Brief Introduction to Natural Language Processing

EE599 Deep Learning

Keith M. Chugg
Spring 2020

USC University of Southern California

Overview of NLP Topics

- Some useful packages for NLP
- Word embeddings
    - word2vec example
- RNNs for NLP tasks
- Attention-based Approaches
    - Transformers and BERTs

Useful NLP Packages

Natural Language Tool Kit (NLTK) https://www.nltk.org/book/

NLTK book on Amazon

"shallow" NLP

tokenize, word counts, regular expressions, etc

Useful NLP Packages

[GenSim]

- Corpora and Vector Spaces
  - From Strings to Vectors
  - Corpus Streaming – One Document at a Time
  - Corpus Formats
  - Compatibility with NumPy and SciPy
- Topics and Transformations
  - Transformation interface
  - Available transformations
- Similarity Queries
  - Similarity interface
  - Where next?
- Experiments on the English Wikipedia
  - Preparing the corpus
  - Latent Semantic Analysis
  - Latent Dirichlet Allocation
- Distributed Computing
  - Why distributed computing?
  - Prerequisites
  - Core concepts
  - Available distributed algorithms

has some more advanced functionality than NLTK, but does not replicate everything useful in NLTK…

e.g., topic identification and text summarization

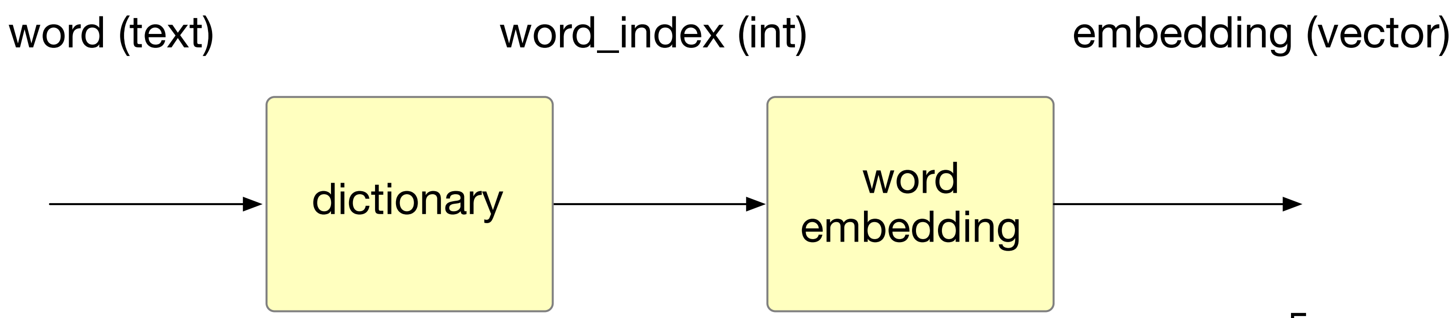
Word Embeddings

NLTK and Gensim have tools for preparing text data to be processed…

To use deep-learning for NLP tasks we need to convert text to numerical data

this is the role of an embedding

Word Embeddings



“house”

92

$$\mathbf{v}_{92} = \begin{bmatrix} -1.32 \\ +2.13 \\ 4.28 \\ \vdots \\ -3.12 \end{bmatrix}$$

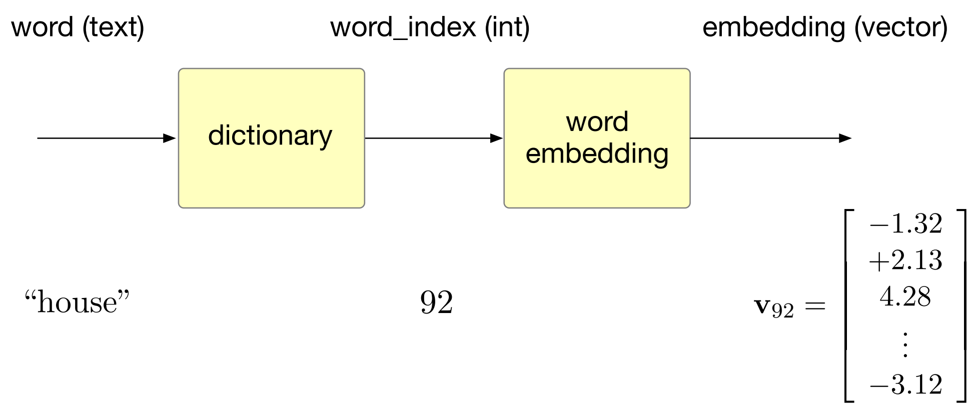
int between 1 and V
(one-to-one)

text from a vocabulary of size V (e.g., V=10,000)

word embedding (vector)
dim = D (e.g., 300)

this is essentially feature extraction for text
(could be done at the character or gram-level as well, but usually word embeddings)

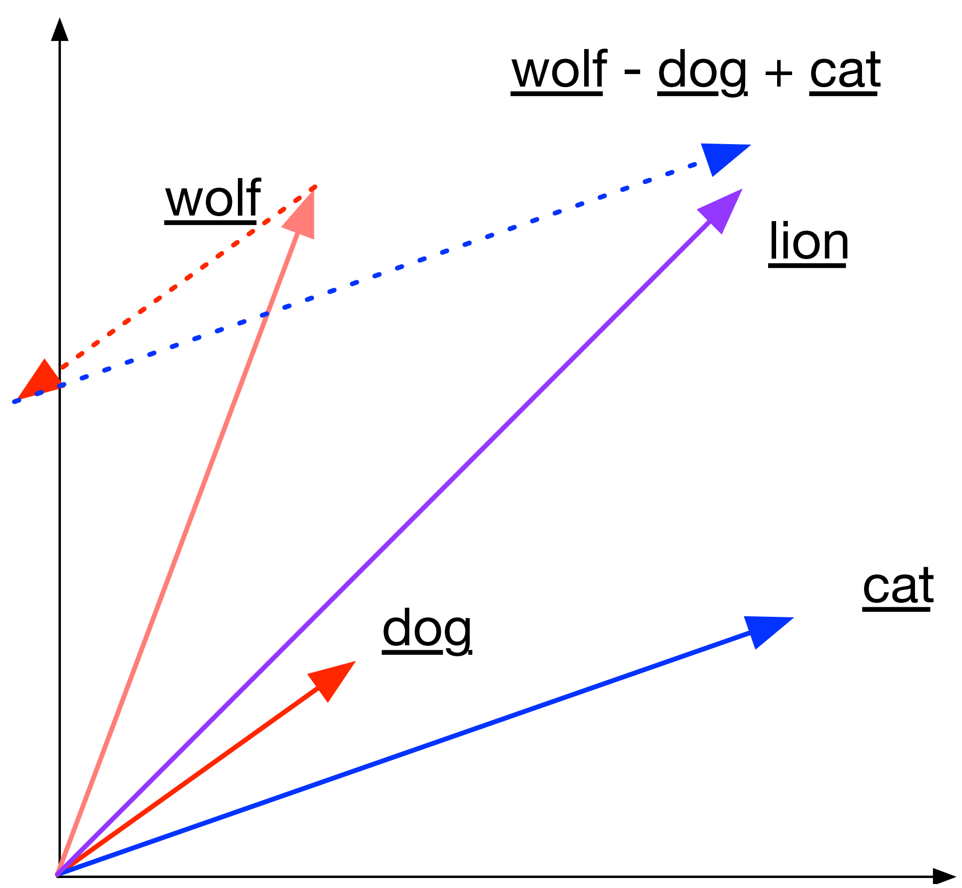
Word Embeddings

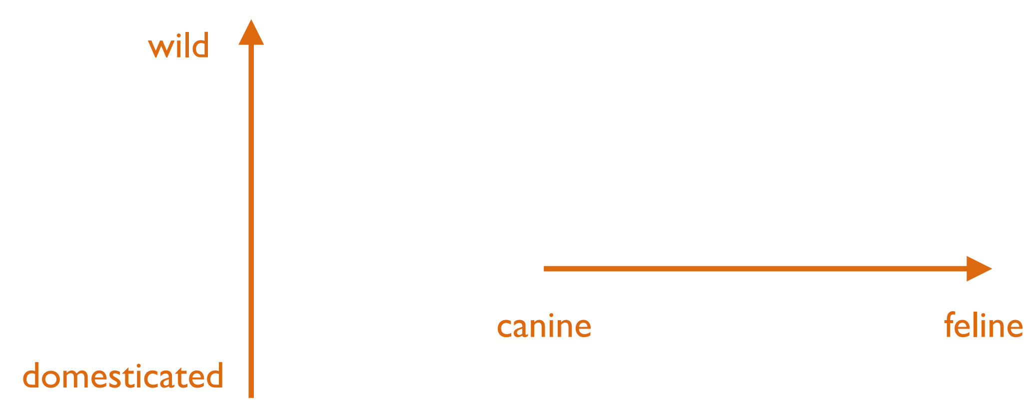


how to choose the vectors?

ideally it would capture relations and context

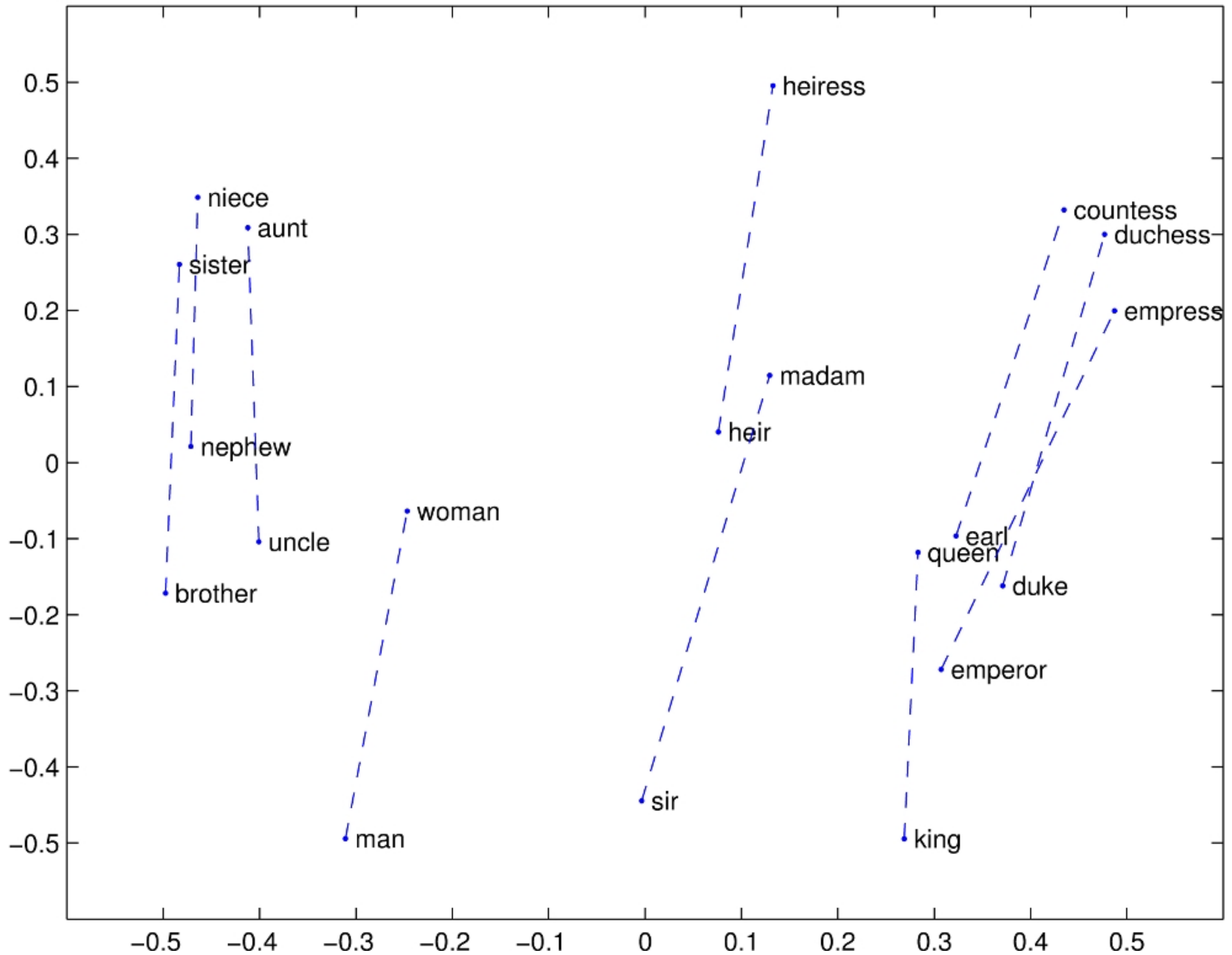
and allow for a sort of word-math





PCA used to reduce from D to 2 dimensions

Word Embeddings

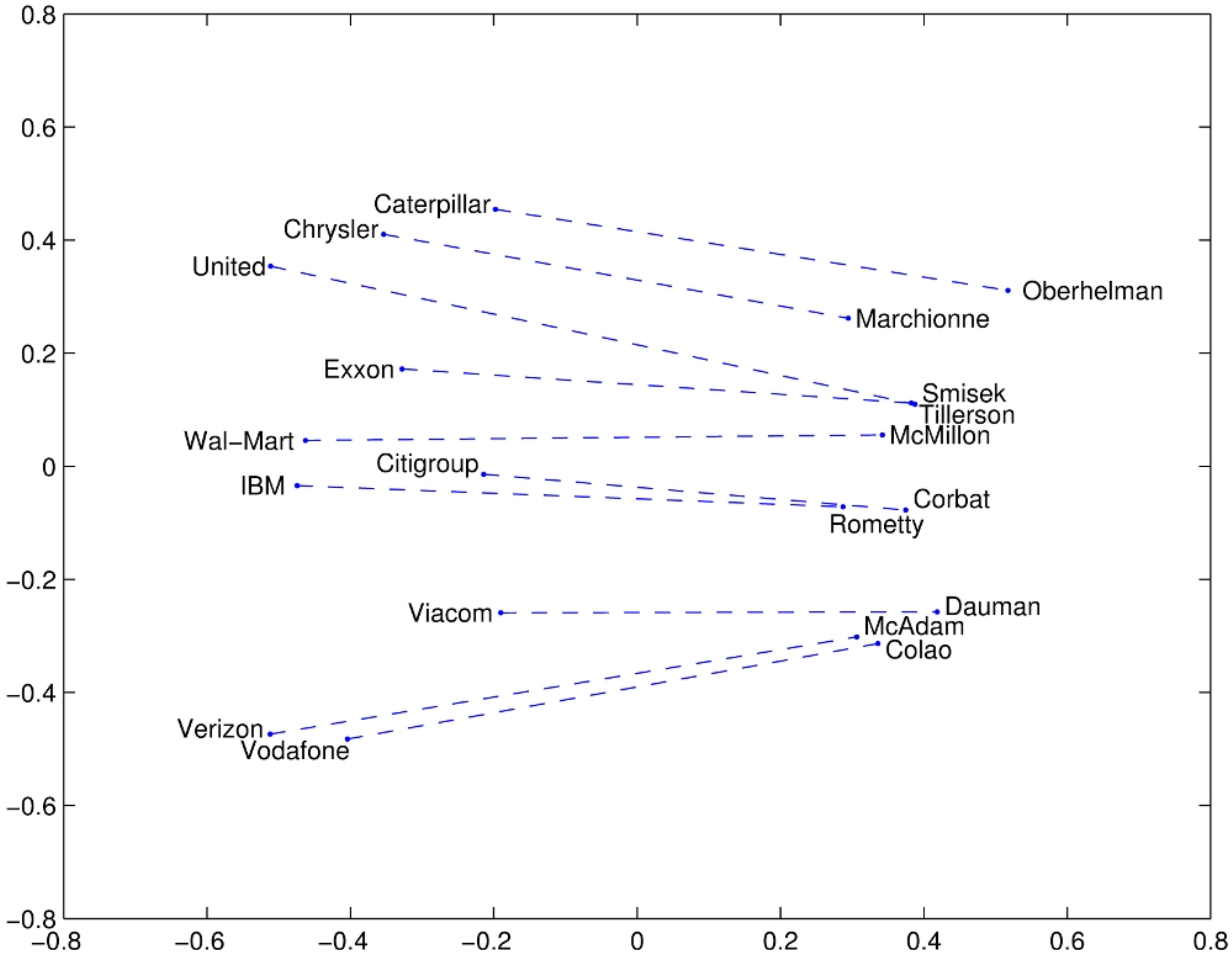


PCA used to reduce from D to 2 dimensions

Gender ~ status/familiarity

https://nlp.stanford.edu/projects/glove/

Word Embeddings

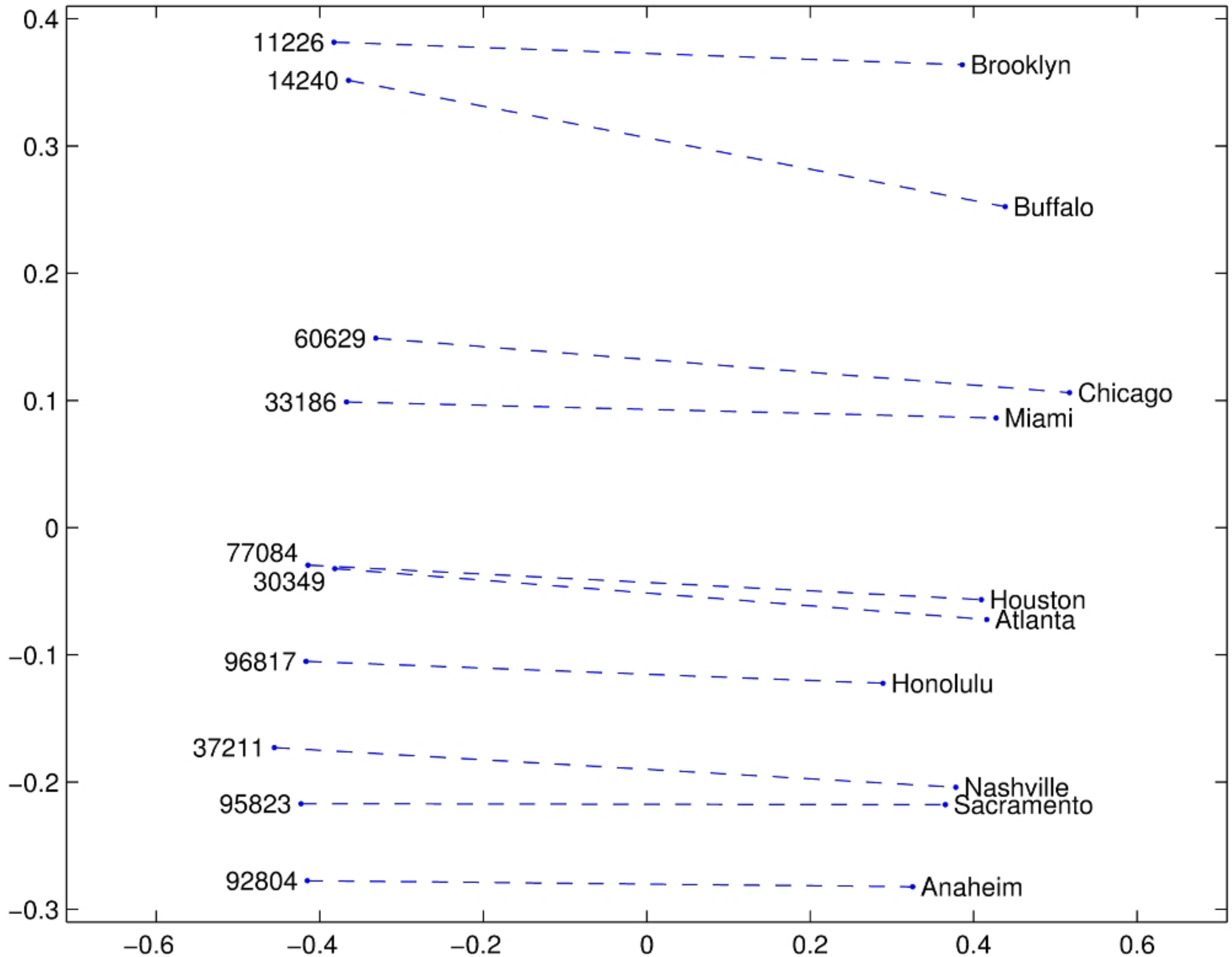


PCA used to reduce from D to 2 dimensions

company ~ eco

https://nlp.stanford.edu/projects/glove/

Word Embeddings

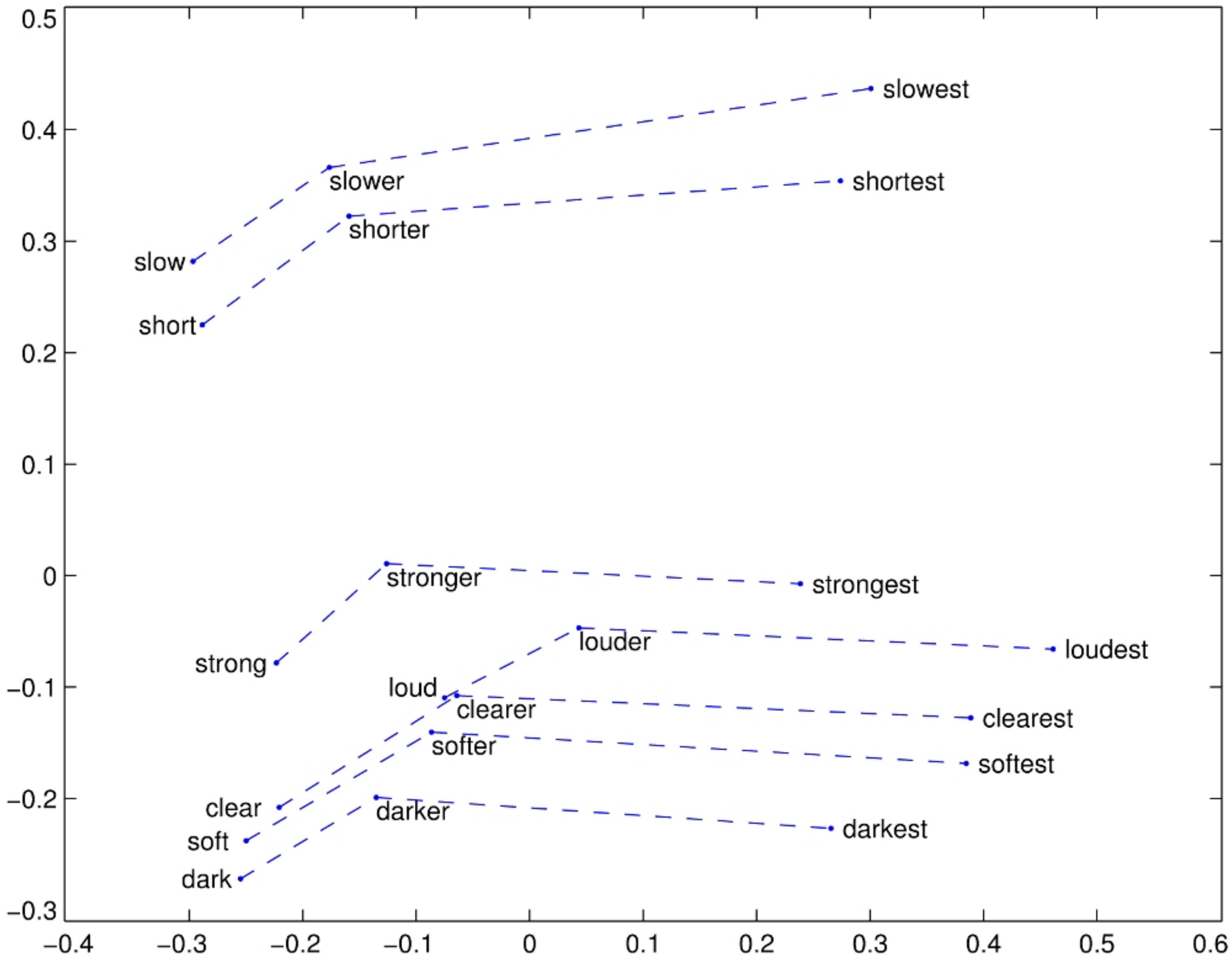


PCA used to reduce from D to 2 dimensions

city ~ zip code

https://nlp.stanford.edu/projects/glove/

Word Embeddings



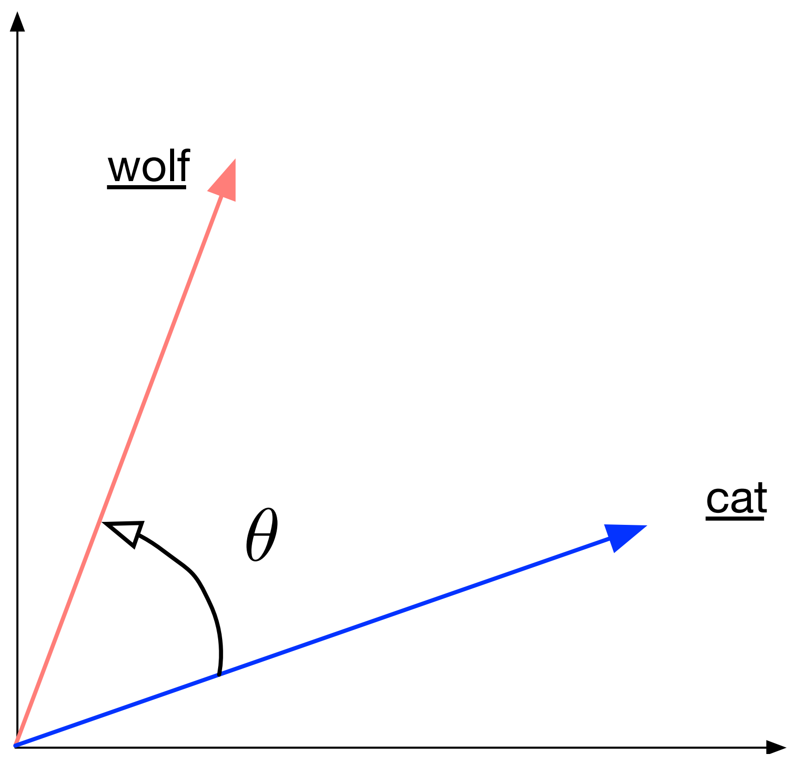
PCA used to reduce from D to 2 dimensions

comparative ~ superlative

https://nlp.stanford.edu/projects/glove/

Word Embeddings

measure the similarity of two vectors



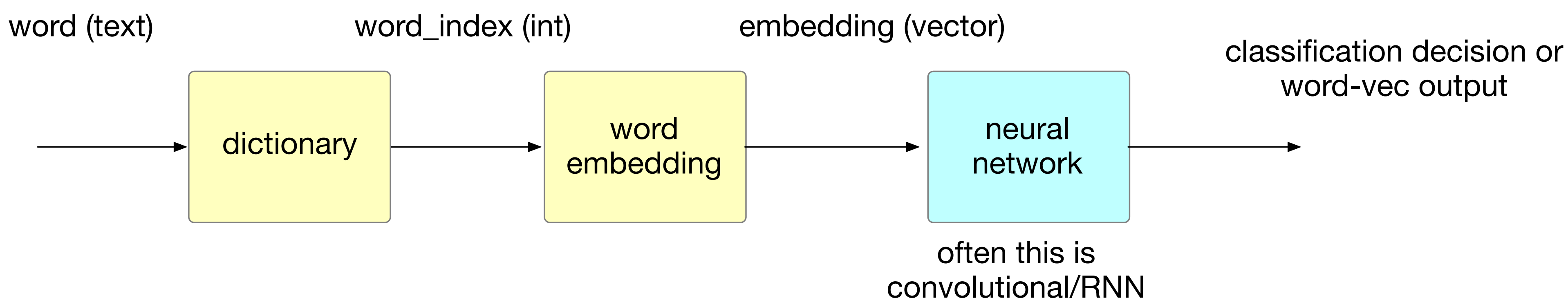
$$\cos\theta_{x,y} = \rho_{xy} = \frac{\mathbf{y}^{\mathrm{t}}\mathbf{x}}{\|\mathbf{x}\|\|\mathbf{y}\|}$$

this is standard linear algebra
(Euclidian geometry)

called the "cosine distance" in the ML world

(not limited to 2D)

Word Embeddings



lots of applications — full NLP beyond scope:

topic identification (e.g., sports article vs. politics)

sentiment analysis (e.g., happy vs. angry tweet)

language translation (e.g., English to Spanish)

text generation (e.g., GAN)

Word Embeddings - Methods

most methods rely on context

the young child **smiled** when she saw the clown

context size 2 for “smile” here is: **{young, child, when she}**

context allows words with similar roles to be identified

e.g., “smiled” and “frowned” may be observed to be similar words in terms of their roles

Word Embeddings - word2vec

Google's word2vec - "skip-gram" model

minimize this over a parametrized p(.|.) function

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

use a softmax for the probability…

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^\top v_{w_I}\right)}$$

the v-values are the desired embeddings

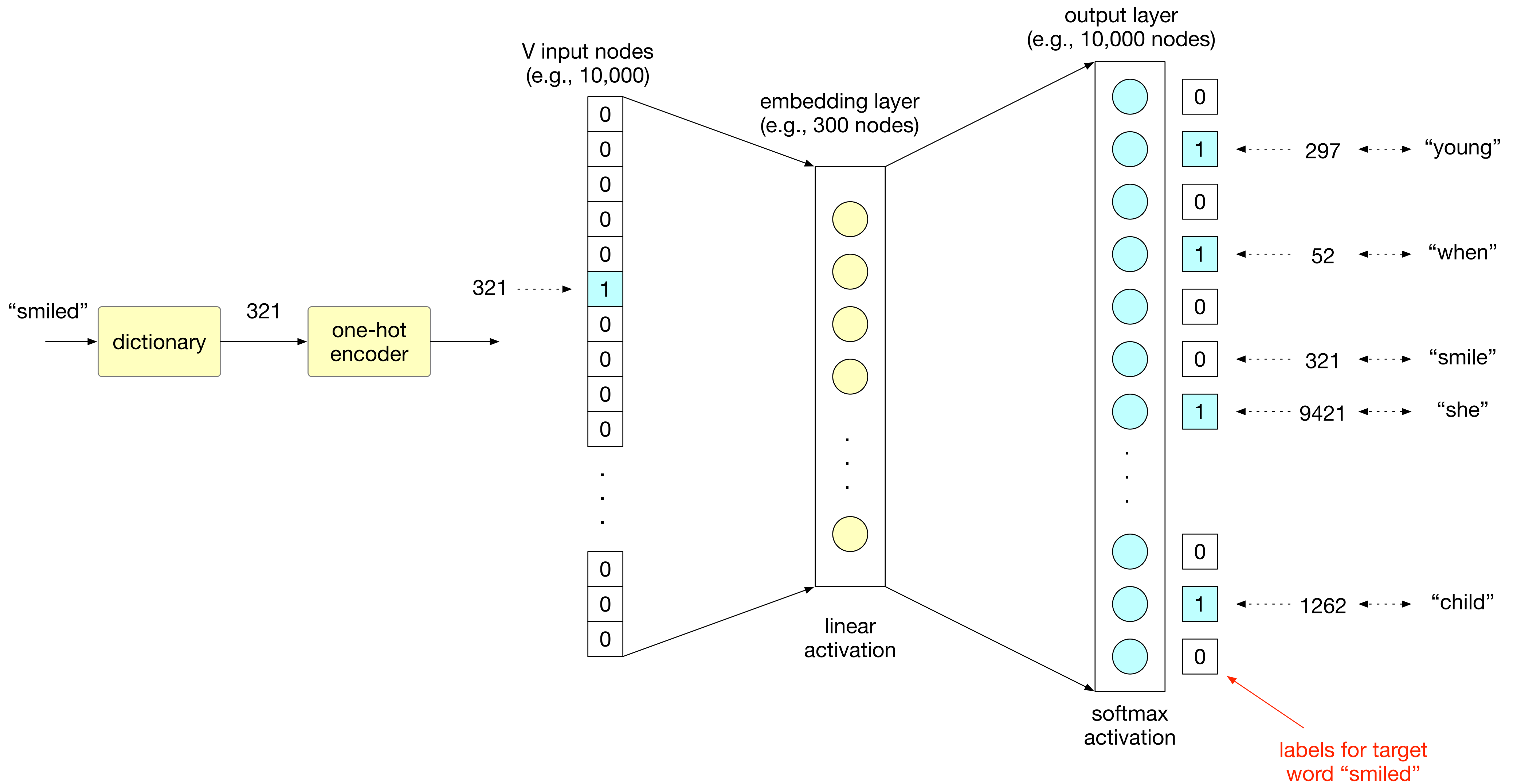
the v-prime-values are associated with context

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

https://code.google.com/archive/p/word2vec/

Word Embeddings - word2vec

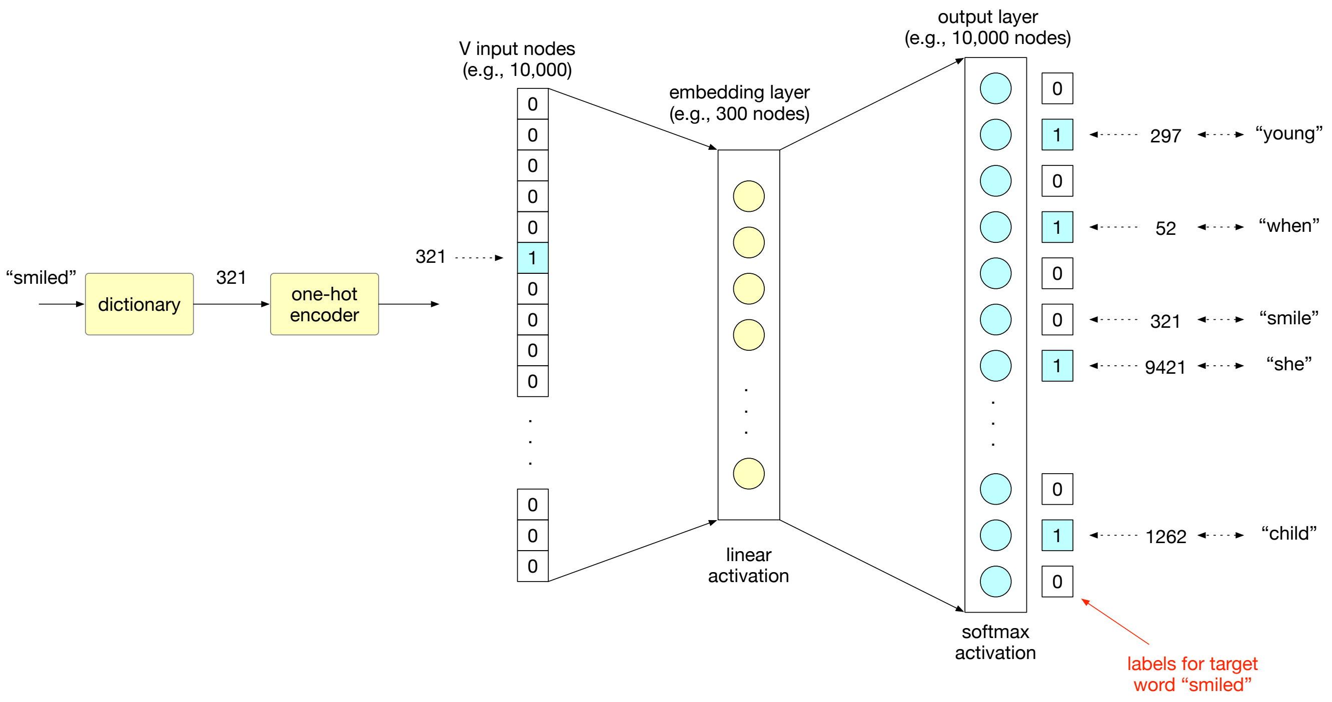
Google's word2vec - "skip-gram" model



train a single hidden layer MLP to predict the context for a given target word (e.g., smile)

Word Embeddings - word2vec

Google's word2vec - "skip-gram" model



$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

rows for the first-junction **W** matrix are the vector representations for the target (**v**)

train this network and throw-out the output soft:

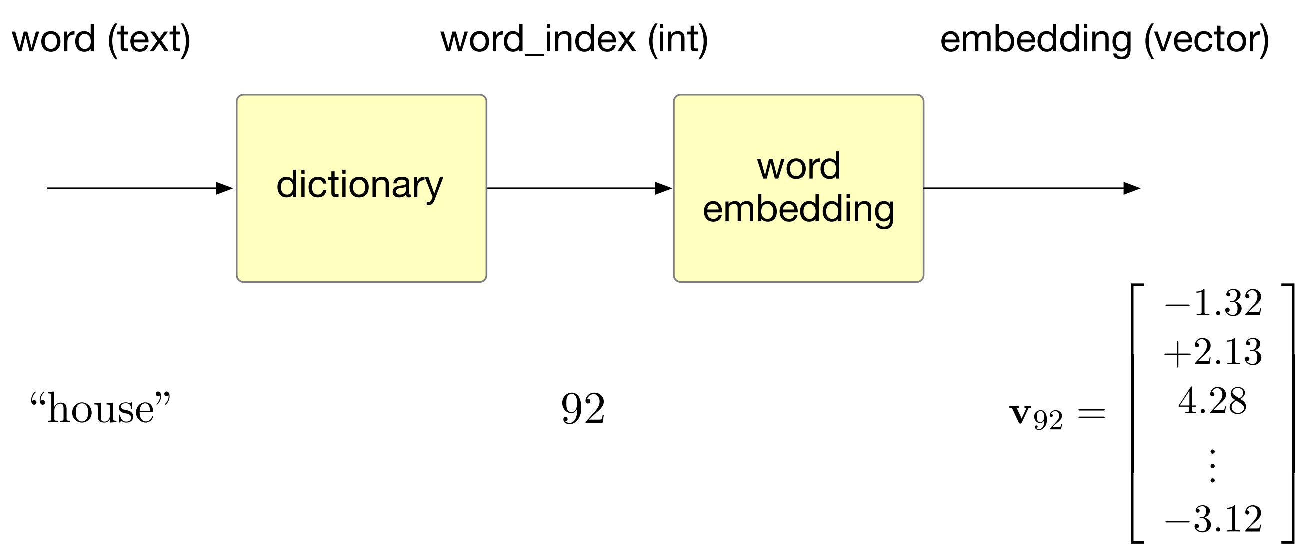
the embedding is the combination of the dictionary, one-hot encode, and first layer weight vectors

Word Embeddings - word2vec

Google’s word2vec - “skip-gram” model

train this network and throw-out the output layer:

the embedding is the combination of the dictionary, one-hot encode, and first layer weight vectors



Word Embeddings - word2vec

There is another variant of word2vec that

predicts the target (output) from the context (inputs)

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

in this case the embeddings are the v-prime vectors and they are the columns of the output layer

this is called "continuous bag of words" (CBOW)

(I think skip-gram is more widely used)

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

https://code.google.com/archive/p/word2vec/

word2vec - Negative Sampling

Problem:

this can be very complex due to desire to work with large vocabulary

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^\top v_{w_I}\right)}$$

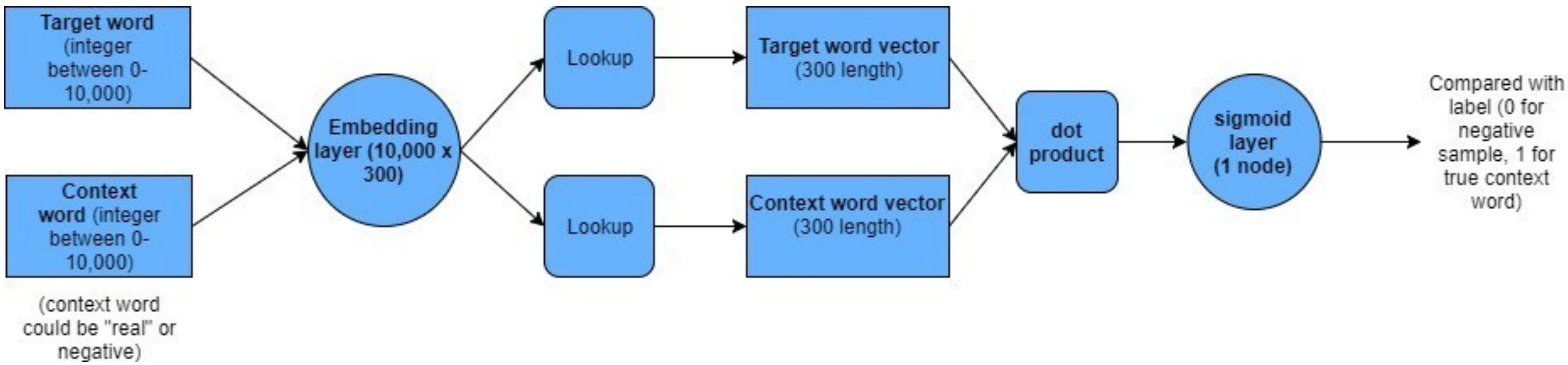
the word2vec paper suggests a complexity-reduction method called **negative sampling**

basic idea is to not label all non-context outputs — just a subset…

don't train all of the weights for each example…

word2vec - Negative Sampling

Problem:



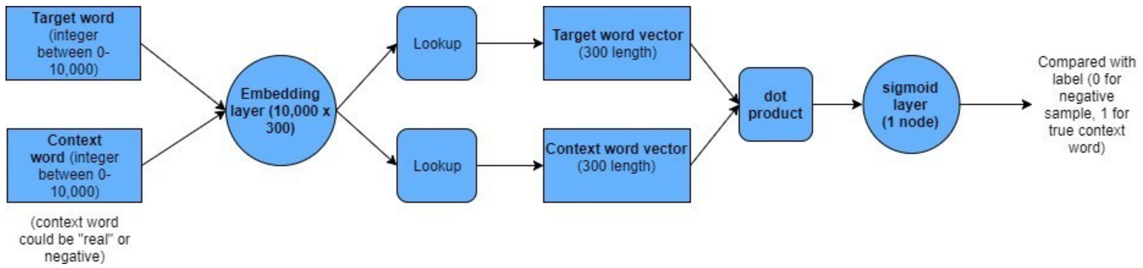
a "negative context" word is a word that is not a context word for the target

let's try this out… keras_word2vec.py

https://adventuresinmachinelearning.com/word2vec-keras-tutorial/

https://github.com/adventuresinML/adventures-in-ml-code

word2vec - Negative Sampling



keras_word2vec.py

obviously, this is a concept demo and not ready for prime time…

Not sure of the best platform to train a word2vec platform in a serious manner…

Gensim has pre-trained word2vec capability — typical use pattern is to use pre-trained embeddings…

https://adventuresinmachinelearning.com/gensim-word2vec-tutorial/

https://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/

Google source code

https://code.google.com/archive/p/word2vec/

Word Embeddings - tf.keras

tf.keras has an **Embedding** layer

```
tf.keras.layers.Embedding(
    input_dim, output_dim, embeddings_initializer='uniform',
    embeddings_regularizer=None, activity_regularizer=None,
    embeddings_constraint=None, mask_zero=False, input_length=None, **kwargs
)
```

V **input_dim**: int > 0. Size of the vocabulary, i.e. maximum integer index + 1

D **output_dim**: int >= 0. Dimension of the dense embedding.

must be the first layer (feature extraction)

can be trainable or non-trainable

trainable: train your embedding for the specific application

non-trainable: use a pre-trained embedding since:

- *in many cases word relations are similar across applications*
- *take a long time to train!*

similar to using some pre-trained convolutional layers from large CV networks

Word Embeddings - tf.keras

keras blog: using pre-trained word embeddings in keras

had sample code (should be easy to update to tf.keras)

Word Embeddings - Glove

Glove has pre-trained embeddings available

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

P(j|i) = probability that word j is context word for word i

approach is based on observation that ratios of these probabilities are highly informative

regression cost function:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

https://nlp.stanford.edu/projects/glove/

Word Embeddings - Glove

Glove has pre-trained embeddings available

regression cost function:

$$J = \sum_{i,j=1}^{V} f\left(X_{ij}\right)\left(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij}\right)^2$$

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}.$$

(to handle X[i,j] ~~0)

X[i,j] = count of how many times word j was seen as context for word i

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

https://nlp.stanford.edu/projects/glove/

NLP and Word Embeddings

two examples using the Glove pre-trained embeddings…

Sentiment analysis of movie reviews in IMBD database:

[Chollet] Francois Chollet, Deep Learning with Python, Manning, 2018.

https://github.com/fchollet/deep-learning-with-python-notebooks/blob/master/6.1-using-word-embeddings.ipynb

tries to use very small amount of training data and LSTM with mixed results..

classify newsgroup posts by newsgroup

https://github.com/keras-team/keras/blob/master/examples/pretrained_word_embeddings.py

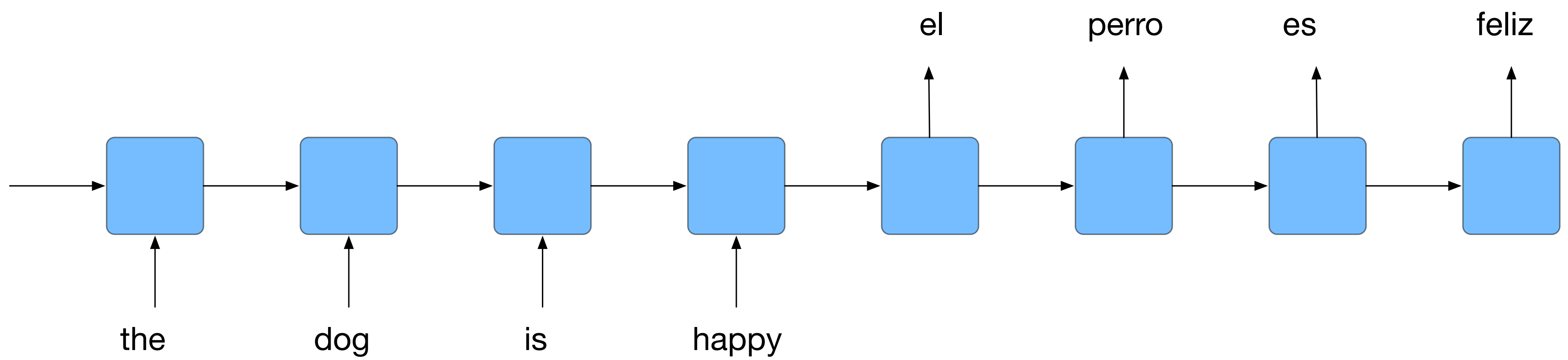
uses 1D convolutional layers — haven't tried this yet…

Overview of NLP Topics

- Some useful packages for NLP
- Word embeddings
  - word2vec example
- RNNs for NLP tasks
- Attention-based Approaches
  - Transformers and BERTs

RNNs for NLP Tasks

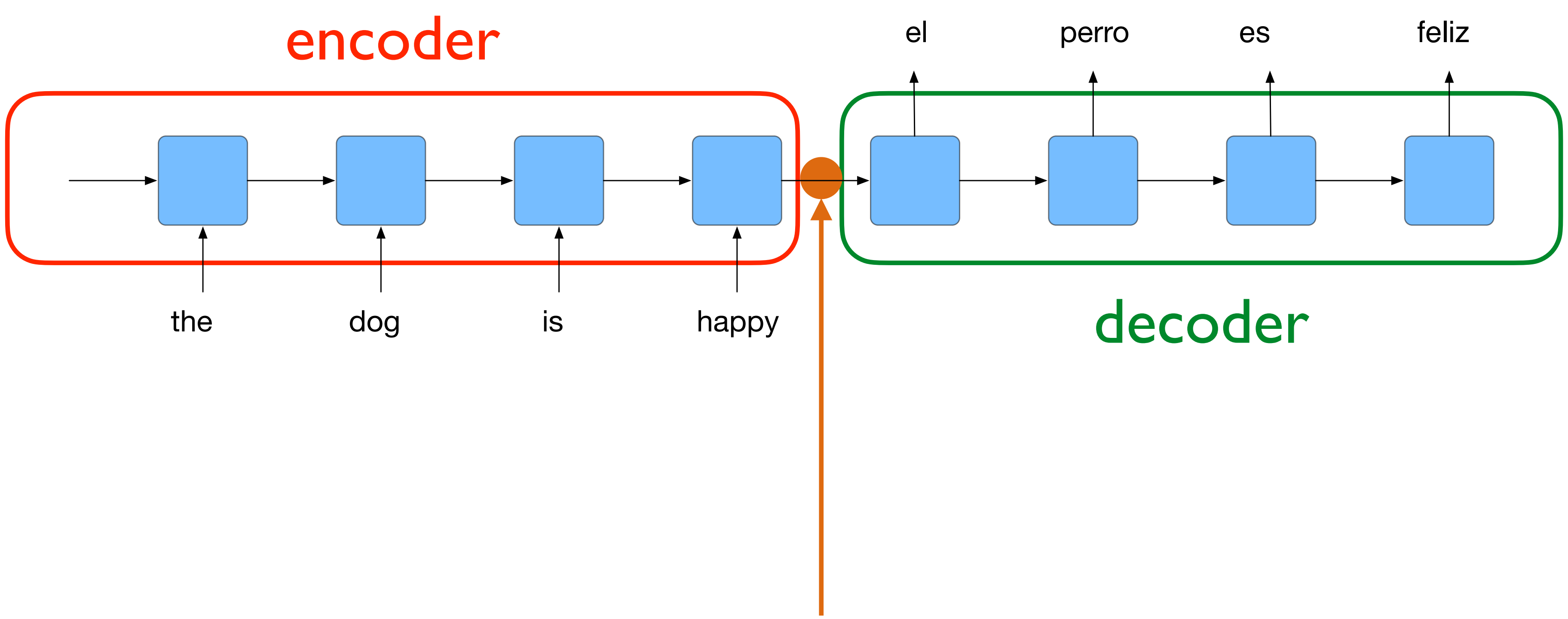
when we talked about RNNs, I showed this example



machine translation (common and challenging NLP task)

RNNs for NLP Tasks

when we talked about RNNs, I showed this example



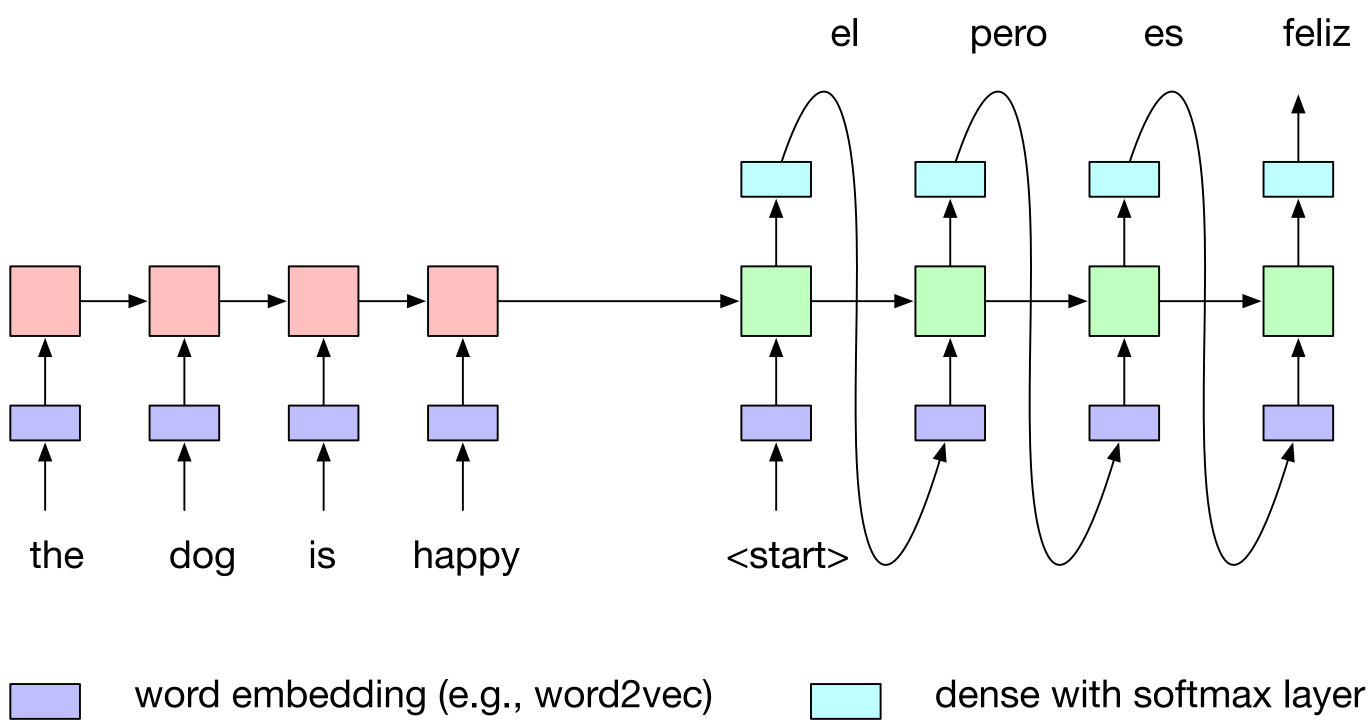
embedding for the entire sentence (aka context)

this seems to be asking a lot of the single vector embedding!

(especially for long sentences)

RNNs for NLP Tasks

let's show some more of the details of this baseline

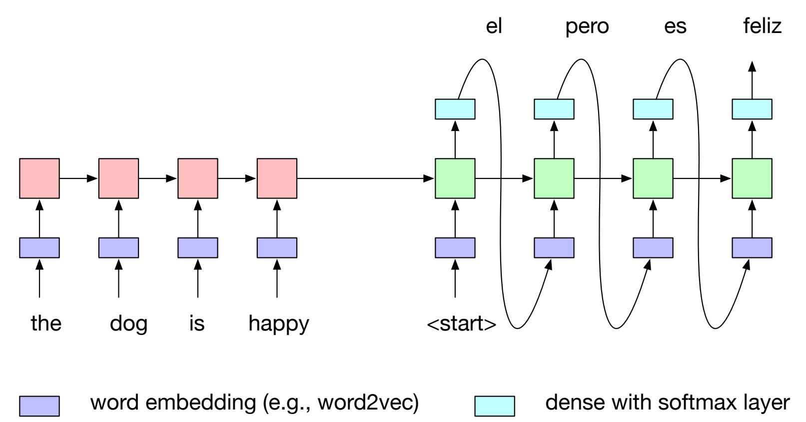


goal is to produce: $$p(\mathbf{w}) = \prod_{n=0}^{N-1} p(w_n | w_{n-1}, w_{n-2} \dots w_0)$$

Decoder RNN out at location n: ~ $$p(w_n | \mathbf{w}_{<n})$$

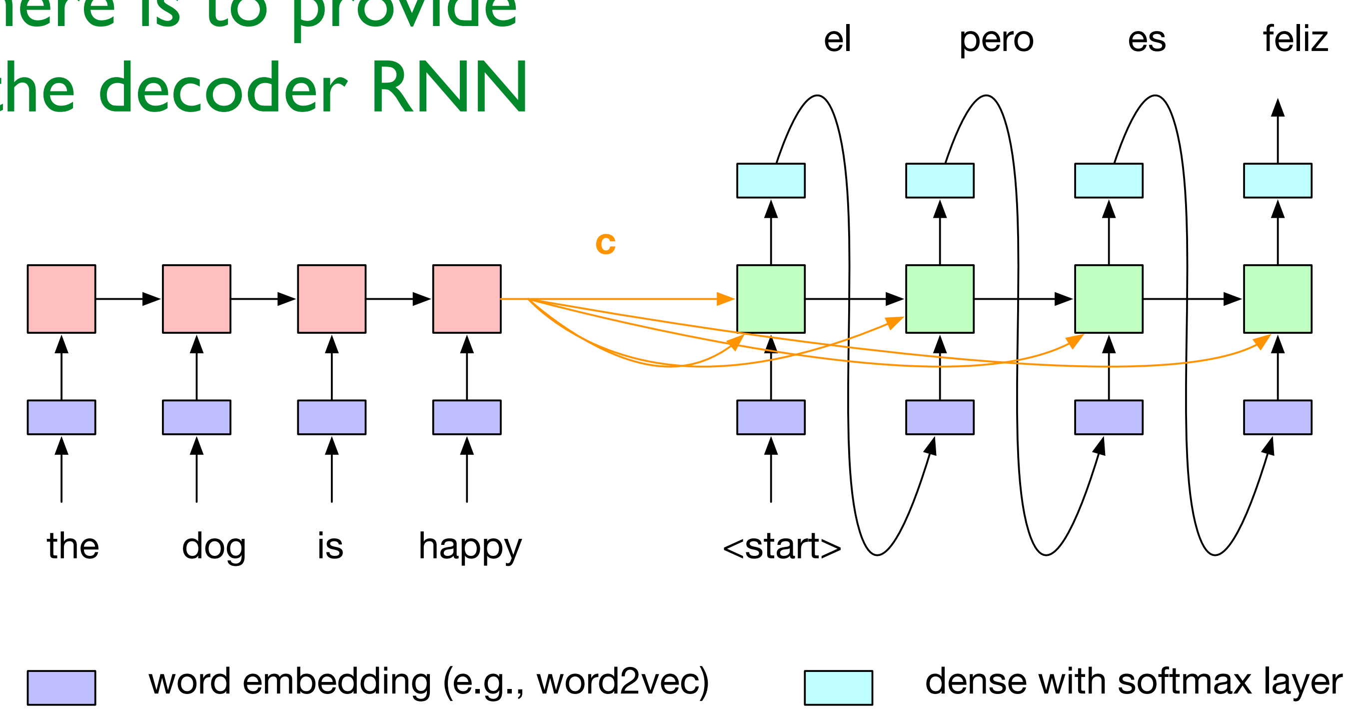
Limitations for RNNs

1. Embedding into final state only
2. This embedding must propagate through the decoder RNN
3. Sequential dependency of state machine means that it is hard to use parallelization in training



A partial solution to the second issue is here is to provide the sentence embedding to each step of the decoder RNN

Does not address issues 1 or 3



Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

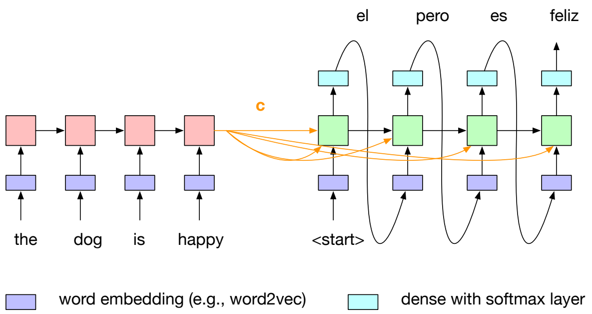
(this is the GRU paper too)

Limitations for RNNs

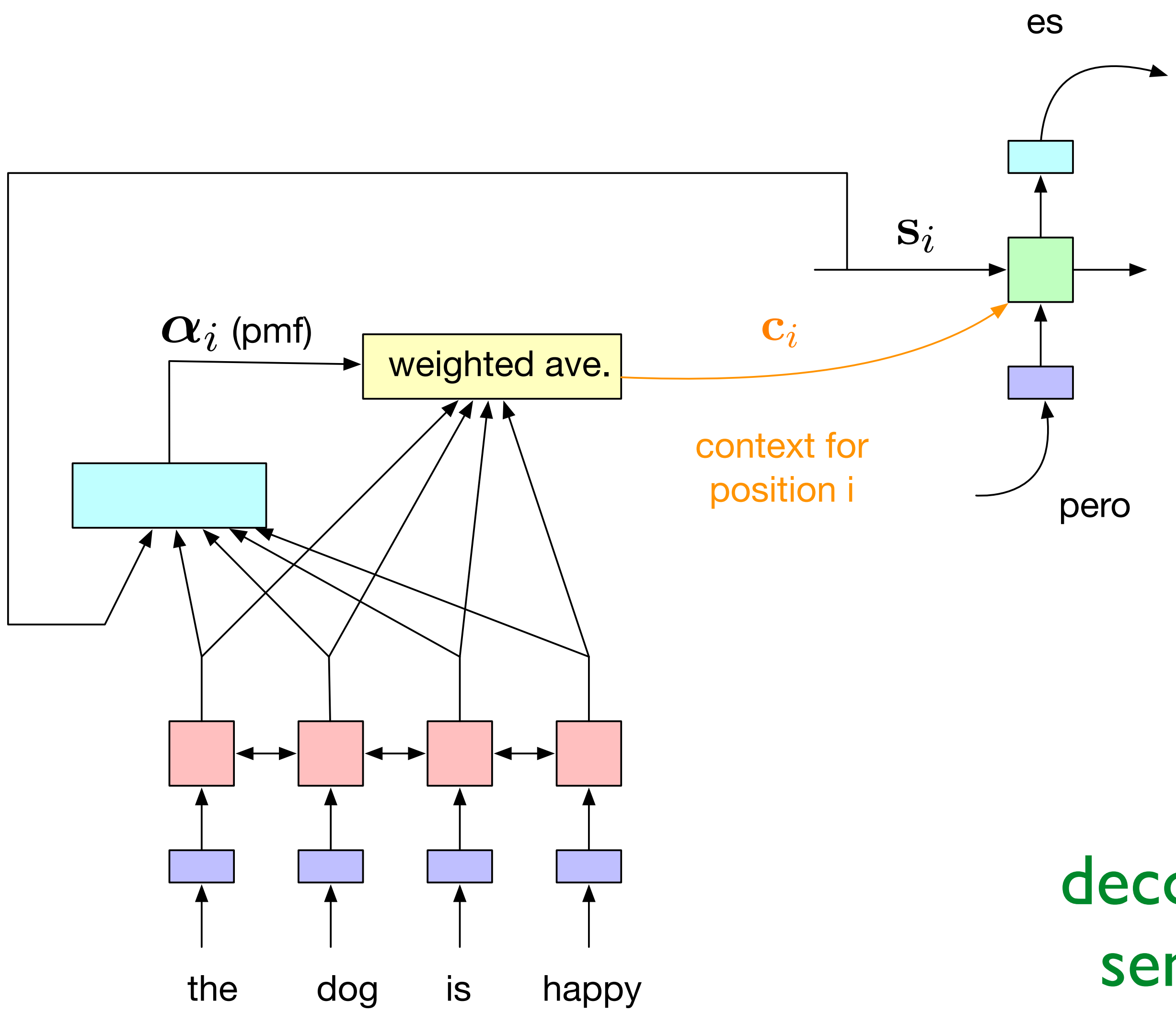
1. **Embedding into final state only**

2. ~~This embedding must propagate through the decoder RNN~~

3. **Sequential dependency of state machine means that it is hard to use parallelization in training**



This is repeated for each index in the decoder



Encoder RNN out is concatenated bidirectional state

The context for decoder position i is a learned average (E{.}) over all of the encoder sates

attention model

decoder is learning where to look in the input sentence for each output sentence location

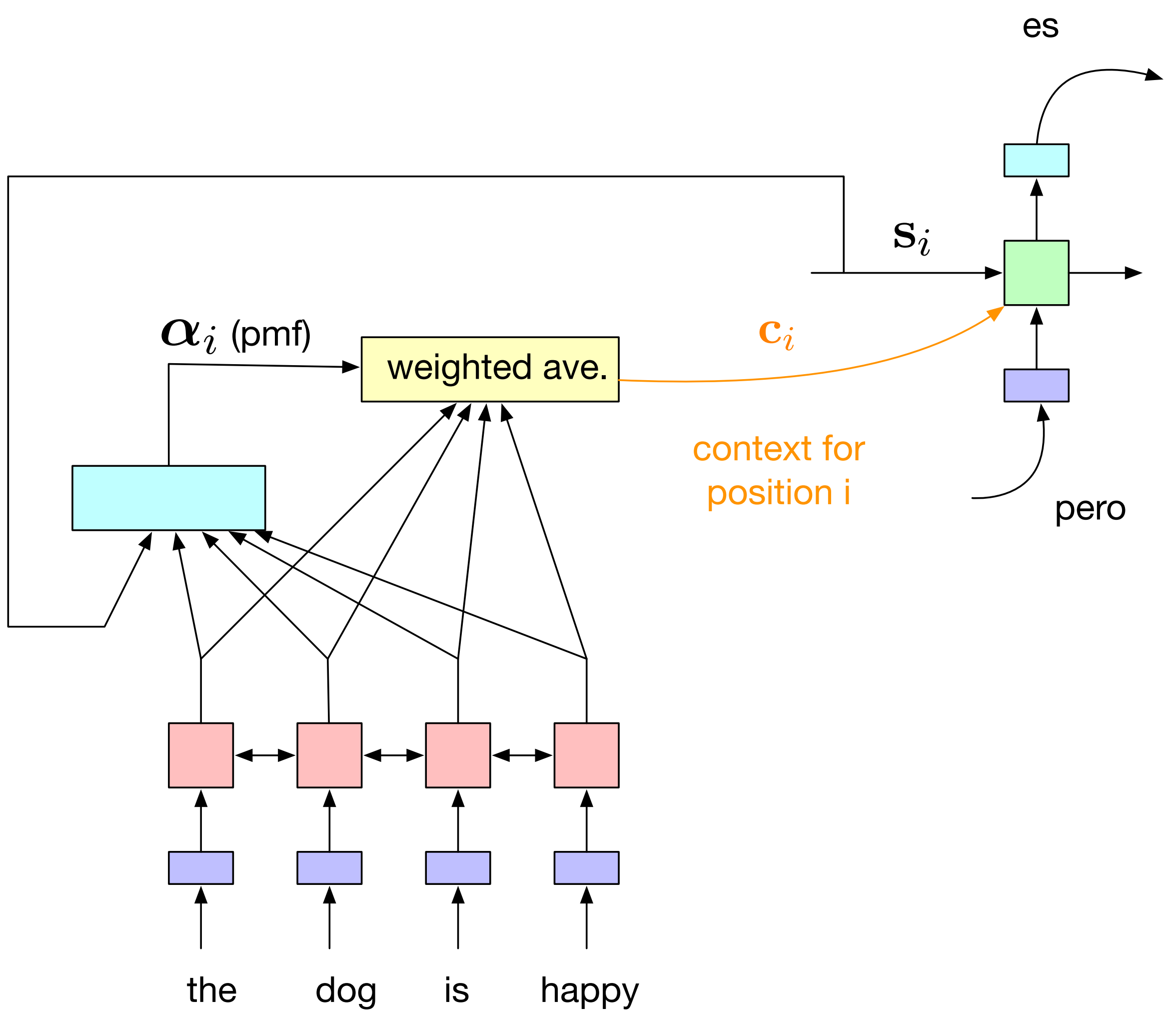
Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

Limitations for RNNs

1. ~~Embedding into final state only~~

2. ~~This embedding must propagate through the decoder RNN~~

3. **Sequential dependency of state machine means that it is hard to use parallelization in training**



This is effective for RNN-based models

The third issue remains — can we do something to get rid of the sequential nature of an RNN while still capitalizing on the "time" ordering traits?

Transformers

Based on an Encoder-Decoder architecture

No recursive computations, replace recurrent state machines with:

- **Positional encoding**
  - embed the position in sentence of word
- **Attention, attention, attention**
  - self-attention in encoder
  - self-attention in decoder
  - decoder-to-encoder attention

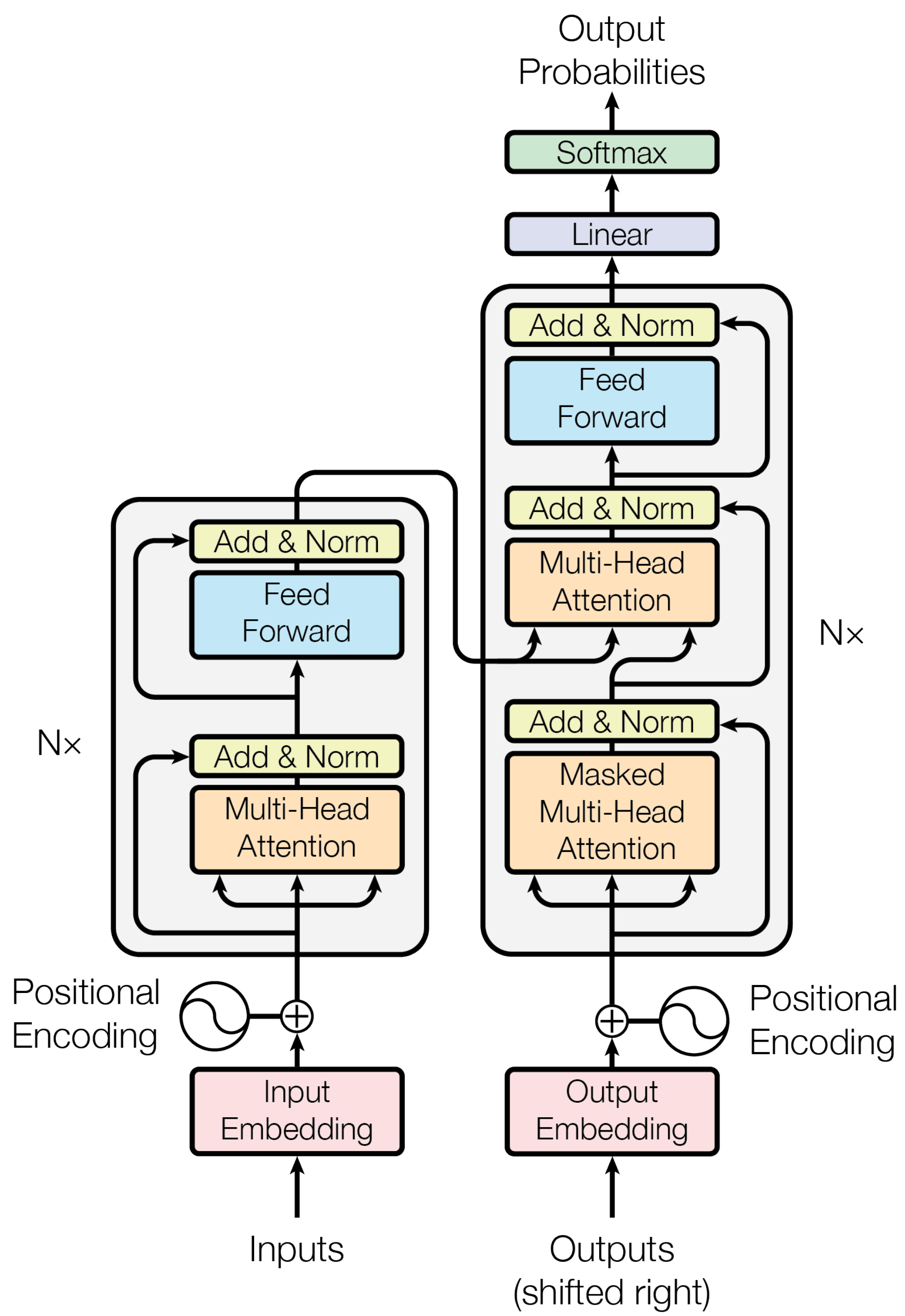


Figure 1: The Transformer - model architecture.

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

Transformers

I found the paper to be a little unclear at points and many of the "tutorials" just repeat the paper…

very nice video tutorial:

https://www.youtube.com/watch?v=z1xs9jdZnuY

Transformer (Attention is all you need)
Minsuk Heo 허민석

let's watch that!

Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

Transformers

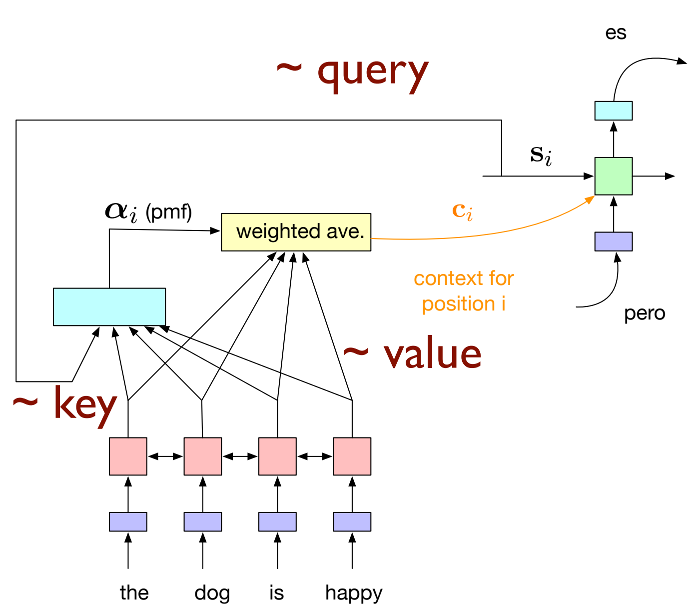
Notes from the video…

The key, value, and query are three different **embeddings** of the original words (and positional info) that utilize the entire sentence structure

query: (embedding for) word for which you would like to find attention region

key: (embedding for) candidate attention words

value: context embeddings to be averaged by softmax(**q . k**)



interpreting the previous attention-based RNN encoder, the key and the value are the same

Transformers

For each input position (word), i, we have: $\mathbf{q}_i \quad \mathbf{k}_i \quad \mathbf{v}_i$

For word 0, form dot product of q-0 with all keys: $\begin{bmatrix} \mathbf{q}_0^{\mathrm{t}}\mathbf{k}_0 & \mathbf{q}_0^{\mathrm{t}}\mathbf{k}_1 & \mathbf{q}_0^{\mathrm{t}}\mathbf{k}_2 & \cdots & \mathbf{q}_0^{\mathrm{t}}\mathbf{k}_{L-1} \end{bmatrix}$

Induce a pmf to average the values:

$$\mathbf{p}_0 = \mathrm{softmax}\left(\begin{bmatrix} \mathbf{q}_0^{\mathrm{t}}\mathbf{k}_0 & \mathbf{q}_0^{\mathrm{t}}\mathbf{k}_1 & \mathbf{q}_0^{\mathrm{t}}\mathbf{k}_2 & \cdots & \mathbf{q}_0^{\mathrm{t}}\mathbf{k}_{L-1} \end{bmatrix}\right)$$

$$= \begin{bmatrix} p_{0,0} & p_{0,1} & p_{0,2} & \cdots & p_{0,L-1} \end{bmatrix}$$

$$\mathbf{a}_0 = \sum_{i=0}^{L-1} p_{0,i}\mathbf{v}_i$$

packed matrix notation (column vectors)

$$\mathbf{B} = \begin{bmatrix} \mathbf{b}_0 & \mathbf{b}_1 & \mathbf{b}_2 & \cdots & \mathbf{b}_{L-1} \end{bmatrix}$$

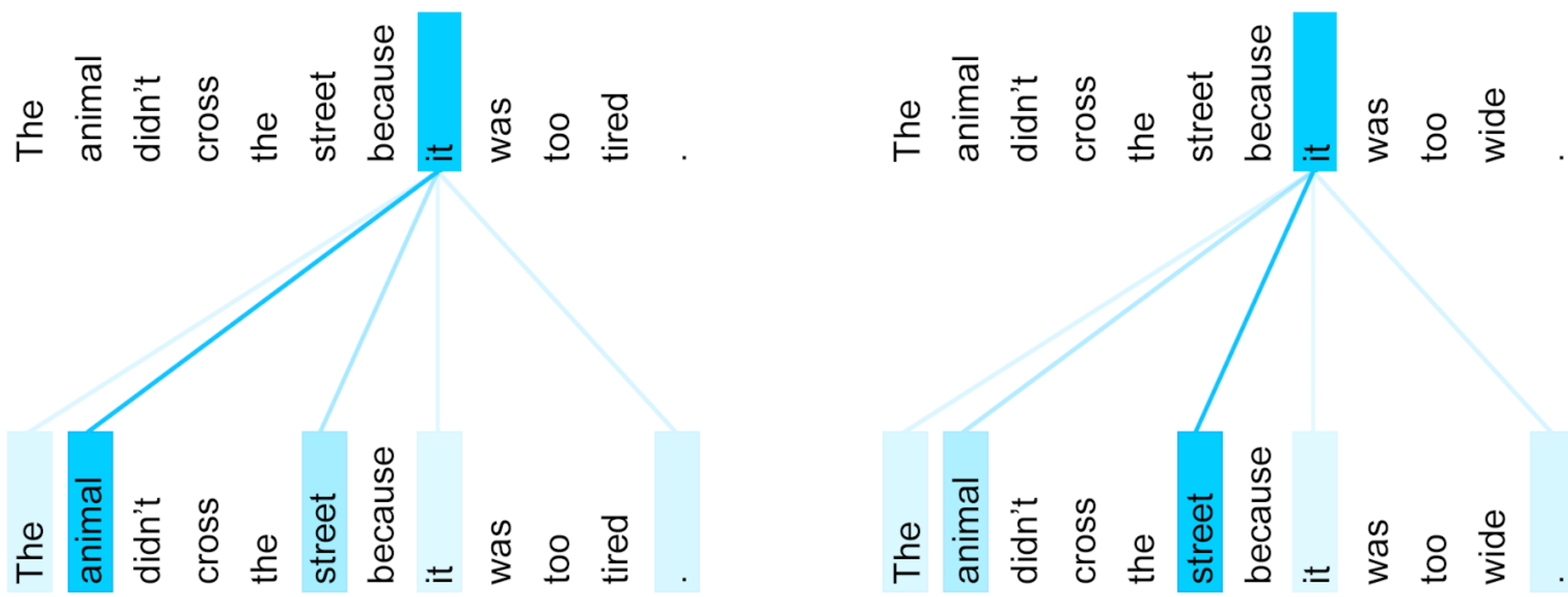
$$\mathbf{P}^{\mathrm{t}} = \mathrm{softmax}_{\mathrm{rows}}\left(\mathbf{Q}^{\mathrm{t}}\mathbf{K}\right)$$

$$\mathbf{A} = \mathbf{V}\mathbf{P} = \mathbf{V}\mathrm{softmax}_{\mathrm{rows}}\left(\mathbf{Q}^{\mathrm{t}}\mathbf{K}\right)$$

from paper (row vectors?): $\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$

Transformers

Why multihead attention?



Because of ambiguity in what "it" refers to in this sentence, different members multihead attention can look at different regions

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Transformers

Overview of the information flow

there is a GIF on
this page :-)

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Transformers

where can I get one?

Huggingface — implementations of transformers and related architectures

Transformers are now built into PyTorch (vers. >= 1.2)

eXtra Long Transformers

Transformers-XL

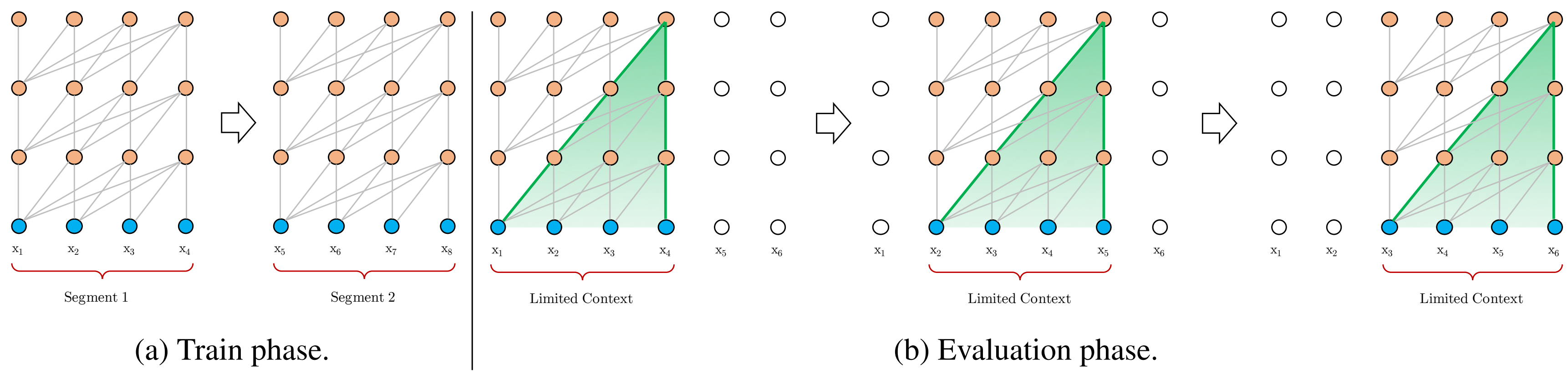


Figure 1: Illustration of the vanilla model with a segment length 4.

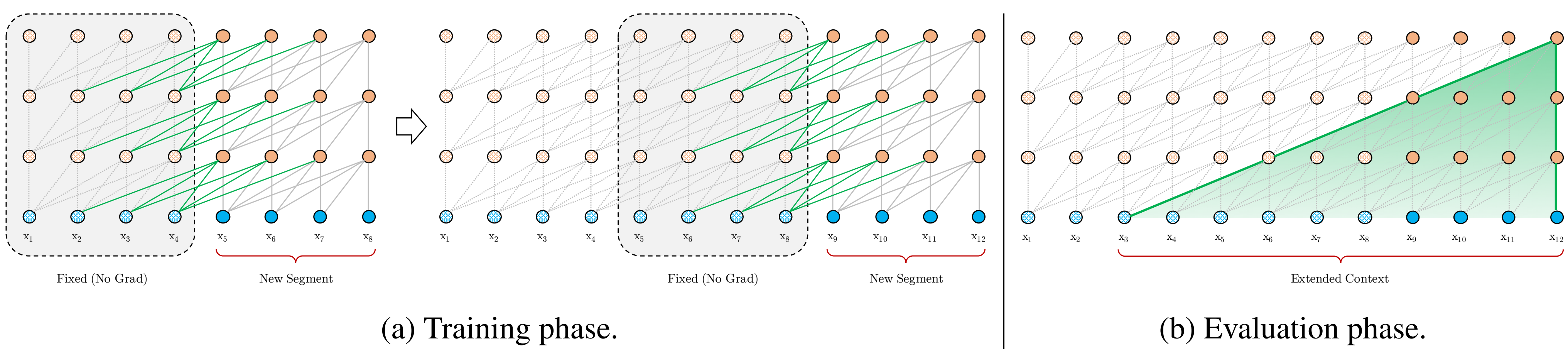


Figure 2: Illustration of the Transformer-XL model with a segment length 4.

Dai, Zihang, et al. "Transformer-xl: Attentive language models beyond a fixed-length context." arXiv preprint arXiv:1901.02860 (2019).

eXtra Long Transformers

Transformers-XL

Main advantages:

Lower complexity in inference mode (reuse computations)

Has longer term context

Dai, Zihang, et al. "Transformer-xl: Attentive language models beyond a fixed-length context." arXiv preprint arXiv:1901.02860 (2019).

Transformer-XL

note consistent language and long-term context

Context:
Kershaw started the 2010 season by posting a 3.07 ERA in April, but did so by walking 22 batters in 29 innings. On May 4, he had his worst start of his career against the Milwaukee Brewers at Dodger Stadium, throwing just 57 pitches in 11 / 3 innings, while retiring only four of the 13 batters he faced — including the pitcher. He was booed loudly upon being pulled from the game. Kershaw said after the game, " I didn't give our team any kind of chance. It's just not a good feeling to let your teammates down, let everybody down. It stings, it hurts. I 've got to figure things out. " Kershaw rebounded his next start by pitching an 8 inning two-hitter and out-dueling the then undefeated Ubaldo Jiménez. He credited his control of the slider being the major turning point for him. Later in the season, he was suspended for five games after hitting Aaron Rowand of the Giants with a pitch in a game on July 20. The incident occurred after both teams were given a warning following Giants ace Tim Lincecum hitting Matt Kemp earlier in the game. He threw his first career complete game shutout on September 14, 2010 also against San Francisco and finished the season with a record of 13 – 10 and a 2.91 ERA in 32 starts, pitching 2041 / 3 innings and recording 212 strikeouts.
= = = = 2011 season: 1st Cy Young Award = = = =
After finishing the 2010 season strong, the Dodgers named Kershaw as the Opening Day Starter for the 2011 season. On May 29, he pitched the second complete-game shutout of his career, striking out 10 while winning a two-hitter against the Florida Marlins, 8 – 0; he also had two singles and an RBI, scoring twice in the game. He produced his third career shutout on June 20, a two-hit, 11-strikeout effort against the Detroit Tigers. Kershaw became the first Dodgers starter to strike out the side in the 9th inning since Sandy Koufax's perfect game. In his next start, on June 26, Kershaw pitched another complete game (against the Los Angeles Angels of Anaheim). He became the first Dodger starter to have back-to-back complete game victories since Jeff Weaver in the 2005 season and the first Dodger to have double-digit strikeouts in consecutive starts since Chan-Ho Park in the 2000 season. He was awarded the National League Player of the Week award for the week of June 20 – 26 as a result of those two starts. Midway through June, Kershaw had amassed 32 career victories

XL Generation:
, tying him with Luis Castillo for the third best start total in the league and trailing only Johnny Bench and Eddie Mathews (37). The Dodgers signed Kershaw to a one-year deal on July 14, 2011, with a player option for 2012, in a deal worth just under $ 300,000. The Dodgers entered the 2011 season as the major league leaders, and Kershaw had the best record in the National League in saves (39), games started (78) and walks (120); he also had the lowest win / loss ratio in the league (1.87), and he had the worst on-base percentage (.349). He had the best ERA (2.00), most wins (16), or most shutouts (29) in his career (25), while leading the league in losses per nine innings pitched (4.01). Following the 2011 season, the Dodgers traded Kershaw to the Kansas City Royals on July 29 for pitcher Javier Torres.
= = = Kansas City Royals = = =
= = = = 2012 season = = = =
During spring training, Kershaw played very well. He was selected to spring training as a relief pitcher for the Royals for the 2012 season. After an injury to closer Javier Vázquez, he was activated on April 29 to replace Matt Holliday in the Royals ' starting rotation. In his only start with the Royals, on August 6, 2012, Kershaw struck out five batters in seven innings pitched to help the Royals to their first victory in franchise history. On September 27, 2012, it appeared Kershaw was going to pitch a complete game shutout against the Detroit Tigers, but did not manage to do so since the Tigers won 3 – 1. At the conclusion of the season, Kershaw was named Major League Baseball's Most Valuable Player, was chosen to the All-Star Game at Busch Stadium and was named to the All-Star Game as the starting pitcher at shortstop. The Royals announced on February 4, 2013 that Kershaw would spend 2013 as starting pitcher, though he was expected to miss the season as a result of a shoulder injury.
= = = = 2013 season = = = =
On May 17, 2013, Kershaw sustained another back injury and did not start in August and October 2013. He appeared in 22 starts, all starts, finishing with a strikeout-to-walk ratio of 1.50 and a 2.91 ERA. He also had the third most strikeouts in the league: 10. On May 20, 2013, he

Reference:
, a 3.15 ERA and 593 career strikeouts in 568.2 innings. According to the Elias Sports Bureau, Kershaw was the first 23-year-old pitcher to have that many victories, an ERA that low and an average of more than one strikeout per inning since ERA became an official statistic in 1910. Kershaw was selected to the National League team for the 2011 Major League Baseball All-Star Game, his first All-Star selection. In the month of July, Kershaw was 4 – 1 with a 2.02 ERA and NL-leading 45 strikeouts, earning him the National League Pitcher of the Month Award. On August 23, he struck out Matt Holliday of the St. Louis Cardinals for his 200th strikeout of the season and became the 10th Dodger pitcher to record back-to-back 200 strikeout seasons and the first since Chan-Ho Park did it in the 2001 season. Kershaw finished the 2011 season by leading the NL with 21 wins, 248 strikeouts and a 2.28 ERA, winning the NL pitching Triple Crown, the first Triple Crown winner since Jake Peavy of the 2007 San Diego Padres and the first Dodger since Sandy Koufax won it in the 1966 season. Justin Verlander of the Detroit Tigers won the American League Triple Crown the same season, marking the first major-league season since 1924 to feature Triple Crown-winning pitchers in both leagues. Kershaw's 21 wins were the most by a Dodger pitcher since Orel Hershiser won 23 during the 1988 season. His ERA was the lowest by a Dodger since Hershiser's 2.03 in the 1985 season, his strikeouts were the most by a Dodger since Koufax's 317 in 1966 and his 233 1 / 3 innings pitched were the most since Chan Ho Park pitched 234 in 2001. Since 1965 when Koufax did it, Peavy and Kershaw are only two pitchers in the National League have led the league in wins, strikeouts, ERA, and WHIP (walks plus hits per inning pitched). Kershaw also became just the second <unk> to have a 240-plus strikeouts in a season before the age of 24, joining Vida Blue. After the season, Kershaw was awarded the Warren Spahn Award as the best left-handed pitcher in 2011, the Players Choice Award for Most Outstanding National League pitcher, the Gold Glove Award as the top fielding pitcher in the NL and the Sporting News (TSN) National League Pitcher of the Year. He was additionally selected as the starting pitcher for the TSN NL All-Star Team. On November 17, he was honored with the National League Cy Young Award, making him the youngest Cy Young winner since Dwight Gooden

Table 11: Example 1 – 500 tokens generated by XL using a snippet from the Wikitext-103 test set as initial context. The sample is randomly generated without any cherry picking.

Original Wikipedia page: https://en.wikipedia.org/wiki/Clayton_Kershaw

There are many interesting observations from this example:

- Firstly, Kershaw never went to Royals in real life. Despite that, Transformer-XL stays on the fully imagined topic and keeps hallucinating the experience of Kershaw in Royals across the generated text.
- Secondly, notice that XL correctly tracks the chronological order from 2011 to 2012 and to the finally 2013 season in the section titles.
- In addition, notice that Transformer-XL accurately uses the the phrase "another back injury" in the 2013 season paragraph, since it has talked about one earlier injure in the 2012 season. This shows again Transformer-XL's ability of capturing long-term dependency.

Dai, Zihang, et al. "Transformer-xl: Attentive language models beyond a fixed-length context." arXiv preprint arXiv:1901.02860 (2019).

BERTS

Bidirectional Encoder Representations from Transformers

Train a transformer encoder for two generic language tasks to capture general language representations

Use fine-tuning to adapt this baseline model to solve a large number of specific NLP tasks

Fine-tuning the publicly-available baseline models produces SOTA performance on a wide range of NLP tasks

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

BERTS

Bidirectional Encoder Representations from Transformers

Pre-training (original) training takes ~ week on multiple GPUs

Fine-tuning for a specific task can be achieved in ~ 2 hours on a GPU

Fine-tuning involves training the entire network (not just an add-on layer)

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

BERTS

Pre-training involves two tasks:

1. Masked Language Model (MLM)
2. Next Sentence Prediction (MLM)

can accommodate one- or two-sentence inputs

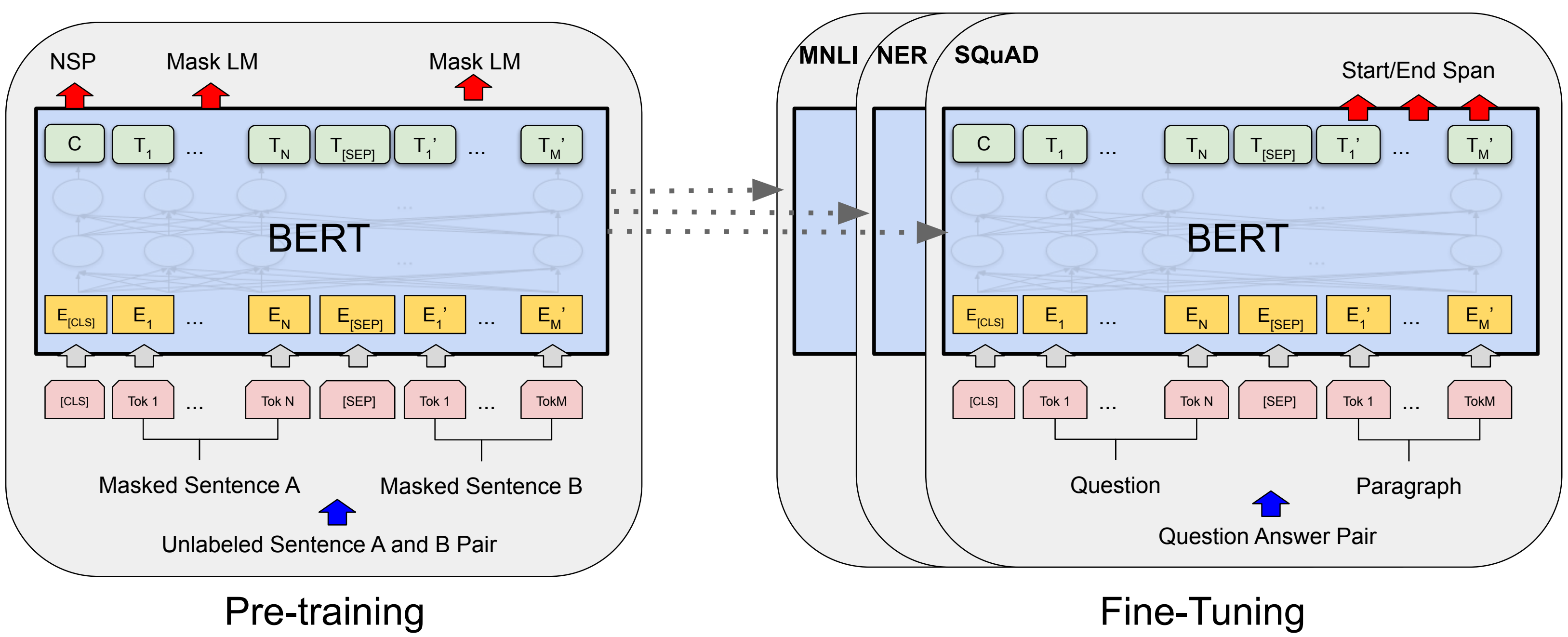


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

BERTS

Masked Language Model (MLM)

```
Input: the man went to the [MASK1] . he bought a [MASK2] of milk.
Labels: [MASK1] = store; [MASK2] = gallon
```

mask 15% of words in sentences

Note that self-attention method in the Transformer encoder make this a truly bidirectional model at all layers ~ **"deeply bidirectional"**

$p(w_n|\mathbf{w}_{<n})$ left-to-right LM

$p(w_n|\mathbf{w}_{>n})$ right-to-left LM

$p(w_n|\mathbf{w}_{!=n})$ bidirectional LM

https://github.com/google-research/bert

BERTS

Example of why deep bidirectionality is powerful

I made a bank deposit

I made a bank shot from the free-throw line

I made a bank along the creek

BERT is "deeply bidirectional" since all of the self-attention layers utilize bidirectional context

https://github.com/google-research/bert

BERTS

Masked Language Model (MLM)

```
Input: the man went to the [MASK1] . he bought a [MASK2] of milk.
Labels: [MASK1] = store; [MASK2] = gallon
```

mask 15% of words in sentences

Since [mask] token will not be in fine-tuning data, some of the deleted words are replaced with randomly selected words instead of [mask]

https://github.com/google-research/bert

BERTS

Next Sentence Prediction

```
Sentence A: the man went to the store .
Sentence B: he bought a gallon of milk .
Label: IsNextSentence
```

```
Sentence A: the man went to the store .
Sentence B: penguins are flightless .
Label: NotNextSentence
```

This is a binary classification problem

https://github.com/google-research/bert

BERTS

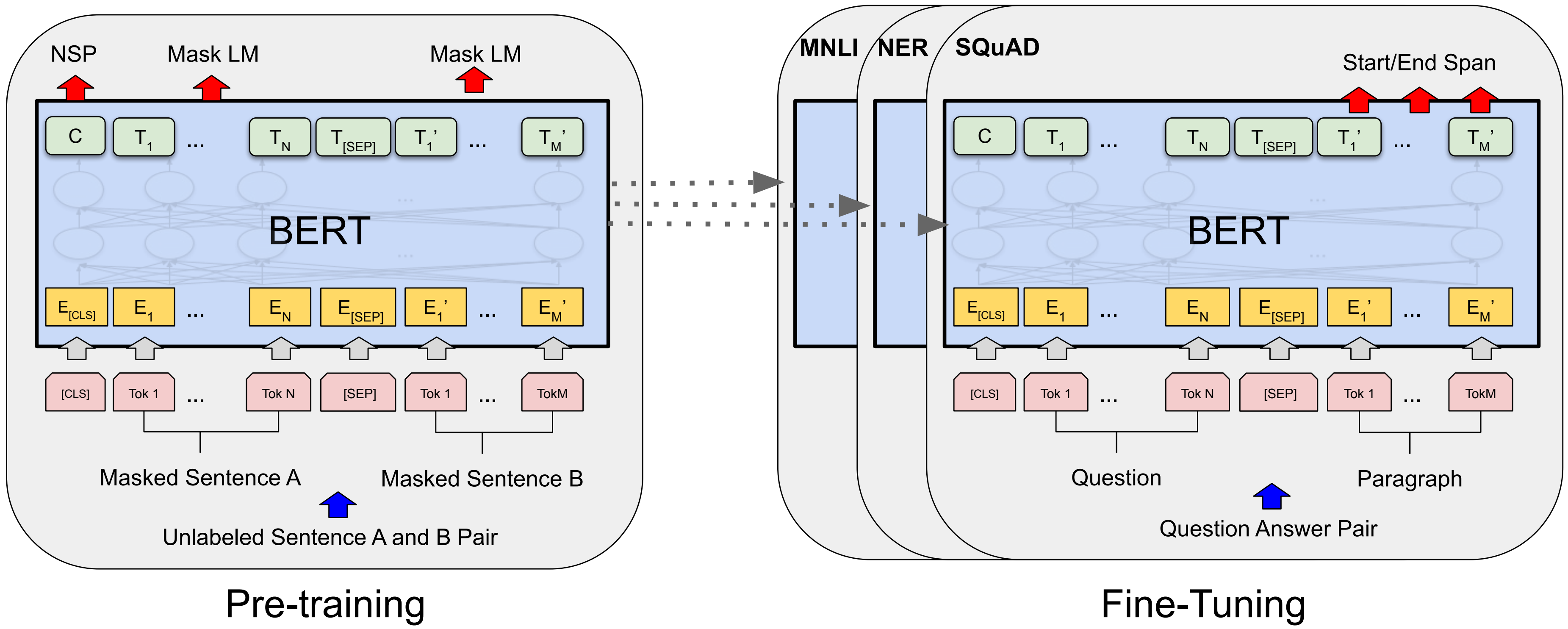


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

can be trained on **unlabeled** text corpus

[Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).]

BERTS

models:

BERT_base: 12 Transformer blocks
768 dimensional k/v/q
12 attention heads

110 Million parameters

BERT_large: 24 Transformer blocks
1024 dimensional k/v/q
16 attention heads

340 Million parameters

similar in size to the larger CNNs, but no parameter reuse, so should have lower computational complexity

other variants since then (see GitHub page)

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

BERTS

	H=128	H=256	H=512	H=768
L=2	**2/128 (BERT-Tiny)**	2/256	2/512	2/768
L=4	4/128	**4/256 (BERT-Mini)**	**4/512 (BERT-Small)**	4/768
L=6	6/128	6/256	6/512	6/768
L=8	8/128	8/256	**8/512 (BERT-Medium)**	8/768
L=10	10/128	10/256	10/512	10/768
L=12	12/128	12/256	12/512	**12/768 (BERT-Base)**

L: number of transformer blocks

H: k/v/q vector dimension

A: number of attention heads

- `BERT-Large, Uncased (Whole Word Masking)` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Large, Cased (Whole Word Masking)` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Uncased` : 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Large, Uncased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Cased` : 12-layer, 768-hidden, 12-heads , 110M parameters
- `BERT-Large, Cased` : 24-layer, 1024-hidden, 16-heads, 340M parameters
- `BERT-Base, Multilingual Cased (New, recommended)` : 104 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Multilingual Uncased (Orig, not recommended)` **(Not recommended, use `Multilingual Cased` instead)**: 102 languages, 12-layer, 768-hidden, 12-heads, 110M parameters
- `BERT-Base, Chinese` : Chinese Simplified and Traditional, 12-layer, 768-hidden, 12-heads, 110M parameters

https://github.com/google-research/bert

BERTS

pre-training data:

Books Corpus (800 million words)

English Wikipedia (2,500 million words)

important that sentences come from contiguous text (no shuffling sentences) for the next-sentence prediction task

pre-training computation:

"4 days on 4 to 16 parallel TPUs"

TPU = tensor processor unit ~ google's highly optimized GPU

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

BERTS

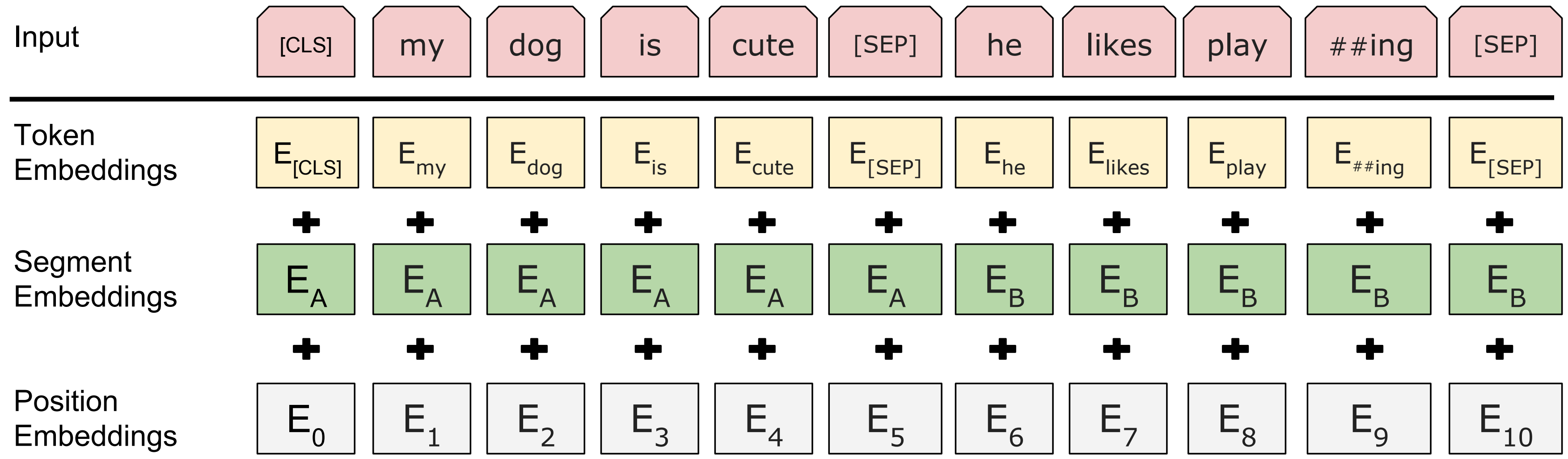


Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

positional and segment embeddings added to word (token) embedding

uses WordPiece embeddings (which use some word fragments)

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

BERTS

GLUE (General Language Understanding Evaluation):

fine-tuning add a classification layer (H x N_classes)

"diverse natural language understanding tasks"

System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

BERTS

SQuAD (Stanford Question and Answer Dataset):

Question with a passage that contains the answer

introduce **start** and **stop** markers in training to delineate the answer

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

System	Dev EM	Dev F1	Test EM	Test F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	86.3	89.0	86.9	89.5
#1 Single - MIR-MRC (F-Net)	-	-	74.8	78.0
#2 Single - nlnet	-	-	74.2	77.1
Published				
unet (Ensemble)	-	-	71.4	74.9
SLQA+ (Single)	-		71.4	74.4
Ours				
$\text{BERT}_{\text{LARGE}}$ (Single)	78.7	81.9	80.0	83.1

Table 3: SQuAD 2.0 results. We exclude entries that use BERT as one of their components.

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

BERTS

SWAG (Situations With Adversarial Generations):

Given a sentence, choose most plausible continuation

System	Dev	Test
ESIM+GloVe	51.9	52.7
ESIM+ELMo	59.1	59.2
OpenAI GPT	-	78.0
$\mathrm{BERT}_{\mathrm{BASE}}$	81.6	-
$\mathrm{BERT}_{\mathrm{LARGE}}$	**86.6**	**86.3**
Human (expert)†	-	85.0
Human (5 annotations)†	-	88.0

Table 4: SWAG Dev and Test accuracies. †Human performance is measured with 100 samples, as reported in the SWAG paper.

Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

NLP Oriented Projects in EE599

Session 4A: Natural Language Processing and Social Network Analysis **Chair: Arnab Sanyal** **Zoom meeting: 954 5825 6362**			
4:00 PM	Transformer and self-attention mechanism	Chengwei Wei, Shiming Gao	AS
4:20 PM	Named Entity Recognition with BERT and Transformers	Wenjing Lin, Jiaqi Liu	OA
4:40 PM	Rumor Detection on Social Media with Graph Convolutional Network	Murong Yue, Dejia Hao, Meng-Ju Lee	KH
5:00 PM	Sentiment Extraction on Social Media	Jianing Luo, Xuejing Tan, Qian Wang	KH
5:20 PM	Deep Grammar Corrector	Zixi Liu, Mingxi Lei	KC