CITY NAME

DEV DAY

DEV DAY
CITY NAME

MLT3

Natural Language Processing: concepts, algorithms & use cases

Speaker Name
Job Title
Company/Org Name

What to expect

1 – Word Vectors

2 – Algorithms

- Word2Vec
- GloVe
- FastText, BlazingText
- ELMo
- BERT
- XLNet

3 – Use cases & demos

- Word vectors
- Word similarity
- Word analogy
- Text classification
- Sentiment analysis

4 – Getting started

Problem statement

- NLP is a major field in AI
  - Text classification, machine translation, text generation, chat bots, vocal assistants, etc.
  - You could even say that strong AI requires efficient NLP
- NLP apps require a language model in order to predict the next word
  - Given a sequence of words (w_1, … , w_n), predict w_{n+1} that has the highest probability
- Vocabulary size can be hundreds of thousands of words
  … in millions of documents
- Can we build a compact mathematical representation of language, that will help with a variety of downstream NLP tasks?

« *You shall know a word by the company it keeps* », Firth (1957)

- Word vectors are built from co-occurrence counts
  - Also called word embeddings
  - High dimensional: at least 50, up to 300

- Words with similar meanings should have similar vectors
  - "car" ≈ "automobile" ≈ "sedan"

- The distance between vectors for the same concepts should be similar
  - distance ("Paris", "France") ≈ distance("Berlin", "Germany")
  - distance("hot", "hotter") ≈ distance("cold", "colder")

High-level view

1. Start from a large text corpus (100s of millions of words, even billions)

2. Preprocess the corpus
   - Tokenize: « hello, world! » → « <BOS>hello<SP>world<SP>!<EOS>»
   - Multi-word entities: « Rio de Janeiro » → « rio_de_janeiro »

3. Build the vocabulary
   - Remove very rare words?

4. Learn vector representations for all words

… or simply use (or fine-tune) pre-trained vectors (more on this later)

DEV DAY

Algorithms

aws

Word2Vec (2013)

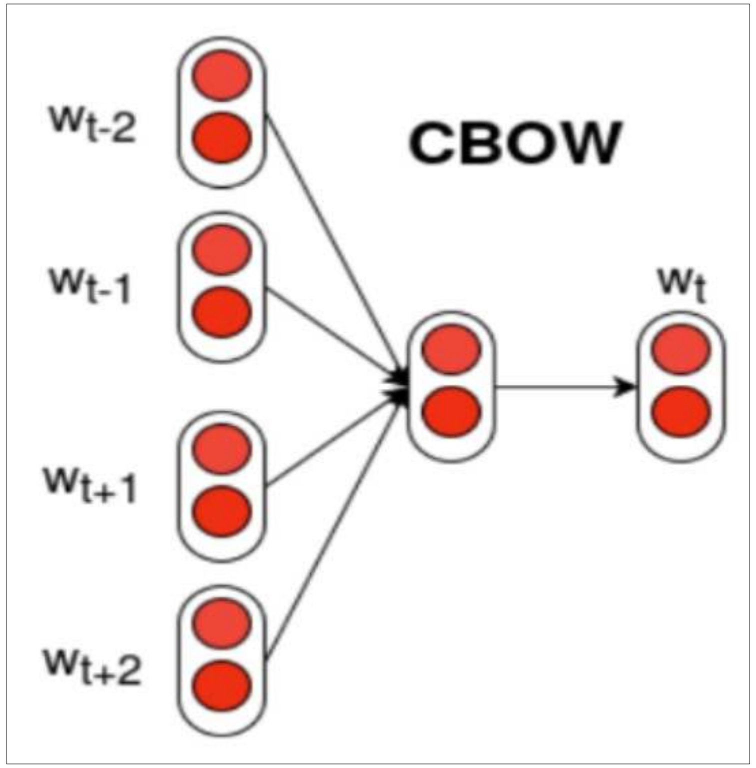
https://arxiv.org/abs/1301.3781
https://code.google.com/archive/p/word2vec/

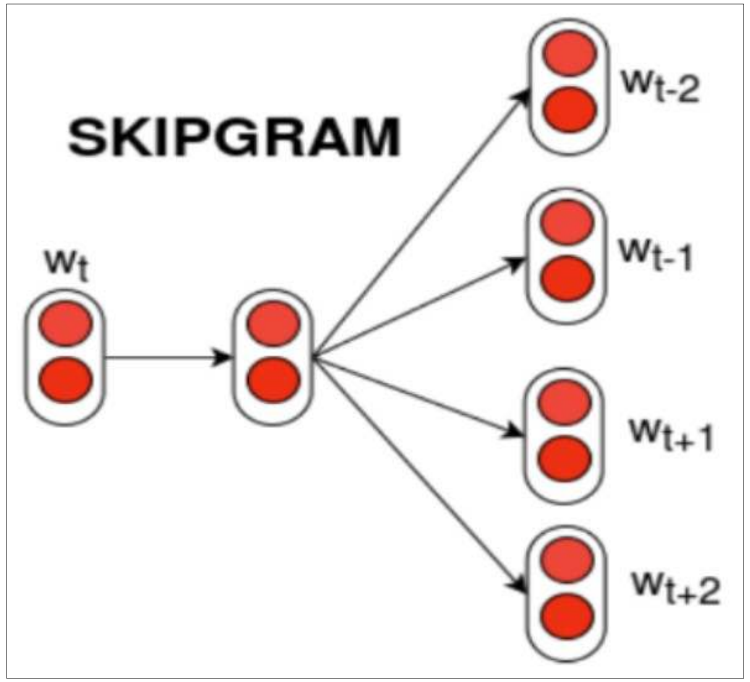
- Continuous bag of words (CBOW):
  - the model predicts the current word from surrounding context words
  - Word order doesn't matter (hence the 'bag of words')

- Skipgram
  - the model uses the current word to predict the surrounding window of context words
  - This may work better when little data is available

- CBOW trains faster, skipgram is more accurate

- C code, based on shallow neural network





Source: Wikipedia

Global Vectors aka GloVe (2014)

https://nlp.stanford.edu/projects/glove/
https://github.com/stanfordnlp/GloVe
https://www.quora.com/How-is-GloVe-different-from-word2vec

- Performance generally similar to Word2Vec
- Pre-trained models: up to 840 billion tokens, 2.2 million vocabulary, 300 dimensions
- C code, based on matrix factorization

FastText (2016)

https://arxiv.org/abs/1607.04606
https://arxiv.org/abs/1802.06893
https://fasttext.cc/
https://www.quora.com/What-is-the-main-difference-between-word2vec-and-fastText

- Extension of Word2Vec: each word is treated as a set of subwords aka character n-grams
  - « Computer », n=5 : <START>Comp , compu, omput, mpute, puter, uter<END>
  - A word vector is the sum or average of its subword vectors
- Subwords help with rare/unknown/mispelled words, as they share subwords with known words
  - « Computerization » and « Cmputer » should be close to « Computer »

- Unsupervised learning: compute word vectors, with pre-trained vectors for 294 languages
- Supervised learning: use word vectors for multi-label, multi-class text classification
- Also language detection for 170 languages

- Multithreaded C++ code, with Python API

BlazingText (2017)

https://dl.acm.org/citation.cfm?id=3146354
https://aws.amazon.com/blogs/machine-learning/enhanced-text-classification-and-word-vectors-using-amazon-sagemaker-blazingtext/

- Amazon-invented algorithm, available in Amazon SageMaker
- Extends FastText with GPU capabilities

- Unsupervised learning: word vectors
  - 20x faster
  - CBOW and skip-gram with subword support
  - Batch skip-gram for distributed training
- Supervised learning: text classification
  - 100x faster
  - Models are compatible with FastText

	Word2Vec (unsupervised learning)			Text Classification (supervised learning)
Modes	Skip-gram (supports subwords)	CBOW (supports subwords)	batch_skipgram	supervised
Single CPU instance	✔	✔	✔	✔
Single GPU instance (with 1 or more GPUs)	✔	✔		✔*
Multiple CPU instances			✔	

Limitations of Word2Vec (and family)

- Some words have different meanings (aka polysemy)
  - « *Kevin, stop throwing rocks!* » vs. « *Machine Learning rocks* »
  - Word2Vec encodes the different meanings of a word into the same vector
- Bidirectional context is not taken into account
  - Previous words (left-to-right) and next words (right-to-left)

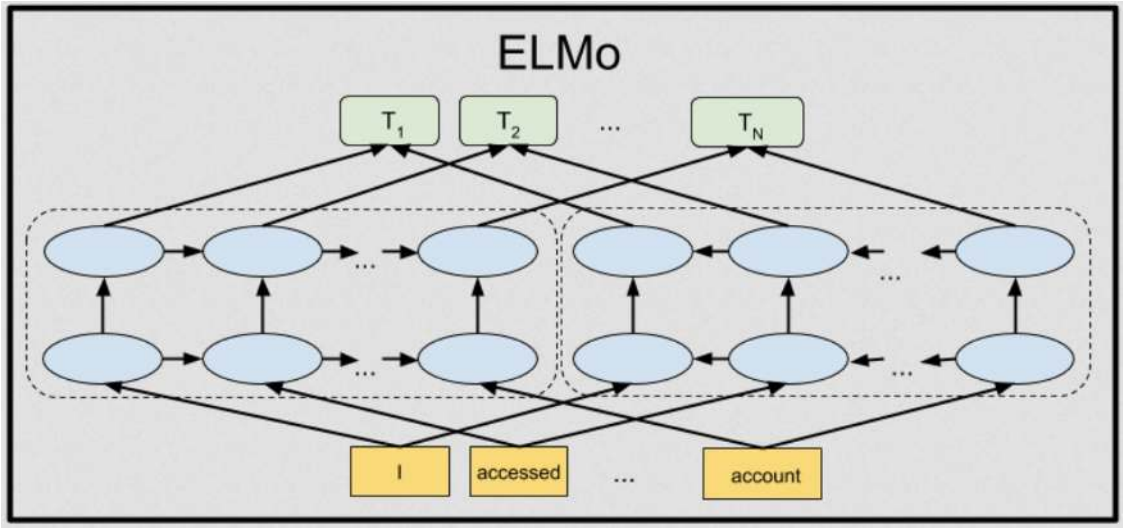
Embeddings from Language Models aka ELMo (02/2018)

https://arxiv.org/abs/1802.05365
https://allennlp.org/elmo

- ELMo generates a context-aware vector for each word
  - Character-level CNN
  - Bidirectional context, with two unidirectional LSTMs
    → No cheating possible (can't peek at the future)
  - "Deep" embeddings, reflecting output from all layers

- No vocabulary, no vector file:
  you need to use the model itself

- Reference implementation with TensorFlow



Source: Google

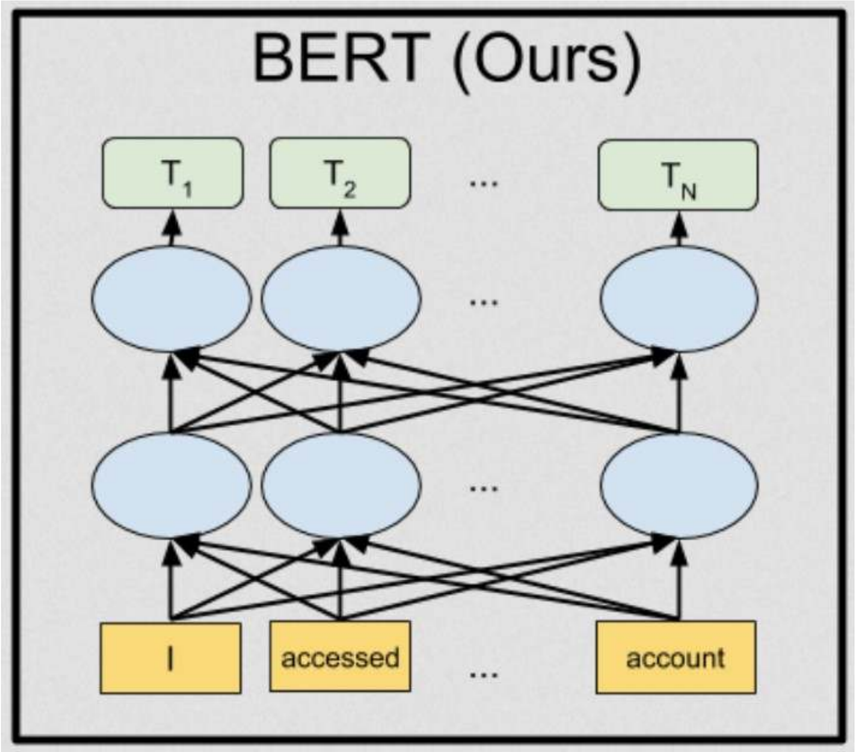
	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Source: ELMo paper

Bidirectional Encoder Representations from Transformers aka BERT (10/2018)

https://arxiv.org/abs/1810.04805
https://github.com/google-research/bert
https://www.quora.com/What-are-the-main-differences-between-the-word-embeddings-of-ELMo-BERT-Word2vec-and-GloVe

- BERT improves on ELMo
  - Replace LSTM with Transformers, which deal better with long-term dependencies
  - Truly bidirectional architecture: left-to-right and right-to-left contexts are learned by the same network
  - Words are randomly masked during training to prevent cheating
- Pre-trained models: BERT Base and BERT Large
  - Masked word prediction
  - Next sentence prediction
- Reference implementation with TensorFlow



Source: Google

Limitations of BERT

- BERT cannot handle more than 512 input tokens
- BERT masks words during training, but not during fine-tuning (aka training/fine-tuning discrepancy)
- BERT isn't trained to predict the next word, so it's not great at text generation
- BERT doesn't learn dependencies for masked words
  - Train « I am going to <MASK> my <MASK> » on « walk » / « dog », « eat » / « breakfast », and « debug » / « code ».
  - BERT could legitimately predict « I am going to eat my code » or « I am going to debug my dog » :-/

XLNet (06/2019)

https://arxiv.org/abs/1906.08237
https://github.com/zihangdai/xlnet

- XLNet beats BERT at 20 tasks
- XLNet uses bidirectional context, but words are randomly permuted
    - No cheating possible
    - No masking required
- XLNet Base and XLNet Large
- Reference implementation with TensorFlow

07/2019: ERNIE 2.0 (Baidu)

beats BERT & XLNet

https://github.com/PaddlePaddle/ERNIE/

Train yourself or not?

- Word2Vec and friends
  - Try pre-trained embeddings first
    - Check that the training corpus is similar to your own data
    - Same language, similar vocabulary
  - Remember that subword models will help with unknown / mispelled words
  - If you have exotic requirements AND lots of data, training is not expensive

- ElMo, BERT, XLNet
  - Training is very expensive: several days using several GPUs
  - Fine-tuning is cheap: just a few GPU hours for SOTA results
  - Fine-tuning scripts and pre-trained models are available: start there!

- In both cases, you'll still have to pre-process data (yeaaaaah)

DEV DAY

Use cases & demos

Demo: training Word2Vec subword vectors with BlazingText on Amazon SageMaker

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/introduction_to_amazon_algorithms/blazingtext_word2vec_subwords_text8

Word similarity

- Words with a similar meaning are expected to have similar vectors
  - 'cosine similarity', i.e. normalized dot product

    -1 → words are not similar

    +1 → words are similar

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

- Using word vectors for your vocabulary:
  - Pick a word
  - Compute the cosine similarity of its vector with respect to all other vectors
  - Keep the top 'k' cosines similarities
  - Return the corresponding words

nearest neighbors of ***frog***	**Litoria**	**Leptodactylidae**	**Rana**	**Eleutherodactylus**
Pictures				

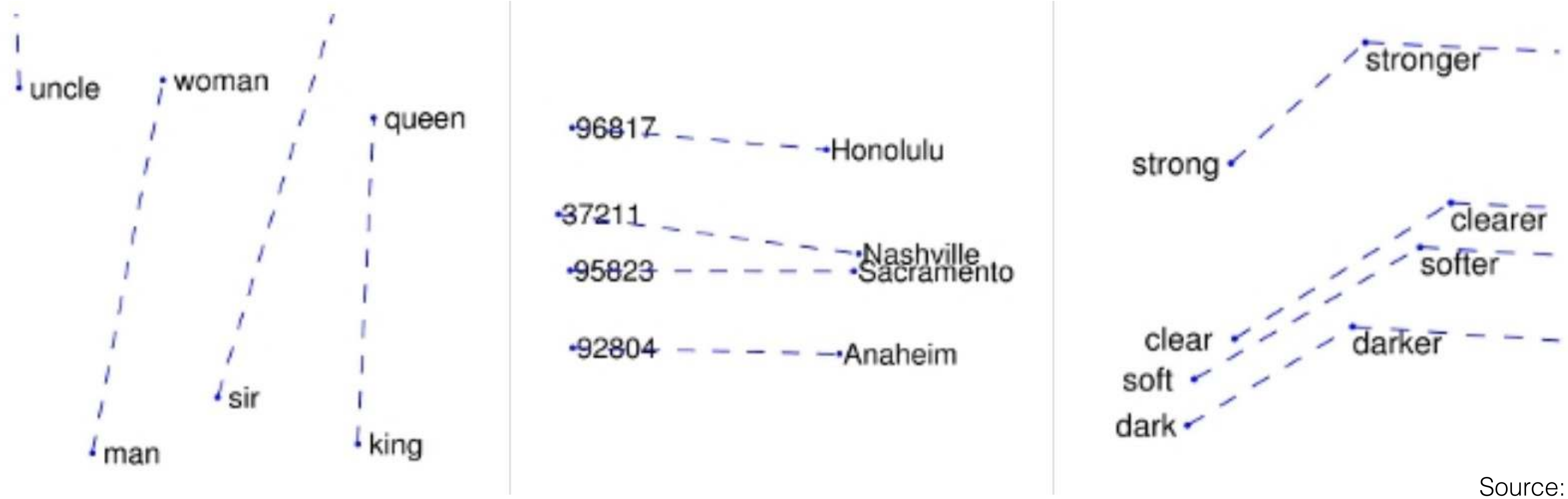
Source: GloVe

Word analogy

- The distance between two word vectors defines the relationship between the two words.
- A similar distance between two other vectors reflects a similar relationship

- « King » - « Man » ≈ « Queen » - « Woman »
- « King » - « Man » + « Woman » ≈ « Queen »
- Meaning: « Man » is to « King » what « Woman » is to « Queen »

- Now we can ask :« Paris » is to « France » what « Rome » is to… ?
- Answer: vector closest to « France » - « Paris » + « Rome », hopefully « Italy » ☺



aws

Source: GloVe

Demo: finding similarities and analogies with Gluon NLP and pre-trained GloVe embeddings

https://gitlab.com/juliensimon/dlnotebooks/gluonnlp/

Demo: embeddings with ELMo on TensorFlow

https://gitlab.com/juliensimon/dlnotebooks/blob/master/nlp/ELMO%20TensorFlow.ipynb

Text classification

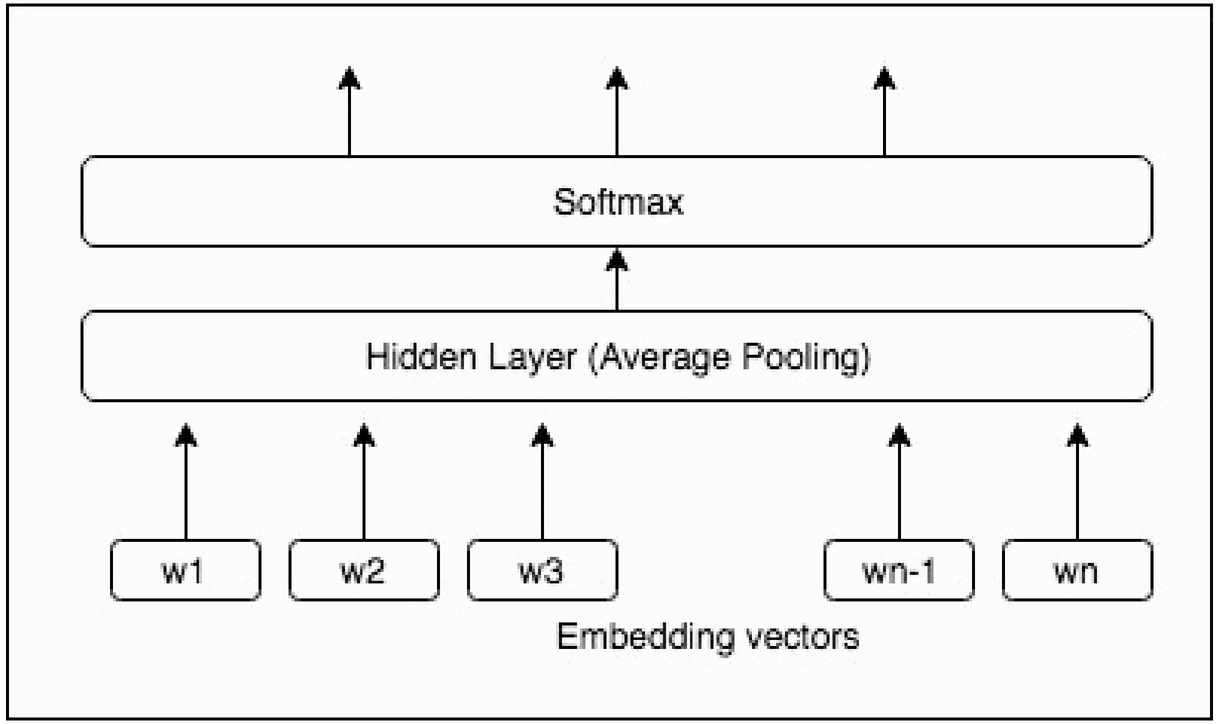
Sentiment analysis, spam detection, sentence pair comparaison, etc.

1. Build a dataset of labeled sentences

2. Grab a pre-trained model, and add a classification layer

3. Convert each sentence to a list of vectors

4. Train or fine-tune the model to predict the correct class



Source: Wikipedia

Demo: sentiment analysis on movie review with ktrain and pre-trained BERT

https://gitlab.com/juliensimon/dlnotebooks/ktrain/

Getting started on AWS

https://ml.aws

https://aws.amazon.com/marketplace/solutions/machine-learning/natural-language-processing

https://aws.amazon.com/sagemaker

https://github.com/awslabs/amazon-sagemaker-examples

Thank you!