

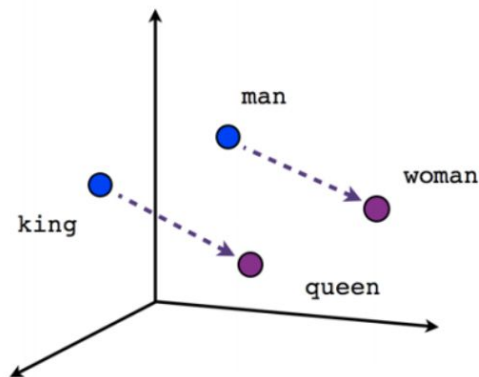
NLP Crash Course II

Indira Sen

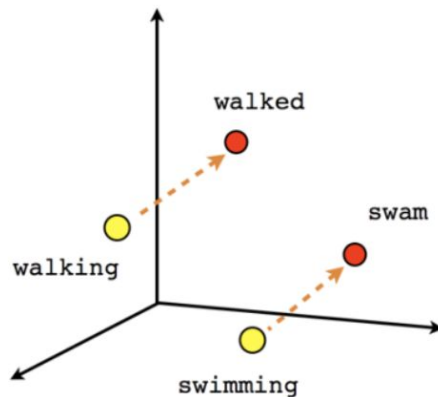
Agenda

- Vector Semantics and Embeddings
- Contextual Embeddings
- Attention
- Transformers
- BERT and Co.

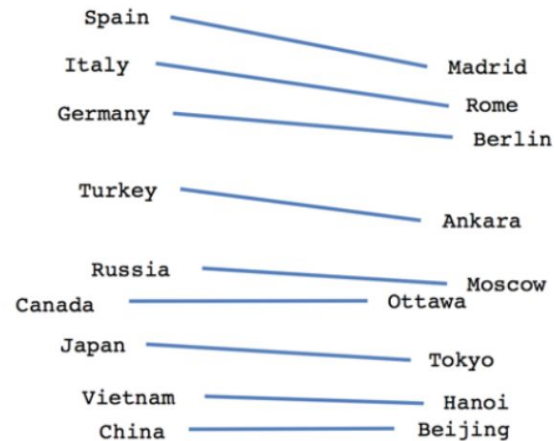
Vector Semantics



Male-Female



Verb tense



Country-Capital

Vector Semantics

“ As Wittgenstein says, ‘the meaning of words lies in their use.’ The day-to-day practice of playing language games recognizes customs and rules. It follows that a text in such established usage may contain sentences such as ‘Don’t be such an ass!’, ‘You silly ass!’, ‘What an ass he is!’ In these examples, the word ass is in familiar and habitual company, commonly collocated with you silly—, he is a silly—, don’t be such an—. **You shall know a word by the company it keeps!** ”

- John Rupert Firth, “A Synopsis of Linguistic Theory” 1957

lemma and wordform

- A **lemma** or **citation form**

- Same stem, part of speech, rough semantics

- A **wordform**

- The “inflected” word as it appears in text

Wordform	Lemma
banks	bank
sung	sing
duerm	sleep

Dan Jurafsky, <https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html>

Lemmas have senses

- One lemma “bank” can have many meanings:

Sense 1: ■ ...a **bank** can hold the investments in a custodial account...

Sense 2: ■ “...as agriculture burgeons on the east **bank** the river will shrink
even more”

2

- **Sense (or word sense)**
 - A discrete representation
of an aspect of a word’s meaning.
- The lemma **bank** here has two senses

Handcrafted Resources: WordNet 3.0

- A hierarchically organized lexical database
- On-line thesaurus + aspects of a dictionary
- Some [other languages](#) available or under development

- (Arabic, Finnish, German, Portuguese...)

Category	Unique Strings
Noun	117,798
Verb	11,529
Adjective	22,479
Adverb	1,161

- Dan Jurafsky, <https://web.stanford.edu/~jurafsky/NLPCourseraSlides.html>

How is “sense” defined in WordNet?

- The **synset (synonym set)**, the set of near-synonyms, instantiates a sense or concept, with a **gloss**
- Example: **chump** as a noun with the **gloss**:
“a person who is gullible and easy to take advantage of”
- This sense of “chump” is shared by 9 words:
chump¹, fool², gull¹, mark⁹, patsy¹, fall guy¹, sucker¹, soft touch¹, mug²
- Each of **these** senses have this same gloss
 - (Not **every** sense; sense 2 of gull is the aquatic bird)

Word Similarity

- **Synonymy**: a binary relation
 - Two words are either synonymous or not
- **Similarity (or distance)**: a looser metric
 - Two words are more similar if they share more features of meaning
- Similarity is properly a relation between **senses**
 - The word “bank” is not similar to the word “slope”
 - Bank¹ is similar to fund³
 - Bank² is similar to slope⁵
- But we’ll compute similarity over both words and senses

Two classes of similarity algorithms

- Thesaurus-based algorithms
 - Are words “nearby” in hypernym hierarchy?
 - Do words have similar glosses (definitions)?
- Distributional algorithms
 - Do words have similar distributional contexts?

Problems with thesaurus-based meaning

- We don't have a thesaurus for every language
- Even if we do, they have problems with **recall**
 - Many words are missing
 - Most (if not all) phrases are missing
 - Some connections between senses are missing
 - Thesauri work less well for verbs, adjectives
 - Adjectives and verbs have less structured hyponymy relations

Distributional models of meaning

- Also called vector-space models of meaning
- Offer much higher recall than hand-built thesauri
 - Although they tend to have lower precision
- Zellig Harris (1954): “**oculist** and **eye-doctor** ... occur in almost the same environments.... **If A and B have almost identical environments we say that they are synonyms.**
- Firth (1957): “You shall know a word by the company it keeps!”

Intuition of distributional word similarity

- Example:

A bottle of *tesgüino* is on the table
Everybody likes *tesgüino*
Tesgüino makes you drunk
We make *tesgüino* out of corn.

- From context words humans can guess *tesgüino* means
 - an alcoholic beverage like **beer**
- Intuition for algorithm:
 - Two words are similar if they have similar word contexts.

Term-document matrix

- Each cell: count of term t in a document d : $tf_{t,d}$
 - Each document is a **count vector** in \mathbb{N}^V : a column below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Term-document matrix

- Two documents are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

The words in a term-document matrix

- Each word is a **count vector** in \mathbb{N}^D : a row below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

The words in a term-document matrix

- Two **words** are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

The Term-Context matrix

- Instead of using entire documents, use smaller contexts
 - Paragraph
 - Window of 10 words
- A word is now defined by a vector over counts of context words

Term-context matrix for word similarity

- Two **words** are similar in meaning if their context vectors are similar

	aardvark	computer	data	pinch	result	sugar	...
apricot	0	0	0	1	0	1	
pineapple	0	0	0	1	0	1	
digital	0	2	1	0	1	0	
information	0	1	6	0	4	0	

Should we use raw counts?

- For the term-document matrix
 - We used **tf-idf** instead of raw term counts
- For the term-context matrix
 - **Positive Pointwise Mutual Information (PPMI)** is common

TF-IDF: Term Frequency-Inverse Document Frequency

A transformation to represent a term as function of how frequently it appears in a document but accounting for its frequency in the overall dataset

- TF for a given term = the number of times the term appears
- IDF for a given term = the number of documents in collection / number of documents that contain term

Formalizing

- Term frequency ($tf_{t,d}$) = the number of times term t occurs in document d ;
- Inverse document frequency = inverse fraction of number of documents containing (D_t) among total number of documents N

$$tfidf(t, d) = tf_{t,d} \times \log \frac{N}{D_t}$$

Pointwise Mutual Information

- **Pointwise mutual information:**

- Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X, Y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

- **PMI between two words:**

(Church & Hanks 1989)

- Do words x and y co-occur more than if they were independent?

$$\text{PMI}(\text{word}_1, \text{word}_2) = \log_2 \frac{P(\text{word}_1, \text{word}_2)}{P(\text{word}_1)P(\text{word}_2)}$$

Is any of this still relevant?

- Find distinctive words using TF-IDF and PPMI

Analyzing Polarization in Social Media: Method and Application to Tweets on 21 Mass Shootings

Dorottya Demszy¹ Nikhil Garg¹ Rob Voigt¹ James Zou¹

Matthew Gentzkow¹ Jesse Shapiro² Dan Jurafsky¹

¹Stanford University ²Brown University

{ddemszky, nkgarg, robvoigt, jamesz, gentzkow, jurafsky}@stanford.edu
jesse.shapiro.1@brown.edu

Abstract

We provide an NLP framework to uncover four linguistic dimensions of political polarization in social media: topic choice, framing, affect and illocutionary force. We quantify these aspects with existing lexical methods, and propose clustering of tweet embeddings as a means to identify salient topics for analysis across events; human evaluations show that our approach generates more cohesive topics than traditional LDA-based models. We apply our methods to study 4.4M tweets on 21 mass shootings. We provide evidence that the discussion of these events is highly polarized

2016) and Facebook (Bakshy et al., 2015). Prior NLP work has shown, e.g., that polarized messages are more likely to be shared (Zafar et al., 2016) and that certain topics are more polarizing (Balasubramanyan et al., 2012); however, we lack a more broad understanding of the many ways that polarization can be instantiated linguistically.

This work builds a more comprehensive framework for studying linguistic aspects of polarization in social media, by looking at topic choice, framing, affect, and illocutionary force.

1.1 Mass Shootings

Latent Semantic Analysis (LSA)

- a technique analyzing relationships between a **set of documents** and **the terms they contain** by producing a set of concepts related to the documents and terms.
- Simplified description:
 - Find the term-document matrix
 - Apply Singular Value Decomposition

[Why?]

Latent Semantic Analysis (LSA)

- a technique analyzing relationships between a **set of documents** and **the terms they contain** by producing a set of concepts related to the documents and terms.
- Simplified description:
 - Find the term-document matrix
 - Apply Singular Value Decomposition
- For dimensionality reduction
- https://en.wikipedia.org/wiki/File:Topic_model_scheme.webm
- We can now use cosine similarity (or other similarity metrics) to find the similarity between documents!

[Why?]

Practical Uses

LSA: Topic modeling, similarity between documents

Finding distinctive or discriminative words in subsets of data

Example:
How is hate
speech and
toxicity
defined in
different
languages in
current
datasets?



The timeline so far: Supervised NLP

Classical NLP:
Train separate
models for
separate tasks

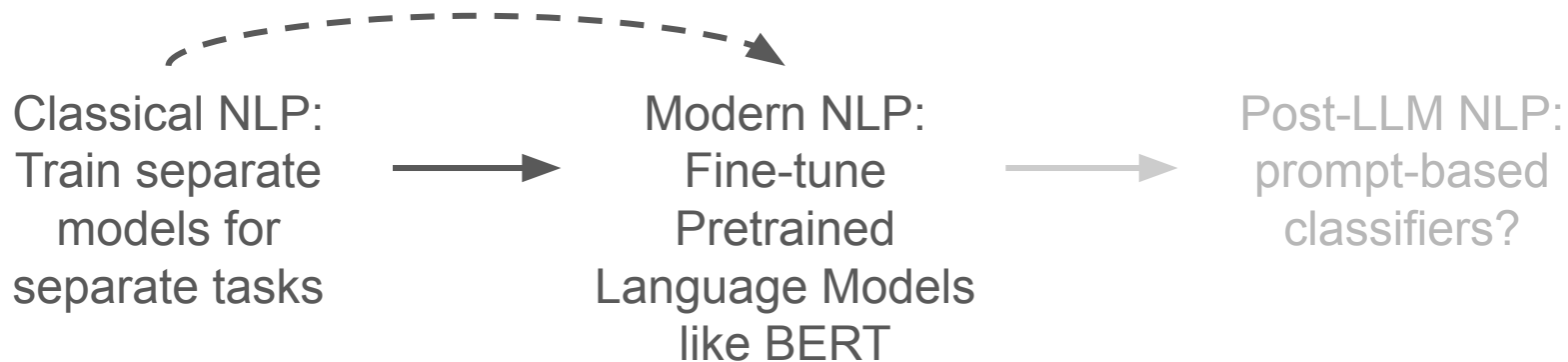


Modern NLP:
Fine-tune
Pretrained
Language Models
like BERT

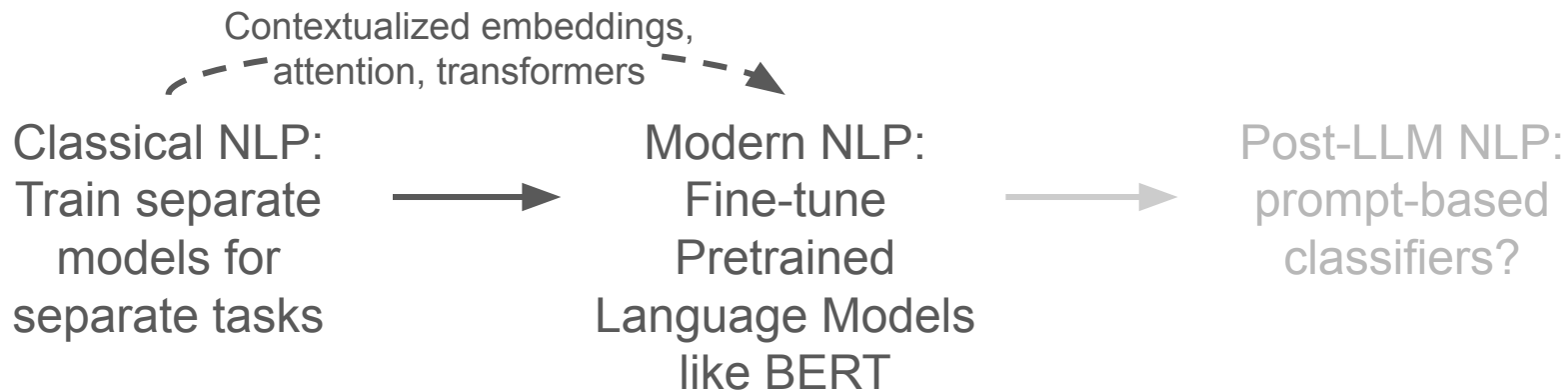


Post-LLM NLP:
prompt-based
classifiers?

The timeline so far: Supervised NLP



The timeline so far: Supervised NLP



Learn Representations of Words

- So far: all methods based on counting; no learning or prediction

- Now: Learn low-dimensional representations ('vectors') through prediction:
using context to predict words in a surrounding window

- Transform this into a supervised prediction problem
- similar to language modeling but ignore order within the context window

Word2Vec

- **Learned parameters:** word vectors
- **Goal:** make each vector “know” about the contexts of its word
- **How:** train vectors to predict possible contexts from words (or, alternatively, words from contexts)

Important: These are word type vectors

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.

Jeffrey Dean
Google Inc.

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

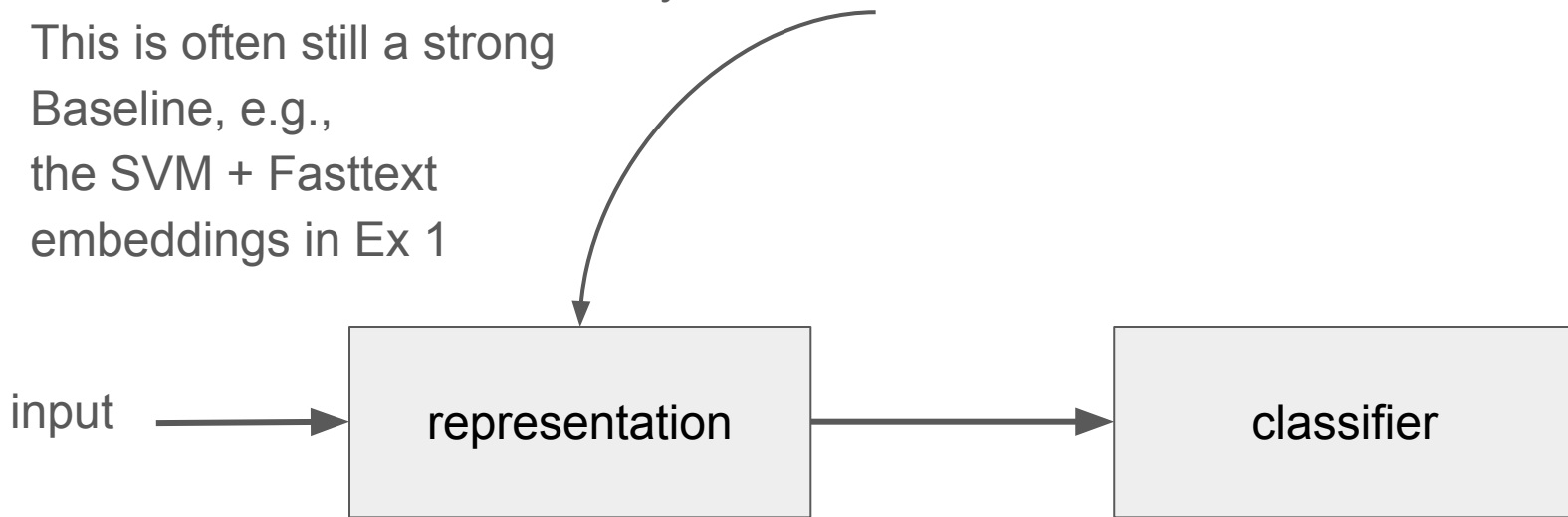
Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

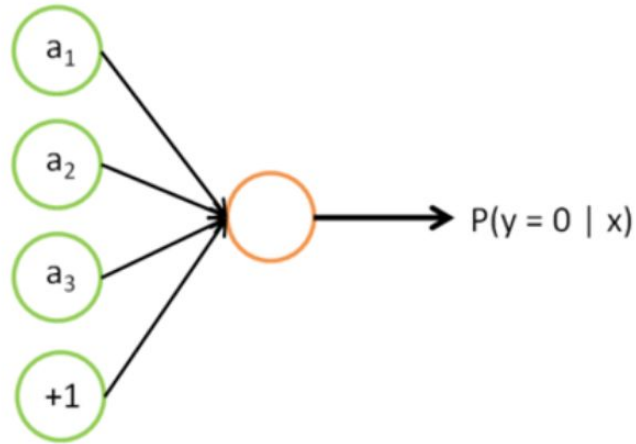
We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

What can you do with these word vectors?

- Besides many applications like document search, word similarity, you can also use them as features for your classifier!
- This is often still a strong Baseline, e.g., the SVM + Fasttext embeddings in Ex 1

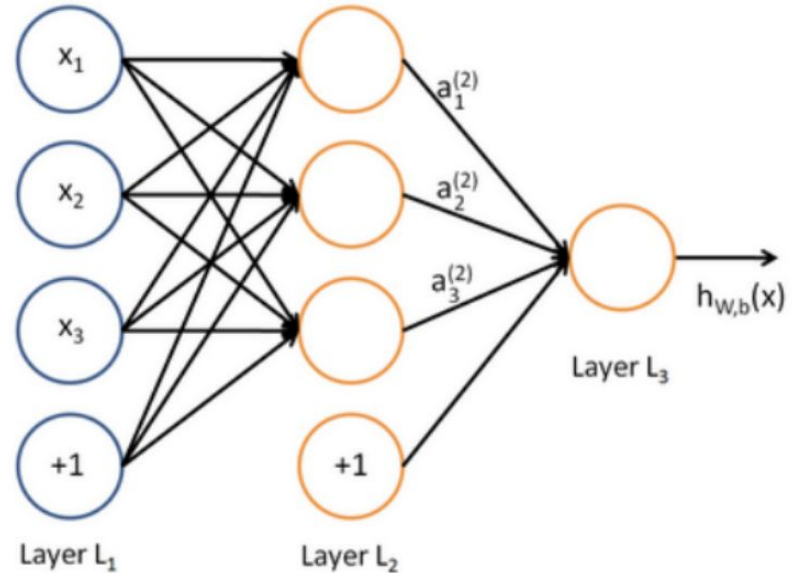


Detour: Neural Networks and Deep Learning



Input
(features) Logistic
classifier

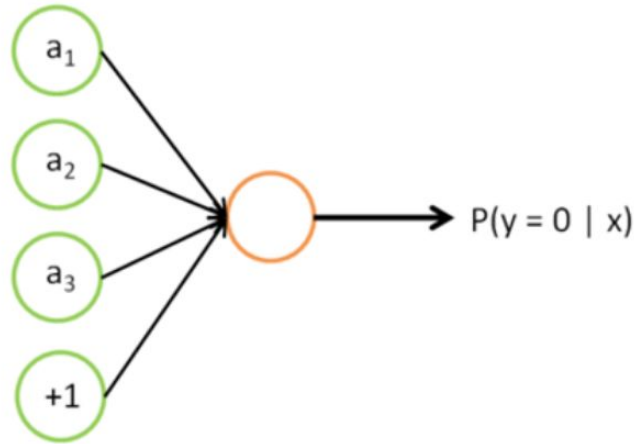
Logistic Regression



Neural Network

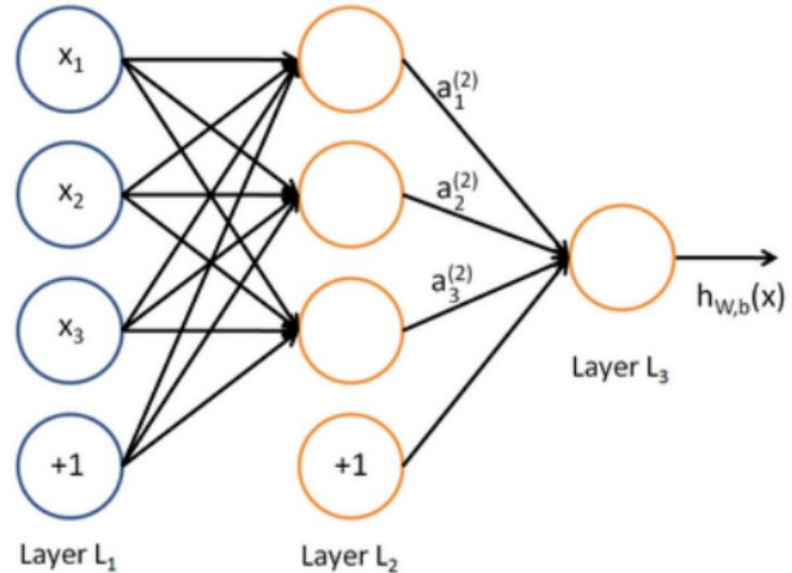
Detour: Neural Networks and Deep Learning

No need for feature engineering, the network learns higher or lower dimensions on itself



Input
(features) Logistic
classifier

Logistic Regression



Neural Network

Detour: Neural Networks

What do we mean by
feature engineering?

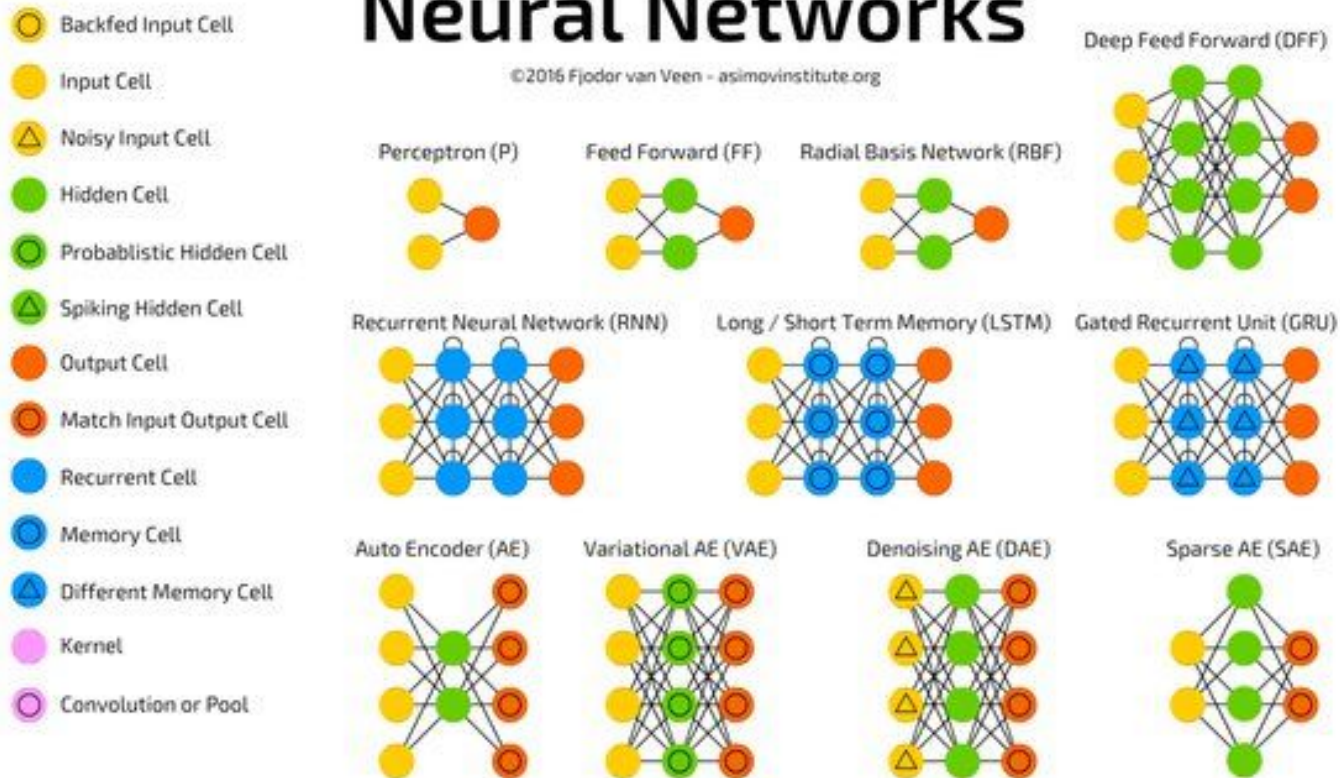
Say for sentiment analysis,

Table II. Summary of the Articles Employed a Supervised Method to Address TSA

Study	Task	Algorithms	Features	Dataset
Go et al. [2009]	TSA	NB, MaxEnt, SVM	unigrams, bigrams, POS	STS
Pak and Paroubek [2010]	TSA	MNB, SVM, CRF	unigrams, bigrams, trigrams, POS	own
Barbosa and Feng [2010]	TSA	SVM	meta-features (POS, polarity-MPQA), tweet syntax (i.e., retweet, hashtags, emoticons, links etc.)	own
Davidov et al. [2010]	TSA	kNN	word and n-gram based, punctuation-based, pattern-based	OC
Bakliwal et al. [2012]	TSA	SVM, NB	words' polarity, unigrams, bigrams, emoticons, hashtags, URLs, targets etc.	STS, Mejaj [Bora 2012]
Mohammad et al. [2013]	TSA	SVM	word/character n-grams, POS, caps, lexicons, punctuation, negation, tweet-based	SemEval-2013
Kiritchenko et al. [2014]	TSA	linear kernel SVM, MaxEnt	word/character n-grams, POS, caps, punctuation, emoticons, automatic sentiment lexicons, polarity, emphatic lengthening	SemEval-2013
Asiaee et al. [2012]	TSA	dictionary learning, WSVM, NB, kNN, SVM		DETC
Agarwal et al. [2011]	TSA	SVM	POS, unigrams, DAL lexicon, caps, exclamation etc.	own
Aisopos et al. [2011]	TSA	MNB, C4.5 tree	n-grams	own
Kouloumpis et al. [2011]	TSA	AdaBoost	N-gram with lexicon features, twitter-based, POS	STS, ETC
Saif et al. [2012b]	TSA	NB	unigrams, POS, sentiment-topic, semantic features	STS, HCR, OMD
Hamdan et al. [2013]	TSA	SVM, NB	unigrams, concepts (DBPedia), verb groups/adjectives (WordNet) and senti-features (SentiWordNet)	SemEval-2013
Jiang et al. [2011]	entity-TSA	SVM	unigrams, emoticons, hashtags, punctuation the General Inquirer lexicon	own
Aston et al. [2014]	TSA	Perceptron with Best Learning	character n-grams	Sanders

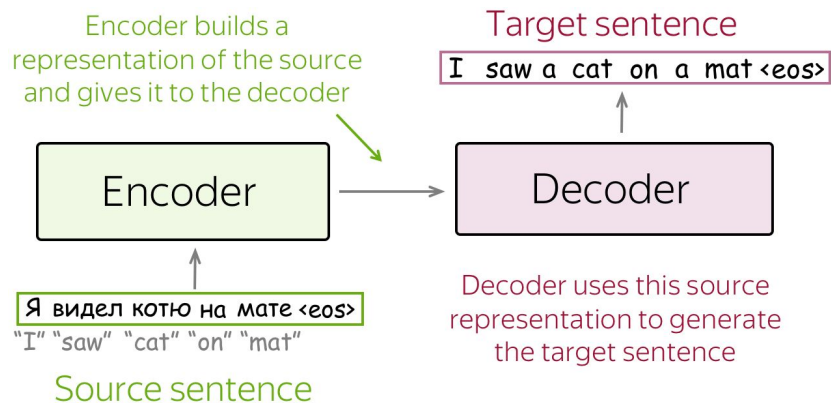
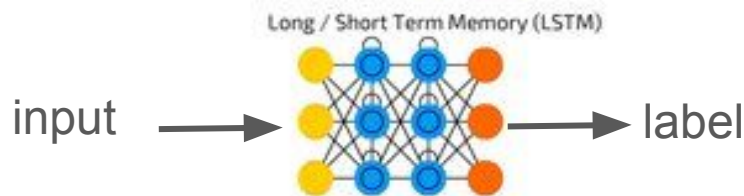
[Like It or Not: A Survey of Twitter Sentiment Analysis Methods](#), Anastasia Giachanou And Fabio Crestani, 2016

Detour: Neural Networks and Deep Learning



Deep Learning in NLP (till mid 2010's)

- Precludes need for explicit feature engineering
- Automatically creates complex representations
- Popular Classifiers are RNNs (GRUs and LSTMs)
- Classification is one strand of problems
- Another: sequence-to-sequence output, e.g., Machine Translation



Contextualized Embeddings (ELMo, CoVe)

Deep contextualized word representations

Matthew E. Peters[†], Mark Neumann[†], Mohit Iyyer[†], Matt Gardner[†],
{matthewp, markn, mohiti, mattg}@allenai.org

Christopher Clark^{*}, Kenton Lee^{*}, Luke Zettlemoyer^{†*}

The idea behind Contextualized Embeddings:
Word *token* vectors instead of word *type*, with
neural networks modeling left and right
contexts

vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP

et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Contextualized Embeddings (ELMo, CoVe)



Deep contextualized word representations

Matthew E. Peters[†], Mark Neumann[†], Mohit Iyyer[†], Matt Gardner[†],
{matthewp, markn, mohiti, mattg}@allenai.org

Christopher Clark^{*}, Kenton Lee^{*}, Luke Zettlemoyer^{†*}

Contextualized Embeddings:
word token vectors instead of word *type*, with
neural networks modeling left and right
contexts

vary across linguistic contexts (i.e., to model polysemy). Our word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. We show that these representations can be easily added to existing models and significantly improve the state of the art across six challenging NLP

et al., 2017), ELMo representations are deep, in the sense that they are a function of all of the internal layers of the biLM. More specifically, we learn a linear combination of the vectors stacked above each input word for each end task, which markedly improves performance over just using the top LSTM layer.

Contextualized Embeddings (ELMo, CoVe)

GloVe, Word2Vec → CoVe, ELMo

Great idea 1

What is encoded



- Lena Voita, NLP Course for you

word token vectors instead of word type vectors —i.e., vectors for words in context, or contextual word vectors— that are pre trained on large corpora.

Contextualized Embeddings (ELMo, CoVe)

GloVe, Word2Vec → CoVe, ELMo

Great idea 1

What is encoded



- Lena Voita, NLP Course for you

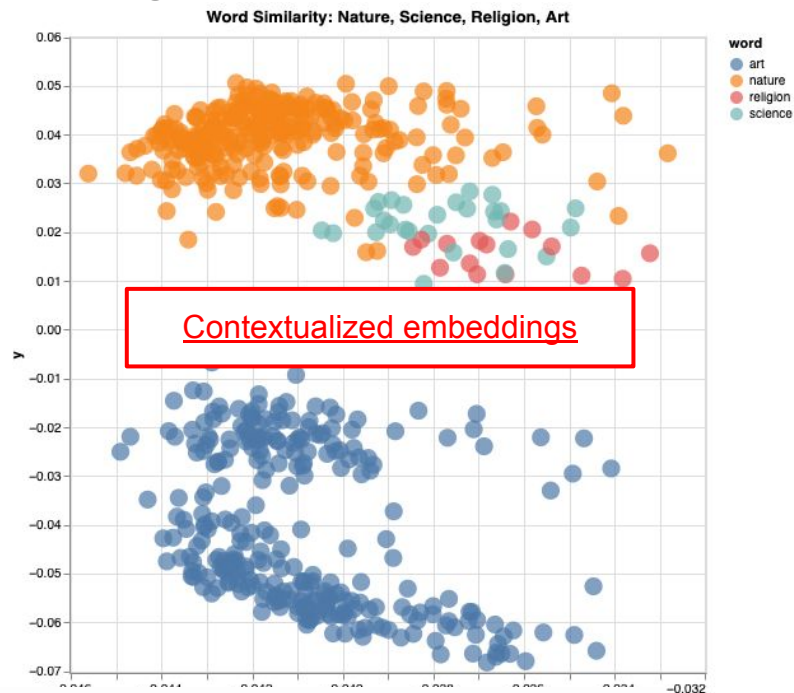
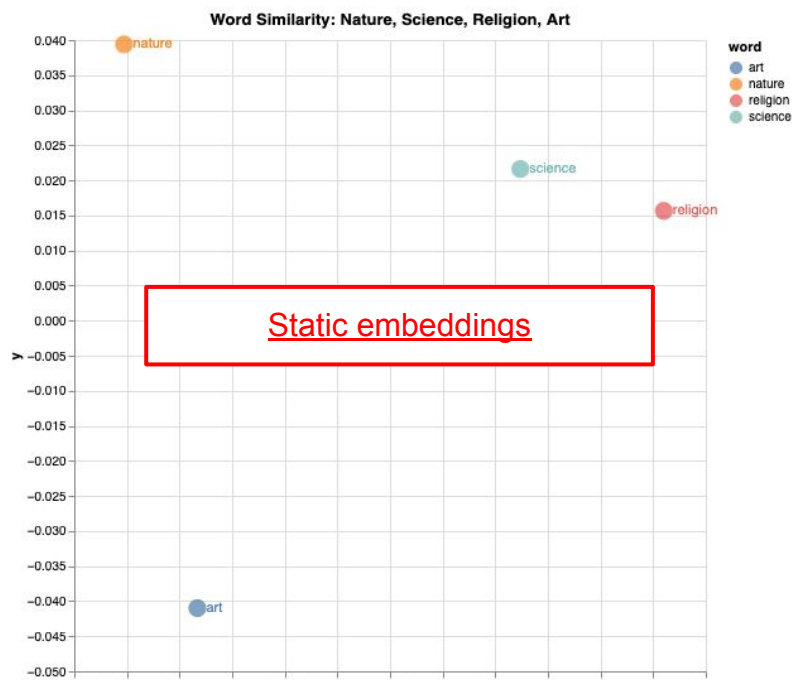
word token vectors instead of word type vectors —i.e., vectors for words in context, or contextual word vectors— that are pre trained on large corpora.

Pre training method: Contextual word vectors are built from both type-level vectors and neural network parameters that “contextualize” each word.

ELMo trains one RNN for left contexts (going back to the beginning of the sentence a token appears in) and another RNN for right contexts (up to the end of the sentence).

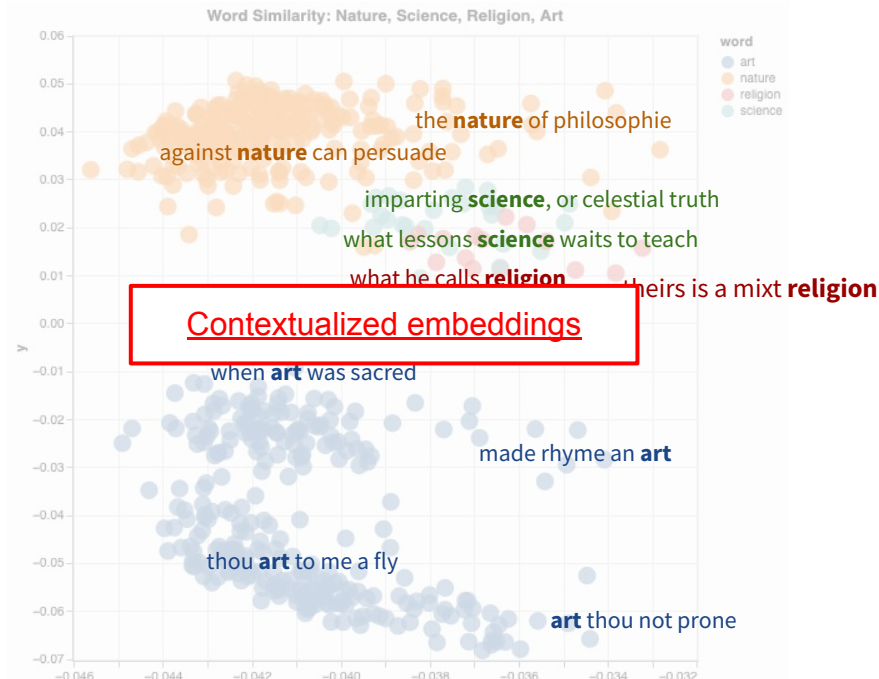
Contextualized Embeddings (ELMo, CoVe)

Result: multiple vectors of a word depending on its context



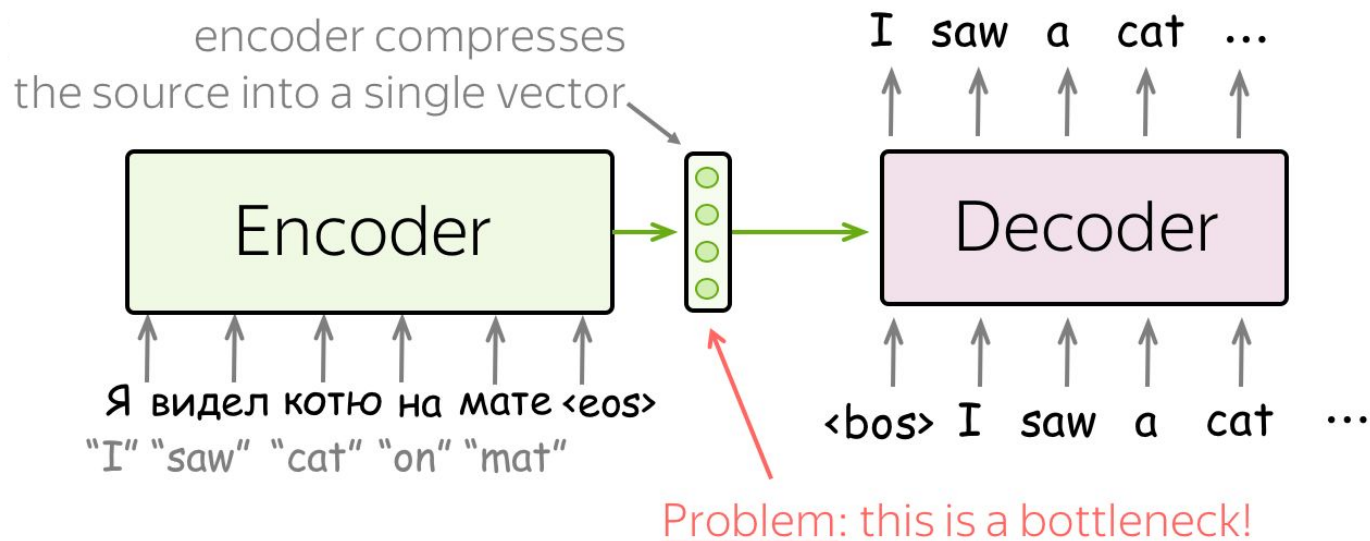
Contextualized Embeddings (ELMo, CoVe)

Result: multiple vectors of a word depending on its context



Attention

The fixed encoding issue



Attention

The fixed encoding

enco
the source int



Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau

Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***

Université de Montréal

ABSTRACT

Neural machine translation is a recently proposed approach to machine translation. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder-decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder-decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search

Attention

The fixed encoding

encod

Published as a conference paper at ICLR 2015

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau

The idea behind 'Attention': At different steps, let a model "focus" on different parts of the input.

Я видел
"I" "saw" "

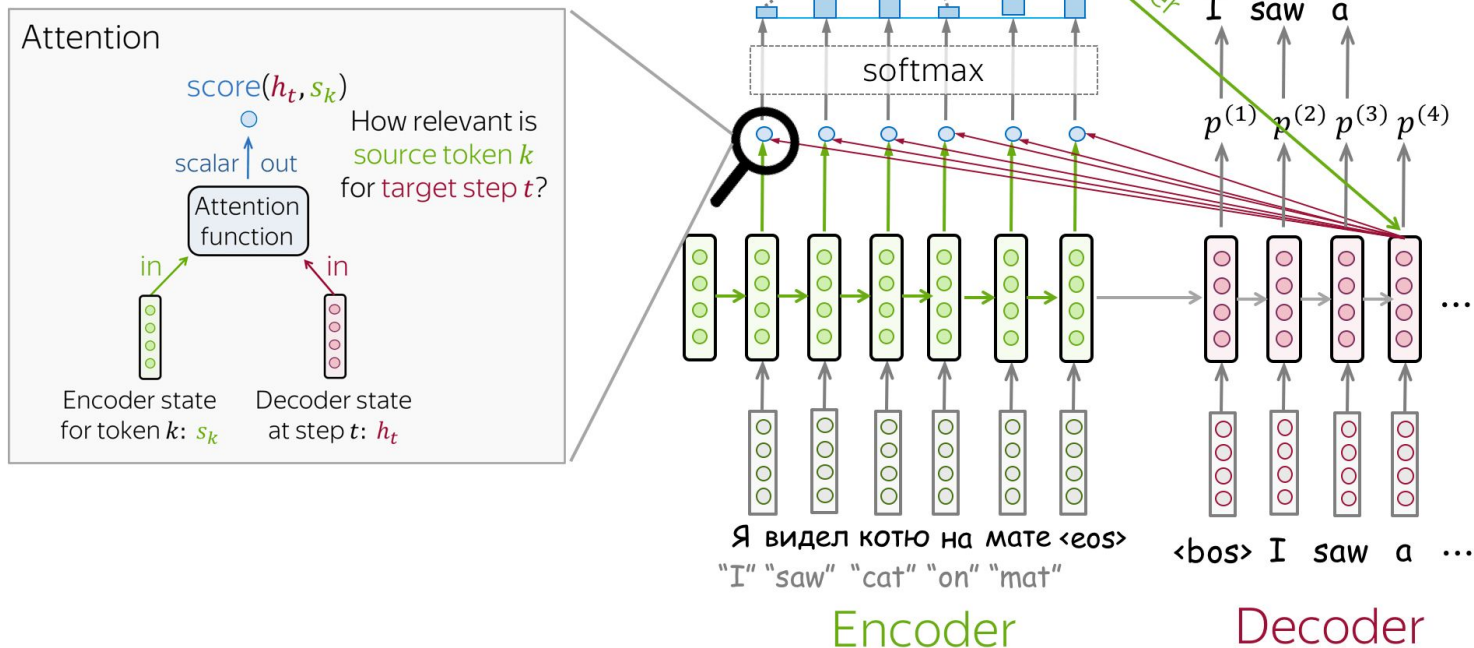
tion. Unlike the traditional statistical machine translation, the neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The models proposed recently for neural machine translation often belong to a family of encoder-decoders and encode a source sentence into a fixed-length vector from which a decoder generates a translation. In this paper, we conjecture that the use of a fixed-length vector is a bottleneck in improving the performance of this basic encoder-decoder architecture, and propose to extend this by allowing a model to automatically (soft-)search

Attention

Attention output: weighted sum of encoder states with attention weights

Attention weights: distribution over source tokens

A model can learn to “pay attention” to the most relevant source tokens for each step



Transformers

- RNNs in architectures have limited context
- Fix: replace all components with attention

	Seq2seq without attention	Seq2seq with attention	Transformer
processing within encoder	RNN/CNN	RNN/CNN	attention
processing within decoder	RNN/CNN	RNN/CNN	attention
decoder-encoder interaction	static fixed- sized vector	attention	attention

Transformers

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

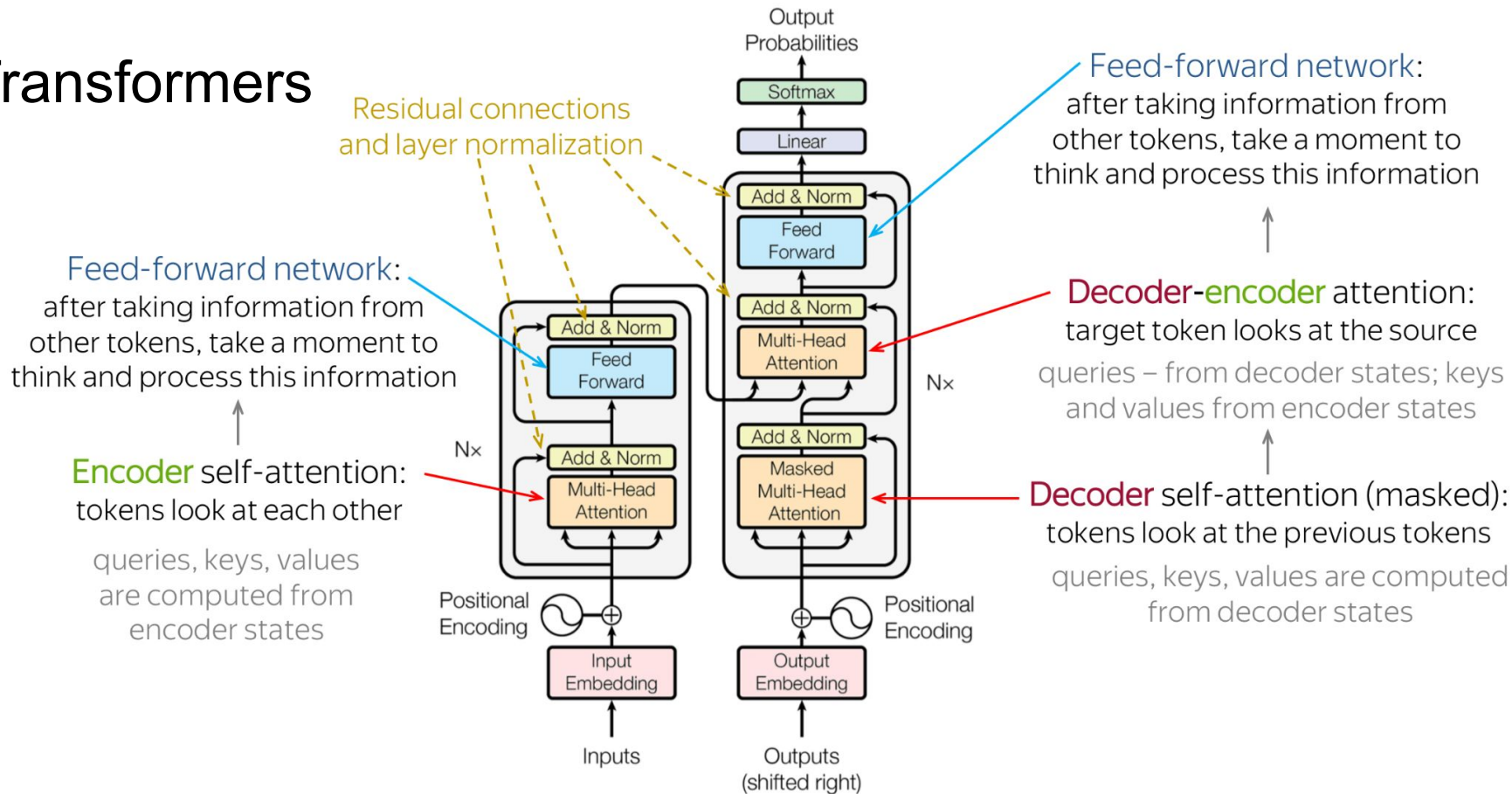
Jakob Uszkoreit*
Google Research
usz@google.com

The idea behind ‘Transformers’: Replace all components with Attention to model context better

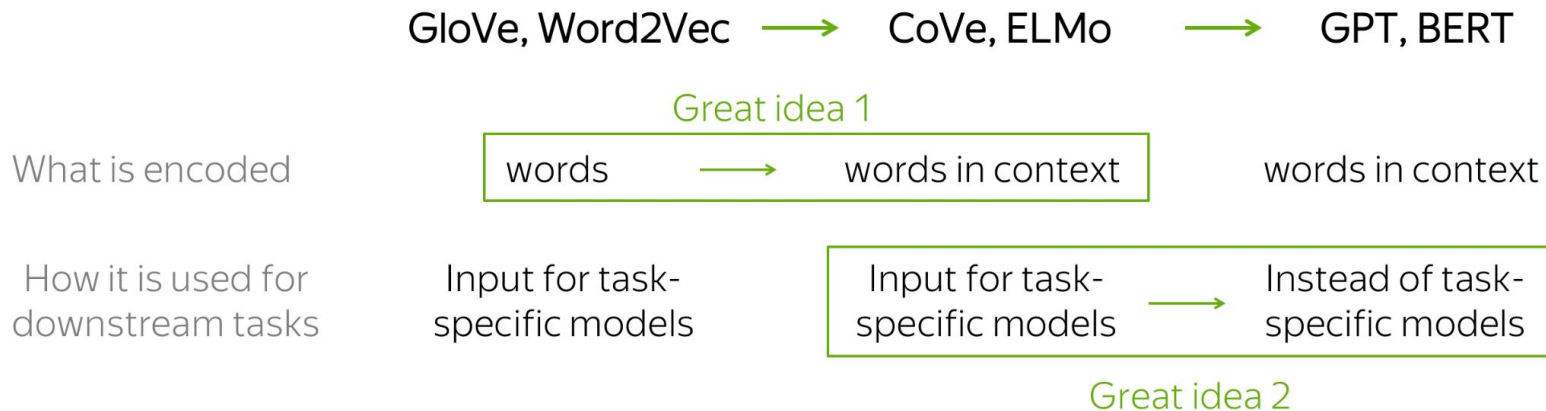
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including

Transformers



BERT, GPT (the first one),



BERT, GPT (the first one),

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

Google AI Language

{jacobdevlin, mingweichang, kentonl, kristout}@google.com

What is encoded

How it is used for
downstream tasks

Abstract

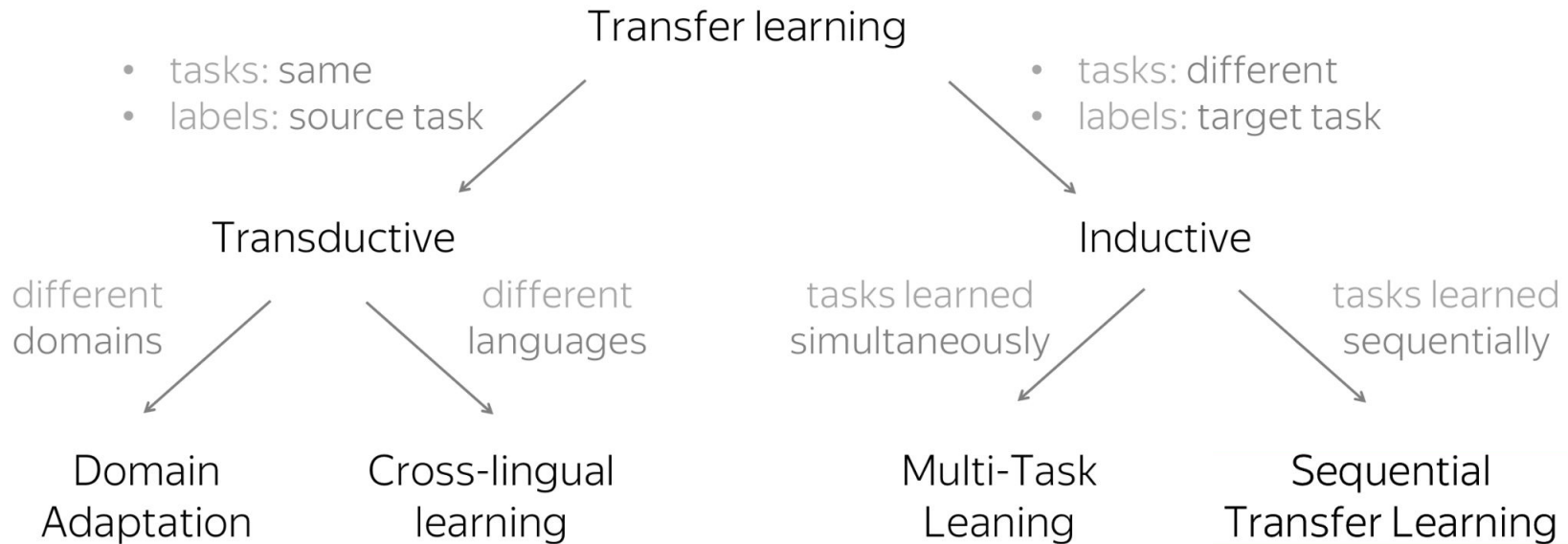
We introduce a new language representation model called **BERT**, which stands for **Bidirectional Encoder Representations from Transformers**. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the

Revisiting Machine Learning

- Usual workflow, train model on your dataset for a certain task.
- The dataset has to represent exactly what task you want to do
- However,
 - Sometimes, you do not have enough labeled data for a very similar task
- Transfer Learning!

Revisiting Machine Learning: Transfer Learning




This taxonomy is from [Sebastian Ruder's blog post](#).

The old vs. 'new' paradigm (maybe revisited due to LLMs)

OLD: Train a model on a task where we have labeled data

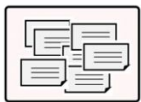
NEW:

1. **Pre-train** a model on a task where we have lots of data
 - *e.g., we have lots and lots of internet data*
 2. **Fine-tune** the model on your downstream task
 - *e.g., a specific, curated dataset that you're studying*
- 
- Transfer learning!**

Transfer Learning via Pretraining

- Train from scratch

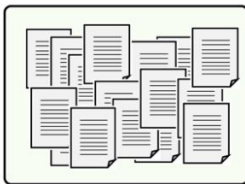
What they will know:



May be not enough to
learn relationships
between words

- Take pretrained
(Word2Vec, GloVe)

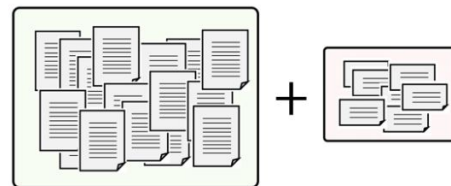
What they will know:



Know relationships between words,
but are **not** specific to the task

- Initialize with pretrained,
then fine-tune

What they will know:



Know relationships between
words and adapted for the task

“Transfer” knowledge from a huge unlabeled corpus to your task-specific model

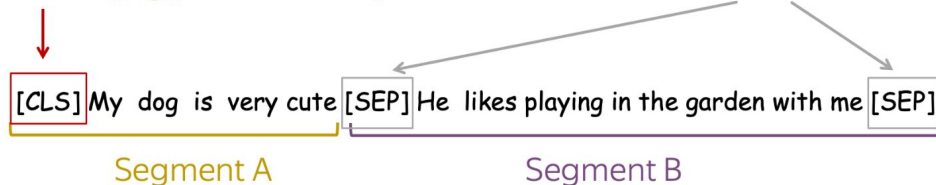
How is BERT Pre-trained?

- Pre-training Input: Pairs of Sentences with Special Tokens separating them

[CLS]: Special token

- Training time: predict if sentences are consecutive or not (Next Sentence Prediction /NSP objective)
- Test time: downstream tasks (e.g., classification)

[SEP]: Special token-separator



Training on pairs of sentences: either consecutive or random (50%/50%)

- Pre-training Objective:

- Next Sentence Prediction (NSP)
- Masked Language Modeling (MLM)

- Lena Voita, NLP Course for you

What is fine-tuning?

- Inputs:
 - pretrained model (usually a transformer, BERT and variants are still very popular)
 - Labeled dataset (rule-of-thumb, at least 1K examples)
- Adds a new layer containing classification parameters.
- All model parameters are updated to maximize the log probability of the labels.
- Output: classifier for your specific task

The timeline so far: Supervised NLP

Classical NLP:
Train separate
models for
separate tasks



Modern NLP:
Fine-tune
Pretrained
Language Models
like BERT



Post-LLM NLP:
prompt-based
classifiers?

Further readings and materials

1. Smith, Noah A. "[Contextual word representations: A contextual introduction.](#)" arXiv preprint arXiv:1902.06006 (2019).
2. Lena Voita, 'NLP Course for you' https://lena-voita.github.io/nlp_course/
3. Jurafsky and Martin, 'Speech and Language Processing':
<https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf>
4. Jacob Eisenstein, 'Natural Language Processing'
<https://cseweb.ucsd.edu/~nnakashole/teaching/eisenstein-nov18.pdf>
5. AI for Humanists, <https://www.aiforhumanists.com/workshops/>