aws SUMMIT ONLINE

Building NLP models with Amazon SageMaker

Antje Barth
Developer Advocate AI/ML
Amazon Web Services

Agenda

Introduction to NLP

Algorithms & Concepts

BERT-family of models

NLP with Amazon SageMaker

feat. TensorFlow, PyTorch, Apache MXNet

Demo

Introduction to NLP

aws SUMMIT ONLINE

Problem statement

- Natural Language Processing (NLP) is a major field in AI
- NLP apps require a language model in order to predict the next word
- 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 domain-specific 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 into tokens

    Tokenize: « hello, world! » → « <BOS>hello<SP>world<SP>!<EOS>»

    Multi-word entities: « Rio de Janeiro » → « rio_de_janeiro »

3. Build the vocabulary from the tokens

4. Learn vector representations for the vocabulary

… or simply use pre-trained models with existing vector representations (more on this later)

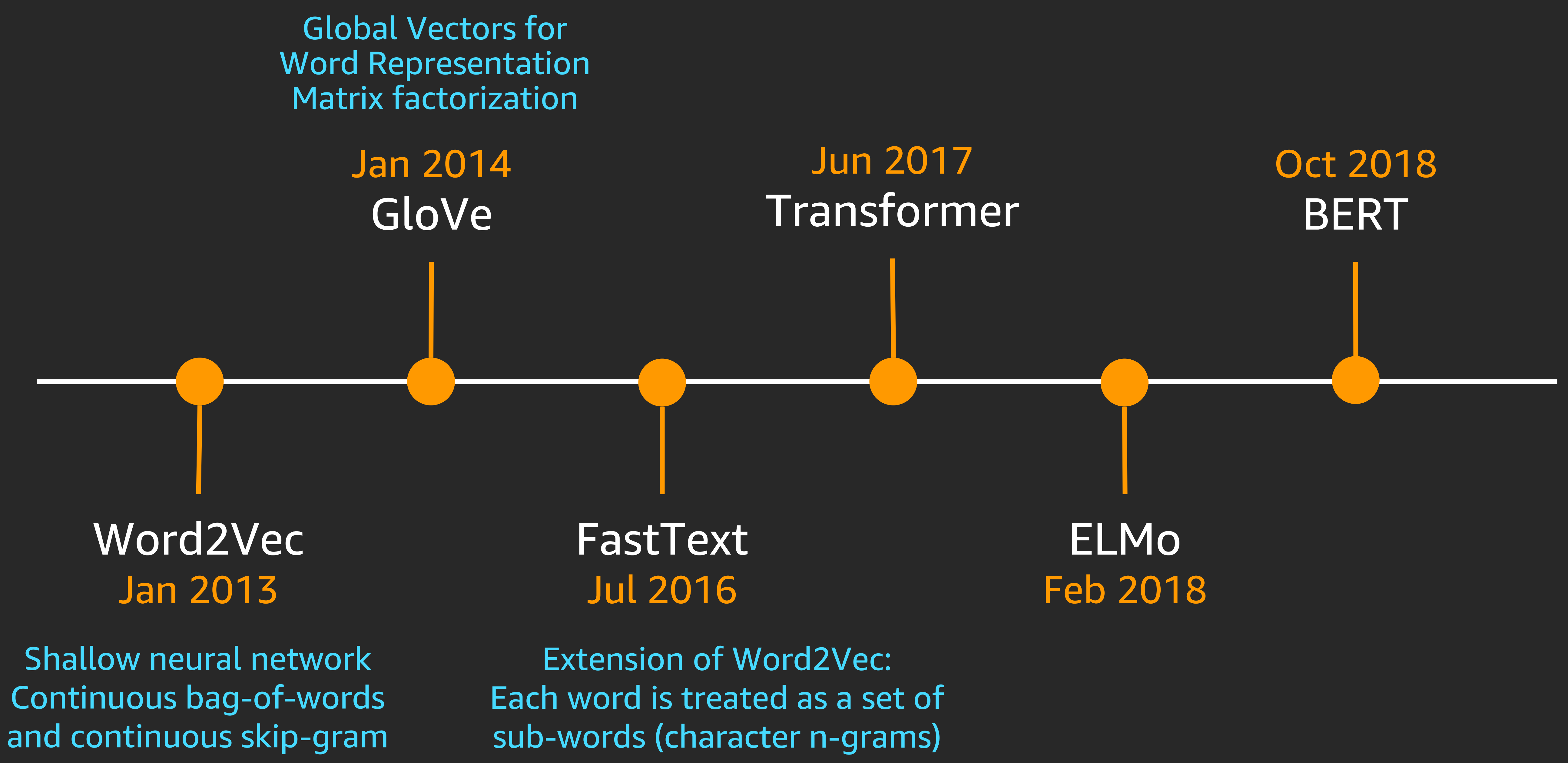
Popular NLP use cases

- Representation learning
- Machine Translation
- Text Classification
- Language Modeling
- Sentiment Analysis
- Named Entity Recognition
- Question Answering

Algorithms & Concepts

aws SUMMIT ONLINE

Evolution of NLP algorithms



Limitations of Word2Vec (and family)

- Some words have different meanings (aka polysemy)

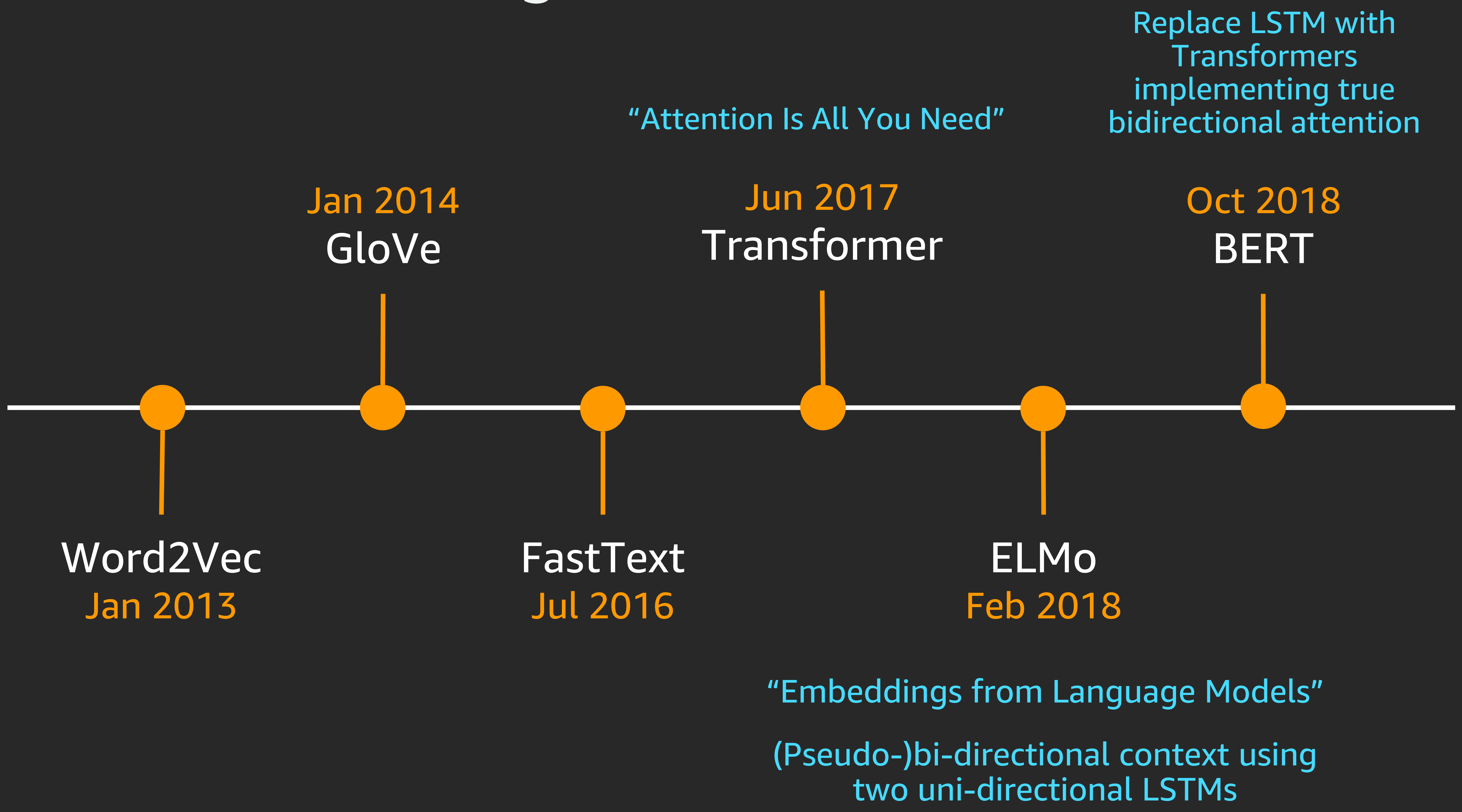
  « *Kevin, stop throwing rocks!* » vs. « *Machine Learning rocks* »

  Word2Vec encodes the different meanings of a word as the same vector

- Bidirectional context is not taken into account

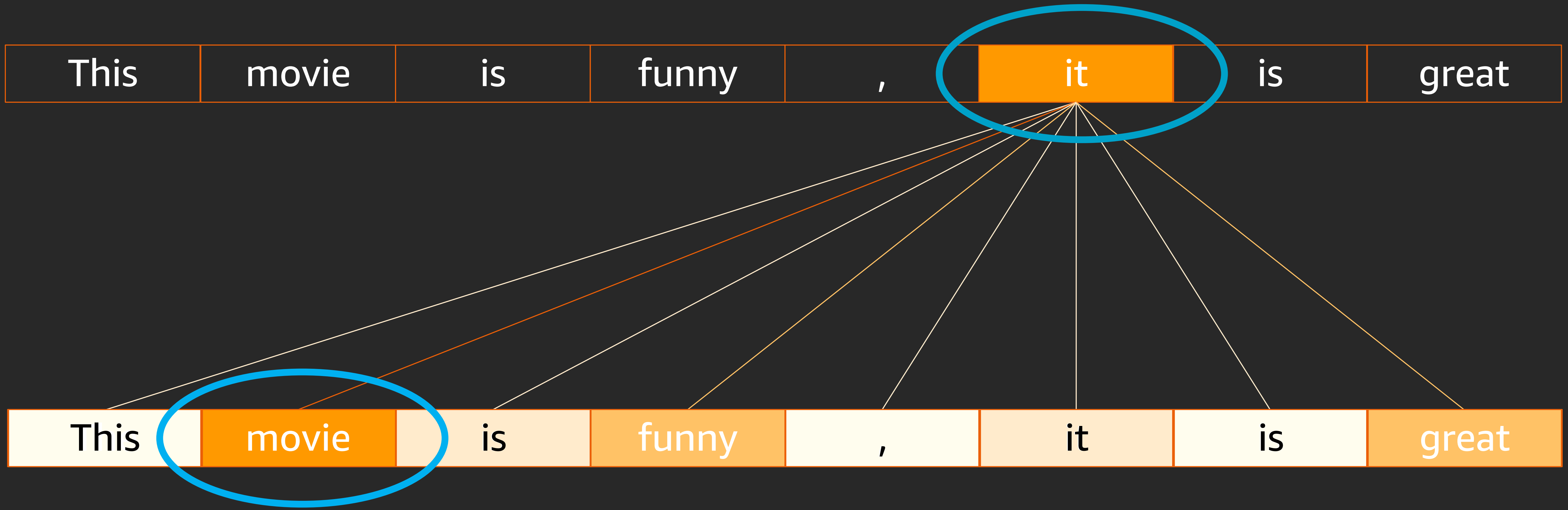
  Previous words (left-to-right) and next words (right-to-left)

Evolution of NLP algorithms



Attention on sentence

"This movie is funny, it is great"



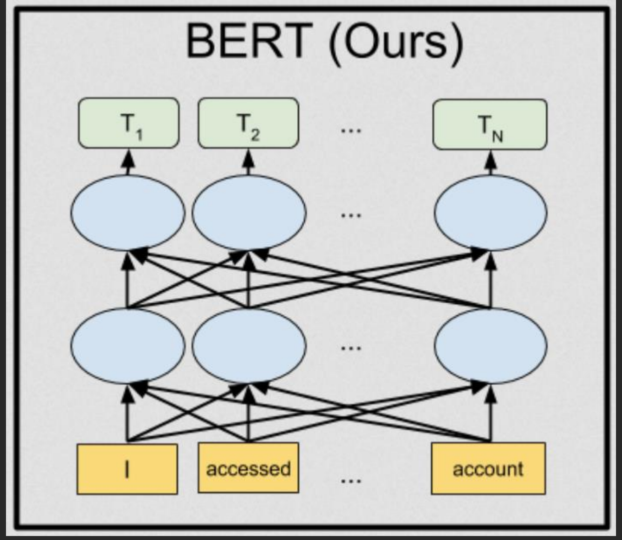
BERT-family of models

BERT Bidirectional Encoder Representations from Transformers

https://arxiv.org/abs/1810.04805

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

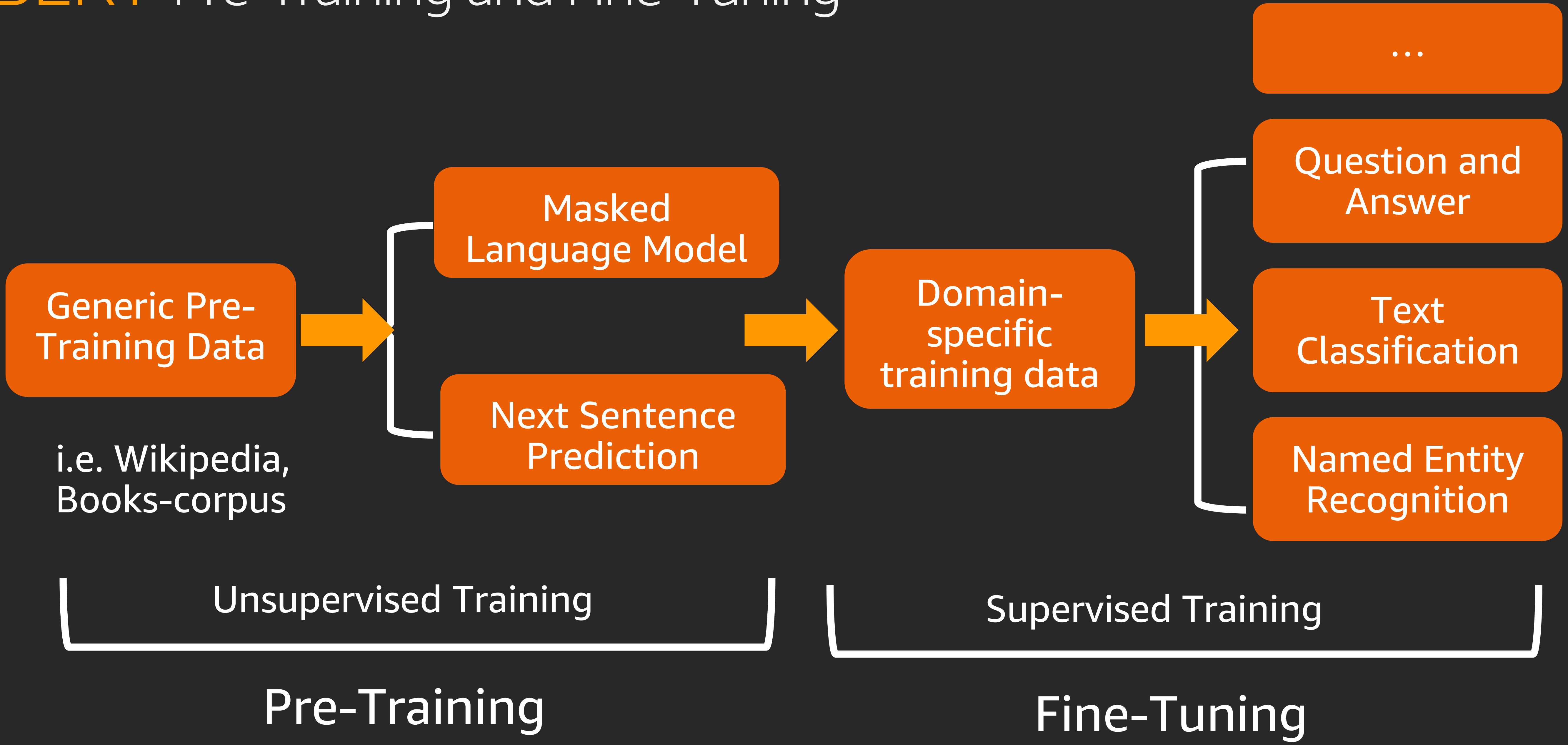
- BERT improves on ELMo
  - Replace LSTM with Transformers, which deal better with long-term dependencies
  - Truly bidirectional architecture: left-to-right and right-to-left contexts are learned by the same network
  - Words are randomly masked during training to improve learning
  - Sentences are randomly paired to improve Next Sentence Prediction (NSP)



- Pre-trained models: BERT Base and BERT Large

	Layers	Hidden Units	Parameters
BERT base	12	768	110M
BERT large	24	1024	340M

BERT Pre-Training and Fine-Tuning



"Pre-Training is the new the training, training becomes fine-tuning."

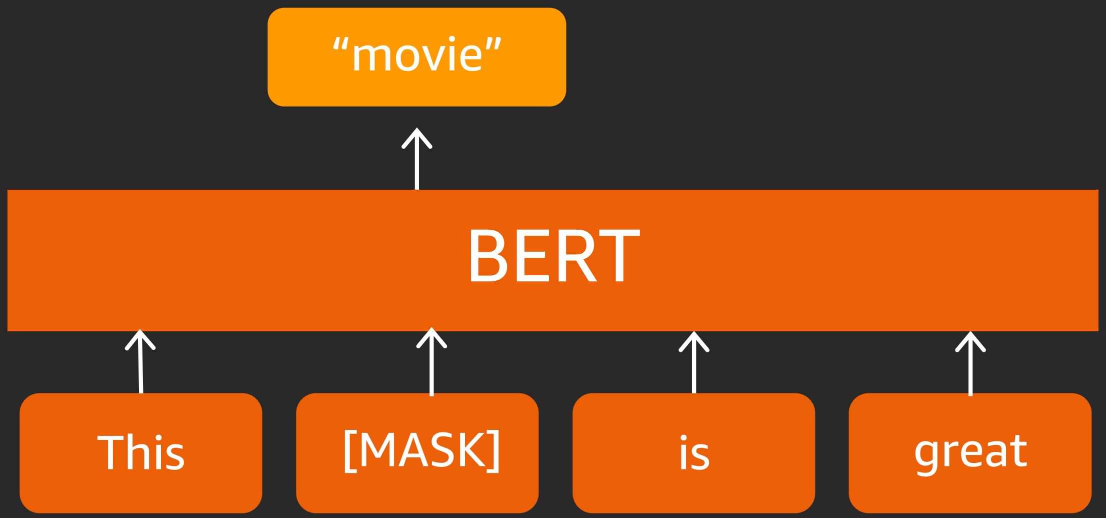
BERT Pre-Training: Masked Language Model (MLM)

Estimate $p(x_i \mid x[1{:}i-1], x[i+1{:}n])$

Randomly mask 15% of all tokens and predict token

This [MASK] is great

Outputs: P(movie| This, [MASK], is, great)



BERT Pre-Training

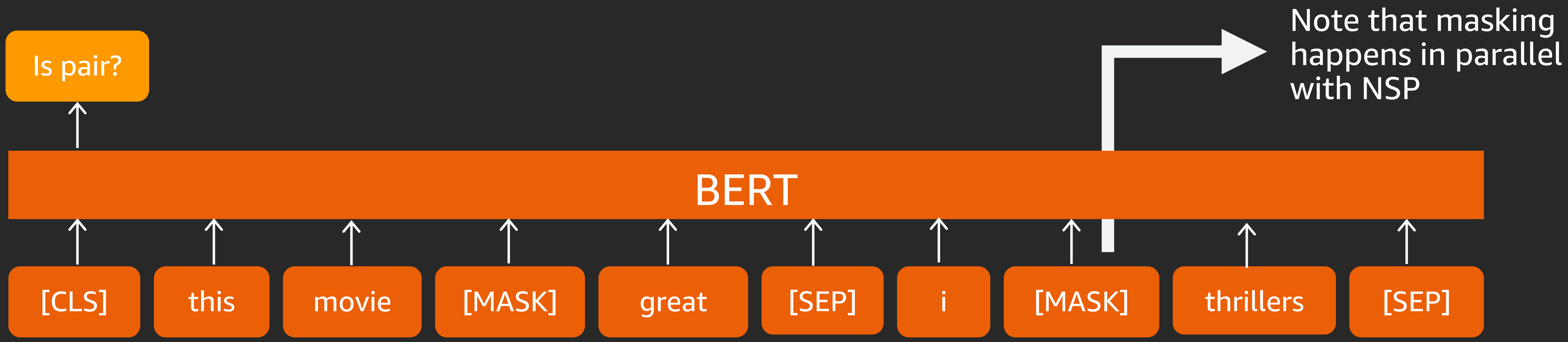
BERT Pre-Training: Next Sentence Prediction (NSP)

Predict next sentence

- 50% of the time, replace one sentence in a sentence pair with another random sentence
- Feed the two-sentence encodings into a dense layer to predict if they are a pair.

Goal: Learn logical coherence

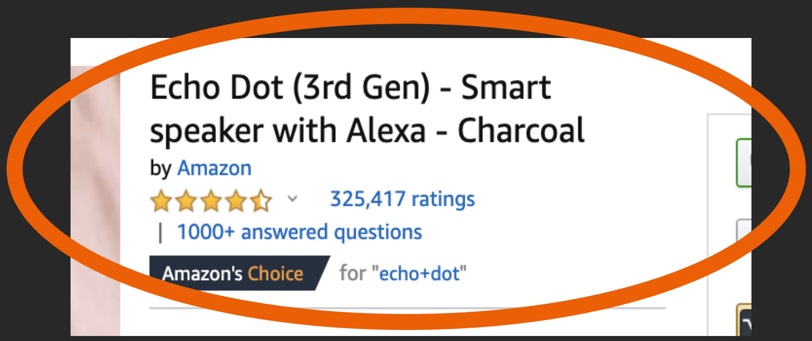
```
<cls> this movie is great <sep> i love thrillers <sep>
<cls> this movie is great <sep> tomorrow is saturday <sep>
```



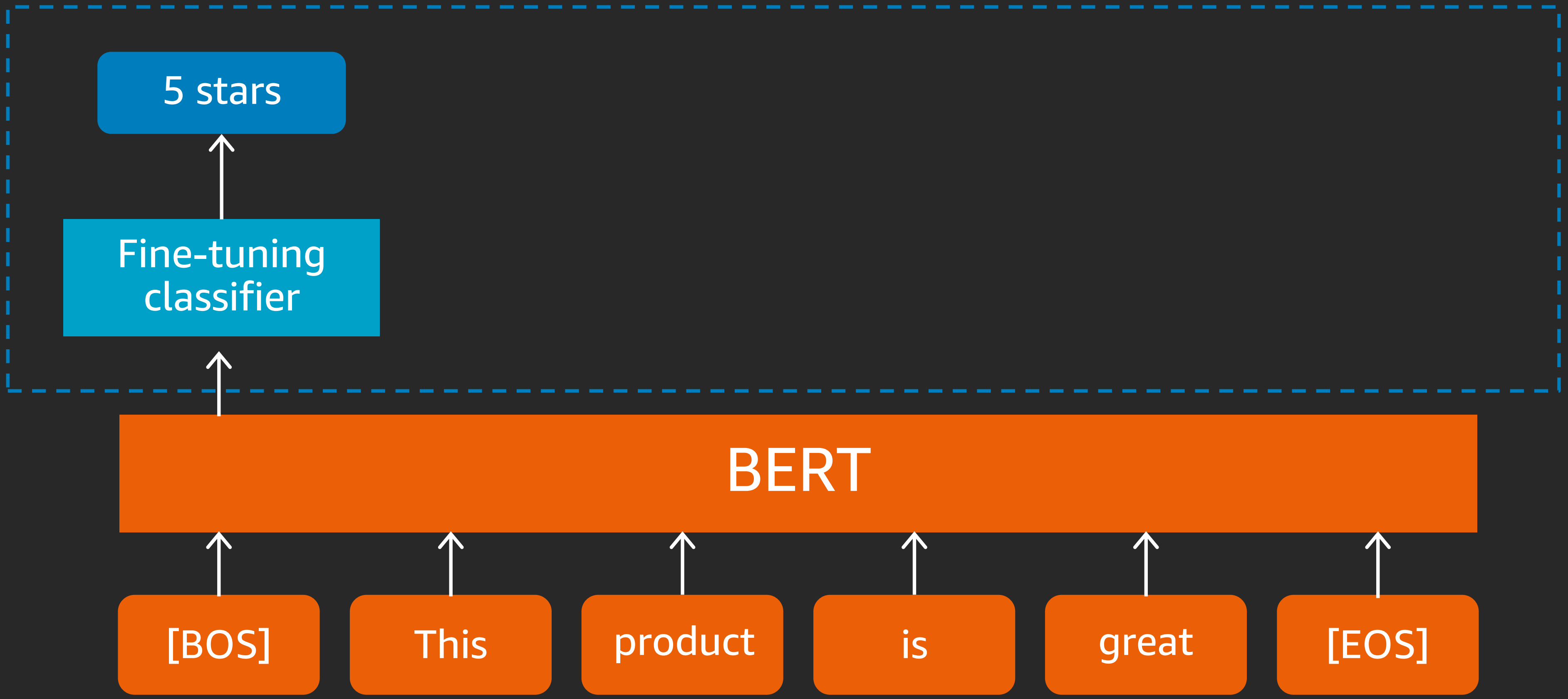
BERT Fine-Tuning

Star Rating Classifier (1 star = bad, 5 stars = good)

Output: 5 stars

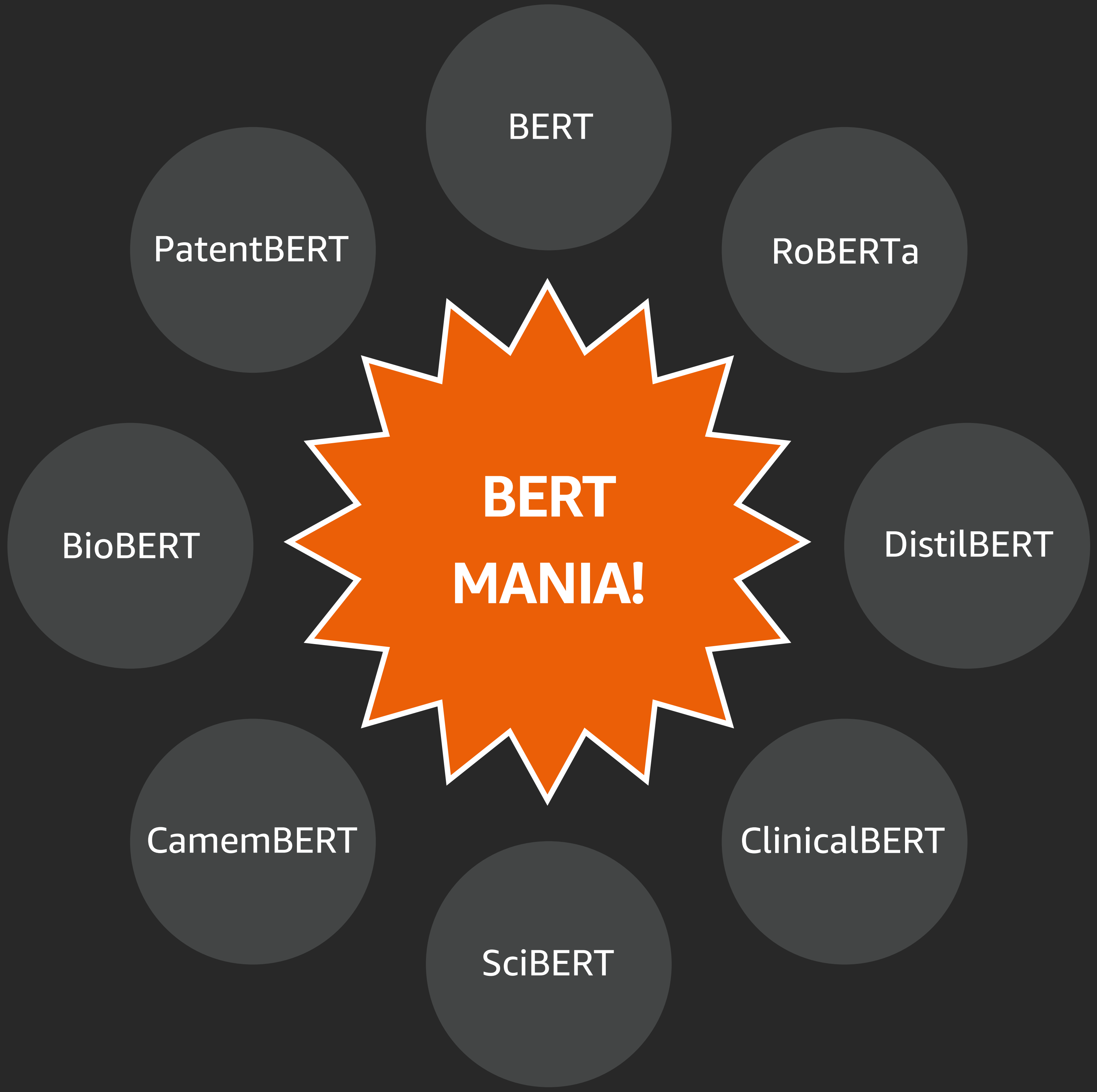


BERT fine-tuning (text classification)



Input:

This product is great


BERT
MANIA!
BERT
RoBERTa
DistilBERT
ClinicalBERT
SciBERT
CamemBERT
BioBERT
PatentBERT

NLP with Amazon SageMaker

AWS SUMMIT ONLINE

Amazon SageMaker helps you build, train, and deploy models

Amazon SageMaker Studio
IDE for ML

Amazon SageMaker Autopilot
Automatically build and train models

Prepare

- **Amazon SageMaker GroundTruth**
  Build and manage training dataset
- **Processing job**
  Supports Python or Spark

Build

- **Amazon SageMaker notebooks**
  One-click notebooks with elastic compute
- **AWS Marketplace**
  Pre-built algorithms, models, and data

Train & tune

- **One-click training**
  Supports supervised, unsupervised & RL
- **Automatic model tuning**
  One-click hyperparameter optimization
- **Amazon SageMaker Experiments**
  Capture, organize, and compare every step
- **Amazon SageMaker Debugger**
  Debug and profile training runs

Deploy & manage

- **One-click deployment**
  Supports real-time, batch & multi-model
- **Amazon SageMaker Model Monitor**
  Automatically detect concept drift
- **Amazon SageMaker Neo**
  Train once, deploy anywhere
- **Amazon Elastic Inference**
  Auto scaling for 75% less
- **Amazon Augmented AI**
  Add human review of model predictions

Popular deep learning frameworks

TensorFlow

PyTorch

mxnet

TensorFlow on AWS

aws SUMMIT ONLINE

TensorFlow is a first-class citizen on Amazon SageMaker

- Built-in TensorFlow containers for training and prediction
  - Code available on GitHub: https://github.com/aws/sagemaker-tensorflow-containers
  - Build it, run it on your own machine, and customize it
  - Versions : 1.4.1 → 1.15, 2.0, 2.1

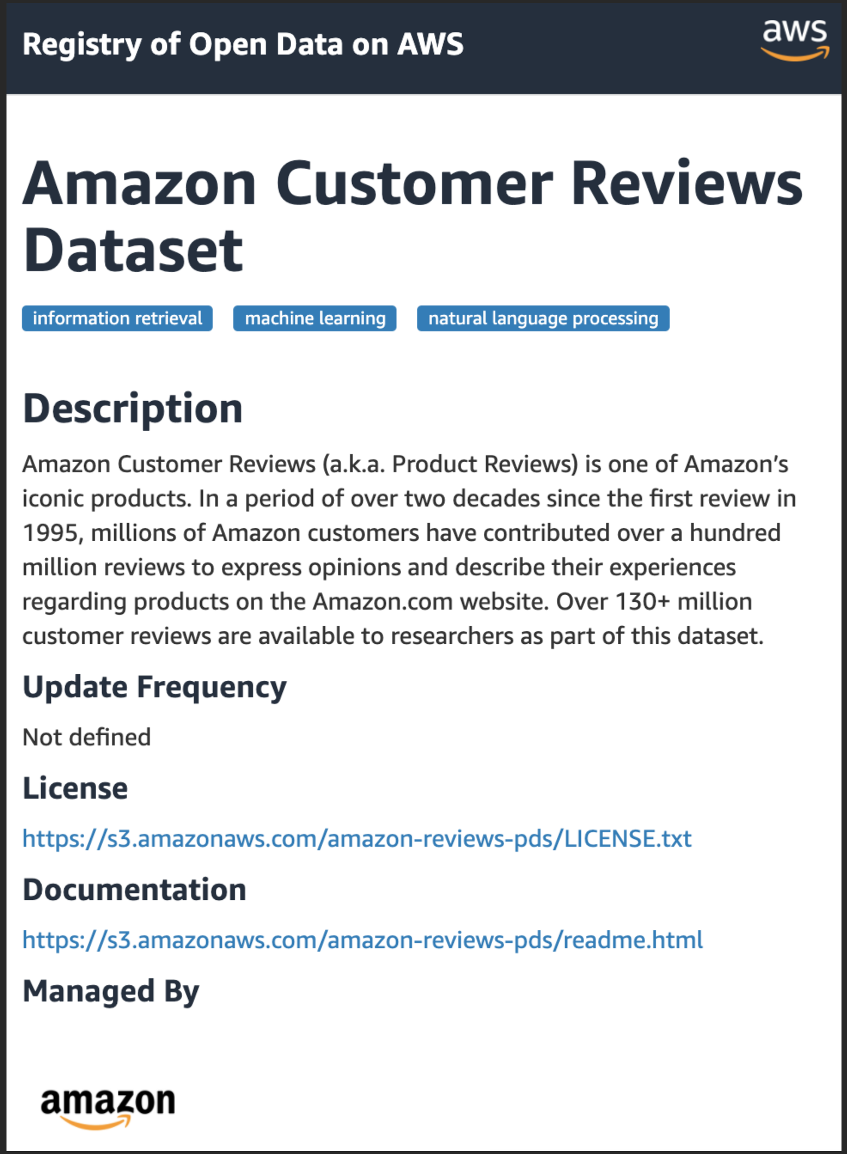
- TensorFlow tooling
  - Standard tools: TensorBoard, TensorFlow Serving
  - Amazon SageMaker features: Local mode, script mode, model tuning, Managed Spot Training, Pipe mode, Amazon EFS & Amazon FSx for Lustre, Amazon Elastic Inference
  - Performance optimizations: GPUs and CPUs (AWS, Intel MKL-DNN library)
  - Distributed training: Parameter Server and Horovod

Demo

https://github.com/data-science-on-aws/workshop/blob/e90f4be78be47f951ec9f0a13617d65e559ddea7/06_train/03_Train_Reviews_BERT_Transformers_TensorFlow_ScriptMode.ipynb

Amazon Customer Reviews Dataset

https://registry.opendata.aws/amazon-reviews/


Registry of Open Data on AWS

Amazon Customer Reviews Dataset

information retrieval · machine learning · natural language processing

Description

Amazon Customer Reviews (a.k.a. Product Reviews) is one of Amazon's iconic products. In a period of over two decades since the first review in 1995, millions of Amazon customers have contributed over a hundred million reviews to express opinions and describe their experiences regarding products on the Amazon.com website. Over 130+ million customer reviews are available to researchers as part of this dataset.

Update Frequency

Not defined

License

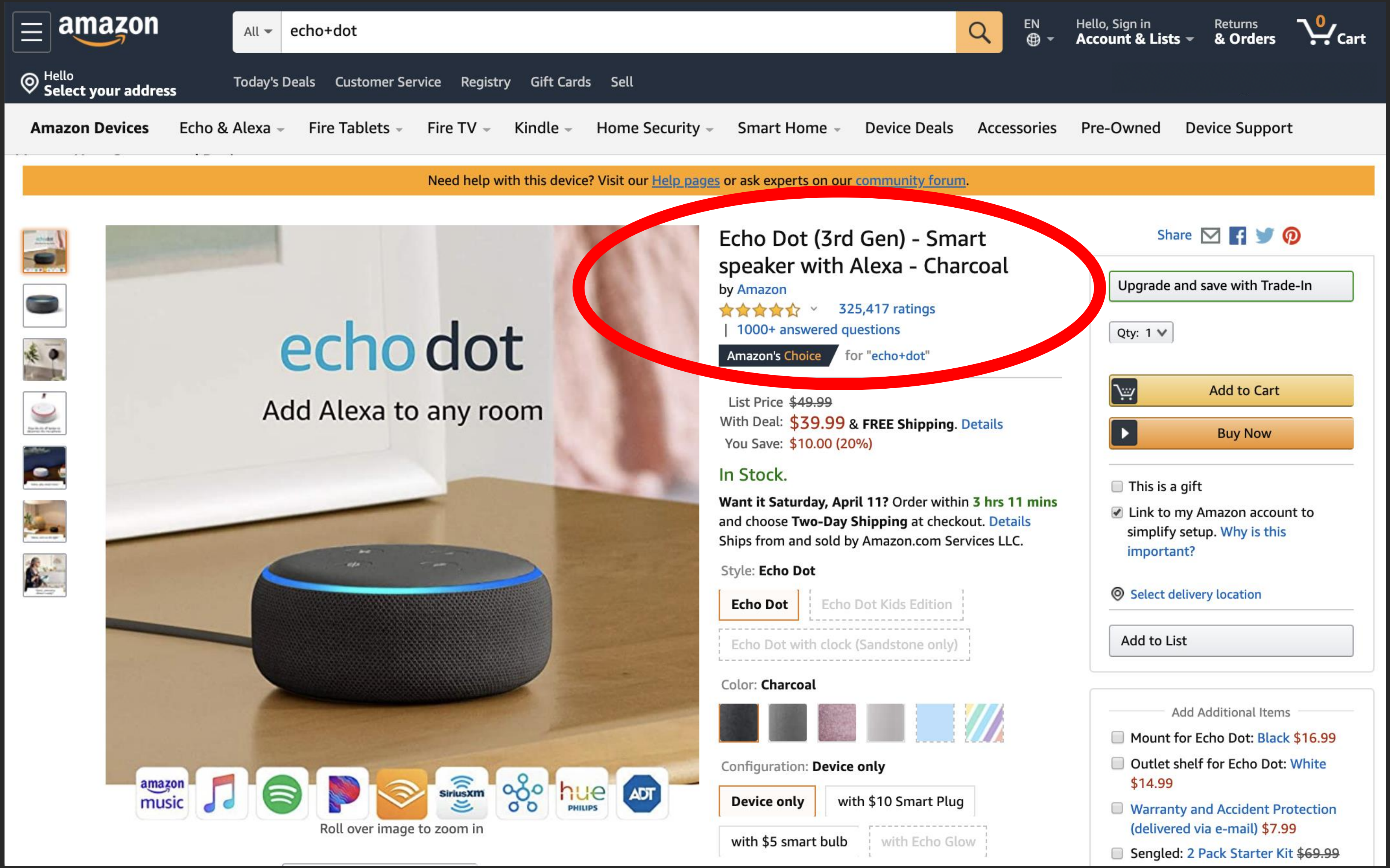
https://s3.amazonaws.com/amazon-reviews-pds/LICENSE.txt

Documentation

https://s3.amazonaws.com/amazon-reviews-pds/readme.html

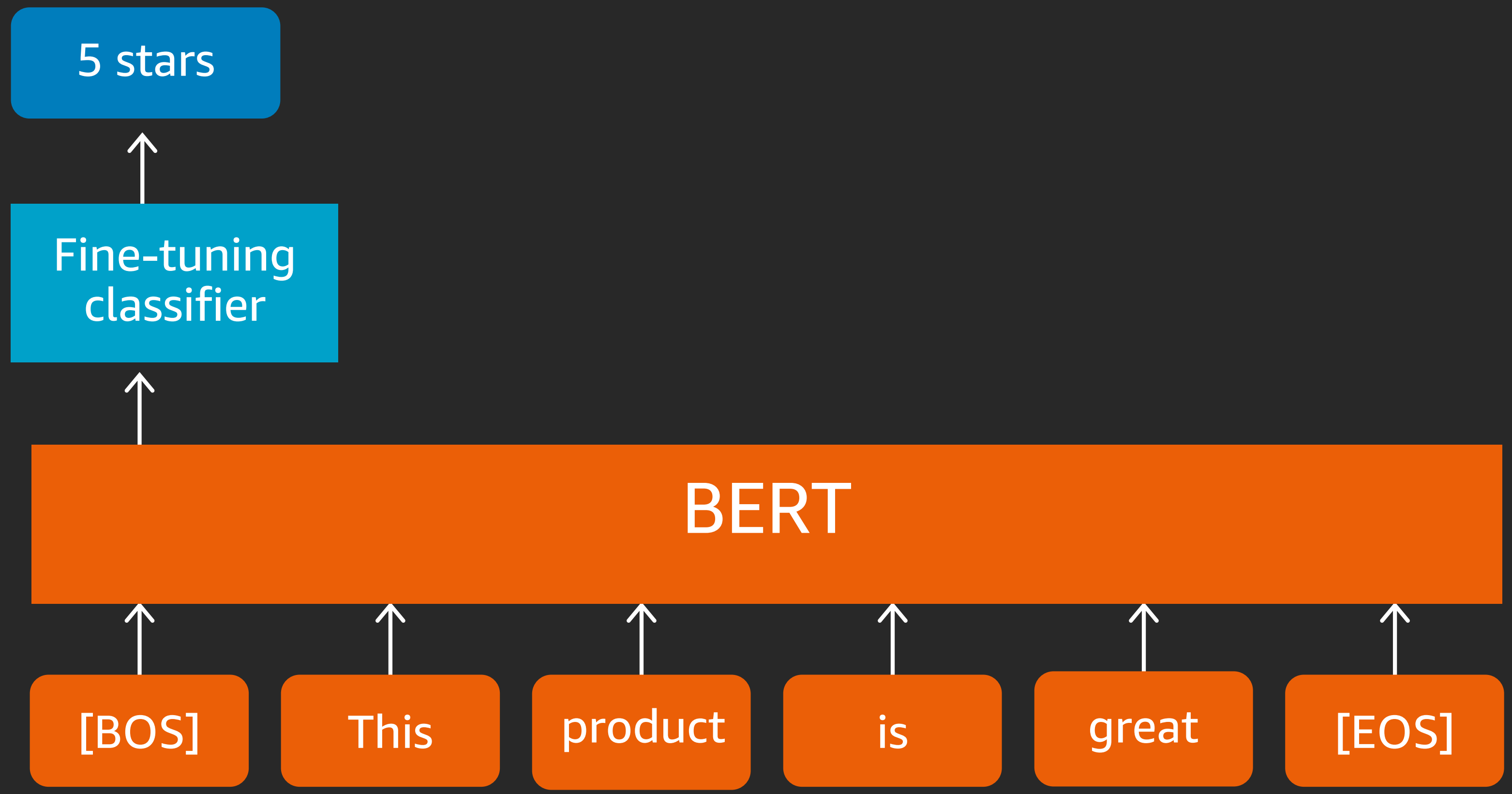
Managed By

amazon



Text Classification (Star Rating)

1. Build a dataset of labeled sentences
2. Grab a pre-trained model (BERT), and add a classification layer
3. Convert each sentence (Amazon review) to a list of vectors using pre-trained tokenizer (BERT tokenizer)
4. Train or fine-tune the model to predict the correct class (star rating) for each review



PyTorch on AWS

aws SUMMIT ONLINE

PyTorch


Eager & graph-based execution
Dynamic neural networks
Distributed training
Hardware accelerated inference
Simplicity over complexity

PyTorch on AWS

```
# Create the estimator
from sagemaker.pytorch import PyTorch

pytorch_estimator = PyTorch( entry_point='pytorch-bert.py’,
                             source_dir=’src’,
                             train_instance_type='ml.p3.2xlarge’,
                             train_instance_count=2,
                             framework_version=‘1.4.0’,
                             hyperparameters = {'epochs': 20, 'batch-size': 64,
                                                'learning-rate': 0.1})

# Train the estimator
pytorch_estimator.fit({'train': 's3://my-data-bucket/path/to/my/training/data’,
                       'test': 's3://my-data-bucket/path/to/my/test/data’})

# Deploy the estimator to a SageMaker Endpoint and get a Predictor
predictor = pytorch_estimator.deploy(instance_type='ml.m4.xlarge’,
                                     initial_instance_count=1)

# `data` is a NumPy array or a Python list, `response` is a NumPy array.
response = predictor.predict(data)
```

BERT with PyTorch

```python
import torch
from transformers import *

# Transformers has a unified API
# for 10 transformer architectures and 30 pretrained weights.
#          Model          | Tokenizer          | Pretrained weights shortcut
MODELS = [(BertModel,       BertTokenizer,       'bert-base-uncased'),
          (OpenAIGPTModel,  OpenAIGPTTokenizer,  'openai-gpt'),
          (GPT2Model,       GPT2Tokenizer,       'gpt2'),
          (CTRLModel,       CTRLTokenizer,       'ctrl'),
          (TransfoXLModel,  TransfoXLTokenizer,  'transfo-xl-wt103'),
          (XLNetModel,      XLNetTokenizer,      'xlnet-base-cased'),
          (XLMModel,        XLMTokenizer,        'xlm-mlm-enfr-1024'),
          (DistilBertModel, DistilBertTokenizer, 'distilbert-base-cased'),
          (RobertaModel,    RobertaTokenizer,    'roberta-base'),
          (XLMRobertaModel, XLMRobertaTokenizer, 'xlm-roberta-base'),
         ]

# To use TensorFlow 2.0 versions of the models, simply prefix the class names with 'TF', e.g. `TFRobertaModel`
```

🤗 Transformers

https://github.com/huggingface/transformers#quick-tour

Deploying PyTorch models in production is a challenge

- No official model server
- Need to write custom code to deploy and predict with the trained models
- For production workloads, need to build your own systems for scaling, monitoring, security, etc.

Introducing TorchServe

NEW

A PyTorch model serving library, built and maintained by AWS in collaboration with Facebook.

Easily deploy PyTorch models in production at scale

Model Training

Train models with PyTorch

TorchServe

Start TorchServe model server and load trained models

Prediction API

One or more models hosted on SageMaker, EC2, EKS, Kubernetes, or any other ML environment

Prediction

Client Application

Key features

- Low latency prediction API provided automatically
- Default handlers for most common applications like object detection, text classification, etc.
- Multi-model serving
- Model versioning for A/B testing
- Monitoring/logging
- RESTful end points that can be accessed via web requests (HTTP)

Get started with TorchServe: https://github.com/pytorch/serve

"AWS enabled us to train BERT models for our enterprise clients in a scalable way"

deepset

How to optimize BERT training?

Milos Rusic, CEO

Malte Pietsch, CTO

Timo Möller, Head of ML

Training transformers costs a lot of time and money

Common training modes

Pre-Training: New language or special domain

Domain adaptation: Similar language, but domain-related differences

Downstream: Question Answering, NER, Classification …

* Using a Tesla V100 GPU

	Time *	Costs	#Runs (typical client)
Pre-Training	2000+ GPU hours	$$$	0-2
Domain Adapt.	100+ GPU hours	$$	1-5
Downstream (QA)	10+ GPU hours	$	> 30 (Experiments, Re-training …)

→ Efficient training is a big cost saver, especially for pre-training

deepset

Optimized training via Amazon SageMaker

Accelerate training

PyTorch's DistributedDataParallel	→ ~ 30% faster
Automatic Mixed Precision (AMP)	→ ~ 80% faster
Lazy Data Loading	→ No up-front preprocessing time

Managed Spot Training

Checkpointing of all objects (model, optimizer, LR schedule, data loader …)

→ ~ 70% savings

→ Open-source code: FARM

→ Marketplace algorithm: Training BERT via FARM (coming soon)

deepset

Apache MXNet on AWS

Apache MXNet (incubating)

https://mxnet.apache.org/

Scalable

Debuggable

Flexible

Optimized libraries

8 front-end languages

Portable

MXNet on AWS

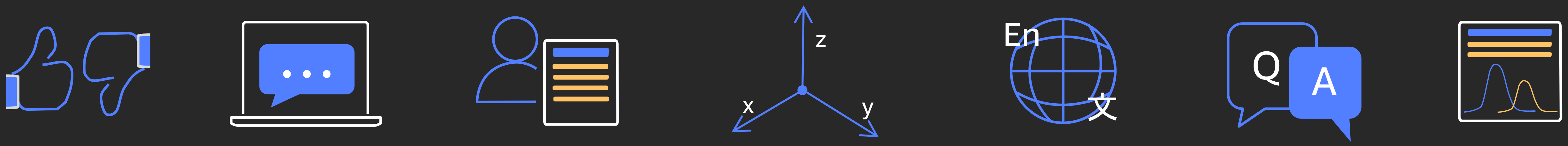
```
# Create an estimator
from sagemaker.mxnet import MXNet

mxnet_estimator = MXNet(‘mxnet-bert.py‘,
                        source_dir=’src’,
                        train_instance_type='ml.p2.xlarge',
                        train_instance_count=1,
                        framework_version='1.3.0',
                        hyperparameters={'batch-size': 100, 'epochs': 10,
                                         'learning-rate': 0.1})


# Train the estimator
mxnet_estimator.fit('s3://my_bucket/my_training_data/’)


# Deploy the estimator to a SageMaker Endpoint and get a Predictor
predictor = mxnet_estimator.deploy(instance_type='ml.m4.xlarge‘,
                                   initial_instance_count=1)
```

GluonNLP Toolkit

https://gluon-nlp.mxnet.io/

- Comprehensive model zoo
- State-of-the-art models
- Out-of-the-box preprocessing
- Many tutorials & examples

Built-in NLP tasks

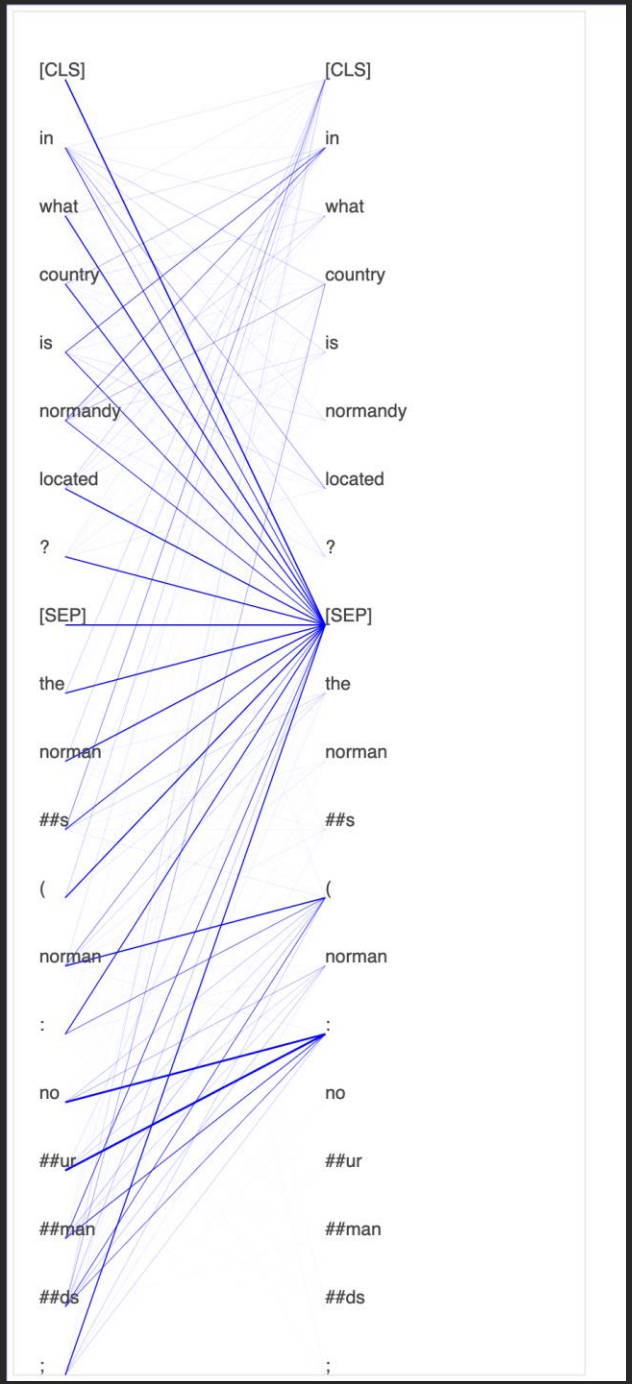


Sentiment Analysis | Text Generation | Named Entity Recognition | Representation Learning | Machine Translation | Question Answering | Language Modeling

Visualizing BERT Attention
Using GluonNLP and SageMaker Debugger

- Download a BERT model from the GluonNLP model zoo and finetune the model on the Stanford Question Answering dataset (”SQuAD“)
- Use Amazon SageMaker Debugger to monitor attentions in BERT model training in real-time.

Notebook:
https://github.com/awslabs/amazon-sagemaker-examples/blob/master/sagemaker-debugger/model_specific_realtime_analysis/bert_attention_head_view/bert_attention_head_view.ipynb



https://arxiv.org/pdf/1904.02679.pdf

Get started

Get 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

https://github.com/data-science-on-aws/workshop

APN **Machine Learning** Competency Partners

accenture

Deloitte.

Visit the Partner Discovery Zone to meet these partners and view the full list of APN Competency Partners

Thank you!

Antje Barth

@anbarth

data-science-on-aws/workshop

aws SUMMIT ONLINE