We have a problem

- At Return Path, we process billions of emails a year, from *tons* of senders
- We want to tag and cluster senders
  - Industry verticals (e-commerce, apparel, travel, etc.)
  - Type of customers they sell to (luxury, soccer moms, etc.)
  - Business model (daily deals, flash sales, etc.)
- It’s too much to do by hand!
What to do?

- Standard approaches aren’t great
  - Bag of words classification model (document-term matrix, LSA, LDA)
    - Have to manually label lots of cases first
    - Difficult with lots of data (especially LDA)
  - Bag of words clustering
    - Can’t easily put one company into multiple categories (i.e. more general tagging)
    - Needs lots of tuning

- How about deep learning neural networks?
  - Very trendy. Let’s try it!
Neural Networks

- Machine learning algorithms modeled after the way the human brain works
- Learn patterns and structure by passing training data through “neurons”
- Useful for classification, regression, feature extraction, etc.
Deep Learning

- Neural networks with *lots* of hidden layers (hundreds)
- State of the art for machine translation, facial recognition, text classification, speech recognition
  - Tasks with real *deep* structure, that humans do automatically but computers struggle with
  - Should be good for company tagging!
Distributed Representations

- Human brain uses distributed representations
- We can use deep learning to do the same thing with words (letters -> words -> phrases -> sentences -> …)
Deep Learning Challenges

- Computationally difficult to train (i.e., slow)
  - Each hidden layer means more parameters
  - Each feature means more parameters
- Real human-generated text has a near-infinite number of features and data
  - i.e., slow would be a problem
- Solution: use word2vec
word2vec

- Published by scientists at Google in 2013
- Python implementation in 2014
  - gensim library
- Learns *distributed vector representations* of words (“word to vec”) using a neural net
  - NOTE for hardcore experts: word2vec does not strictly or necessarily train a deep neural net, but it uses deep learning technology (distributed representations, backpropagation, stochastic gradient descent, etc.) and is based on a series of deep learning papers
What is the output?

• Distributed vector representations of words
  ○ each word is encoded as a vector of floats
  ○ \( \text{vec}_{\text{queen}} = (0.2, -0.3, .7, 0, \ldots, .3) \)
  ○ \( \text{vec}_{\text{woman}} = (0.1, -0.2, .6, 0.1, \ldots, .2) \)
  ○ length of the vectors = dimension of the word representation
  ○ key concept of word2vec: words with similar vectors have a similar meaning (context)
word2vec Features

- Very fast and scalable
  - Google trained it on 100’s of billions of words
- Uncovers deep latent structure of word relationships
  - Can solve analogies like King::Man as Queen::?
    or Paris::France as Berlin::?
  - Can solve “one of these things is not like another”
  - Can be used for machine translation or automated sentence completion
How does it work?

- Feed the algorithm (lots of) sentences
  - totally *unsupervised* learning
- word2vec trains a neural net that encodes the *context* of words within sentences
  - “Skip-grams”: what is the probability that the word “queen” appears 1 word after “woman”, 2 words after, etc.
word2vec at Return Path

- At Return Path, we implemented word2vec on data from our Consumer Data Stream
  - billions of email subject lines from millions of users
  - fed 30 million unique subject lines (300m words) and sending domains into word2vec (using Python)
Grouping companies with word2vec

- Find daily deals sites like Groupon

  [word for (word, score) in model.most_similar('groupon.com', topn = 100) if '.com' in word]


- Find apparel sites like Gap

  [word for (word, score) in model.most_similar('gap.com', topn = 100) if '.com' in word]

More \textit{word2vec} applications

- Find relationships between products
  \begin{verbatim}
  model.most_similar(positive=['iphone', 'galaxy'], negative=['apple']) = 'samsung'
  ie. iphone::apple as galaxy::? samsung!
  \end{verbatim}

- Distinguish different companies
  \begin{verbatim}
  model.doesnt_match(['sheraton','westin','aloft','walmart']) = 'walmart'
  ie. Wal Mart does not match Sheraton, Westin, and Aloft hotels
  \end{verbatim}

- Other possibilities
  \begin{itemize}
  \item Find different companies with similar marketing copy
  \item Automatically construct high-performing subject lines
  \item Many more...
  \end{itemize}
Try it yourself

- C implementation exists, but I recommend Python
  - *gensim* library: [https://radimrehurek.com/gensim/](https://radimrehurek.com/gensim/)
  - tutorial: [http://radimrehurek.com/gensim/models/word2vec.html](http://radimrehurek.com/gensim/models/word2vec.html)
  - webapp to try it out as part of tutorial
  - Pretrained Google News and Freebase models: [https://code.google.com/p/word2vec/](https://code.google.com/p/word2vec/)
  - Only takes 10 lines of code to get started!
Thanks for listening!

- Many thanks to:
  - Data Science Association and Level 3
  - Michael Walker for organizing

- Slides posted on [http://will-stanton.com/](http://will-stanton.com/)
- Email me at will@will-stanton.com