Week 6: Supervised and unsupervised applications

Text Analytics and Natural Language Processing

Instructor: Benjamin Batorsky

Final project **DUE 8/7 (11:59pm)**

- Submission format
  - Github repo or set of files
    - Code for exploration, processing, modeling (notebooks, scripts)
    - Data files
    - Write-up
      - All project participants
      - Link to repo (make it public!) or reference to specific files
      - All sections from outline, but expanded with results, conclusions, etc
  - MUST include a runnable version
    - Example: Subsample of full data, simplified code
- On Canvas: Submit write-up OR sign-up for a presentation

Presentations (Class time 8/5)

- 10-15 minutes
- Must include all sections from outline (just like write-up!)
- Will need to submit slides by 8/3
- Notify us ahead of time, but ONLY if you're on track to submit slides!
  - If you don't, you'll be taking a slot from another student!
- Takes the place of write-up, but still document your code/submit runnable example

Exciting stuff: Tentative speaker schedule

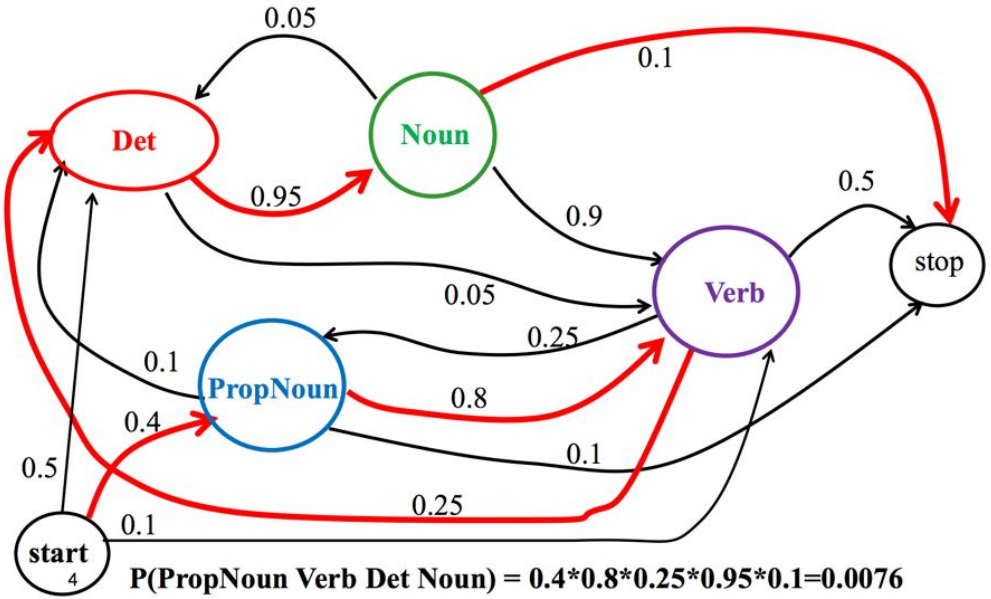
- Week 6:
  - **Wednesday @ 7-8pm: Mady Mantha, Senior Technical Evangelist, Rasa AI**
  - New readings will be posted today/tomorrow!
- Week 7:
  - **Monday @ 6:30-7:30pm**: Maryam Jahanshahi, Research Scientist, TapRecruit

What was the major development in the 1990s-2000s that led to rapid development in statistical language models?

- Growth of the field of linguistics
- Increased focus on semantic-driven processing
- Massive expansion in availability of data
- New statistical techniques

1990s-2000s: Shift towards statistical NLP

- Increasing attention on probabilistic models of language
  - Visual of parsing trees with probabilities, Markov Models
  - Probabilities of certain words appearing, the precursor to modern language models
- Also newly available data sources for developing/testing
  - WordNet (https://wordnet.princeton.edu/)
    - Inventory of English words and their relationships
  - Penn TreeBank (https://catalog.ldc.upenn.edu/LDC99T42)
    - Text parsed for part of speech, dependencies, etc
  - Ontonotes (https://catalog.ldc.upenn.edu/LDC2013T19)
    - Text labelled with entities (e.g. places and persons)
  - The internet!



Midterm review (short answer)

What was the major development in the 1990s-2000s that led to rapid development in statistical language models?

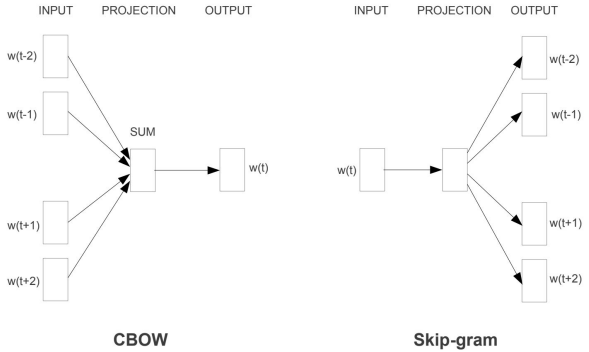
- Growth of the field of linguistics
- Increased focus on semantic-driven processing
- Massive expansion in availability of data
- New statistical techniques

What is one of the major *differences* between GloVe embeddings and an embedding layer in a neural network?

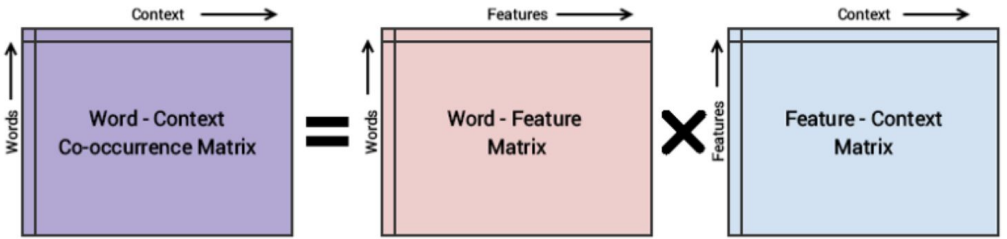
1. GloVe embeddings have more dimensions
2. How each is trained
3. Speed of training
4. The embedding layer includes context

Review of what we've covered

- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
- 2000s: Neural Language models
- 2008: Multi-task learning
- 2013: Word embeddings
- 2014: Expansion of Neural models
- 2015: Attention
- 2018 and beyond: Language model advancements



https://arxiv.org/pdf/1301.3781.pdf



Conceptual model for the GloVe model's implementation

Implementing Deep Learning Methods and Feature Engineering for Text Data: The GloVe Model

What is one of the major differences between GloVe embeddings and an embedding layer in a neural network?

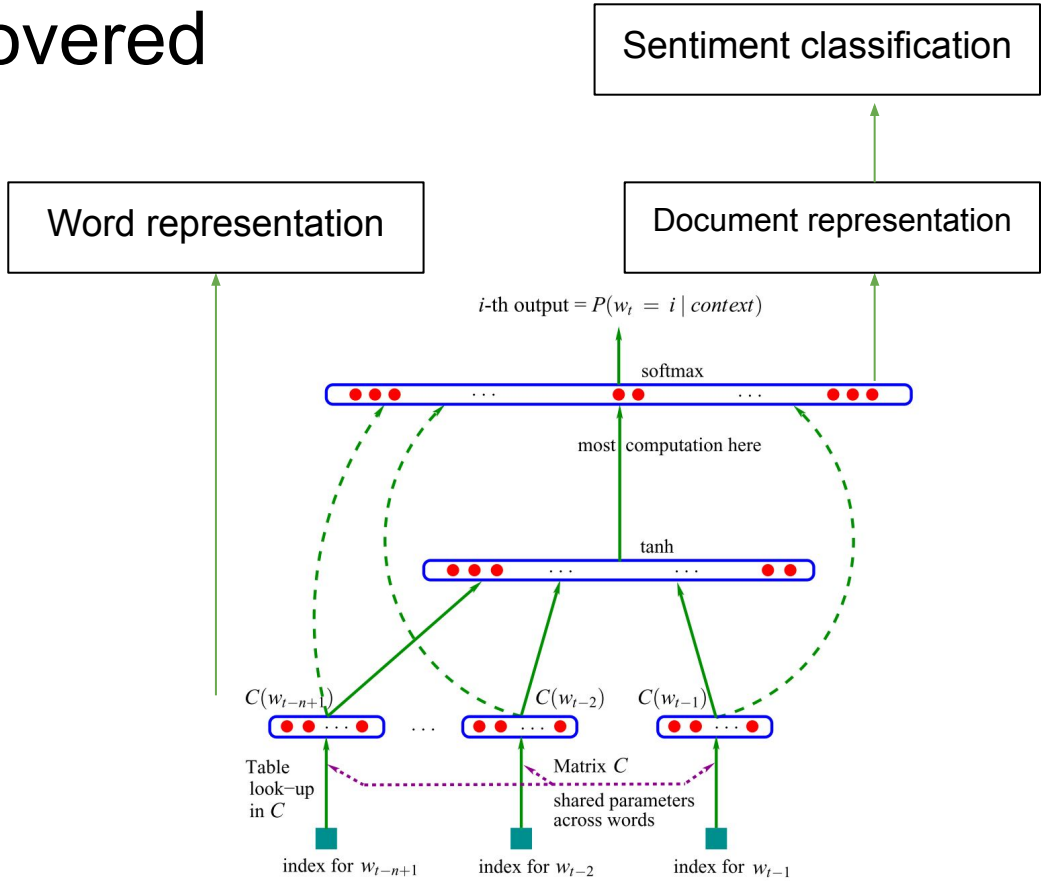
1. GloVe embeddings have more dimensions
2. How each is trained
3. Speed of training
4. The embedding layer includes context

What best describes “fine-tuning” in the context of pre-trained language models?

1. Changing the tokenization strategy for the model in order to better fit the task
2. Using pre-trained representations as input to a task-specific model
3. Adjusting the train/validation/test split to better reflect the pre-trained model
4. Adjusting the model hyperparameters to improve performance

Review of what we've covered

- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
- 2000s: Neural Language models
- 2008: Multi-task learning
- 2013: Word embeddings
- 2014: Expansion of Neural models
- 2015: Attention
- 2018 and beyond: Language model advancements



A Neural Probabilistic Language Model
(https://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf)

What best describes “fine-tuning” in the context of pre-trained language models?

1. Changing the tokenization strategy for the model in order to better fit the task
2. **Using pre-trained representations as input to a task-specific model**
3. Adjusting the train/validation/test split to better reflect the pre-trained model
4. Adjusting the model hyperparameters to improve performance

Distilling text vectors with matrix factorization

X ~ UV

Where:

X is an NxD matrix

U is an Nxd matrix

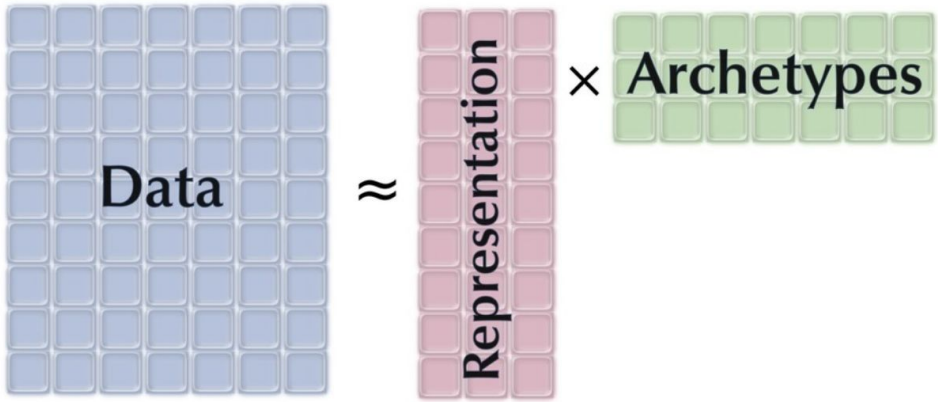
V is an dxD matrix

(Note: Matrix notation can be read as # of rows x # of columns)

What is D?

What is d?

Visual of factorization



Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{D} \text{Loss}\left(X_{ij}, (UV)_{ij}\right)$$

Leland McInnes: Bluffer's Guide to Matrix Factorization
https://www.youtube.com/watch?v=9iol3Lk6kyU
https://speakerdeck.com/lmcinnes/a-guide-to-dimension-reduction

Bi-directional LSTMs

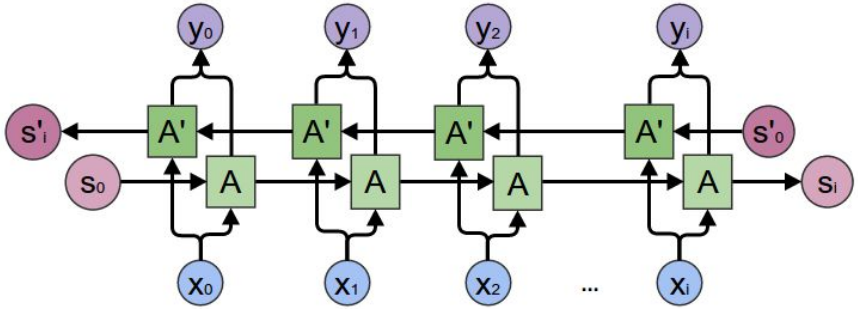
What is the objective of a language model?

If this Bi-directional LSTM was being trained as a language model, what would y be?

What is X?

How does this relate to ELMo's architecture?

Bi-directional LSTM



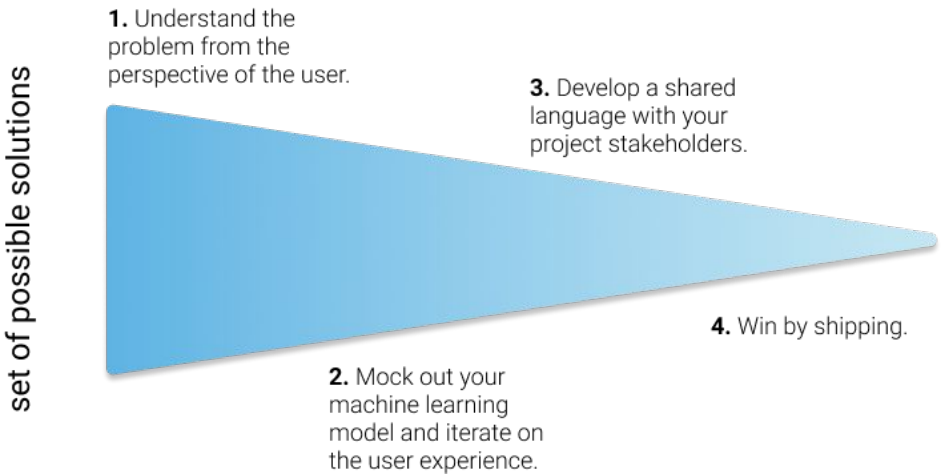

```
LSTM(embedding_dim, hidden_dim, n_layers,
     dropout=dropout_prob, batch_first=True,
     bidirectional=True)
```

http://colah.github.io/posts/2015-09-NN-Types-FP/

Finishing up week 5’s material

Scoping an NLP project

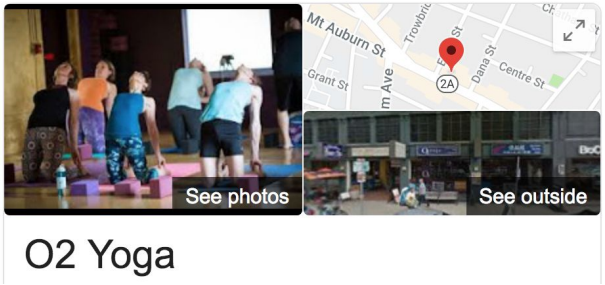
- Setup
  - Understand the problem
  - Inventory of solutions
    - Impact
    - Feasibility
    - Requirements
  - Setting up code base
- Data collection/labelling/sourcing
- Model exploration
- Deployment
- (throughout) Debugging and testing



https://www.jeremyjordan.me/ml-requirements/

Unsupervised use-case: Customer segmentation

- Small/Mid-sized businesses that straddle multiple categories
- Customer questions
  - Sales: “Which businesses are similar to this lead?”
  - Marketing: “How do we better personalize ad campaign messaging?”
  - Guidance: “How do we surface the most relevant recommendations to customers?”



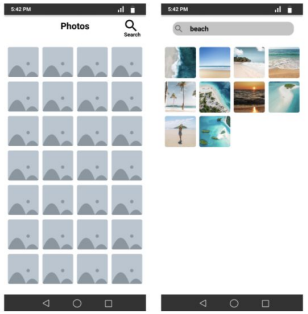
“...offers classes 7 days a week. Our vegan cafe opened in July of 2013... We also have a retail store selling a limited selection of US-made yoga gear...peruse the retail, enjoy the cafe, or get a massage with one of the body workers in the Wellness Center…”

Yoga studio, cafe AND retail?!

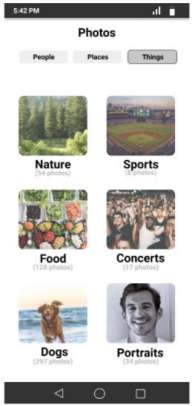
Understanding the problem

Review filtering mechanism

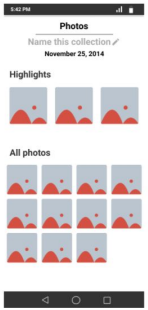
- Who are the end users?
  - Viewers
  - Creators
  - Internal
- Is this an NLP problem?
  - Are there text features?
  - Are the text features informative?
- What are some possible solutions?



Text query



Topic-centric query



Group photos and select highlights

https://www.jeremyjordan.me/ml-requirements/

Understanding the problem: Customer segmentation

- Who are the end users?

- Is this an NLP problem?
  - Are there text features?
  - Are the text features informative?

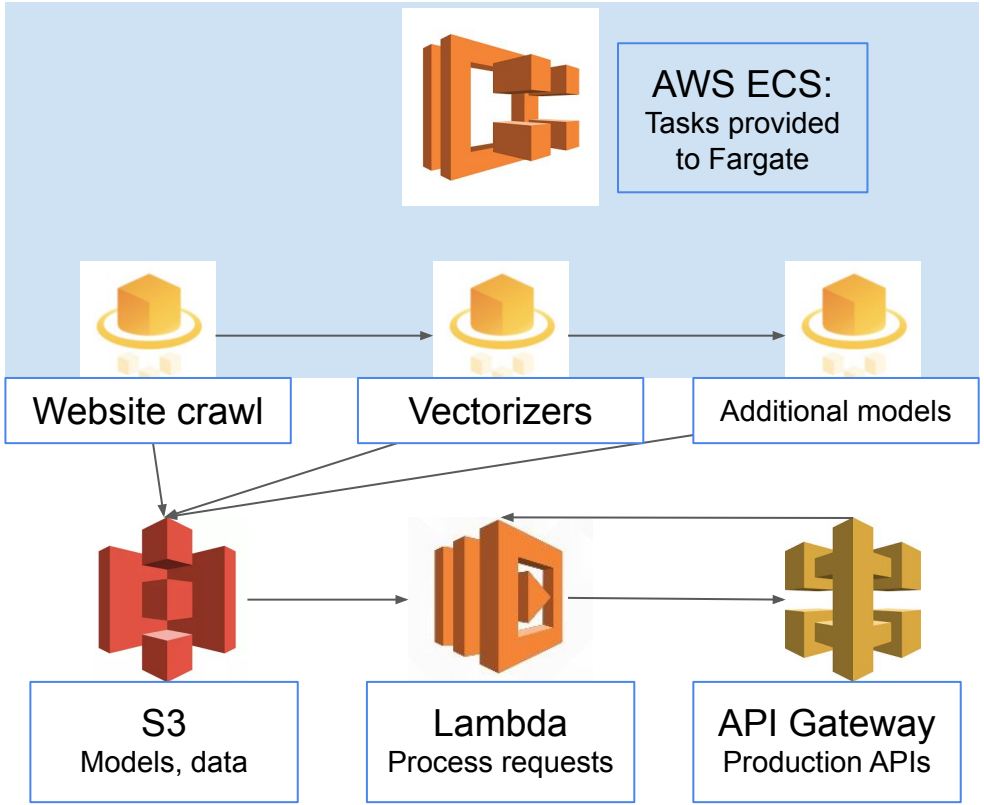
- What are some possible solutions?

Possible solutions

Solution	Data source	Metric of interest	Feasibility	Impact
Clustering based on text features	Social media posts	Clustering metric (inertia) Density of clusters Weak supervised metric	Availability of posts High-frequency low-frequency Acquisition Noise-to-signal	Search for business, get others in cluster Descriptive “segments” Important “features”
Combine structure + unstructured	Social media structured info			
	Reviews	Topic, sentiment		

Implemented solution: Segmentation Pipeline

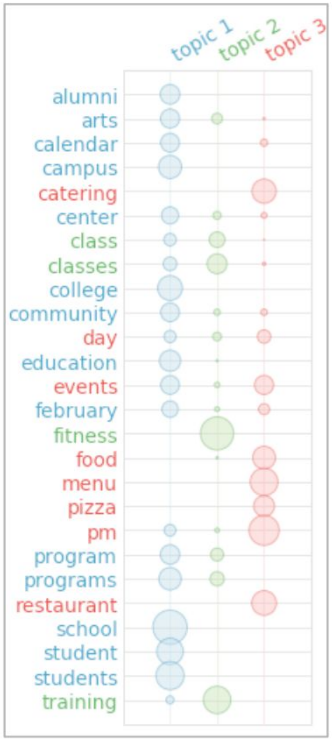
- Data ingestion
  - Daily crawl of new websites
- Preprocessing
  - Stripping punctuation
  - Number/URL tokens
- Vocabulary generation
  - Gensim phrase identification
    - E.g. “ice”, “cream” converted to “ice cream”
  - Vocabulary generated from presence across industry
- Information extraction/vectorization
  - Term Frequency-Inverse Document Frequency

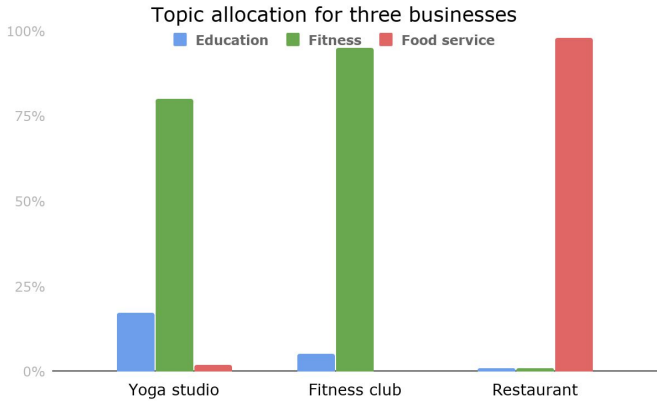


Segmentation Pipeline (continued)

- Topic modelling
  - Non-negative matrix factorization (NMF)
  - Identify interpretable topics and relevant words
- Output
  - TF-IDF + NMF transform on processed website text
  - Calculate cosine similarity across all businesses
  - Identify populations ranked by similarity
- Feedback
  - Feedback from internal users on similarity of results
  - Feedback used to reweight pairwise similarity and train new models

Product similarity

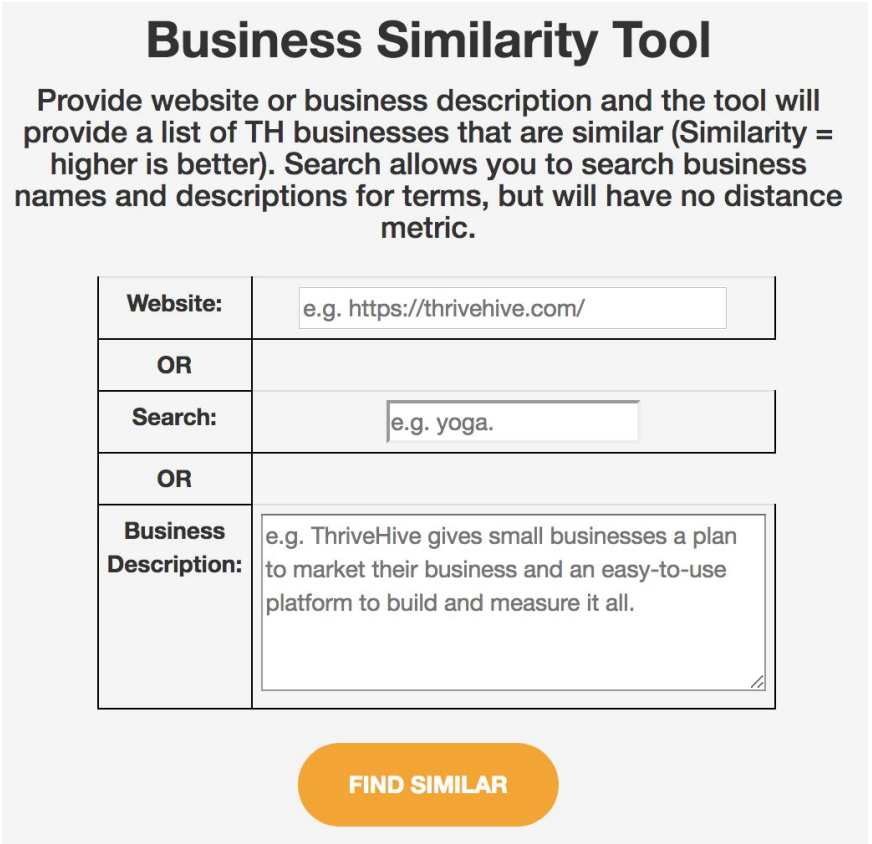




Circles are sized according to "relevance" to each topic

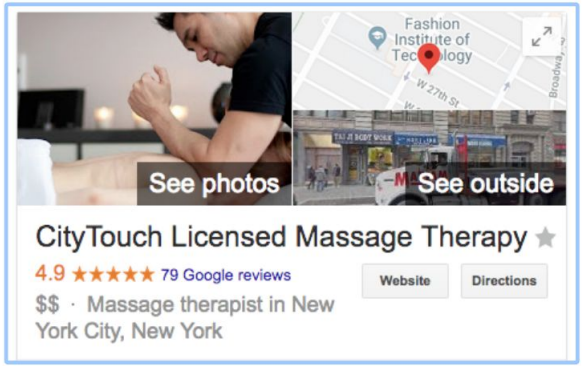
Personalized targeting for marketing/sales

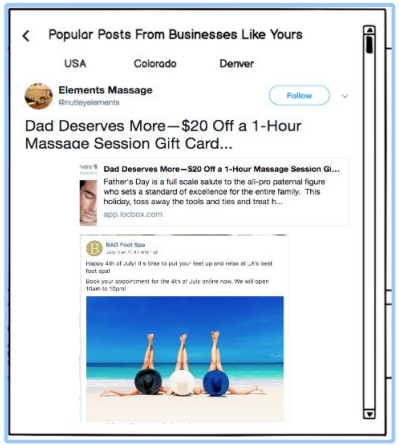
- Interactive web application over NLP API
- Three options
  - Website - On-request text extraction
  - Description - Treated as document
  - Keyword search - Simple regex search
- Similar business plus success metrics
- Sales
  - Regular usage during lead calls/prospecting
  - Source of feedback for similarity
- Marketing
  - Ad-hoc building of targeting populations
  - “Success stories” for similar businesses



Personalized guidance for businesses

- Content generation a major issue for business owners
  - “What do I talk about?”
- Surface top-performing posts from similar businesses
- Initial testing: 80% of content identified as relevant
  - 75% of content identified as relevant claimed to “give ideas for new content”





Implemented solution

Solution	Data source	Metric of interest	Feasibility	Impact
Customer “similarity” score	Website text	Customer feedback (internal/external)	High: Limited modelling required, light-weight process	High: Enables comparing business without relying on categories

Notebook use-case: Recommending similar movies

Goal: Can we recommend similar movies to a particular movie or set of movies?

Solution	Data source	Metric of interest	Feasibility	Impact
	Movie reviews IMDB all information User review patterns			
Content-based recommendation system	Plot synopsis Scripts	Compare content-based vs user-based	Size of the data Availability (new	1.0 -> same genre/same director

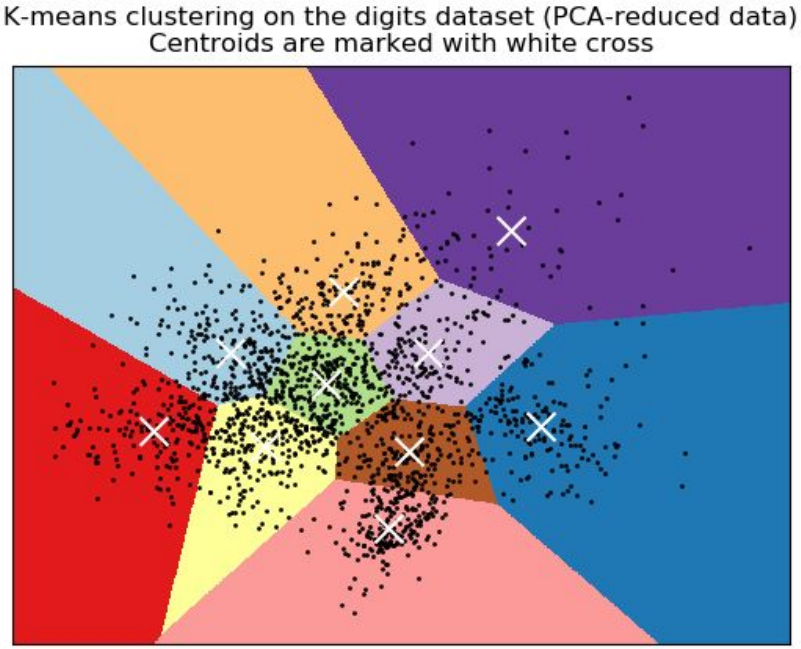
Notebook use-case: Categorizing movies

Goal: Can we create informative groupings of movies?

Solution	Data source	Metric of interest	Feasibility	Impact

K-means clustering

- Clustering: Unsupervised approach
  - “Identify patterns in data”
- K-means
  - Initialize K center points
  - “Draw” segment around center point
  - New center = center of the segment
  - Repeat until no significant change
- Optimizes for “inertia”
  - Sum of squared distances from center of cluster



https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html

Notebook use-case: Identifying actors/locations

Goal: Can we extract the actors and locations mentioned in reviews?

Solution	Data source	Metric of interest	Feasibility	Impact
Pre-trained model-based NER	Reviews, tagged with entities	Number of actors and locations Accuracy, Precision, recall, f1-score	Might be hard to have a tagged set Pre-trained might not work with actor names	Model-based, flexible
Have an inventory of actors and locations, pattern matching	Inventories Reviews	Recall	Potential to get complicted	Inflexible

Named-Entities

- Named-entity: A real-world named object (e.g. person, place, organization)
  - New York City is different than just an assembly of three words “new”, “york” and “city”
- To identify these
  - Dictionaries
  - Pattern-matching
  - Models

In June 2020 [DATE], I took a course at Harvard Extension School [ORG] in Cambridge [LOC].

Why is NER difficult?

- Ambiguity about entity boundaries
- Good performance typically requires a lot of quality labelled data
  - CoNLL 2003 task: ~21k labelled sentences
  - OntoNotes 5.0: 1.4 M articles
- Task complexity
  - Each token needs one tag and one entity type, decision between X tags * Y types
- Performance evaluation
  - What if the method achieves partial match?
  - Are we interested in performance on new data? Out-of-inventory entities?

Harvard [U-ORG (or B-ORG)]

Extension [B-ORG (or I-ORG)]

School [L-ORG]

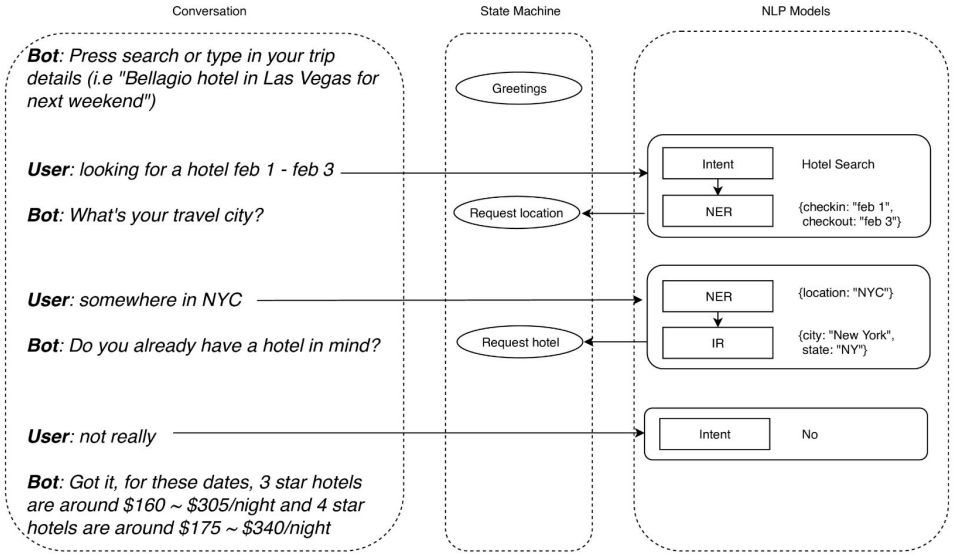
in [O]

Cambridge [U-LOC]

CoNLL 2003: https://www.aclweb.org/anthology/W03-0419/
OntoNotes 5.0: https://catalog.ldc.upenn.edu/LDC2013T19

Why is NER important?

- Named-entities are fundamentally different from other tokens
  - “New York City” != “new” + “york” + “city”
- Having a “complete” inventory is extremely rare
- NER is part of many Machine Learning applications
  - Content recommendation/Search
  - Chatbots
  - Translation



https://www.groundai.com/project/real-world-conversational-ai-for-hotel-bookings/1

NER use-case

- Dataset: Full-text for 22k court cases
- Current NER approach:
  - Regular expressions using inventories
    - Inventories likely to be incomplete and spellings/mentions likely to vary
  - Manual review and hand-labelling
    - Extremely time-consuming
    - Needs to be repeated for new data/new questions
- Proposed approach:
  - Model-based NER
  - Compare versus current approach
    - Performance on new data
    - Performance on entities out-of-inventory

What are the regulatory agencies involved in food safety enforcement?

"China FDA brought a suit against…"

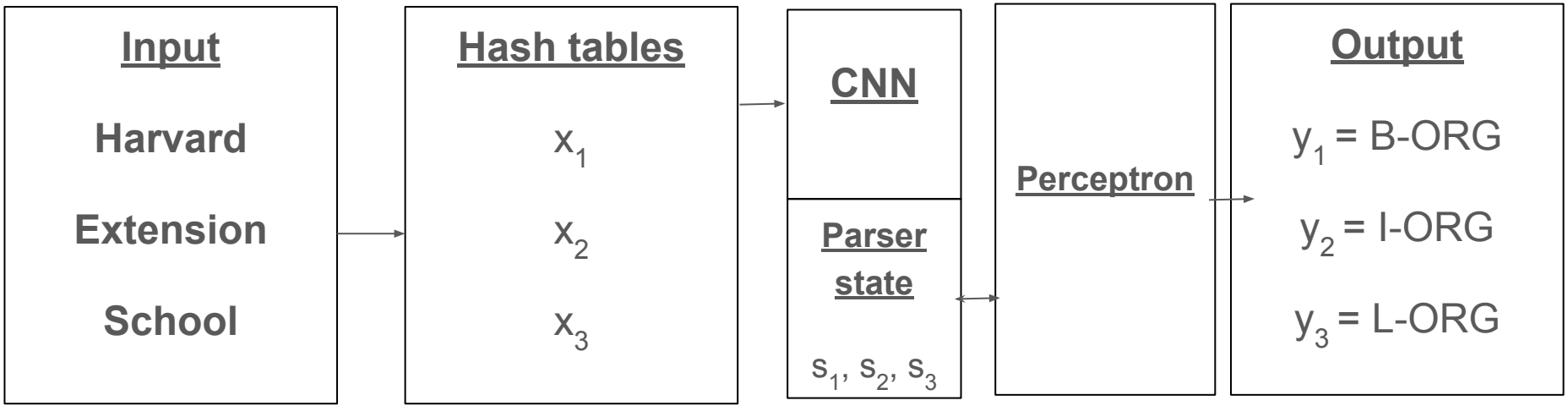
What types of products do they oversee?

"...for selling tainted pork products…"

What is their jurisdiction?

"...in Hangzhou Province."

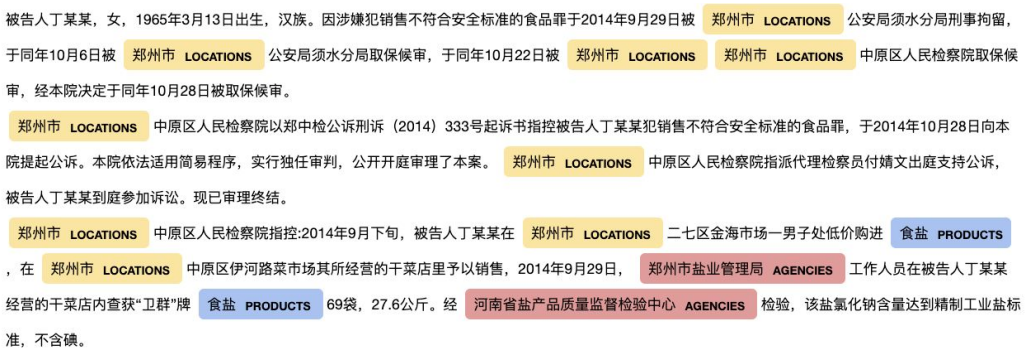
SpaCy's NER model



Matthew Honnibal on spaCy's NER model: https://spacy.io/universe/project/video-spacys-ner-model

Our data and inventories

- Dataset
  - Full-text for 22k court cases prosecuted by government agencies
- Inventories
  - Products
    - Sourced from our database of food inspection results
  - Agencies
    - Curated as part of the previous manual process
  - Locations
    - A selection of cities and all prefectures and provinces in China



Using SpaCy's PhraseMatcher

- PhraseMatcher
  - Given a pattern, extracts matches and can pass to callback function
    - (Match id, start token, end token)
- Custom callback
  - Default entity attribute can't handle overlap
  - Used custom attribute for Doc objects
    - Doc._.entities
  - For overlapping entities
    - If different types, choose higher priority one
    - If same types, go with longer entity

```
# intialize model
nlp = Chinese()
# initialize the matcher with model vocab
matcher = PhraseMatcher(nlp.vocab)
# add entity inventory as Doc objects from model
or i, c in enumerate(ENTITY_TYPES):
    matcher.add(c, add_entity, *parsed_ents[i])
```

Callback function

Constructing training and test datasets

- Year-split
  - How well will the model perform on future court case data?
  - Train on cases before 2017, test on after
  - Baseline: Inventory with only entities before 2017
- Excluded entities
  - How well does the model identify entities it hasn't seen?
  - Train on cases with 30% of entities from each inventory removed
  - Baseline: Inventory with remaining 70%

Method	Train		Test	
	Docs	Unique entities	Docs	Unique entities
Year-split	13,501	6,707	7,722	4,378
Excluded entities	14,859	8,548	6,364	2,564

Training the model

- Available hyperparameters
  - Dropout
  - Batch size
  - Optimizer
- Outputs loss with each epoch
  - Based on the model predicted tags (e.g. entity/non-entity)*
- Important to note
  - Plateauing: I haven't seen much movement in loss after 15-20 epochs
  - If updating: Not enough to just put new examples, model likely to forget what it's learned

```
nlp_model = Chinese()
ner = nlp_model.create_pipe('ner')
nlp_model.add_pipe(ner)
for l in labels:
    ner.add_label(l)
optimizer = nlp_model.begin_training()
sizes = compounding(1.0, 4.0, 1.001)
epoch = 15
for itn in range(epoch):
    random.shuffle(train_data)
    batches = minibatch(train_data, size=sizes)
    for batch in batches:
        texts, annotations = zip(*batch)
        nlp_model.update(texts, annotations, sgd=optimizer, drop=0.35, losses=losses)
    print("Losses", losses)
```

Losses {'ner': 169404.9403350675}

Losses {'ner': 79686.03164099755}

*More details on NER model loss: https://github.com/explosion/spaCy/issues/3360

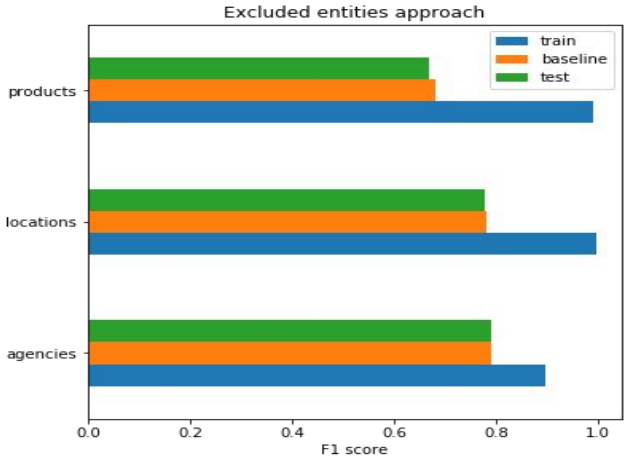
Scoring the result

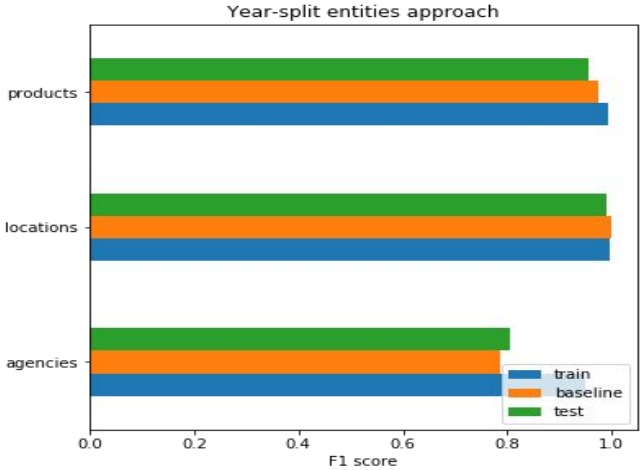
- Built-in SpaCy Scorer
  - Compare model parsing result to “gold standard”
  - Provides entity-level precision (“p”), recall (“r”) and F1-score (“f”)
  - Adapted to get baseline performance
- How does this model perform on data it has seen?
  - F1-score on training data
- How does this model perform on unseen data?
  - F1-score on test data
- How well does this model identify entities it hasn’t seen?
  - Recall of excluded entities

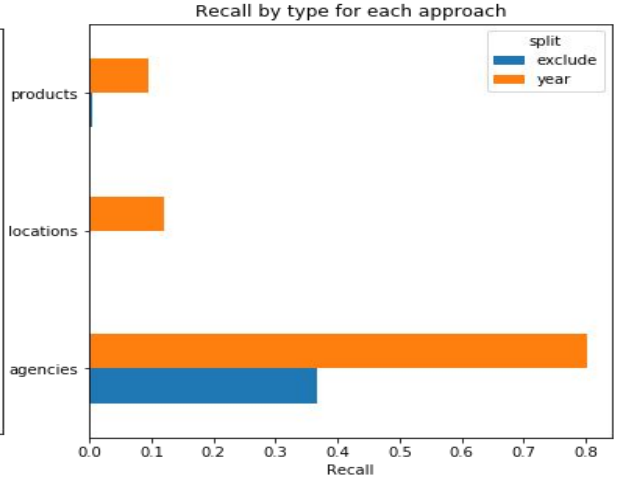
```
scorer = Scorer()
for doc, annot in test_data:
    doc_to_test = full_model(doc)
    gold_text = nlp(doc)
    gold = GoldParse(gold_text, entities=annot.get("entities"))
    scorer.score(doc_to_test, gold)
```

```
{'uas': 0.0,
 'las': 0.0,
 'las_per_type': {'': {'p': 0.0, 'r': 0.0, 'f': 0.0}},
 'ents_p': 97.69166443143628,
 'ents_r': 55.78143651884051,
 'ents_f': 71.0141561506281,
 'ents_per_type': {'agencies': {'p': 88.99794567450354,
   'r': 70.3917674670518,
   'f': 78.60887096774192},
  'locations': {'p': 99.61005302327793,
   'r': 63.799037524366476,
   'f': 77.7805627500673},
  'products': {'p': 97.63231014366795,
   'r': 51.04266349059916,
   'f': 67.03768671561359}},
 'tags_acc': 0.0,
 'token_acc': 100.0,
 'textcat_score': 0.0,
 'textcats_per_cat': {}}
```

Model-based NER has better performance on agencies than other entity types







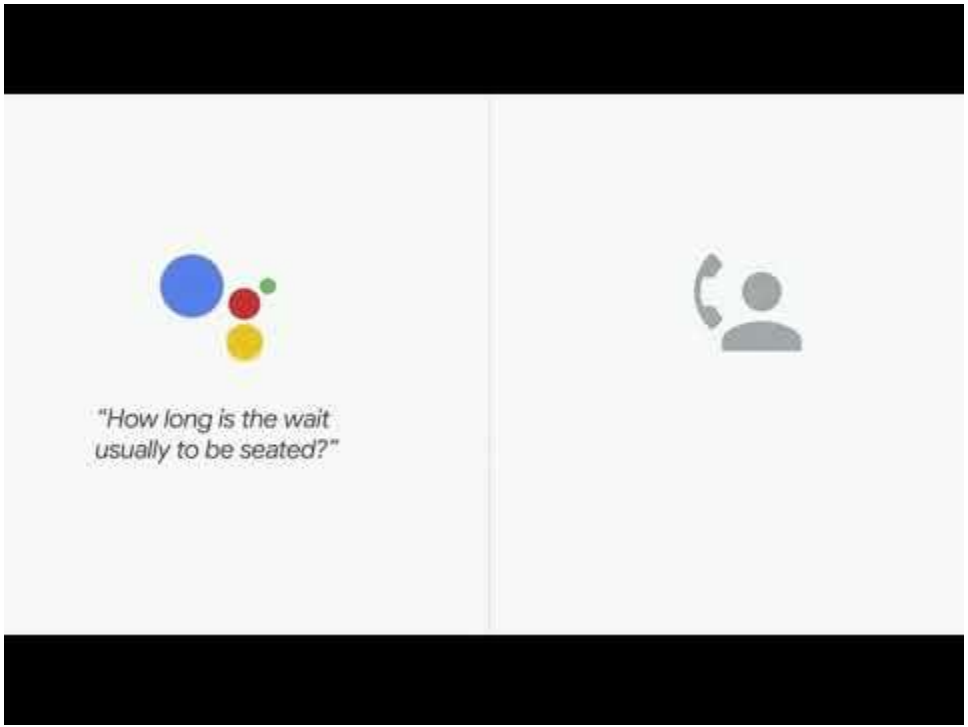
NER model for identifying actors/locations

Language generation - Cool stuff

Write with Transformer from HuggingFace:
https://transformer.huggingface.co/doc/distil-gpt2

Language generation - useful stuff

Google Assistant making a reservation



Models “learning” language

100 epochs

tyntd-iafhatawiaoihrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e plia tklrgd t o idoe ns,smtt h ne etie h,hregtrs nigtike,aoaenns lng

700 epochs

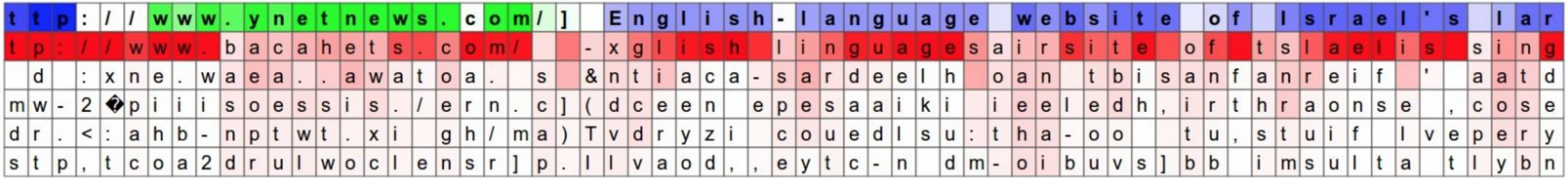
Aftair fall unsuch that the hall for Prince Velzonski's that me of her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort how, and Gogition is so overelical and ofter.

2000 epochs

"Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had oftened him. Pierre aking his soul came to the packs and drove up his father-in-law women.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Neurons as feature learners



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Chatbot example with Rasa

Natural Language Understanding (NLU)

```
"I am looking for a Mexican restaurant in the center of town"
```

```
{
  "intent": "search_restaurant",
  "entities": {
    "cuisine" : "Mexican",
    "location" : "center"
  }
}
```

Natural Language Generation (NLG)

Template-based

intents:

- greet

responses:

- utter_greet
  - text: “Hello {name}”

Model-based (external to Rasa)

FastAPI call to your NLG model

Adapting our sentiment model for NLG