A Pragmatic Introduction to Natural Language Processing models

Julien Simon
Global Evangelist, AI & Machine Learning

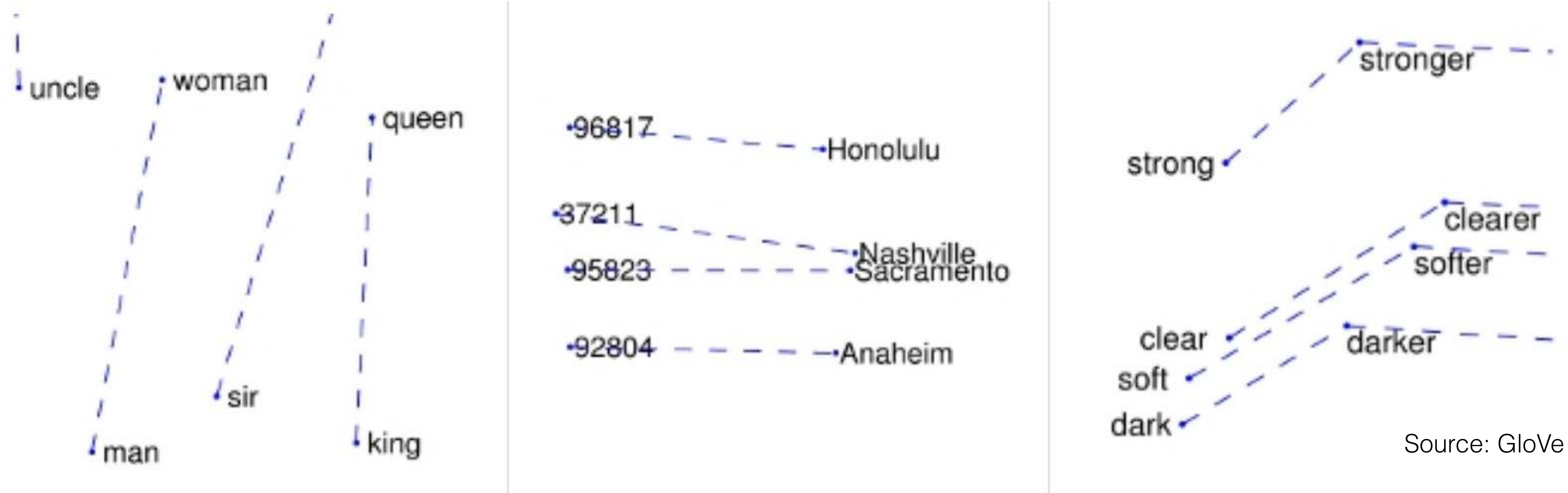
@julsimon
https://medium.com/@julsimon

Problem statement

- Since the dawn of AI, NLP has been a major research field
  - Text classification, machine translation, text generation, chat bots, voice assistants, etc.
- 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 would help us with a variety of downstream NLP tasks?

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

- Initial idea: build word vectors from co-occurrence counts
  - Also called embeddings
  - High dimensional: at least 50, up to 300
  - Much more compact than an *n*n* matrix, which would be extremely sparse
- Words with similar meanings should have similar vectors
  - "car" ≈ "automobile" ≈ "sedan"
- The distance between related vectors should be similar
  - distance ("Paris", "France") ≈ distance("Berlin", "Germany")
  - distance("hot", "hotter") ≈ distance("cold", "colder")



Source: GloVe

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

4. Learn vector representations for all words

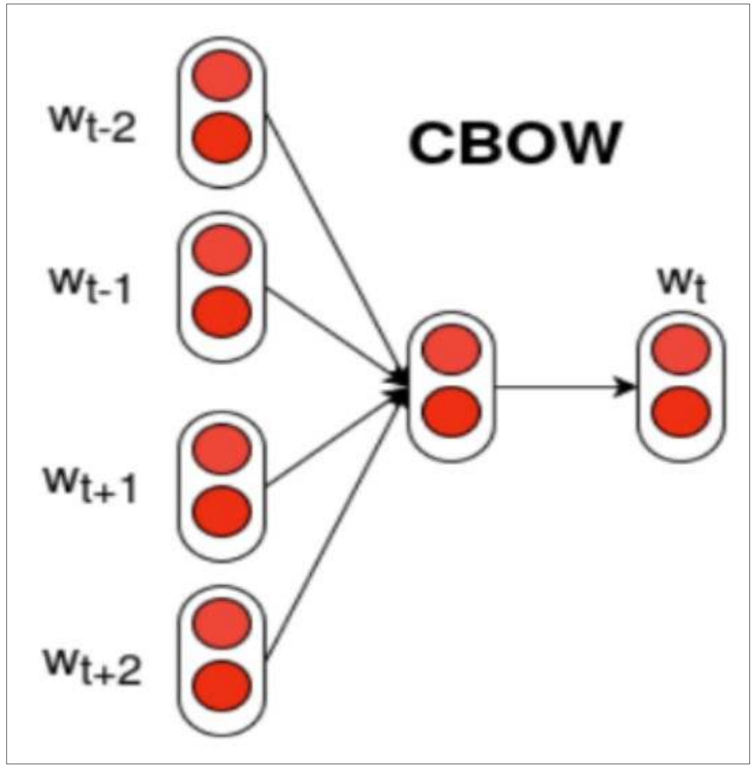
… or simply use (or fine-tune) pre-trained vectors (more on this later)

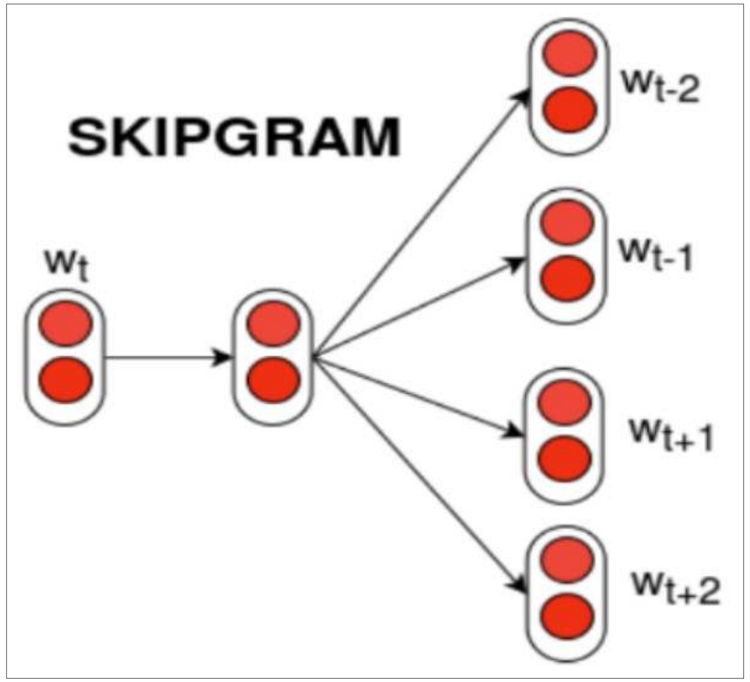
Algorithms

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

- Several pre-trained embedding collections:
  up to 840 billion tokens, 2.2 million vocabulary, 300 dimensions

- C code, based on matrix factorization

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

Source: GloVe

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 a set of subwords, aka n-grams
  - « Computer », n=5 : <START>Comp , compu, omput, mpute, puter, uter<END>
  - A word vector is the average of its subword vectors
- Good for unknown/mispelled words, as they share subwords with known words
  - « Computerized » and « Cmputer » should be close to « Computer »
- Three modes
  - Unsupervised learning: compute embeddings (294 languages available)
  - Supervised learning: use / fine-tune embeddings for multi-label, multi-class 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)

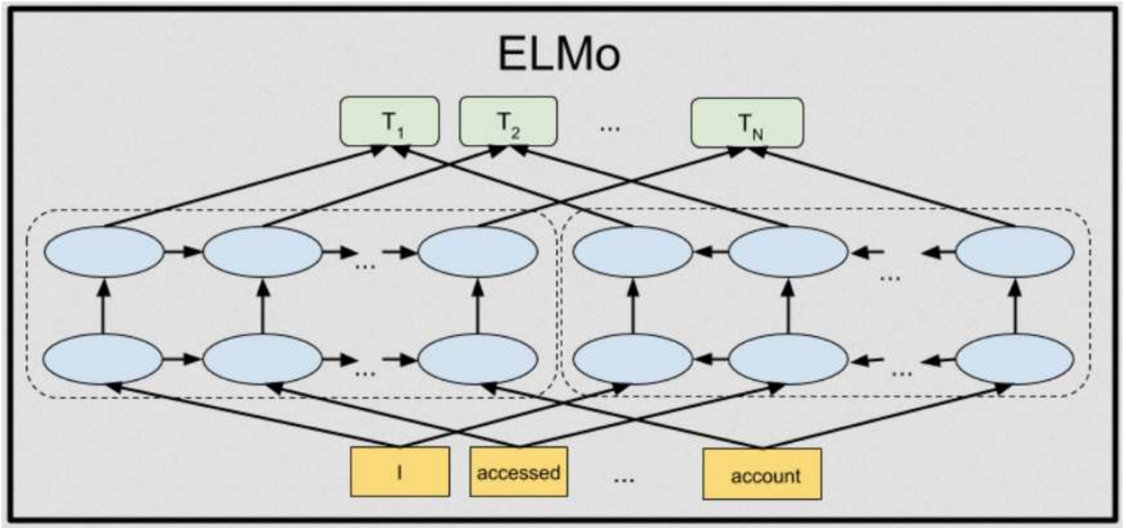
Embeddings from Language Models aka ELMo (02/2018

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

- ELMo generates a context-aware vector for each word
  - Character-level CNN + two unidirectional LSTMs
  - Bidirectional context, no cheating possible (can't peek at the next word)
  - "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

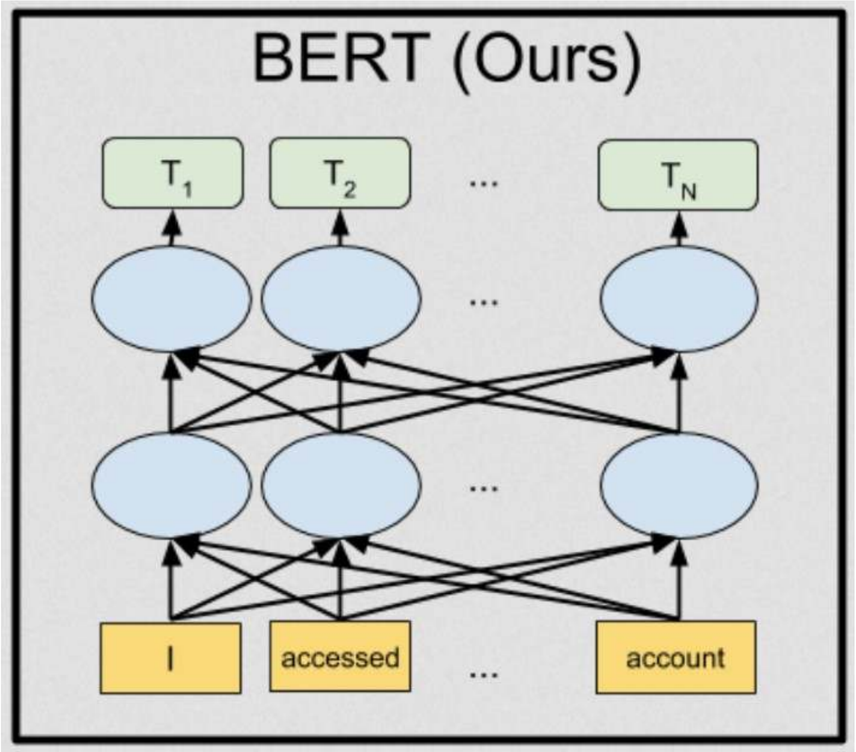
	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round 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
  - True bidirectional architecture: left-to-right and right-to-left contexts are learned by the same network
  - 15% of 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/

Summing things up

- 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!

- The ”best model” is the one that works best on your business problem

Demos

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

Getting started on AWS

https://ml.aws

https://aws.amazon.com/marketplace/solutions/machine-learning/natural-language-processing → off the shelf algos and models that might just save you from this madness ;)

https://aws.amazon.com/sagemaker

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

Thank you!

Julien Simon
Global Evangelist, AI & Machine Learning

@julsimon
https://medium.com/@julsimon