaluation

20% of the corpora (testing dataset) <-> ad-hoc non biased dataset

## Another dataset for optimization

20% of the corpora (optimization of linear interpolation)

## Which model can be used?

Identify the possible models to be used (experiment) & implement them
(use a TDD approach to save time – unit test the created models)

## Model Evaluation & Optimization

How to evaluate the model? How to optimize the model?
Which one is the best model?
How to optimize the model for a mobile app? Memory/ processing constraints?

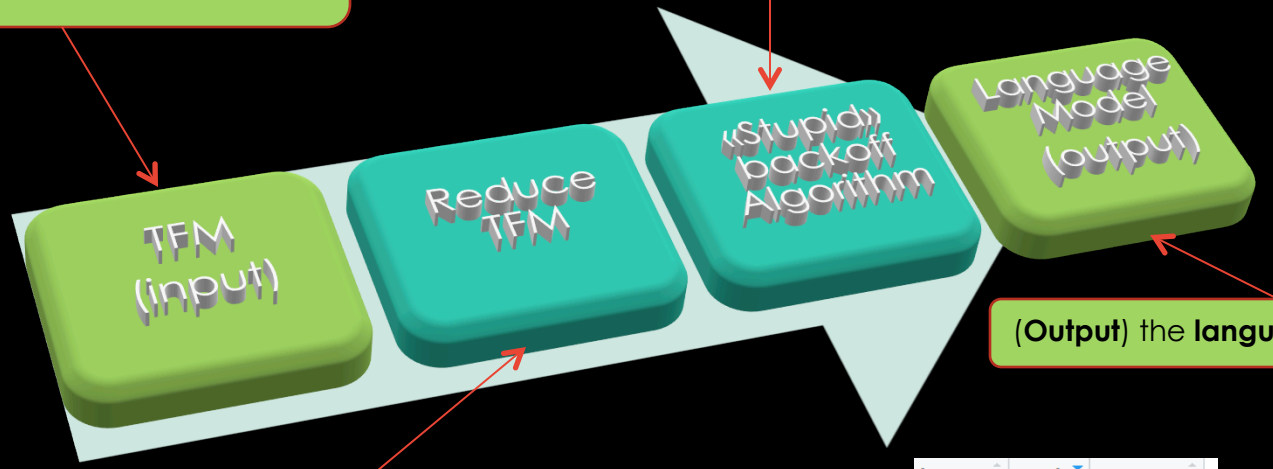
# The Model Creation ....

Summary Info About NGrams - Original Memory Footprint

| ng | noOfEntries_V | N | memory Size |
|---|---|---|---|
| 1 | 362734 | 56859890 | 25.1 Mb |
| 2 | 6905615 | 53549688 | 520 Mb |
| 3 | 20496858 | 50239484 | 1.6 Gb |

(**input**) **Term Frequency Matrices (TFM)** for unigrams, bigrams and trigrams.

Calculate the «Stupid» backoff score and create an ad-hoc data structure representing the language the model




(**Output**) the **language model**

Reducing the vocabulary in order to decrease the memory footprint of the model

Summary Info About NGrams - Reduced Memory Footprint

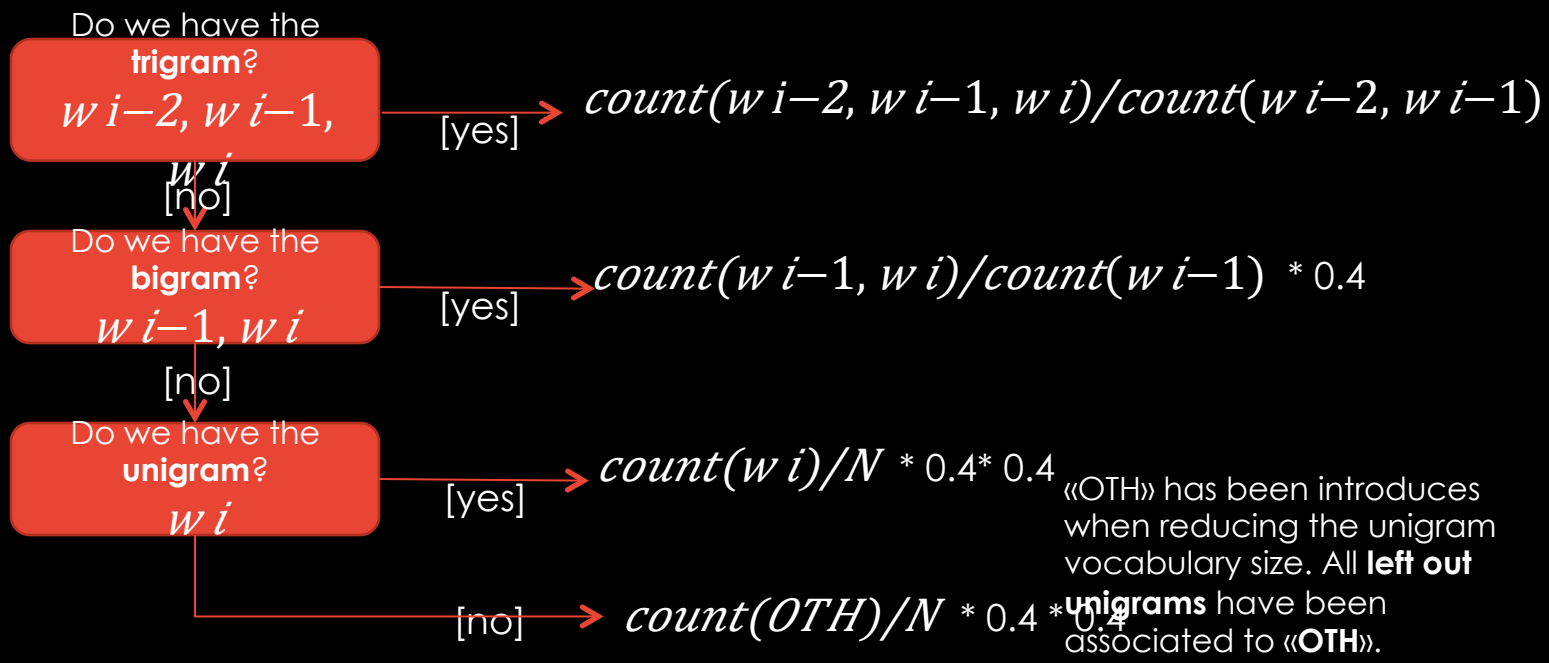
| ng | noOfEntries_V | N | memory Size |
|---|---|---|---|
| 1 | 9132 | 56859890 | 1.1 Mb |
| 2 | 368697 | 40162258 | 26.8 Mb |
| 3 | 229289 | 15071838 | 10.2 Mb |

| term | count | score |
|---|---|---|
| OTH | 515050 | 7.706193e-03 |
| the | 477030 | 7.137337e-03 |
| to | | |
| and | | |
| a | | |
| of | | |

| term | next.word | b.count | u.count | score |
|---|---|---|---|---|
| of | the | 43491 | 201072 | 0.086518262 |
| in | the | 41190 | 165570 | 0.099510781 |
| <s> | i | 29545 | 398243 | 0.029675349 |
| <s> | the | 23721 | 398243 | 0.023825654 |
| to | | | | |
| for | | | | |
| on | | | | |
| to | | | | |

| term | next.word | t.count | b.count | score |
|---|---|---|---|---|
| one of | the | 3494 | 7277 | 0.480142916 |
| a lot | of | 2967 | 4708 | 0.630203908 |
| <s> thanks | for | 2736 | 4864 | 0.562500000 |
| thanks for | the | 2260 | 4281 | 0.527914039 |
| to be | a | 1776 | 16152 | 0.109955423 |

# The Language Model: «Stupid» backoff....




Different models have been implemented: n-grams (n = 1,2,3), linear interpolation (n-grams, n = 1,2,3) with Good Turing smoothing and "Stupid" backoff (with no discount). The model evaluations has been done using the **perplexity measurement** and an **ad-hoc testing dataset** (around 40 sentences).

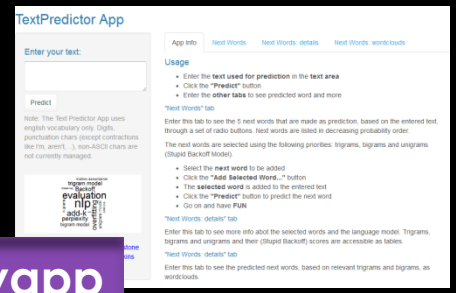
The **"Stupid" backoff model** was the one able to **minimize the perplexity measurement.**

# THE TECHNOLOGY STACK

R markdown

RStudio

Rpres




shinyapp

base {pkg}

OpenNLP

ggplot2

wordcloud

tm

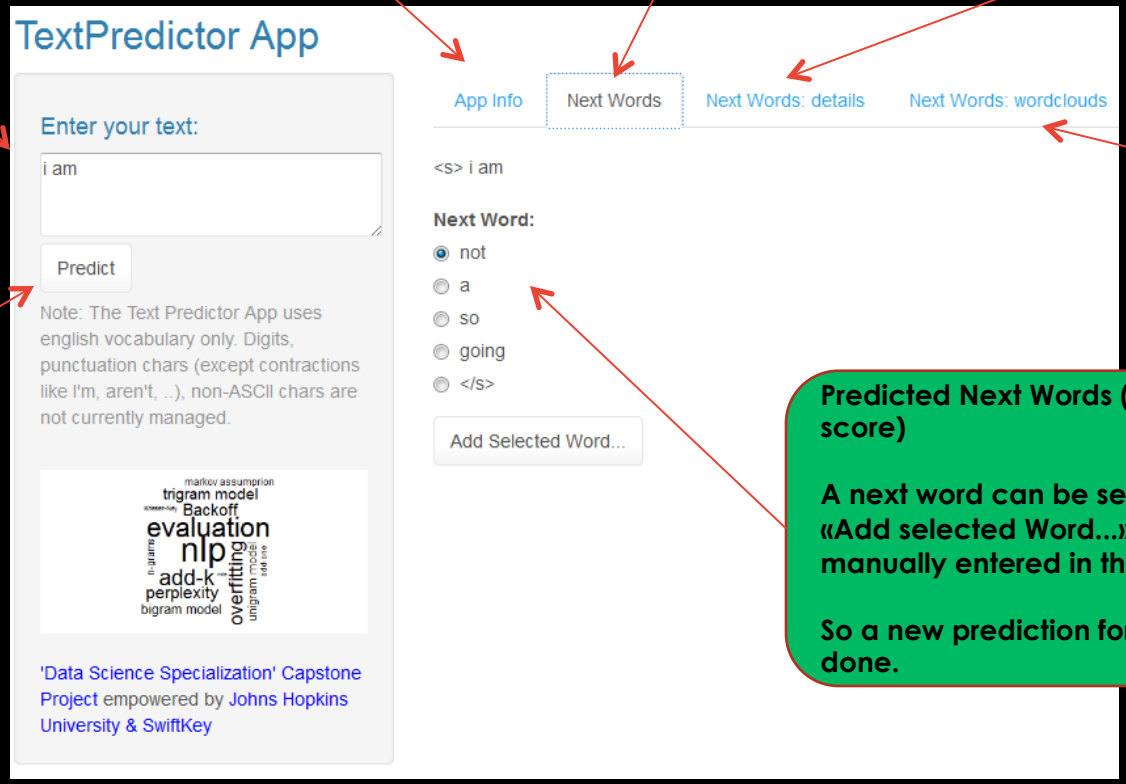
Rweka

[Message Area] Area where to type your message ...

Basic Info on how to use the App

(probable) Next Words – 5 most likely

Details (from the model) about the predicted next words




Next Word Wordclouds (for bigrams & trigrams)

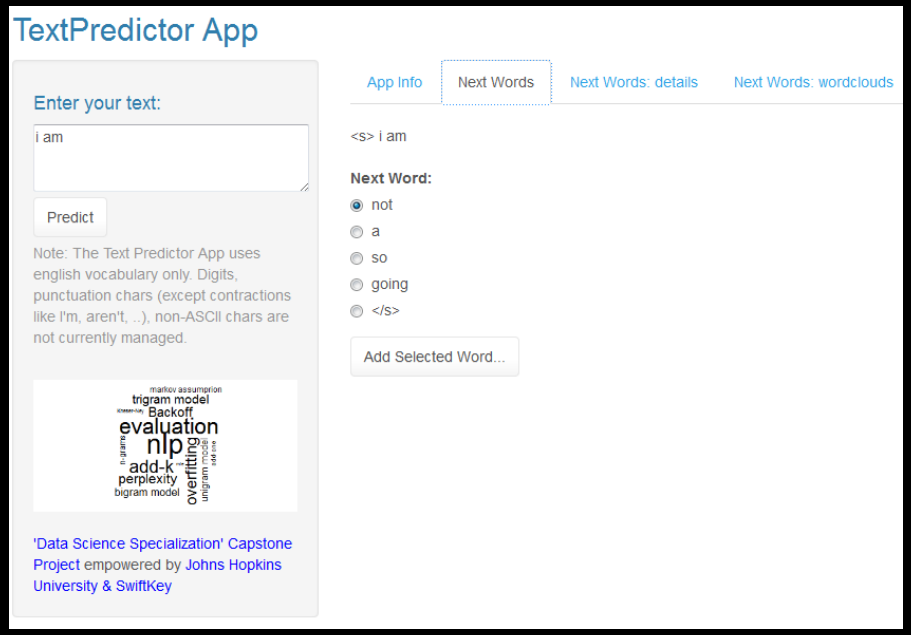
To make a prediction for next words

Predicted Next Words (ordered by decreasing score)

A next word can be selected/ added using «Add selected Word...» button/ or can be manually entered in the [Message Area].

So a new prediction for the next word can be done.

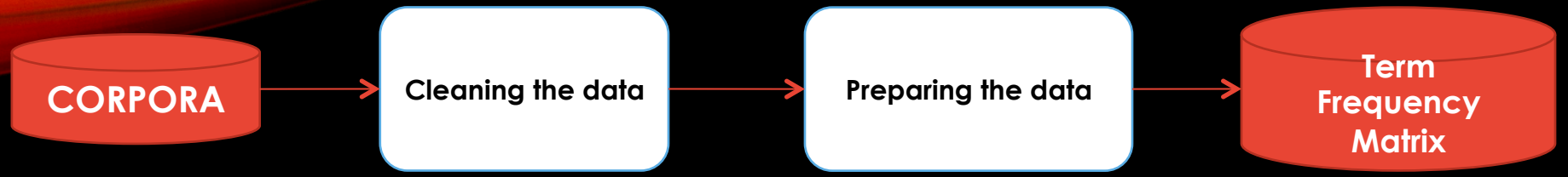
# LIVE DEMO


TextPredictor App
Enter your text:
i am
Predict
Note: The Text Predictor App uses english vocabulary only. Digits, punctuation chars (except contractions like I'm, aren't, ..), non-ASCII chars are not currently managed.
markov assumption
trigram model
Backoff
evaluation
nlp
add-k
perplexity
bigram model
overfitting
unigram model
'Data Science Specialization' Capstone Project empowered by Johns Hopkins University & SwiftKey
App Info
Next Words
Next Words: details
Next Words: wordclouds
<s> i am
Next Word:
not
a
so
going
</s>
Add Selected Word...

Uderstanding the data

Exploratory Analysis

Data Structures
(performance, memory footprint)




```
[1] "and u didnt give it to him? lol"
[2] "what's good. I see the success you got poppin in yo area."
[3] "RT : Consumers are visual. They want data at their finger tips. Mobile is the only way to deliver this, 24/7."
[4] "u welcome"
[5] "It is #RHONJ time!!"
[6] "The key to keeping your woman happy= attention, affection, treat her like a queen and sex her like a pornstar!"
```

| terms | twitter.count | news.count | blogs.count | total |
|---|---|---|---|---|
| one of the | 564 | 1461 | 1469 | 3494 |
| a lot of | 627 | 1144 | 1196 | 2967 |
| <s> thanks for | 2667 | 1 | 68 | 2736 |
| thanks for the | 2239 | 5 | 16 | 2260 |
| to be a | 617 | 495 | 664 | 1776 |

| gs.count | total |
|---|---|
| 18800 | 43491 |
| 15339 | 41190 |

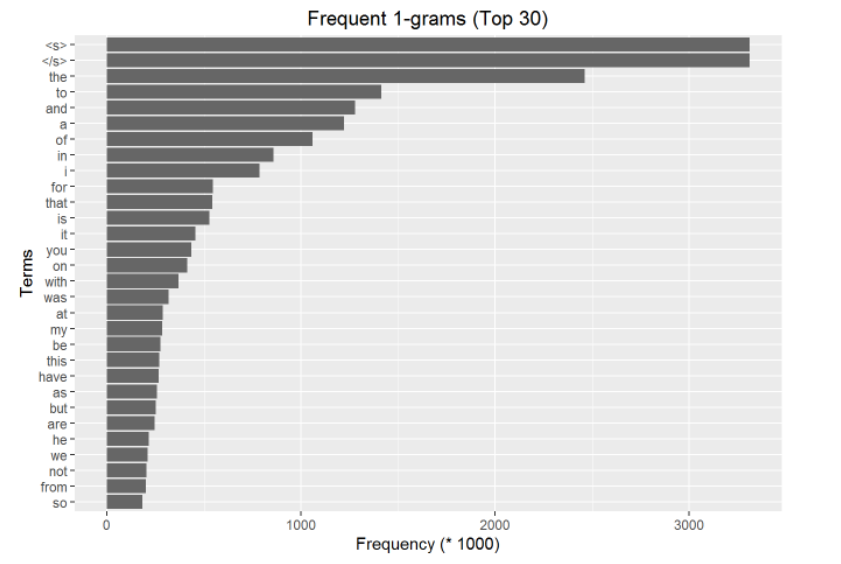
| | | | | |
|---|---|---|---|---|
| <s> i | 19626 | 2769 | 7150 | 29545 |
| <s> the | 4556 | 12897 | 6268 | 23721 |
| to the | 4524 | 8635 | 8719 | 21878 |
| for the | 7278 | 6830 | 5843 | 19951 |

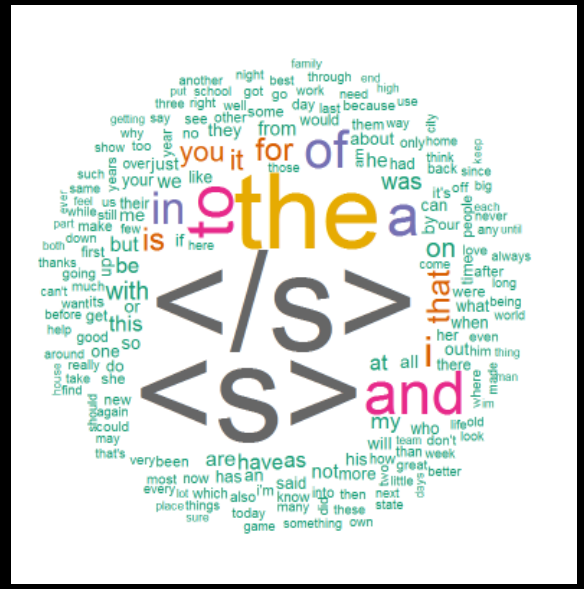
| news.count | blogs.count | total |
|---|---|---|
| 198235 | 185340 | 477030 |
| 98395 | 82909 | 398243 |
| 98395 | 82909 | 398243 |

| | | | |
|---|---|---|---|
| to | 78501 | 90516 | 106648 | 275665 |
| and | 43629 | 88843 | 108536 | 241008 |

**Memory Footprint**
**1-G: 25.1 Mb**
**2-G: 520 Mb**
**3-G: 1.6 Gb**

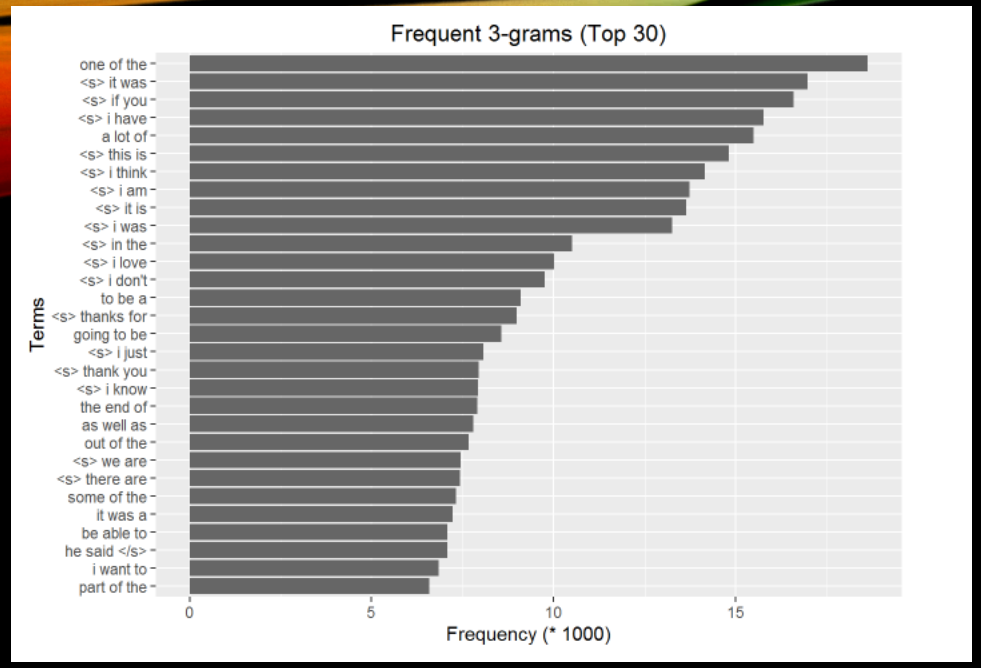
# THE INGESTION PIPELINE – PART 1

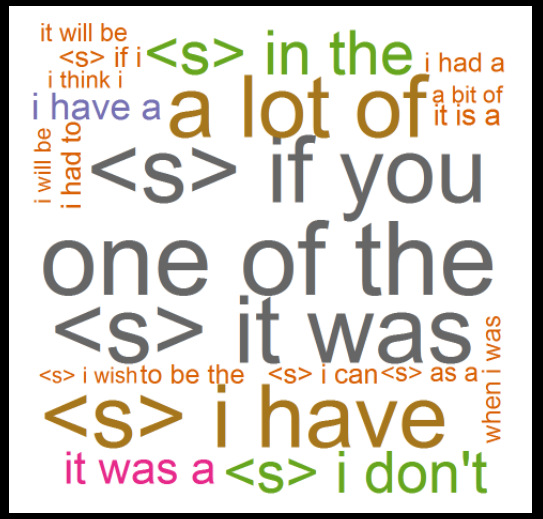



**How many unique words do you need in a frequency sorted dictionary to cover 50% of all word instances in the language? 90%?**

| N = number of tokens | V = vocabulary size | 50% coverage | 90% coverage |
|---|---|---|---|
| 56859890 | 362734 | 81 | 6141 |




**How many unique words do you need in a frequency sorted dictionary to cover 50% of all word instances in the language? 90%?**

| N = number of tokens | V = vocabulary size | 50% coverage | 90% coverage |
|---|---|---|---|
| 50239484 | 20496858 | 1924741 | 15472910 |

Models: strategy | Model Impl. & Tuning | Model Evaluations


**Memory Footprint**
Small when moving into the app (e.g. Mobile, web)

- "Stupid backoff" (Brants *et al*. 2007)
- No discounting, just use relative frequencies

$$S(w_i \mid w_{i-k+1}^{i-1}) = \begin{cases} \dfrac{\text{count}(w_{i-k+1}^{i})}{\text{count}(w_{i-k+1}^{i-1})} & \text{if } \text{count}(w_{i-k+1}^{i}) > 0 \\ 0.4\,S(w_i \mid w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{\text{count}(w_i)}{N}$$

| term | count | score |
|---|---|---|
| OTH | 515050 | 7.706193e-03 |
| the | 477030 | 7.137337e-03 |
| to | | |
| and | | |
| a | | |
| of | | |

| term | next.word | b.count | u.count | score |
|---|---|---|---|---|
| of | the | 43491 | 201072 | 0.086518262 |
| in | the | 41190 | 165570 | 0.099510781 |
| <s> | i | 29545 | 398243 | 0.029675349 |
| <s> | the | 23721 | 398243 | 0.023825654 |
| to | | | | |
| for | | | | |
| on | | | | |
| to | | | | |

| term | next.word | t.count | b.count | score |
|---|---|---|---|---|
| one of | the | 3494 | 7277 | 0.480142916 |
| a lot | of | 2967 | 4708 | 0.630203908 |
| <s> thanks | for | 2736 | 4864 | 0.562500000 |
| thanks for | the | 2260 | 4281 | 0.527914039 |
| to be | a | 1776 | 16152 | 0.109955423 |

**Memory Footprint (small)**
**1-G: 1.1 Mb (90%)**
**2-G: 26.8 Mb (75%)**
**3-G: 10.2 Mb (30%)**

# THE MODEL CREATION – PART 2