Identifying “Fake News” with NLP

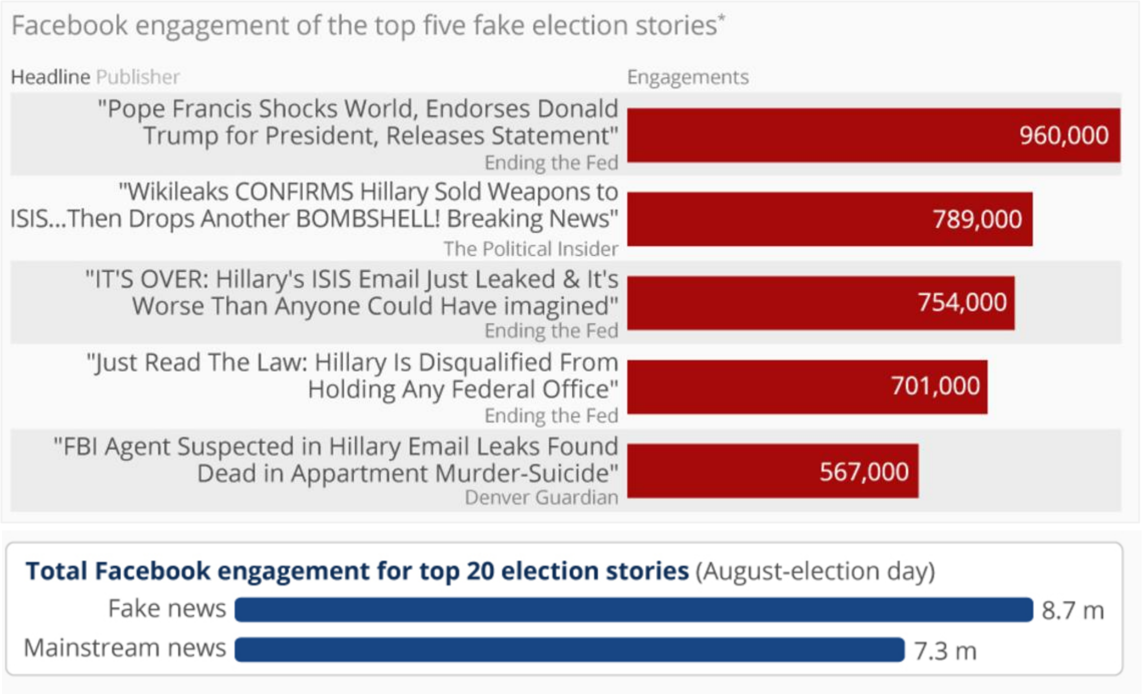
JULIA GOLDSTEIN AND MIKE GHOUL

Agenda

- Introduction & Background
- Workflow
- Data Collection
- Text Preprocessing
- Text to Features
- Classification Models
- Topic Modeling
- Stacked Model
- Conclusion and Future Steps

Introduction & Background

- Fake news often confuses the general public and sows distrust in the media as a whole
- There has been significant analysis on the impact of fake news in the 2016 U.S. presidential election
- From August to November 2016, fake stories earned more shares, reactions, and comments on Facebook than real news stories



*Engagement is measured as total number of shares, comments and reactions
Source: http://libguides.pace.edu/fakenews

Project Questions

- Can we use Natural Language Processing to identify and classify fake news articles?

- How do we distinguish between real vs. fake news?

- Are there any similarities between what we consider ”real” and “fake” stories?

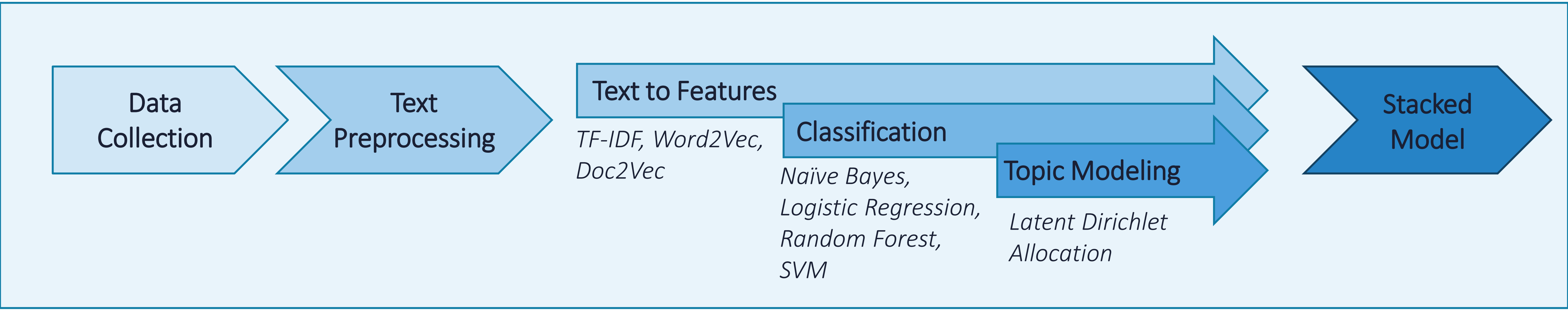
Data Gathering/Classification

- Used Kaggle data set to represent “fake news”:
  - Contains articles from fake news sites (identified with the B.S. Detector browser extension) from November 2016
  - Searches all links on a given webpage for references to unreliable sources

- Collected “real news” from the same website:
  - Collected over 500K articles scraped during November 2016
  - Selected ~24K articles by filtering on known, reputable sources (e.g., Washington Post, CNN, Forbes, NPR, NY Times, NBC News, The Guardian)

- Final data set:
  - 23,571 real articles tagged as “0”
  - 11,040 fake articles tagged as “1”

Workflow



Data Collection

Text Preprocessing

Text to Features

TF-IDF, Word2Vec, Doc2Vec

Classification

Naïve Bayes, Logistic Regression, Random Forest, SVM

Topic Modeling

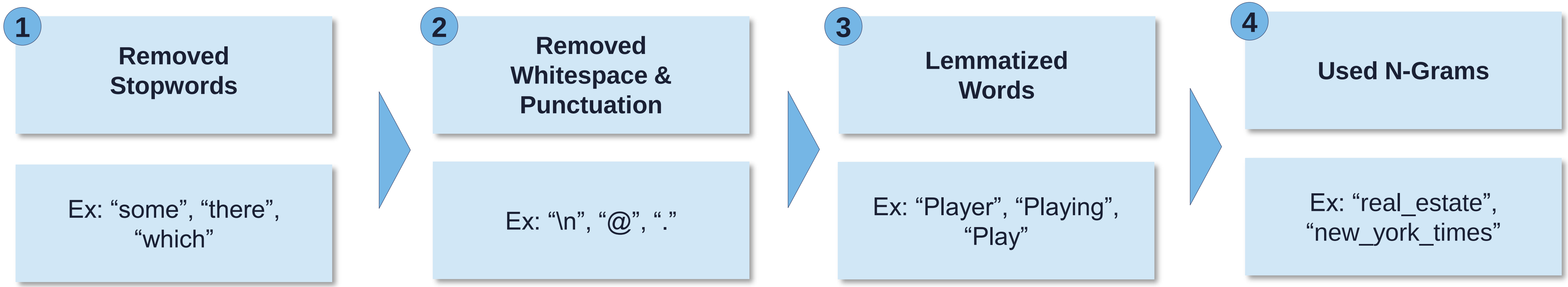
Latent Dirichlet Allocation

Stacked Model

Text Preprocessing

Text classification model performance depends on the type of words used in the corpus and features created for classification.

Commons words and other “noisy” elements increase the number of features in a model but usually reduce its performance. To prepare our text, we tokenized our articles and:



1 Removed Stopwords

Ex: “some”, “there”, “which”

2 Removed Whitespace & Punctuation

Ex: “\n”, “@”, “.”

3 Lemmatized Words

Ex: “Player”, “Playing”, “Play”

4 Used N-Grams

Ex: “real_estate”, “new_york_times”

Example

Before Processing:

“During the campaign, Democrats accused Russia of interfering in a way that helped Trump, the Republican nominee. U.S. intelligence officials have tied the Russian government to the hacking and subsequent leaking of Democrats emails that harmed Hillary Clintons campaign.\nA summary of the brief and informal discussion, provided by the White House, said Obama restated U.S. and allies commitment to Ukraines sovereignty, urged Putin to uphold Russias commitments under the Minsk agreements, and said U.S. Secretary of State John Kerry.”

After:

“democrats accuse_russia interfere way help trump republican_nominee u.s._intelligence_official tie russian_government hacking subsequent leaking democrats email harm hillary_clintons_campaign summary brief informal discussion provide white_house say obama restate u.s._ally commitment ukraines sovereignty urge putin_uphold russias commitment_minsk agreement say u.s. secretary_state_john_kerry”

Converting Text to Features: TF-IDF

- Statistic that reflects how important a word is to a document in a corpus
  - Increase proportionally to the number of times a word appears in a document
  - Offset by number of times the word appears in the overall corpus (i.e., common words)

*count of word **i** in document **j***

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

# of documents

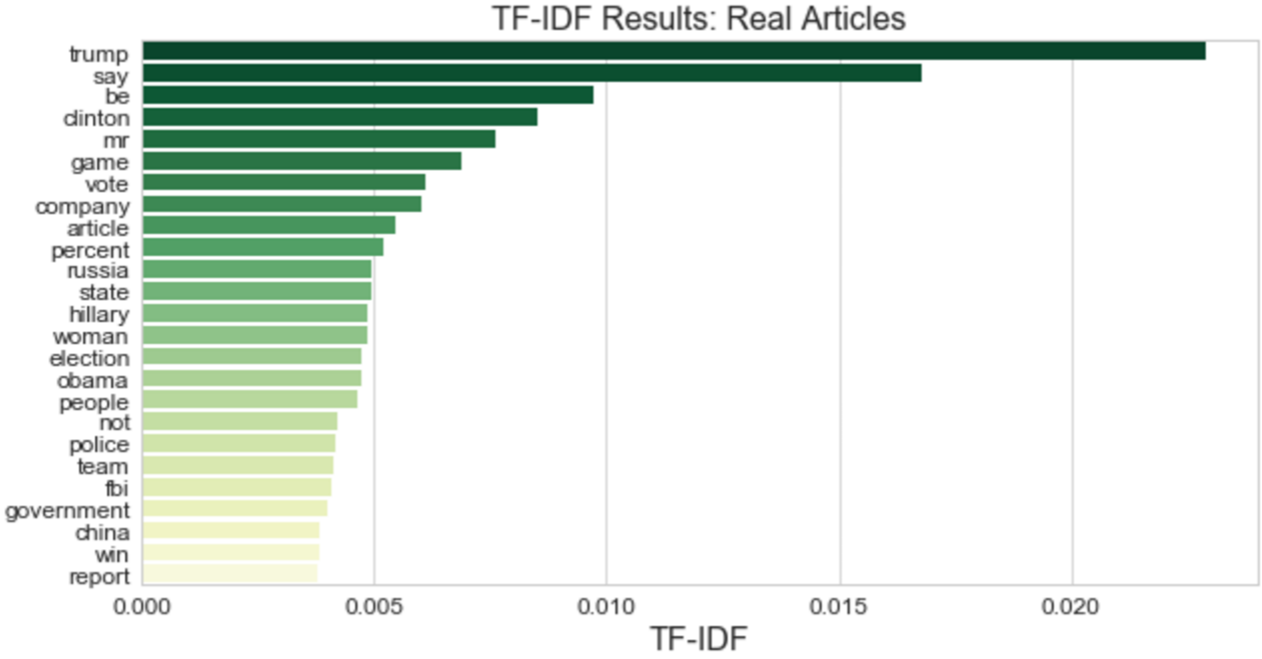
*# of documents containing word **i***

Example Output:

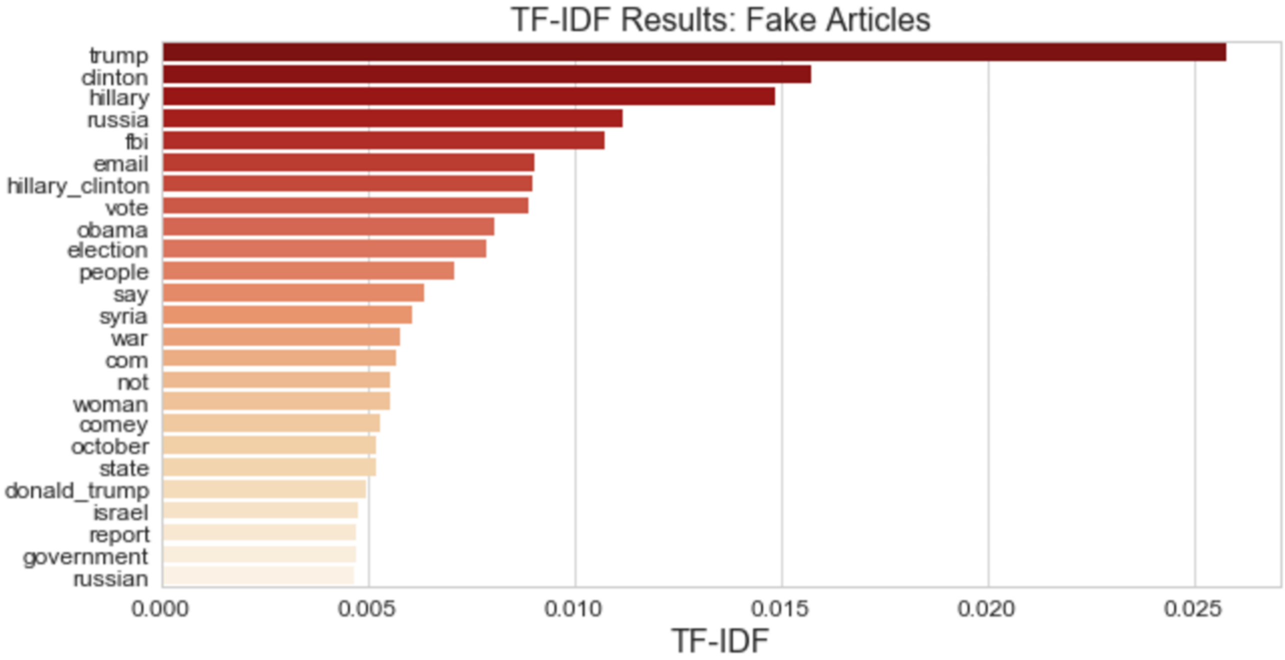
column for each word in corpus

Article #1	.239441	0	.153457	0	0	.195243	0	.140004	.197255

Term Frequency with TF-IDF



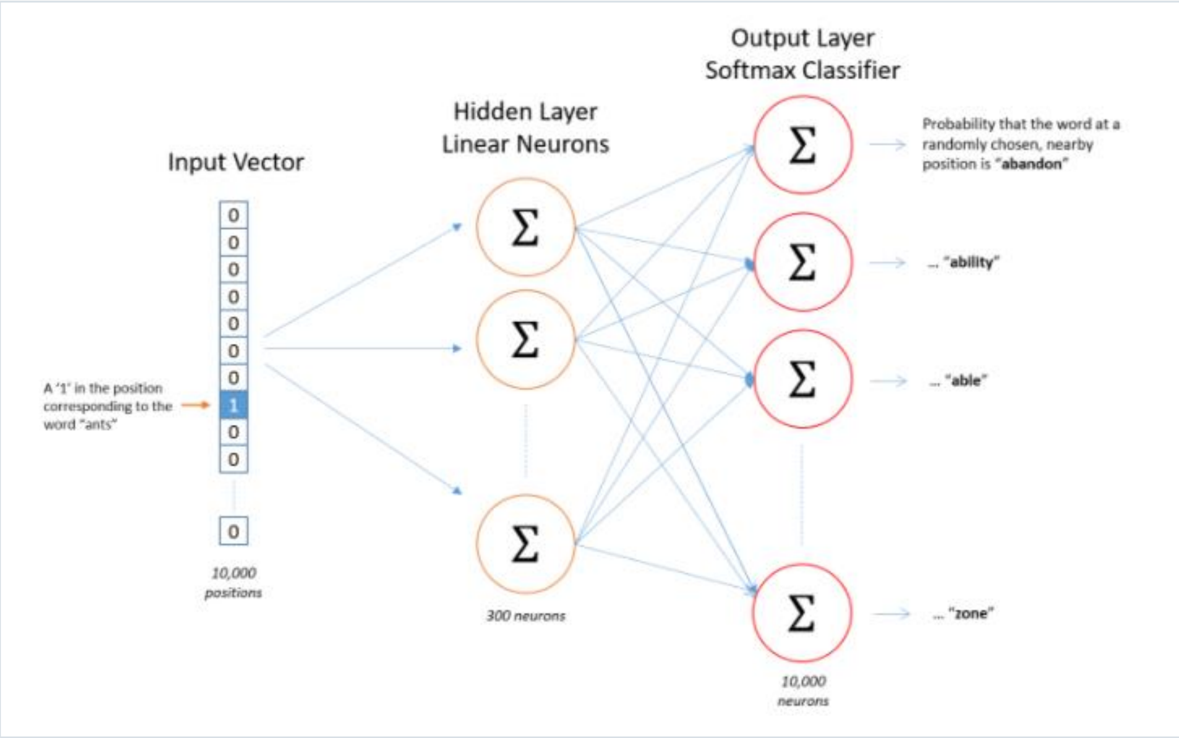
TF-IDF Results: Real Articles



TF-IDF Results: Fake Articles

Converting Text to Features: Word2Vec

- Transforms text into feature vectors and maintains relationships in the corpus
- We used vectors pre-trained on the Google News dataset, containing 300-dimensional vectors for 3 million words and phrases
- We averaged the word vectors for each of our articles to obtain a single vector


Input Vector
0
0
0
0
0
0
0
1
0
0
0
A '1' in the position corresponding to the word "ants"
10,000 positions
Hidden Layer Linear Neurons
Σ
Σ
Σ
300 neurons
Output Layer Softmax Classifier
Σ
Σ
Σ
Σ
10,000 neurons
Probability that the word at a randomly chosen, nearby position is "**abandon**"
... "**ability**"
... "**able**"
... "**zone**"

Source: word2vec-tutorial-the-skip-gram-model

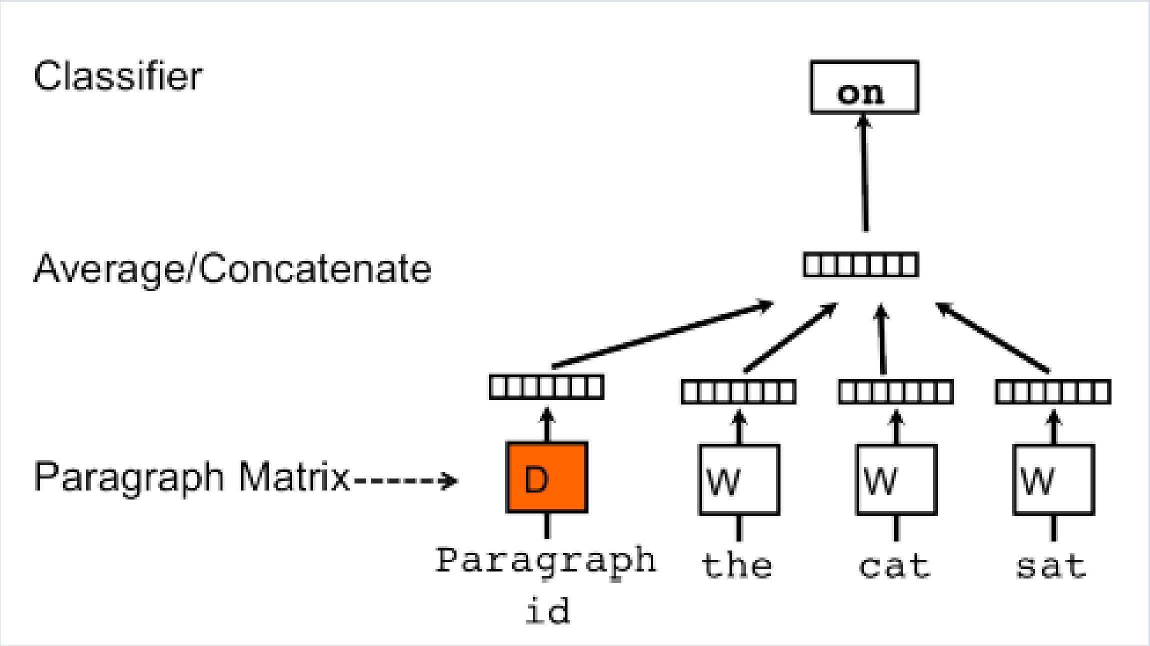
Example Output:

300 columns (based on chosen # of hidden layer neurons)

Article #1	.002029	.020478	-.031536	-.050241	-014399	.027941	-.058338	.058330	.123510

Converting Text to Features: Doc2Vec

- Doc2Vec generates a word vector for each word and a document vector for each document (no need to average results)
- We did not expect these features to perform as well in text classification, but it allowed us to look at the specific words and relationships in our corpus



Source: https://medium.com/towards-data-science/a-gentle-introduction-to-doc2vec-db3e8c0cce5e

Handling Imbalanced Data

There are several metrics that help indicate performance when dealing with imbalanced data:

1. Confusion Matrix:

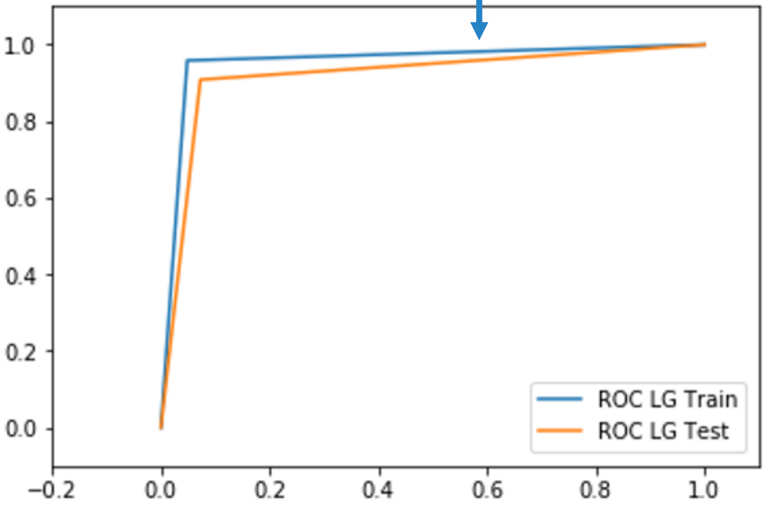
	Predicted Real	Predicted Fake
Actual Real	True Negative (TN)	False Positive (FP)
Actual Fake	False Negative (FN)	True Positive (TP)

2. F1 Score:
   - Strikes a balance between precision and recall
   - Scoring metric used for model optimization

3. ROC AUC Score:
   - Graphs TPR (Sensitivity) and False Positive Rate
   - Can help us clearly understand the performance of a classifier

Model Results Using TF-IDF

Model	**Train**	**Test**			
	ROC AUC	ROC AUC	Precision	Recall	F1 Score
Naïve Bayes	.9073	.8856	0.88	0.82	0.85
Logistic Regression	.9551	.9180	0.85	0.91	0.88
Random Forest	.9995	.8893	0.90	0.82	0.86
Support Vector Machine	.9300	.9079	0.83	0.90	0.87

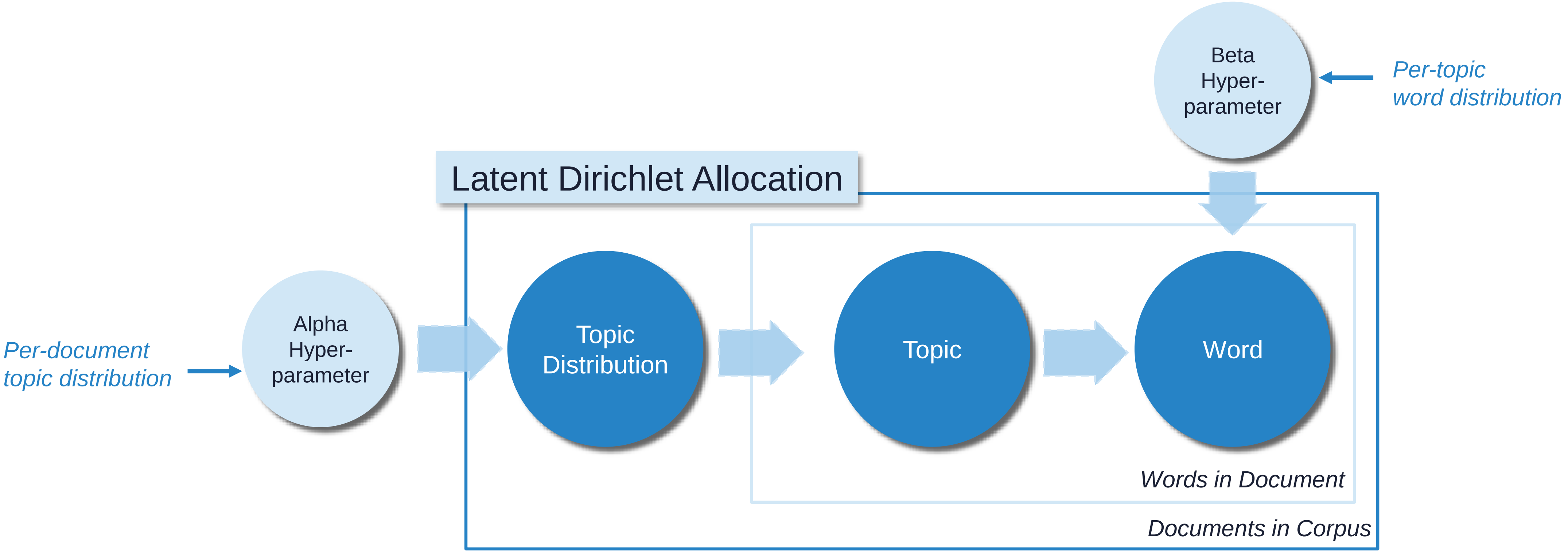


Model Results Using Word2Vec & Doc2Vec

	Model	**Train**	**Test**			
		ROC AUC	ROC AUC	Precision	Recall	F1 Score
Word2Vec	Logistic Regression	0.8356	0.8336	0.70	0.83	0.76
	Random Forest	0.9786	0.8021	0.78	0.69	0.74
	Support Vector Machine	0.8422	0.8414	0.71	0.84	0.77
Doc2Vec	Logistic Regression	0.7464	0.7397	0.56	0.75	0.64

Topic Modeling

- Topic modeling can be used to describe and summarize the documents in a corpus

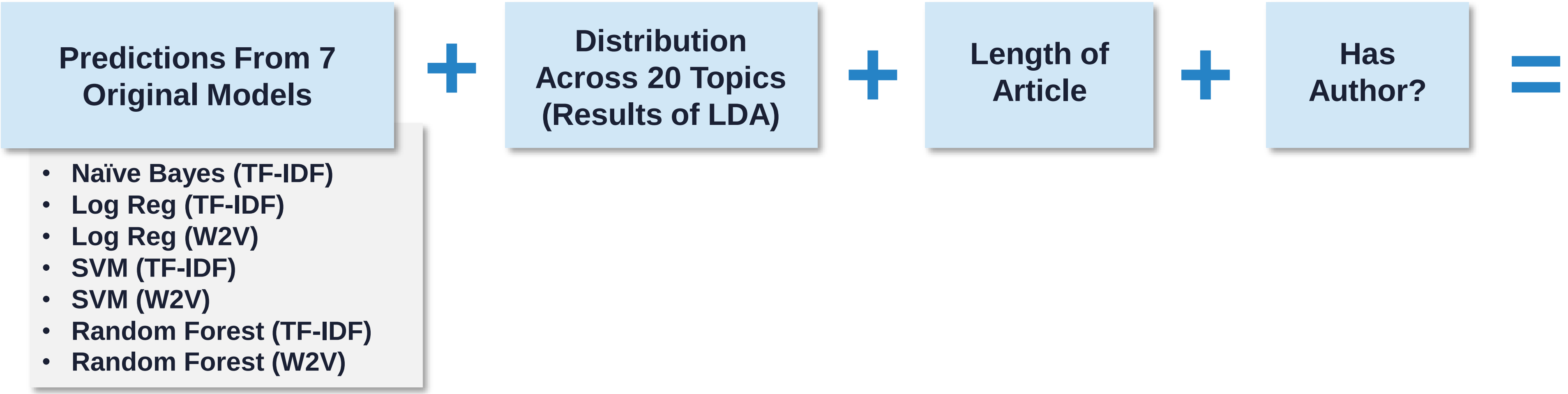


Example Output:

distribution across 20 topics (%)

Article #1	.52534	.19638	0	.07166	0	.05812	0	0	.14790

Stacked Model Results



Predictions From 7 Original Models

- Naïve Bayes (TF-IDF)
- Log Reg (TF-IDF)
- Log Reg (W2V)
- SVM (TF-IDF)
- SVM (W2V)
- Random Forest (TF-IDF)
- Random Forest (W2V)

+

Distribution Across 20 Topics (Results of LDA)

+

Length of Article

+

Has Author?

=

	Predicted Real	Predicted Fake
Actual Real	4665	50
Actual Fake	34	2174

Model	Train	Test			
	ROC AUC	ROC AUC	Precision	Recall	F1 Score
Logistic Regression	0.9859	0.9876	0.98	0.98	0.98
Naïve Bayes	0.9212	0.9215	0.82	0.94	0.87

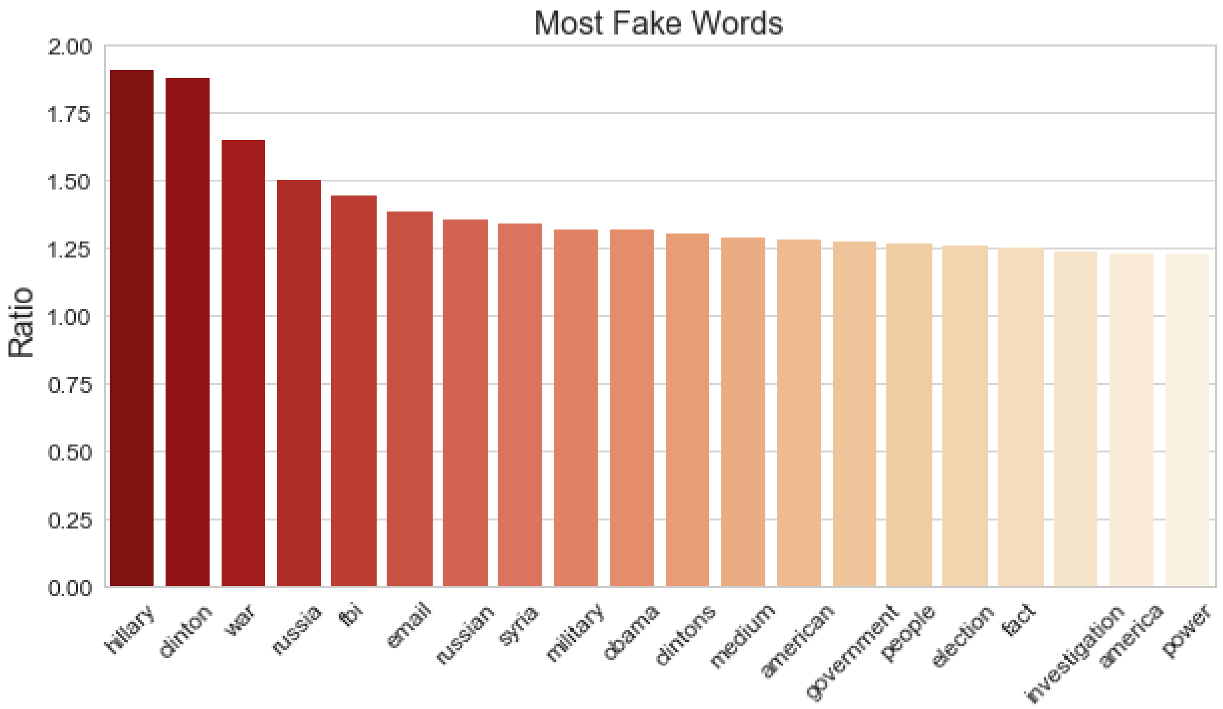
Conclusion and Future Steps

Conclusion:

- Fake news is a global problem that large tech companies are still struggling to address
- Line between misleading statement and falsehood can be difficult to determine
- Concentration of specific topics in fake news makes it difficult to identify real news

Future Steps:

- Gather more articles to expand topic distribution and time range
- Re-process text to take writing style into account (capitalization, punctuation)
- Build an app that determines whether a new article is Real or Fake



Sources We Used

- Ultimate Guide to Understand & Implement Natural Language Processing
- An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec
- Modern NLP in Python
- Word2Vec Tutorial – The Skip-Gram Model
- A Gentle Introduction to Doc2Vec
- This Analysis Shows How Viral Fake Election News Stories Outperformed Real News On Facebook
- OpenSources
- The Joy of Topic Modeling