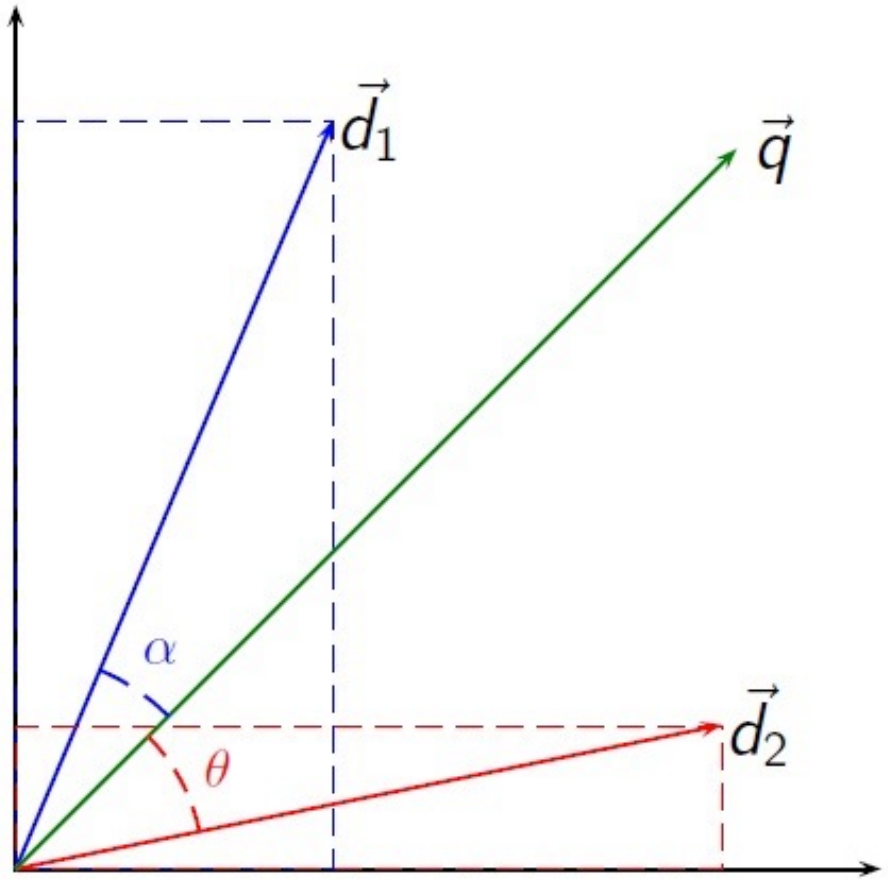
Lecture 9: Vector Semantics

Mark Hasegawa-Johnson

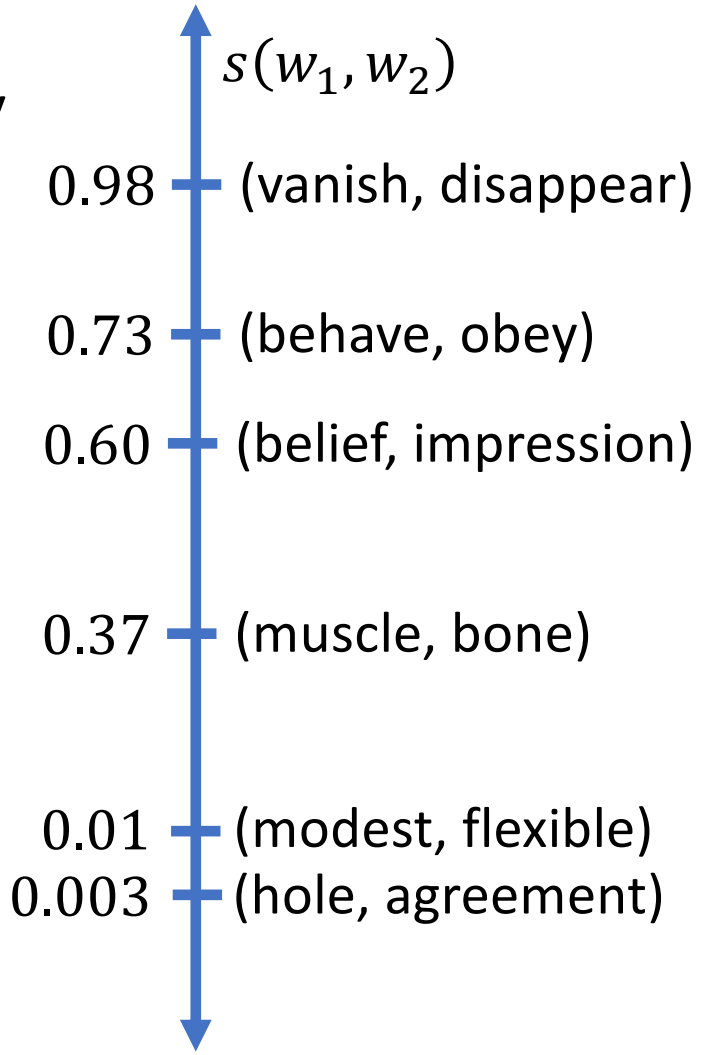


Outline

- Vector semantics: CBOW and skip-gram
- Calculating probabilities using the logistic sigmoid
- Dot-product similarity
- Noise contrastive estimation (NCE)
- Visualizations

Synonymy and similarity

- Words are "synonyms" if they have exactly the same meaning.
- No words ever have ***exactly*** the same meaning, so no two words are ever exactly synonyms.
- We prefer to talk about word similarity, $0 \leq s(w_1, w_2) \leq 1$
  - $s(w_1, w_2) = 1$: w_1 and w_2 are perfect synonyms. Never happens in practice, but sometimes close.
  - $s(w_1, w_2) = 0$: w_1 and w_2 are completely different.



Skip-gram: “You shall know a word by the company it keeps”

- The key idea of skip-gram is that “You shall know a word by the company it keeps” (J.R. Firth, 1957)
- The words “vanish” and “disappear” should be considered similar if they can occur in the same contexts:

“The ship will vanish into the mists.”

”The ship will disappear into the mists.”

Review: Naïve Bayes: the "Bag-of-words" model

We can estimate the likelihood of an e-mail by pretending that the e-mail is just a bag of words (order doesn't matter).

With only a few thousand spam e-mails, we can get a pretty good estimate of these things:

- $P(W = \text{"hi"} | Y = \text{spam})$, $P(W = \text{"hi"} | Y = \text{ham})$
- $P(W = \text{"vitality"} | Y = \text{spam})$, $P(W = \text{"vitality"} | Y = \text{ham})$
- $P(W = \text{"production"} | Y = \text{spam})$, $P(W = \text{"production"} | Y = \text{ham})$

Then we can approximate $P(X|Y)$ by assuming that the words, W, are ***conditionally independent of one another given the category label***:

$$P(X = x | Y = y) \approx \prod_{i=1}^{n} P(W = w_i | Y = y)$$

Similarity: The Internet is the database

Similarity = words can be used interchangeably in most contexts

How do we measure that in practice?

Answer: extract examples of word w_1, +/- C words (C=2 or 3):

…hot, although iced **coffee** is a popular…

…indicate that moderate **coffee** consumption is benign…

…and of w_2:

…consumed as iced **tea**. Sweet tea is…

…national average of **tea** consumption in Ireland…

The words "iced" and "consumption" appear in both contexts, so we can conclude that $s(\text{coffee}, \text{tea}) > 0$. No other words are shared, so we can conclude $s(\text{coffee}, \text{tea}) < 1$.

Continuous Bag of Words (CBOW)

"Context bag of words" (CBOW) approximates each word's probability, in context, by the product of single-word context probabilities:

$$p(w_t|w_{t-3}, \dots, w_{t+3}) \approx \prod_{\substack{c=-3 \\ c\neq 0}}^{3} p(w_t|w_{t+c})$$

Using this model, two words are considered similar if they have similar CBOW probabilities:

$$p(\text{vanish}|\text{the ship will } - \text{into the mists}) \approx$$
$$p(\text{disappear}|\text{the ship will } - \text{into the mists})$$

Skip-gram

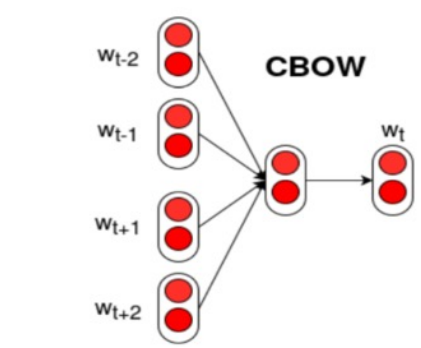
"Skip-gram" approximates the probability of the context given the center word:

$$p(w_{t-3}, \dots, w_{t+3} | w_t) \approx \prod_{\substack{j=-3 \\ j \neq 0}}^{3} p(w_{t+j} | w_t)$$

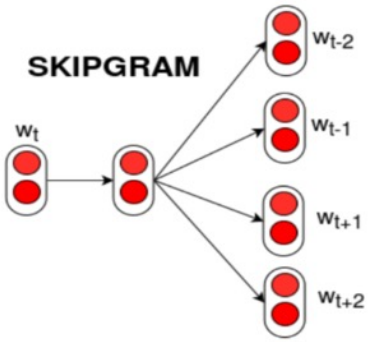
Using this model, two words are considered similar if they have similar skip-gram probabilities:

$$p(\text{the ship will } \_ \text{ into the mists} | \text{vanish}) \approx$$
$$p(\text{the ship will } \_ \text{ into the mists} | \text{disappear})$$

Neural net loss = negative log probability



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_t|w_{t+j})$$



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t)$$

Log probability is easier to differentiate than probability, so:

CBOW:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c, j\neq 0}^{c} \ln P\left(w_t|w_{t+j}\right)$$

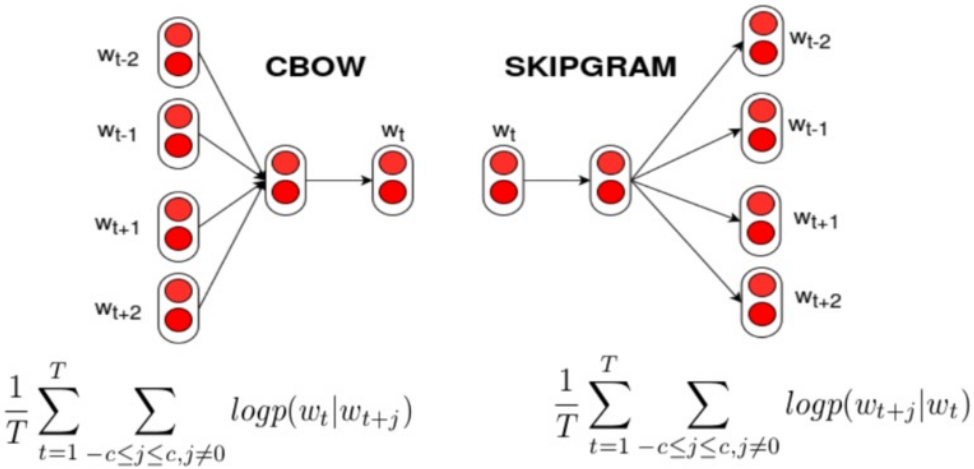
Skip-gram:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c, j\neq 0}^{c} \ln P\left(w_{t+j}|w_t\right)$$

Outline

- Vector semantics: CBOW and skip-gram
- Calculating probabilities using the logistic sigmoid
- Dot-product similarity
- Noise contrastive estimation (NCE)
- Visualizations

Neural net probability estimate?



Let's focus on skip-gram:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c, j\neq 0}^{c} \ln P\left(w_{t+j}|w_t\right)$$

Suppose we want to use a neural net to estimate $P\left(w_{t+j}|w_t\right)$. How can we do that?

The logistic sigmoid function

- Remember the logistic sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

- Since $0 < \sigma(z) < 1$, we can interpret $\sigma(z)$ as a probability.

$$P(w_{t+j}|w_t) = \sigma(z)$$

- Apparently, we want the logit, z, to be a function that measures the similarity or predictability of w_{t+j} and w_t:
  - If w_{t+j} never occurs in the context of w_t, we want $P(w_{t+j}|w_t) \approx 0$, so $z(w_{t+j}, w_t) \approx -\infty$
  - If w_{t+j} always occurs in the context of w_t, we want $P(w_{t+j}|w_t) \approx 1$, so $z(w_{t+j}, w_t) \approx +\infty$

Training the neural net

- Now we can write:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c, j\neq 0}^{c} \ln \sigma(z(w_{t+j}, w_t))$$

- For any pair of words w_1 and w_2, the gradient is:

$$\frac{\partial \mathcal{L}}{\partial z(w_1, w_2)} = -\frac{1}{T}\sum_{w_t = w_2}\sum_{w_{t+j} = w_1} (1 - \sigma(z(w_1, w_2)))$$

- The more often w_1 and w_2 occur in context, the more that gradient descent will push $z(w_1, w_2) \to \infty$

Outline

- Vector semantics: CBOW and skip-gram
- Calculating probabilities using the logistic sigmoid
- Dot-product similarity
- Noise contrastive estimation (NCE)
- Visualizations

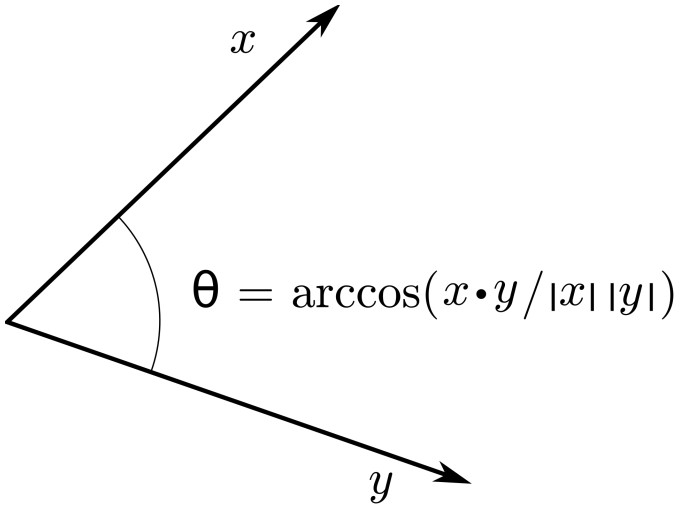
Dot product between vectors

The dot product between two vectors is:

$$\boldsymbol{v}_1^T \boldsymbol{v}_2 = |\boldsymbol{v}_1||\boldsymbol{v}_2| \cos\theta$$

The dot product is a pretty good measure of the similarity between the vectors.

- If $\theta < 90$ degrees, then $\boldsymbol{v}_1^T \boldsymbol{v}_2 > 0$
- If $\theta = 90$ degrees, then $\boldsymbol{v}_1^T \boldsymbol{v}_2 = 0$
- If $\theta > 90$ degrees, then $\boldsymbol{v}_1^T \boldsymbol{v}_2 < 0$



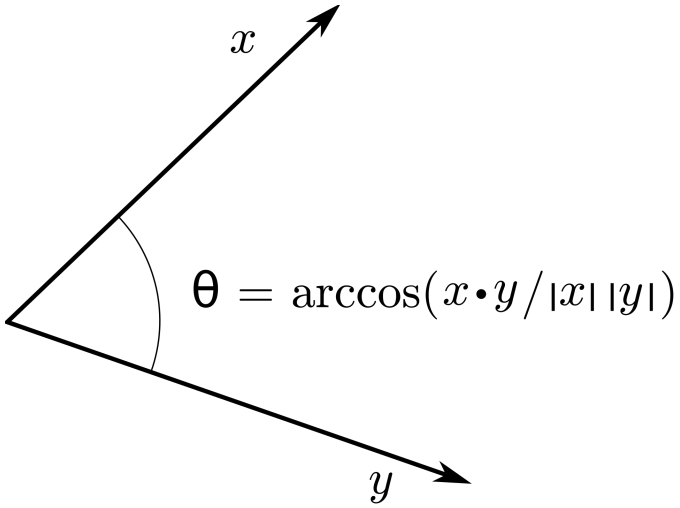
Words as vectors

- Suppose that, for every word in the dictionary w_1, we train a vector $\boldsymbol{v}_1$
- The similarity between two words should be measured by the dot product between their vectors:

$$z(w_1, w_2) = \boldsymbol{v}_1^T \boldsymbol{v}_2$$

- The probability of one word, given the other, is the sigmoid of their dot product:

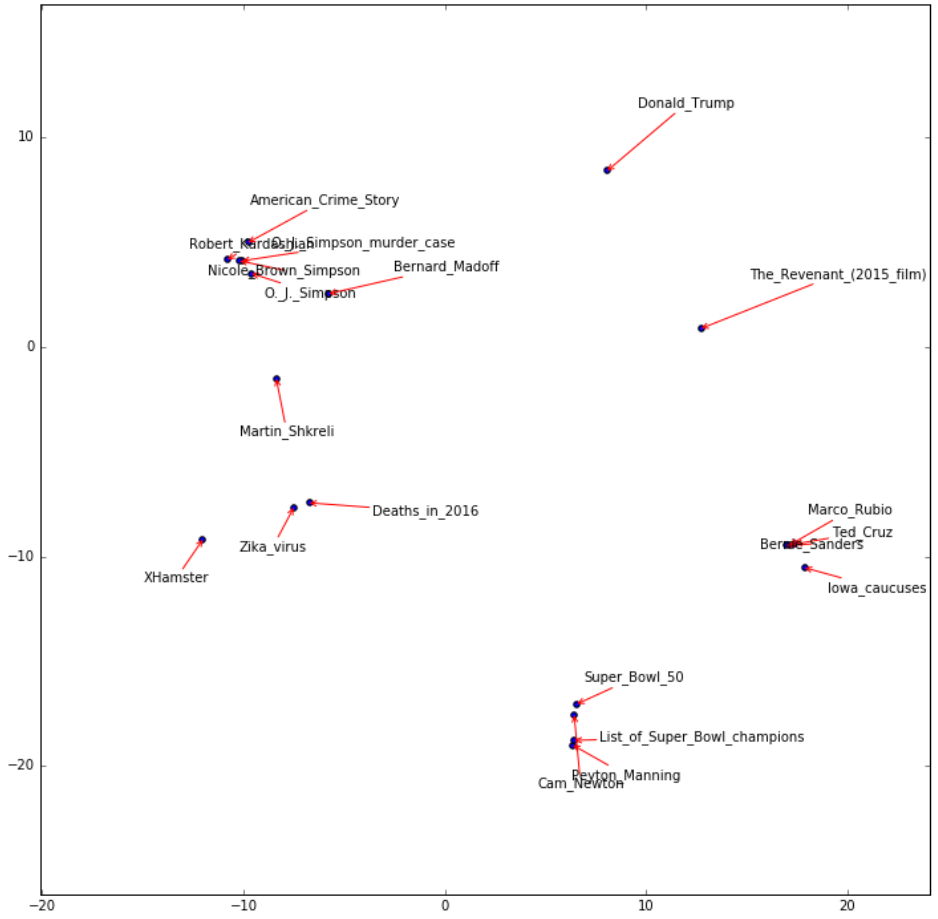
$$P(w_1|w_2) = \frac{1}{1 + e^{-\boldsymbol{v}_1^T \boldsymbol{v}_2}}$$



Words as vectors

t-SNE projection of Word2vec embedding, ewulczyn 2016,

https://commons.wikimedia.org/wiki/File:2016_02_mini_embedding.png



Training the neural net

- Now we can write:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c,j\neq 0}^{c} \ln \sigma(\boldsymbol{v}_1^T \boldsymbol{v}_2)$$

- For every word w with vector $\boldsymbol{v}_c$ the loss gradient is:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = -\frac{1}{T}\sum_{w_t=w}\sum_{j=-c,j\neq 0} \left(1 - \sigma(\boldsymbol{v}_{t+j}^T \boldsymbol{v})\right)\boldsymbol{v}_{t+j}$$

Training the neural net

- Gradient descent updates $\boldsymbol{v}$ as:

$$\boldsymbol{v} \leftarrow \boldsymbol{v} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}}$$

$$= \boldsymbol{v} + \frac{\eta}{T} \sum_{w_t = w} \sum_{j=-c, j \neq 0}^{c} \left(1 - \sigma(\boldsymbol{v}_{t+j}^T \boldsymbol{v})\right) \boldsymbol{v}_{t+j}$$

- In other words, $\boldsymbol{v}$ becomes a weighted average of the vectors $\boldsymbol{v}_{t+j}$ for words w_{t+j} that occur in the context of w.
- The vectors start out random, then vectors that frequently co-occur get pushed together, to become more similar

Outline

- Vector semantics: CBOW and skip-gram
- Calculating probabilities using the logistic sigmoid
- Dot-product similarity
- Noise contrastive estimation (NCE)
- Visualizations

Training the neural net

- This formula has a problem:

$$\boldsymbol{v} \leftarrow \boldsymbol{v} + \frac{\eta}{T} \sum_{w_t = w} \sum_{j=-c, j \neq 0}^{c} \left(1 - \sigma(\boldsymbol{v}_{t+j}^T \boldsymbol{v})\right) \boldsymbol{v}_{t+j}$$

- The problem is that $\left(1 - \sigma(\boldsymbol{v}_{t+j}^T \boldsymbol{v})\right)$ is always positive
- Therefore, $\boldsymbol{v}$ gets gradually bigger and bigger – it never stops growing!!

Fixing the problem

To keep all the vectors from growing forever, we need to subtract something from the right-hand side:

$$\boldsymbol{v} \leftarrow \boldsymbol{v} + \frac{\eta}{T} \sum_{w_t = w} \sum_{j=-c, j \neq 0}^{c} \left(1 - \sigma(\boldsymbol{v}_{t+j}^T \boldsymbol{v})\right) \boldsymbol{v}_{t+j} - SOMETHING$$

…such that, on average, the update will be zero (keeping all vectors small).

- If we still want to say that

$$\boldsymbol{v} \leftarrow \boldsymbol{v} - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}}$$

- …then we need to add something to $\mathcal{L}$ whose derivative will be "SOMETHING"

Noise contrastive estimation

So far, we've said that the probability of $\boldsymbol{v}_{t+j}$ occurring in the context of $\boldsymbol{v}_t$ is:

$$P(\boldsymbol{v}_{t+j}|\boldsymbol{v}_t) = \sigma(\boldsymbol{v}_{t+j}^T \boldsymbol{v}_t)$$

For some randomly chosen "noise" sample, $\boldsymbol{v}_i$, the probability it DOESN'T occur in the context of $\boldsymbol{v}_t$ is:

$$1 - P(\boldsymbol{v}_i|\boldsymbol{v}_t) = 1 - \sigma(\boldsymbol{v}_i^T \boldsymbol{v}_t)$$

We can create a well-normalized loss function by adding these two log probabilities:

$$\mathcal{L} = -\ln \sigma\left(\boldsymbol{v}_{t+j}^T \boldsymbol{v}_t\right) - \ln\left(1 - \sigma(\boldsymbol{v}_i^T \boldsymbol{v}_t)\right)$$

Skipgram NCE

Skigram NCE uses the following update step:

1. Choose a pair $\boldsymbol{v}_t$ and $\boldsymbol{v}_{t+j}$ from the training data
2. Choose k different "noise" examples, $\boldsymbol{v}_i$, uniformly at random from the vocabulary
3. Update $\boldsymbol{v}_t$ according to

$$\mathcal{L} = -\ln \sigma\left(\boldsymbol{v}_{t+j}^T \boldsymbol{v}_t\right) - \frac{1}{k}\sum_{i=1}^{k} \ln\left(1 - \sigma(\boldsymbol{v}_i^T \boldsymbol{v}_t)\right)$$

$$\boldsymbol{v}_t \leftarrow \boldsymbol{v}_t - \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{v}}$$

Quiz

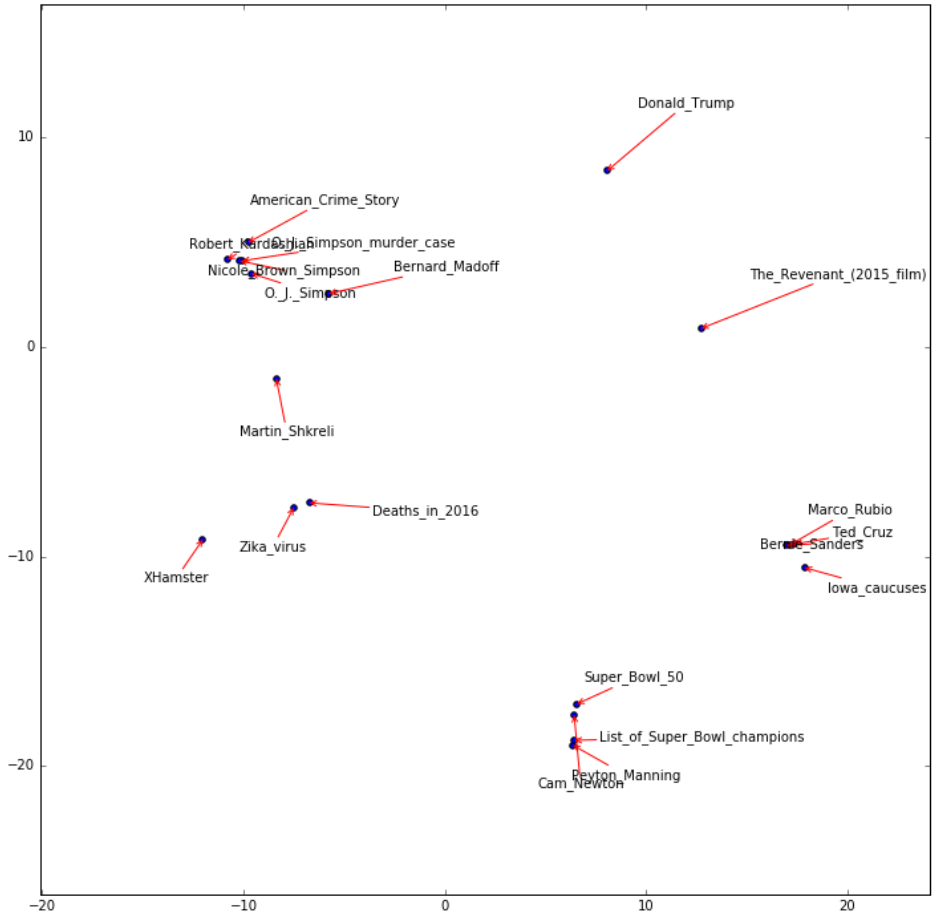
- Try the quiz!

Outline

- Vector semantics: CBOW and skip-gram
- Calculating probabilities using the logistic sigmoid
- Dot-product similarity
- Noise contrastive estimation (NCE)
- Visualizations

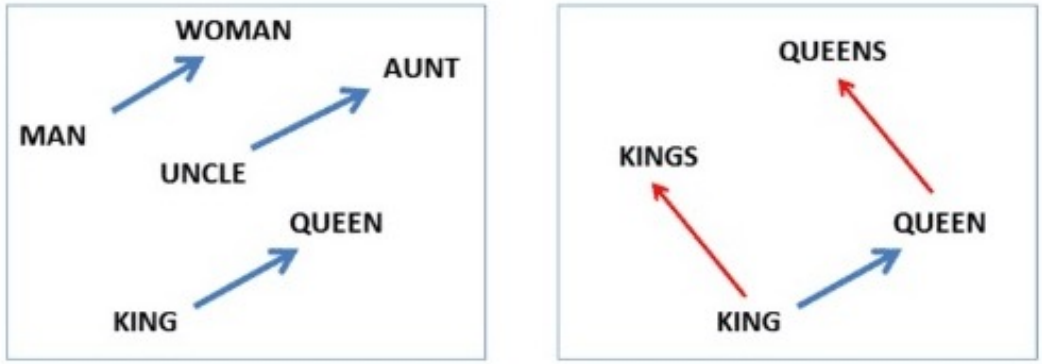
Visualizations: Vectors

One way to visualize the results is with a scatter plot of the trained vectors:



Visualizations: Relatedness

vec(“woman”) - vec(“man”) + vec(“king”) = vec(“queen”)



Christian S. Perone, "Voynich Manuscript: word vectors and t-SNE visualization of some patterns," in *Terra Incognita*, 16/01/2016, http://blog.christianperone.com/2016/01/voynich-manuscript-word-vectors-and-t-sne-visualization-of-some-patterns/.

Mikolov (2013) showed that word2vec captures similarity relationships among words. For example, the difference between the vectors for “woman” and “man” is roughly the same as the difference between the vectors for “queen” and “king.” Perone (2016) showed that this effect works differently depending on the training corpus: in his blog post, he looks at word relatedness in the 15th century Voynich manuscript.

Visualizations: Similarity

Another way to visualize the results is to see which words are nearest neighbors in the vector space:

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated capitulating

Table 6: Examples of the closest tokens given various well known models and the Skip-gram model trained on phrases using over 30 billion training words. An empty cell means that the word was not in the vocabulary.

Learning biased analogies from data

- It's useful that algorithms like word2vec learn appropriate analogies, like "kings → king as queens → queen."
- Unfortunately, it also learns biases. For example, it learns that some occupations are "male," and some are "female!"
- This paper demonstrated the problem, and proposed a really interesting solution:

  "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings," Bolukbasi et al., 2016

The Male-Female vs. Neutral-Specific Space

Here's the resulting 2D space, from Bolukbasi et al., 2016:

Summary

- CBOW:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c,j\neq 0}^{c} \ln P\left(w_t | w_{t+j}\right)$$

- Skip-gram:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c,j\neq 0}^{c} \ln P\left(w_{t+j} | w_t\right)$$

- Dot-product similarity:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c,j\neq 0}^{c} \ln \sigma(\boldsymbol{v}_1^T \boldsymbol{v}_2)$$

- Skip-gram noise contrastive estimation:

$$\mathcal{L} = -\ln \sigma\left(\boldsymbol{v}_{t+j}^T \boldsymbol{v}_t\right) - \frac{1}{k}\sum_{i=1}^{k} \ln\left(1 - \sigma(\boldsymbol{v}_i^T \boldsymbol{v}_t)\right)$$