DEEP LEARNING AND NLP

Word representation

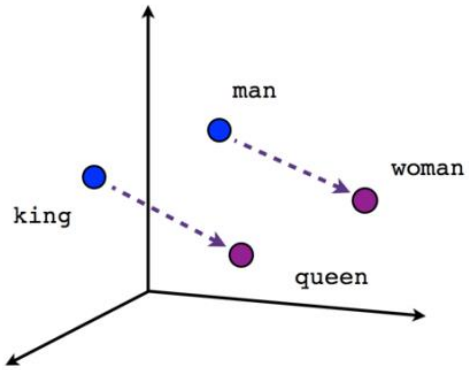
Traditional method

- Uses one hot encoding
- Each word in the vocabulary is represented by one bit position in a huge vector.
- Context information is not utilized

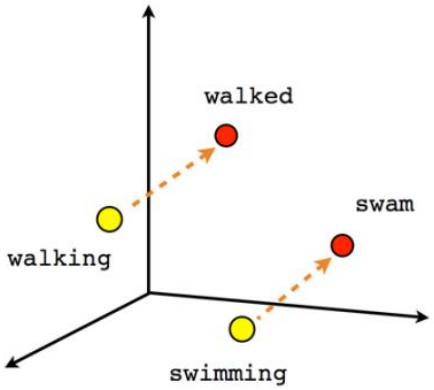
Word embeddings

- Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300)
- Unsupervised, built just by reading huge corpus

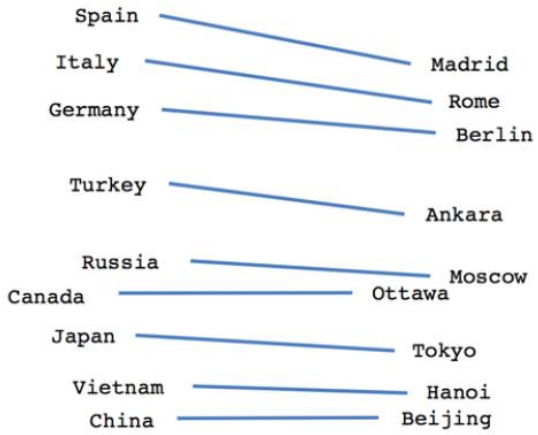
Examples



Male-Female



Verb tense



Country-Capital

Demo (http://w.elnn.kr/search/)

WORD EMBEDDING

http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/

Word Embedding

- A word embedding W:words$\rightarrow\Re^n$ is a paramaterized function mapping words in some language to high-dimensional vectors (perhaps 200 to 500 dimensions). For example, we might find:

$$W(\text{“cat”}) = (0.2,\ \text{-}0.4,\ 0.7,\ \ldots)$$

$$W(\text{“mat”}) = (0.0,\ 0.6,\ \text{-}0.1,\ \ldots)$$

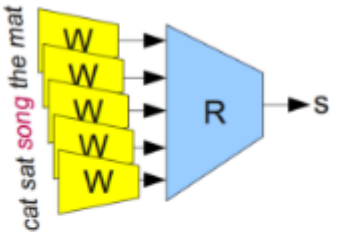
- Typically, the function is a lookup table, parameterized by a matrix, θ, with a row for each word: $W_\theta(w_n) = \theta_n$

Word Embedding Learning

- W is initialized to have random vectors for each word. It learns to have meaningful vectors in order to perform some task.
- Train a network for is predicting whether a 5-gram (sequence of five words) is 'valid.'
  - "cat sat on the mat" vs "cat sat **song** the mat"
- 5-gram -> (W, R) -> 'valid' vs 'broken'

$$R(W(\text{“cat”}),\ W(\text{“sat”}),\ W(\text{“on”}),\ W(\text{“the”}),\ W(\text{“mat”})) = 1$$

$$R(W(\text{“cat”}),\ W(\text{“sat”}),\ W(\text{“song”}),\ W(\text{“the”}),\ W(\text{“mat”})) = 0$$

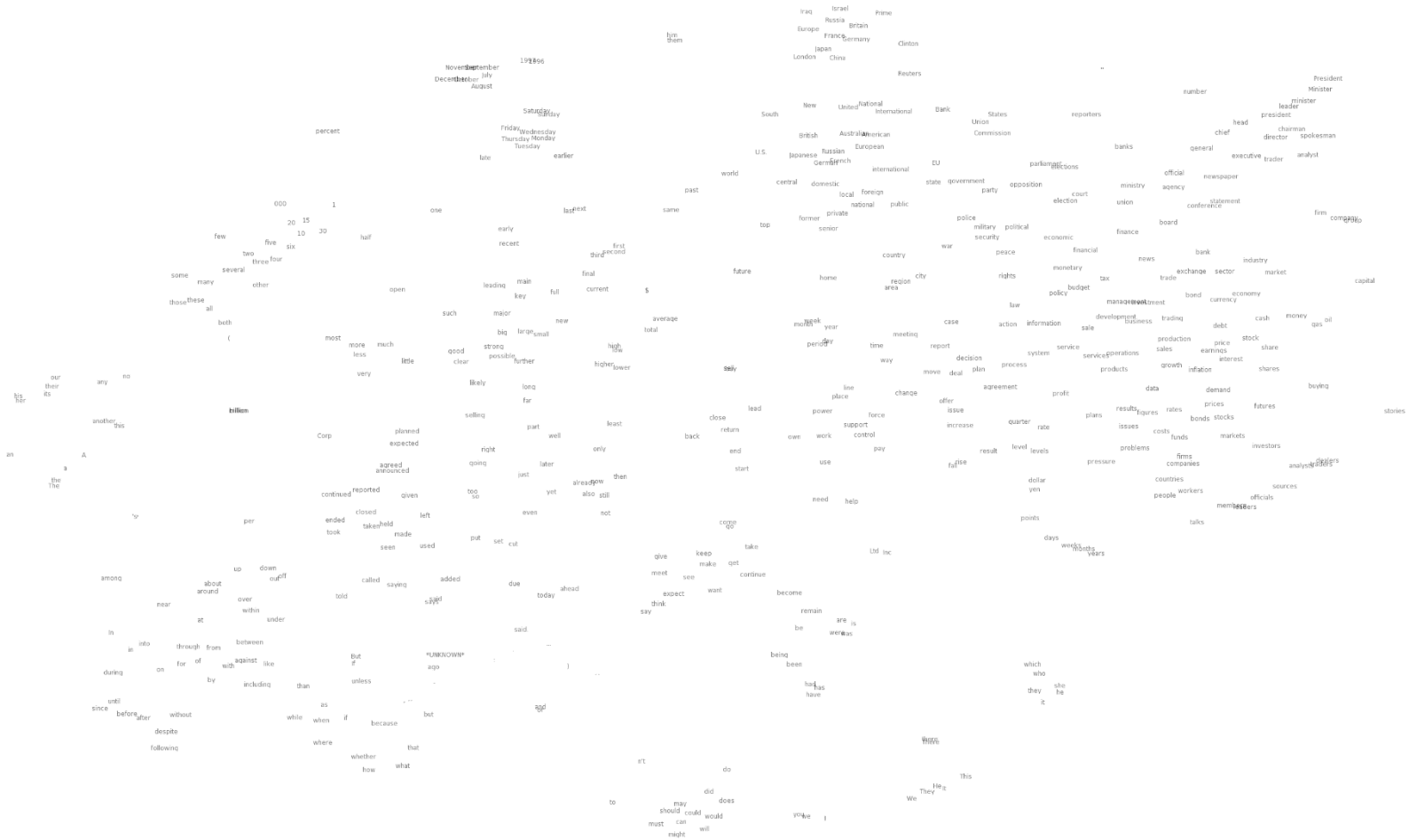


Modular Network to determine if a 5-gram is 'valid' (From Bottou (2011))

Word Embedding Learning

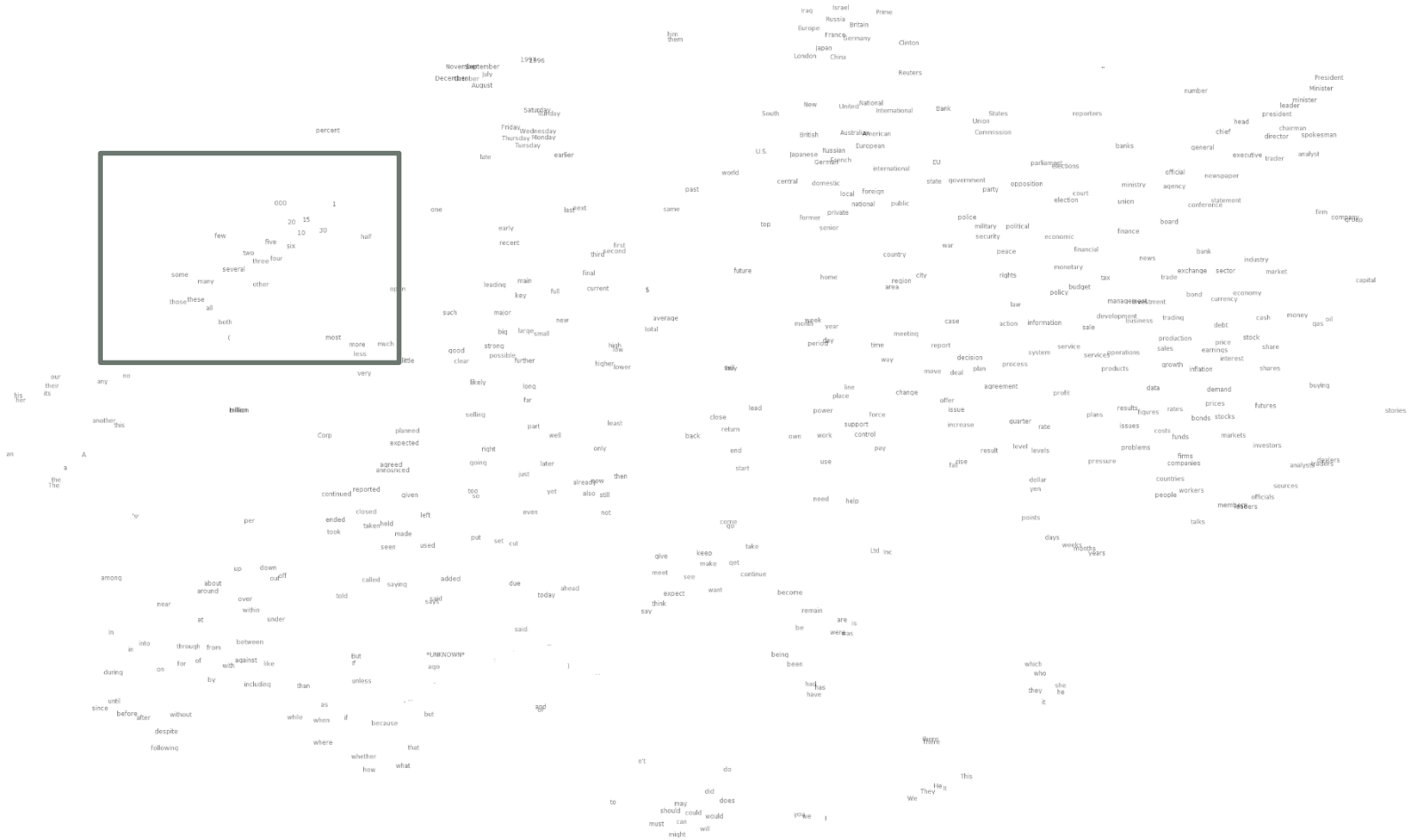
- In order to predict these values accurately, the network needs to learn good parameters for both W and R.

t-SNE Visualization

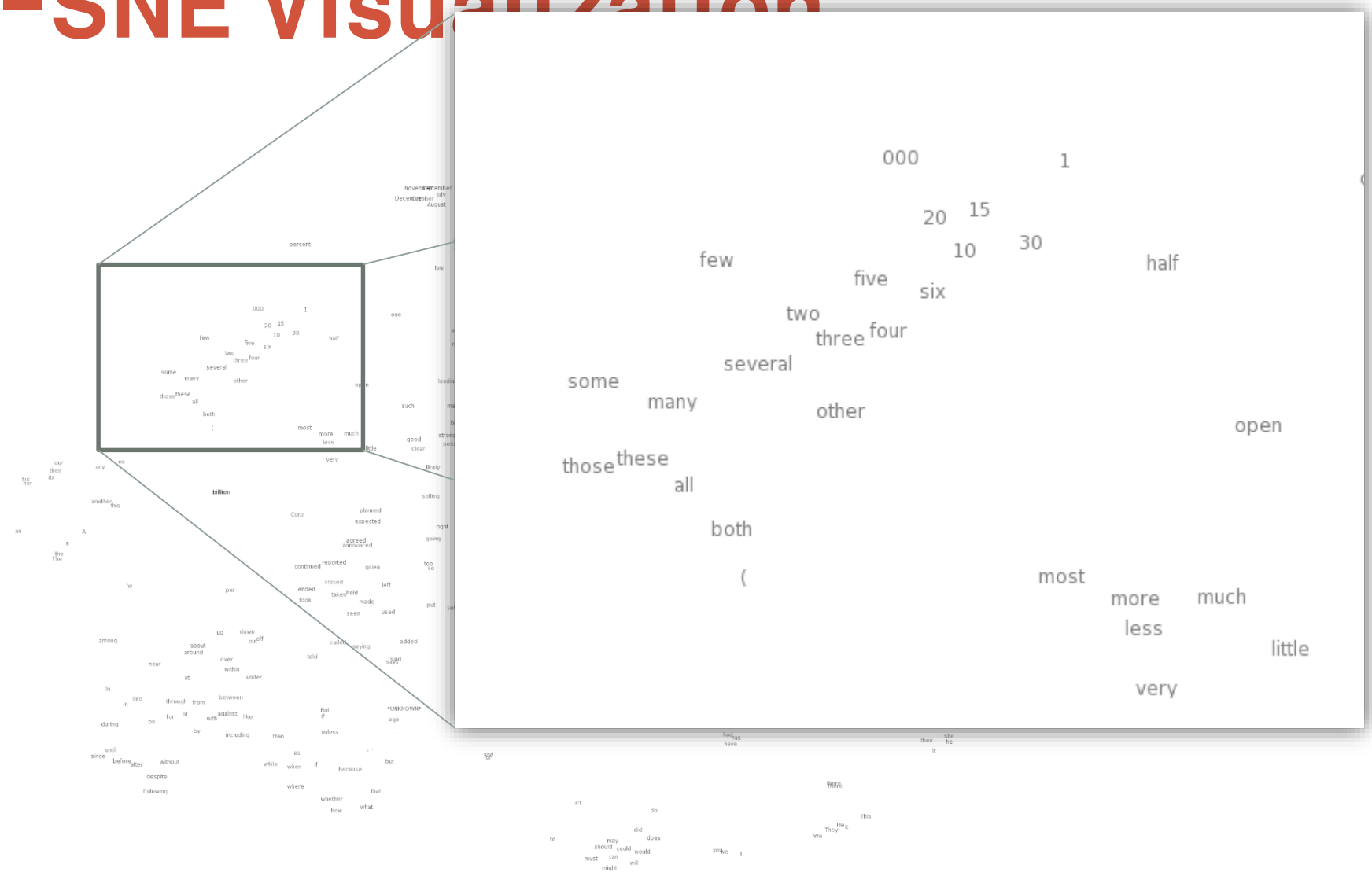


http://metaoptimize.s3.amazonaws.com/cw-embeddings-ACL2010/embeddings-mostcommon.EMBEDDING_SIZE=50.png

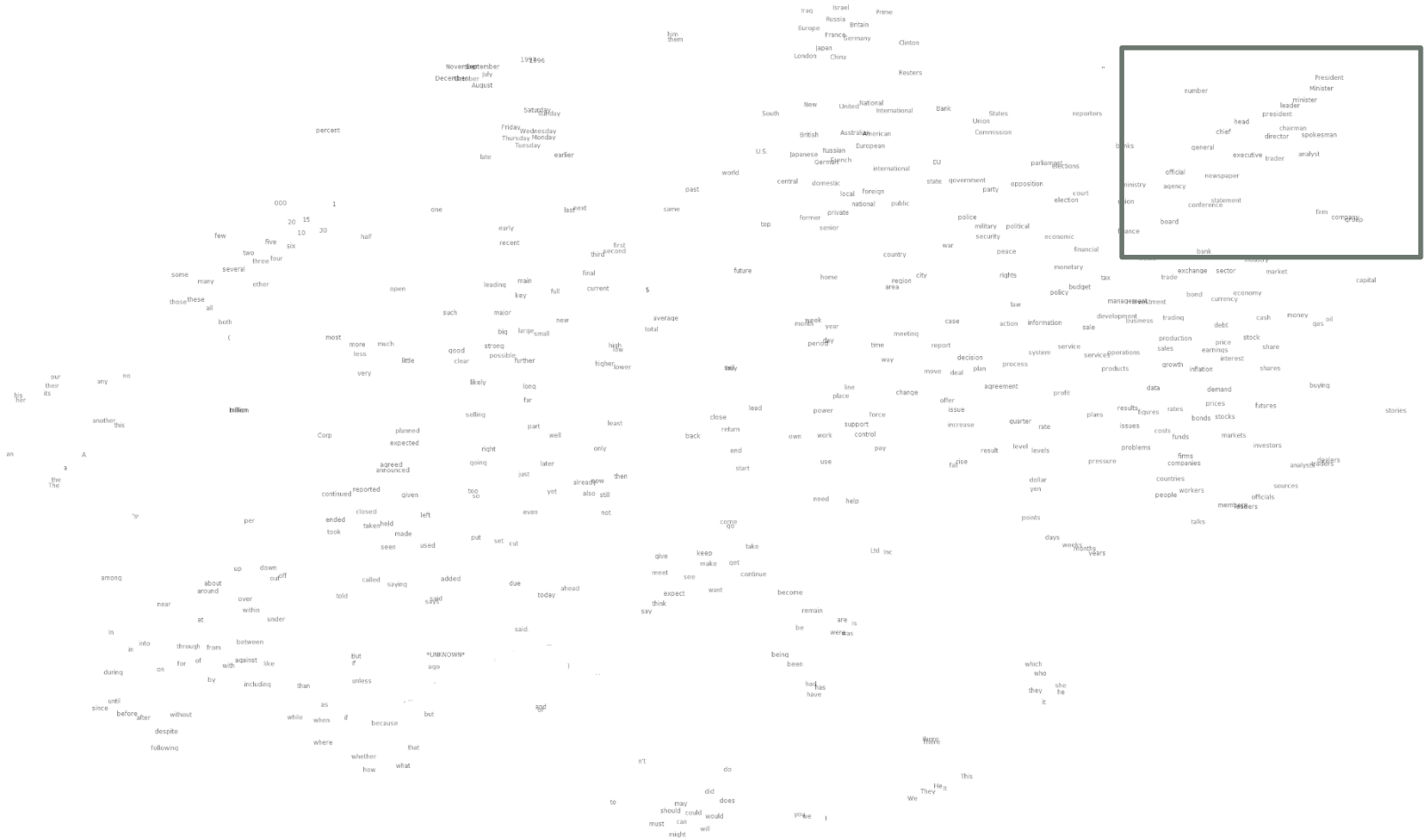
t-SNE Visualization



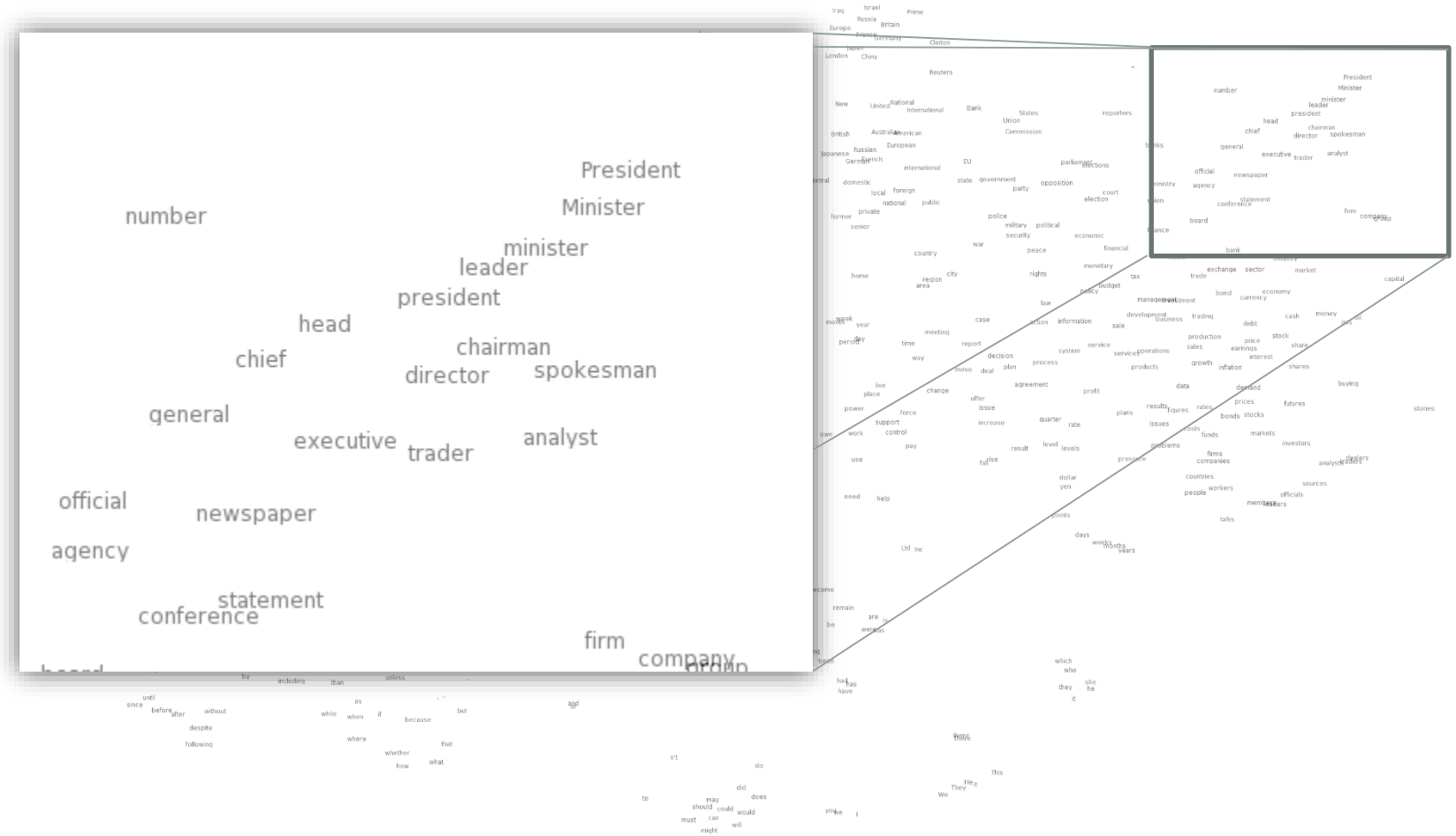
t-SNE Visualization



t-SNE Visualization



t-SNE Visualization



What words have embeddings closest to a given word?

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

What words have embeddings closest to a given word? From Collobert *et al.* (2011)

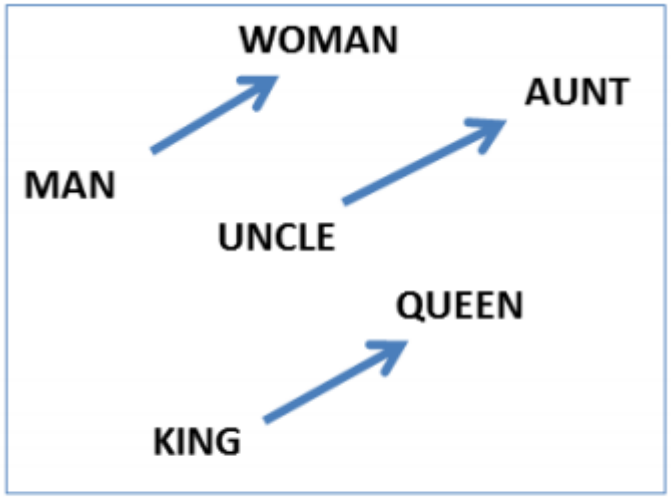
Gender dimension?

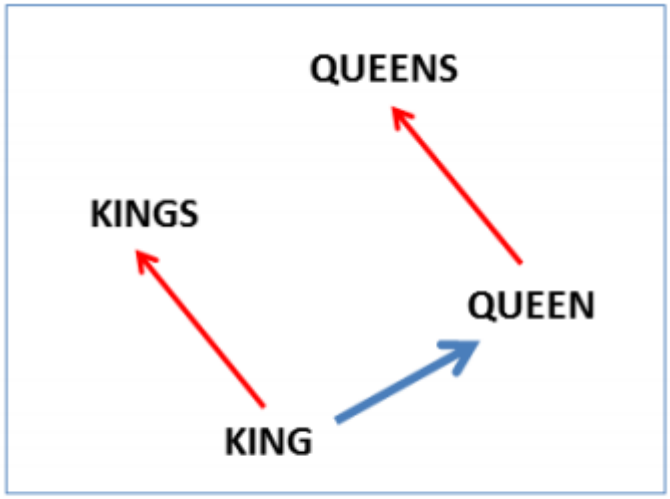
- Word embeddings exhibit an even more remarkable property: analogies between words seem to be encoded in the difference vectors between words. For example, there seems to be a constant male-female difference vector:

$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“aunt”}) - W(\text{“uncle”})$$
$$W(\text{“woman”}) - W(\text{“man”}) \simeq W(\text{“queen”}) - W(\text{“king”})$$

- We say with hindsight, “the word embedding will learn to encode gender in a consistent way. In fact, there’s probably a gender dimension. Same thing for singular vs plural.

Examples





(Mikolov et al., NAACL HLT, 2013)

Much more sophisticated relationships

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Summary

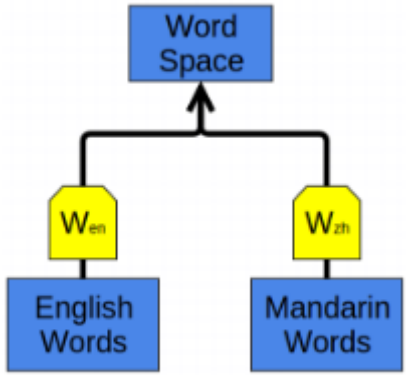
- All of these properties of W are side effects.
  - We didn't try to have similar words be close together.
  - We didn't try to have analogies encoded with difference vectors.
  - All we tried to do was perform a simple task, like predicting whether a sentence was valid. These properties more or less popped out of the optimization process.

- This seems to be a great strength of neural networks
  - They learn better ways to represent data, automatically.
  - Representing data well, in turn, seems to be essential to success at many machine learning problems.
  - Word embeddings are just a particularly striking example of learning a representation.

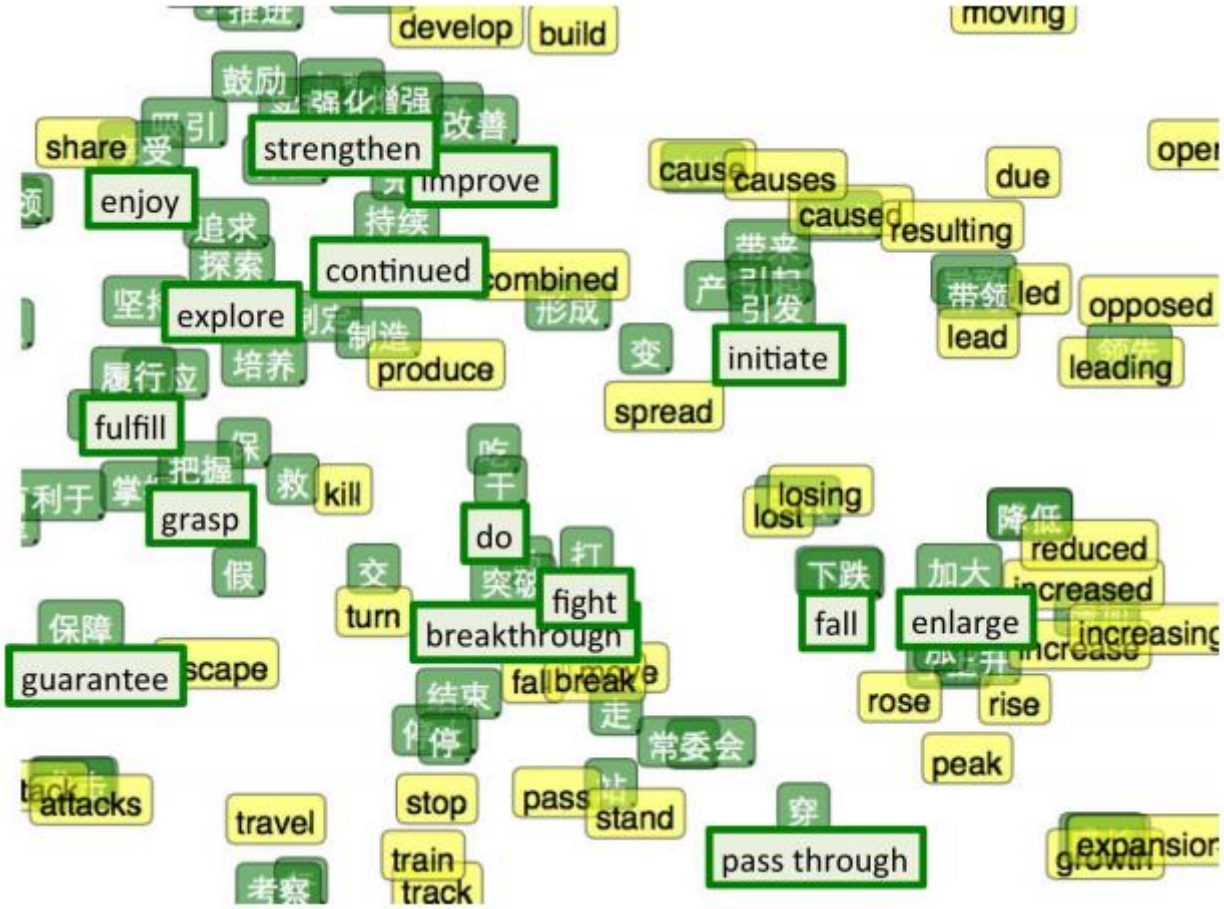
SHARED REPRESENTATIONS

Bilingual word-embedding

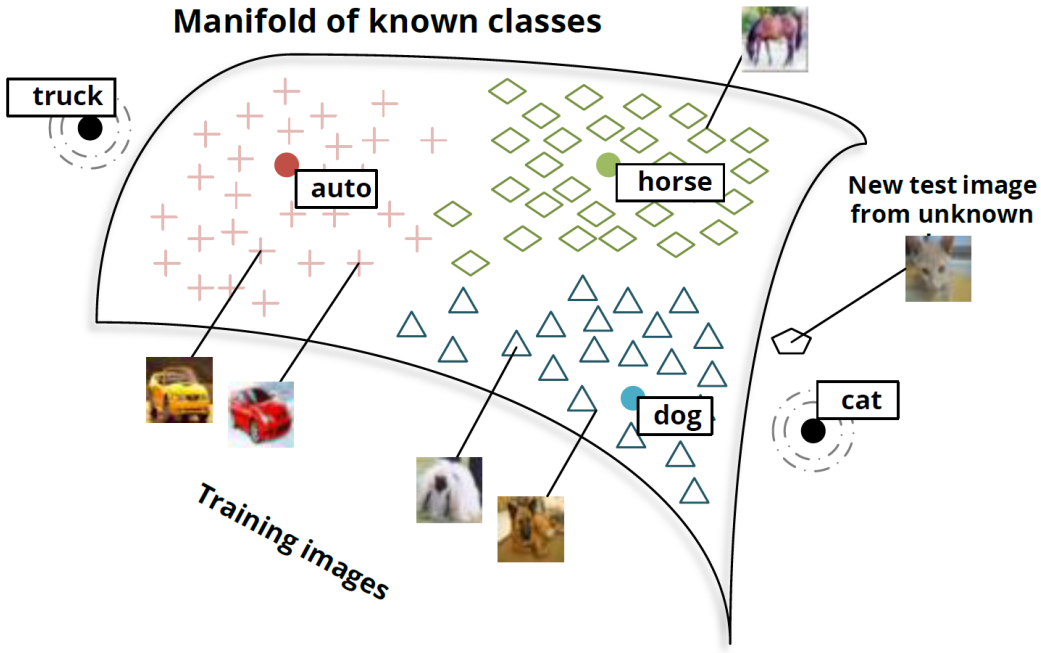
- We train two word embeddings, W_{en} and W_{zh} in a manner similar to how we did above. However, we know that certain English words and Chinese words have similar meanings. So, we optimize for an additional property: words that we know are **close translations should be close together**.

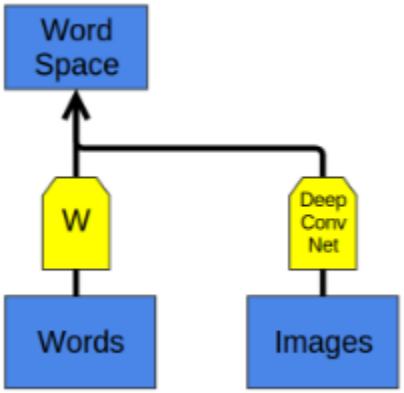


t-SNE visualization of the bilingual word embedding



Word/image embedding





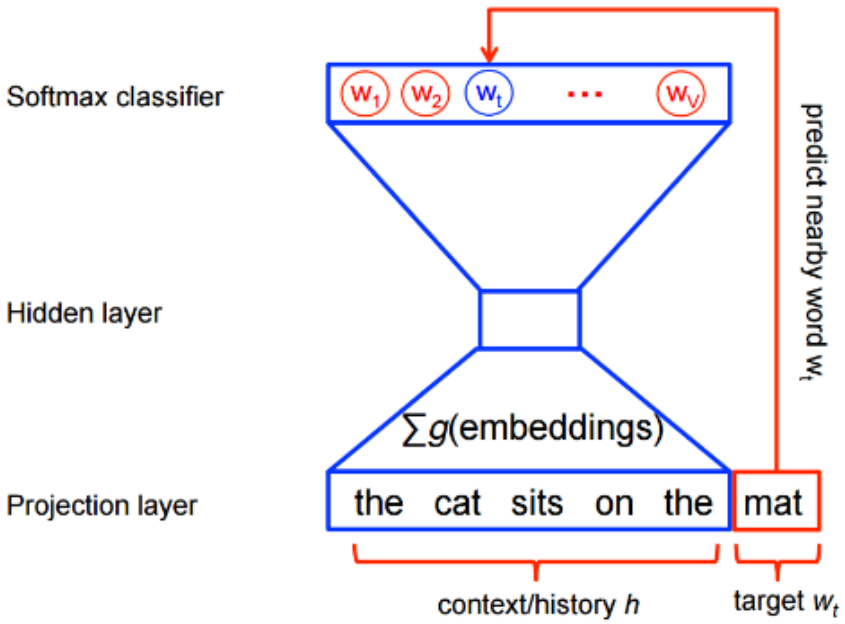
BACKUPS

WORD2VEC MODEL EXAMPLE

Word2vec model

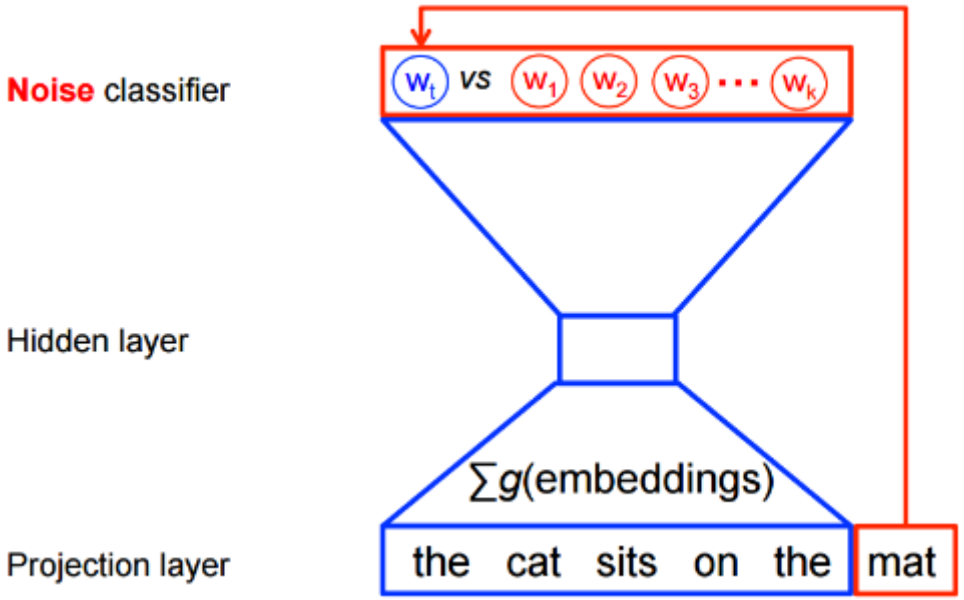
- Word2vec is a particularly computationally-efficient predictive model
- Two methods
  - Continuous Bag-of-Words model (CBOW)
    - CBOW predicts target words (e.g. 'mat') from source context words ('the cat sits on the')
  - Skip-Gram
    - Skip-gram predicts source context-words from the target words model

Basic model



- Neural probabilistic language models are traditionally trained using the ML principle to maximize the probability of the next word w_t (for "target") given the previous words h (for "history") in terms of a softmax function

Scaling up with Noise-Contrastive Training



- This objective is maximized when the model assigns high probabilities to the real words, and low probabilities to noise words

Skip-gram model 1/3

- Example sentence:

  the quick brown fox jumped over the lazy dog

- Context:
  - We could define 'context' in any way that makes sense
  - For now, let's define 'context' as the window of words to the left and to the right of a target word. Using a window size of 1, we then have the dataset of (context, target) pairs.

([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ...

Skip-gram model 2/3

([the, brown], quick), ([quick, fox], brown), ([brown, jumped], fox), ...

- Skip-gram inverts contexts and targets, and tries to predict each context word from its target word, so the task becomes to predict 'the' and 'brown' from 'quick', 'quick' and 'fox' from 'brown', etc. Therefore our dataset becomes (input, output) pairs.

(quick, the), (quick, brown), (brown, quick), (brown, fox), ...

Skip-gram model 3/3

- Training example

Training sample: ([the, brown], quick), ...

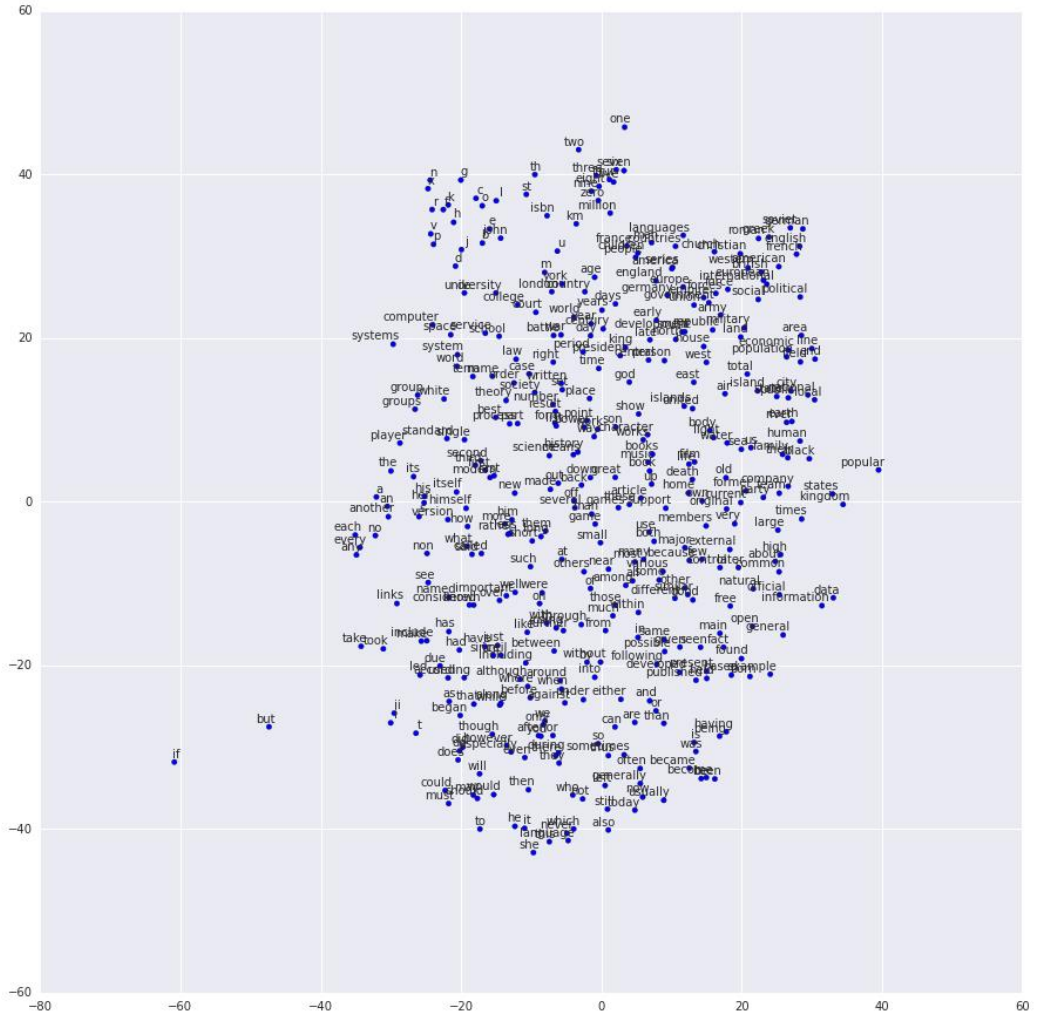
- We have to estimate “the” from “quick”.
- For learning
  - We select of noisy (contrastive) examples by drawing from some noise distribution
    - Let’s assume we select sheep as a noisy example
    - We compute the loss for this pair of observed and noisy examples

“quick” -> “the” vs “quick”-> “sheep”

Implementations

```
….
embeddings = tf.Variable(tf.random_uniform([vocabulary_size, embedding_size], -1.0, 1.0))
….
nce_weights = tf.Variable(
  tf.truncated_normal([vocabulary_size, embedding_size],
                      stddev=1.0 / math.sqrt(embedding_size)))
nce_biases = tf.Variable(tf.zeros([vocabulary_size]))
….
# Placeholders for inputs
train_inputs = tf.placeholder(tf.int32, shape=[batch_size])
train_labels = tf.placeholder(tf.int32, shape=[batch_size, 1])
….
embed = tf.nn.embedding_lookup(embeddings, train_inputs)
….
# Compute the NCE loss, using a sample of the negative labels each time.
loss = tf.reduce_mean(
  tf.nn.nce_loss(nce_weights, nce_biases, embed, train_labels,
                 num_sampled, vocabulary_size))
….
# We use the SGD optimizer.
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1.0).minimize(loss)
```

Results


60
40
20
0
−20
−40
−60
−80
−60
−40
−20
0
20
40
60
one
two
million
but
if
popular

Conclusions

- The representation perspective of deep learning
  - Why are neural networks effective?
    - Because better ways of representing data can pop out of optimizing layered models.