Copyright Notice

These slides are distributed under the Creative Commons License.

DeepLearning.AI makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite DeepLearning.AI as the source of the slides.

For the rest of the details of the license, see https://creativecommons.org/licenses/by-sa/2.0/legalcode

DeepLearning.AI

aws

Practical Data Science

Built-In Algorithms

Machine Learning Workflow

Ingest & Analyze	Prepare & Transform	Train & Tune	Deploy & Manage
Data exploration	Feature engineering	Automated ML	Model deployment
Bias detection	Feature store	Model train and tune	Automated pipelines
Amazon S3 & Amazon Athena	Amazon SageMaker Data Wrangler	Amazon SageMaker Autopilot	Amazon SageMaker Endpoints
AWS Glue	Amazon SageMaker Processing Jobs	Amazon SageMaker Training & Debugger	Amazon SageMaker Batch Transform
Amazon SageMaker Data Wrangler & Clarify	Amazon SageMaker Feature Store	Amazon SageMaker Hyperparameter Tuning	Amazon SageMaker Pipelines

Why use built-in algorithms?

- Implementations are highly-optimized and scalable
- Focus more on domain-specific tasks rather than managing low-level model code and infrastructure
- Trained model can be downloaded and re-used elsewhere

When to choose built-in algorithms vs. custom code

Amazon SageMaker

Built-in Algorithms

Amazon SageMaker

Bring Your Own Script

Amazon SageMaker

Bring Your Own Container

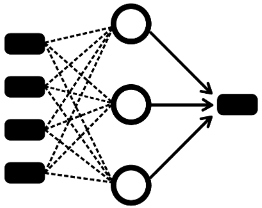
Less Code

More Customizable

Use cases and algorithms

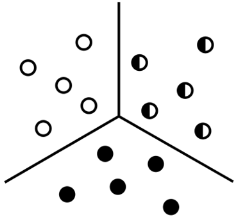
DeepLearning.AI
aws

Popular ML tasks and learning paradigms



Classification & Regression

Supervised



Clustering

Unsupervised

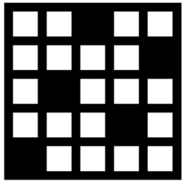
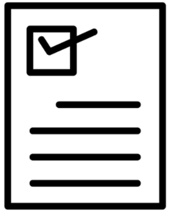


Image Processing

Computer Vision



Text Analysis

NLP / NLU

Classification & regression

Example problems and use cases	Problem types	Input format	Built-in algorithms
Predict if an item belongs to a category: an email spam filter	Binary/multi-class classification	Tabula r	XGBoost, K-Nearest Neighbors (k-NN)

Classification & regression

Example problems and use cases	Problem types	Input format	Built-in algorithms
Predict if an item belongs to a category: an email spam filter	Binary/multi-class classification	Tabula r	XGBoost, K-Nearest Neighbors (k-NN)
Predict a numeric/continuous value: estimate the value of a house	Regression	Tabula r	Linear Learner, XGBoost

Classification & regression

Example problems and use cases	**Problem types**	**Input format**	**Built-in algorithms**
Predict if an item belongs to a category: an email spam filter	Binary/multi-class classification	Tabular	XGBoost, K-Nearest Neighbors (k-NN)
Predict a numeric/continuous value: estimate the value of a house	Regression	Tabular	Linear Learner, XGBoost
Predict sales on a new product based on previous sales data	Time-series forecasting	Tabular	DeepAR Forecasting a supervised learning algorithm for forecasting scalar (one dimentional time series) using Recurrent Neural Networks (RNN)

Clustering

Example problems and use cases	Problem types	Input format	Built-in algorithms
Drop weak features such as the color of a car when predicting its mileage.	Feature engineering: reduce dimensions	Tabular	Principal Component Analysis (PCA)

Clustering

Example problems and use cases	Problem types	Input format	Built-in algorithms
Drop weak features such as the color of a car when predicting its mileage.	Feature engineering: reduce dimensions	Tabular	Principal Component Analysis (PCA)
Detect abnormal behavior	Anomaly detection	Tabular	Random Cut Forest (RCF)

RCF is unsupervised algorithm for detecting anomalous data points within a dataset.
RCF associates an anomaly score with each data point. Low score values indicate that the data point is considered normal. High value indicates the presence anomaly in the data.

Clustering

Example problems and use cases	Problem types	Input format	Built-in algorithms
Drop weak features such as the color of a car when predicting its mileage.	Feature engineering: reduce dimensions	Tabular	Principal Component Analysis (PCA)
Detect abnormal behavior	Anomaly detection	Tabular	Random Cut Forest (RCF)
Group high/medium/low-spending customers from transaction histories	Clustering / grouping	Tabular	K-Means Density Based Clustering algorithm (DBSCAN)

Clustering

Example problems and use cases	Problem types	Input format	Built-in algorithms
Drop weak features such as the color of a car when predicting its mileage.	Feature engineering: reduce dimensions	Tabular	Principal Component Analysis (PCA)
Detect abnormal behavior	Anomaly detection	Tabular	Random Cut Forest (RCF)
Group high/medium/low-spending customers from transaction histories	Clustering / grouping	Tabular	K-Means
Organize a set of documents into topics based on words and phrases	Topic modeling	Text	Latent Dirichlet Allocation (LDA), Neural Topic Model (NTM)

Image processing

Example problems and use cases	Problem types	Input format	Built-in algorithms
Content moderation	Image classification	Image	Image Classification

Image processing

Example problems and use cases	Problem types	Input format	Built-in algorithms
Content moderation	Image classification	Image	Image Classification
Detect people and objects in an image	Object detection	Image	Object Detection

Image processing

Example problems and use cases	**Problem types**	**Input format**	**Built-in algorithms**
Content moderation	Image classification	Image	Image Classification
Detect people and objects in an image	Object detection	Image	Object Detection
Self-driving cars identify objects in their path	Computer vision	Image	Semantic Segmentation It classifies every pixel in the image. (different from object detection and image classification)

This leads to information such as the shapes of the objects contained in the image. The segmentation output is represented as a grayscale image called a segmentation mask that has the same shape as the input image.

Text analysis

Example problems and use cases	Problem types	Input format	Built-in algorithms
Convert Spanish to English	Machine translation	Text	Sequence-to-Sequence

Text analysis

Example problems and use cases	Problem types	Input format	Built-in algorithms
Convert Spanish to English	Machine translation	Text	Sequence-to-Sequence
Summarize a research paper	Text summarization	Text	Sequence-to-Sequence

Text analysis

Example problems and use cases	Problem types	Input format	Built-in algorithms
Convert Spanish to English	Machine translation	Text	Sequence-to-Sequence
Summarize a research paper	Text summarization	Text	Sequence-to-Sequence
Transcribe call center conversations	Speech-to-text	Text	Sequence-to-Sequence

Text analysis

Example problems and use cases	Problem types	Input format	Built-in algorithms
Convert Spanish to English	Machine translation	Text	Sequence-to-Sequence
Summarize a research paper	Text summarization	Text	Sequence-to-Sequence
Transcribe call center conversations	Speech-to-text	Text	Sequence-to-Sequence
Classify reviews into categories	Text classification	Text	BlazingText

Text analysis

Evolution of text analysis algorithms

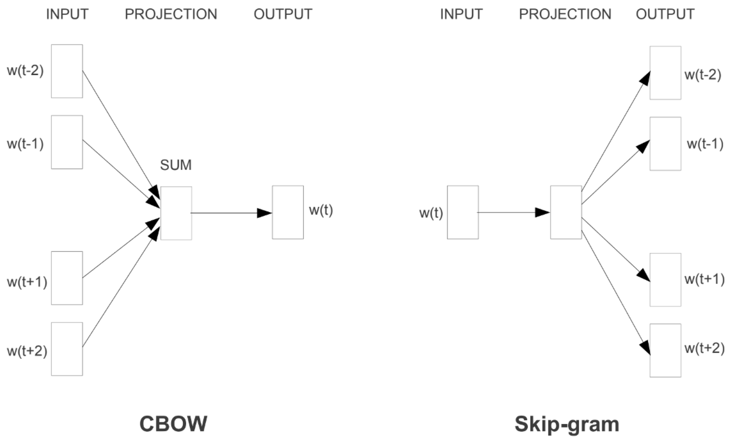
Text analysis algorithm - Word2Vec

Concepts

- Convert text into vectors called “embeddings”
- 300-dimensional vector space
- Perform machine learning on the vectors

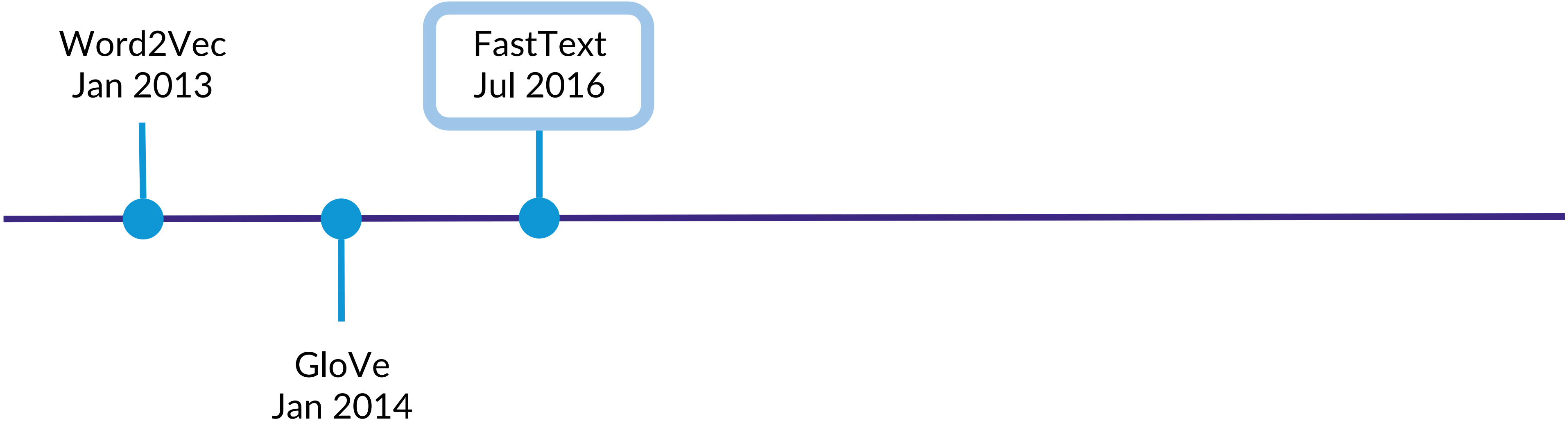
Model architectures to create the embeddings

- Continuous bag-of-words (CBOW)
- Continuous skip-gram



Source: "Efficient Estimation of Word Representations in Vector Space", Mikolov et al., 2013

Evolution of text analysis algorithms



Text analysis algorithm - FastText

Concepts

- Extension of word2vec
- Breaks the word into character sets of length n (n-grams):

```
"amazon" => "a", "am", "ama", "amaz", "amazo", "amazon"
```

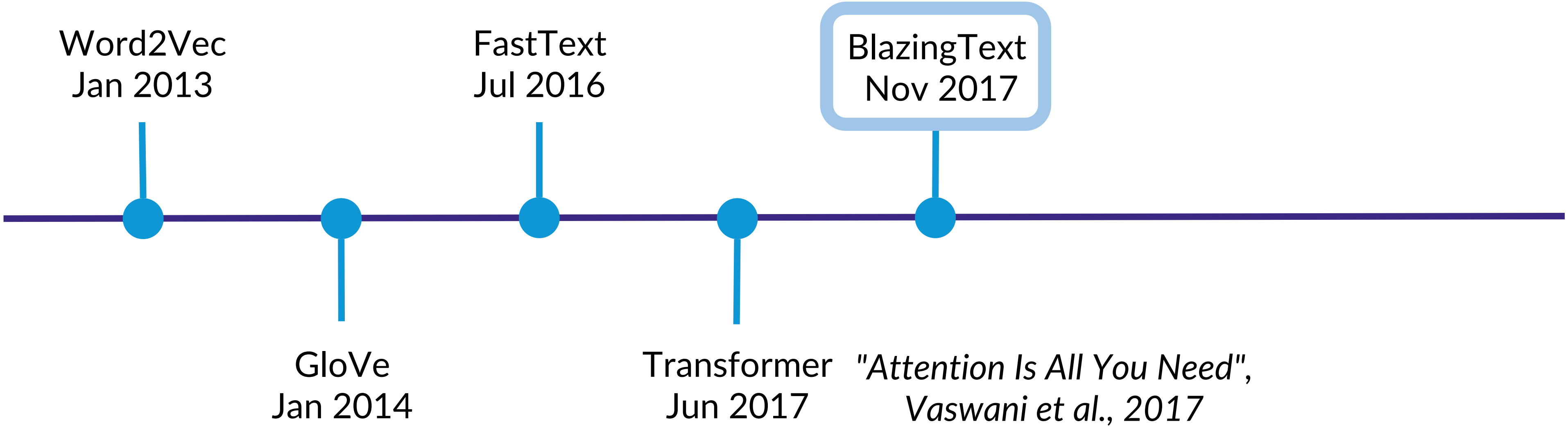
- Embedding for a word is the aggregate of the embedding of each n-gram within the word

Helps with the out-of-vocabulary (OOV) issue with word2vec

Implementation

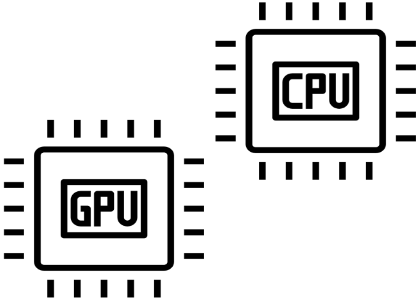
- CBOW and skip-gram models
- Adds text classification

Evolution of text analysis algorithms

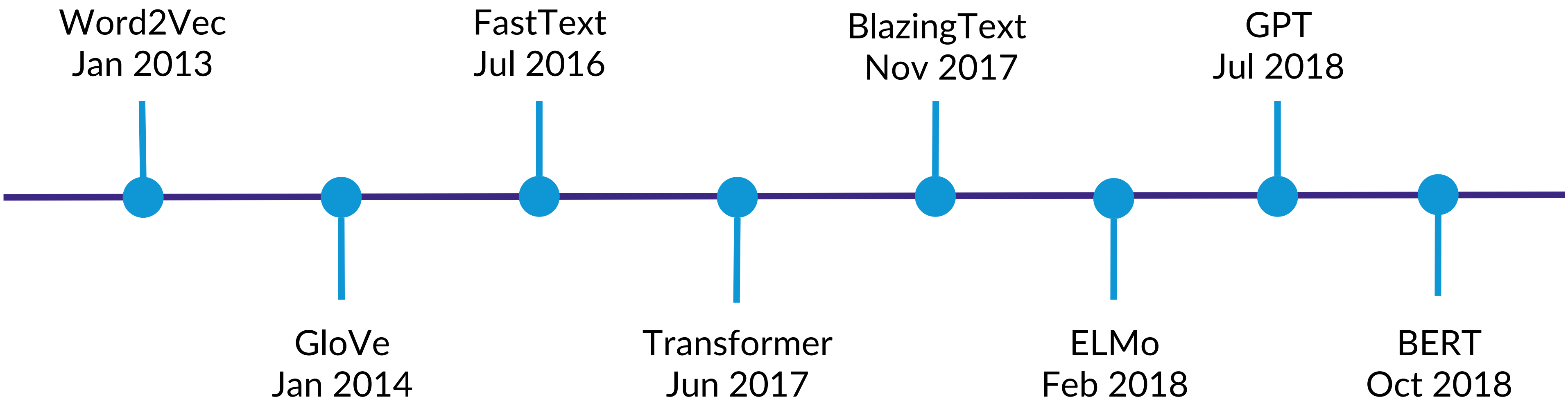


Text analysis algorithm - BlazingText

- Scales and accelerates Word2Vec using multiple CPUs or GPUs for training
- Extends FastText to use GPU acceleration with custom CUDA kernels
- Creates n-gram embeddings using CBOW and skip-gram
- Saves money by early-stopping a training job
  - when the validation accuracy stops increasing
- Optimized I/O for datasets stored in Amazon S3



Evolution of text analysis algorithms



Embeddings from Language models
Words are learned by a deep bidirectional language model. Hence able to capture syntax and semantics across different linguistic contexts.

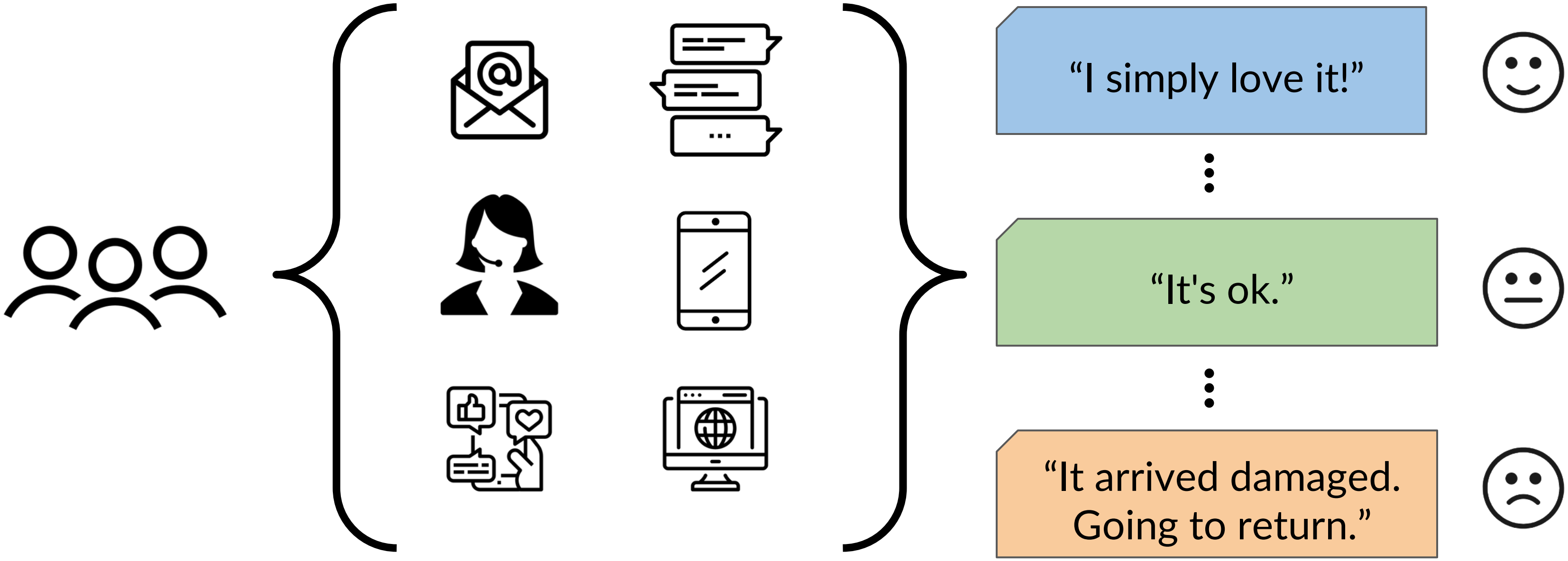
Train a text classifier

with Amazon SageMaker BlazingText

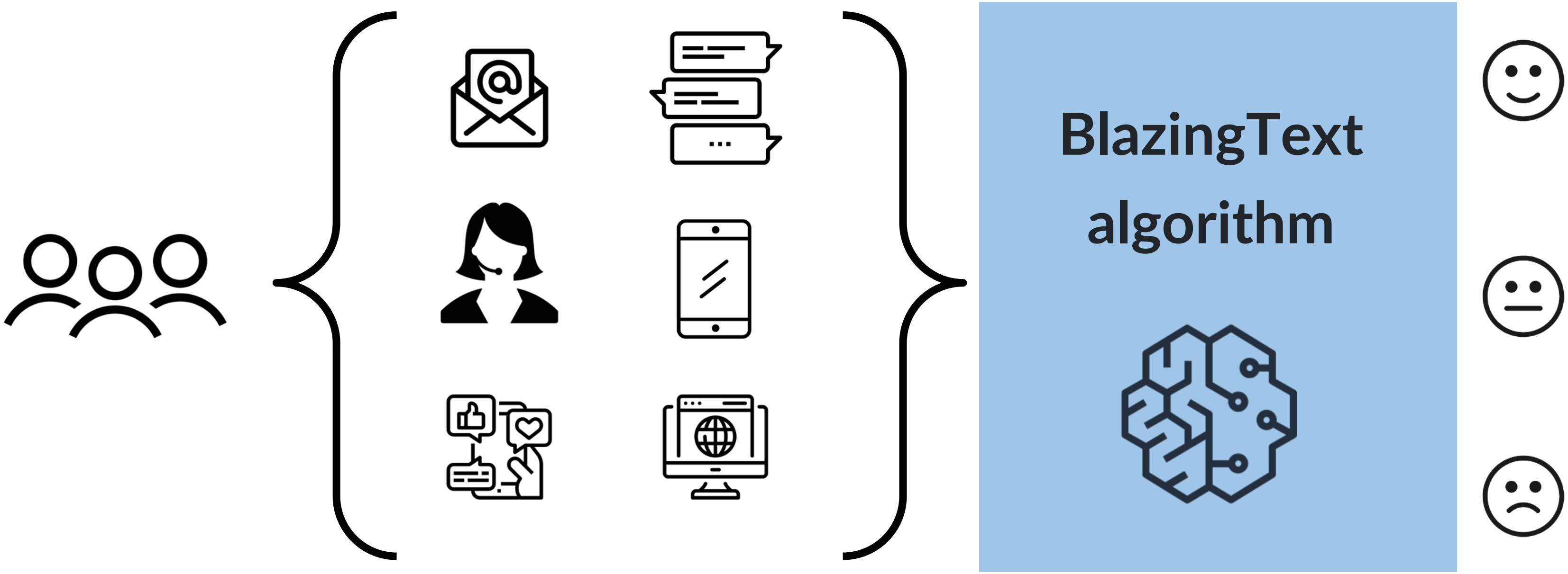
DeepLearning.AI

aws

Multi-class classification for sentiment analysis of product reviews



Multi-class classification for sentiment analysis of product reviews



DeepLearning.AI
aws

Transform raw review data into features

Ingest & Analyze → **Prepare & Transform** → Train & Tune → Deploy & Manage

```
sentiment,review_body
1,"i simply love it"

0,"it's ok"

-1,"it arrived damaged. going to return"
```

```
__label__1 "i simply love it ."

__label__0 "it's ok ."

__label__-1 "it arrived damaged ."
```

Transform raw review data into features

Ingest & Analyze → **Prepare & Transform** → Train & Tune → Deploy & Manage

NLTK

```
def tokenize(review):
    return nltk.word_tokenize(review)
```

```
__label__1 "i simply love it ."

__label__0 "it's ok ."

__label__-1 "it arrived damaged, going to return ."
```

Amazon SageMaker BlazingText hyper-parameters for text classification

Parameter Name	Recommended Ranges or Values	Description
epochs	[5-15]	Number of complete passes through the dataset
learning_rate	[0.005-0.01]	Step size for the numerical optimizer
min_count	[0-100]	Discard words that appear less than this number
vector_dim	[32-300]	Number of dimensions in vector space
word_ngrams	[1-3]	Number of words n-gram features to use
early_stopping	True or False	Stop training if validation accuracy stops improving
patience	[5-15]	Number of epochs before early stopping

Train a text classifier using Amazon SageMaker BlazingText

Ingest & Analyze → Prepare & Transform → **Train & Tune** → Deploy & Manage

```
train_data = sagemaker.inputs.TrainingInput(...)
validation_data = sagemaker.inputs.TrainingInput(...)

data_channels = {
    'train': train_data,
    'validation': validation_data
}

image_uri = sagemaker.image_uris.retrieve(framework='blazingtext', ...)
```

Retrieves Amazon ECR image URIs for pre-built SageMaker Docker images.

Train a text classifier using Amazon SageMaker BlazingText

Ingest & Analyze → Prepare & Transform → **Train & Tune** → Deploy & Manage

```
train_data = sagemaker.inputs.TrainingInput(...)
validation_data = sagemaker.inputs.TrainingInput(...)

data_channels = {
    'train': train_data,
    'validation': validation_data
}

image_uri = sagemaker.image_uris.retrieve(framework='blazingtext', ...)

estimator = sagemaker.estimator.Estimator(image_uri=image_uri, ...)
estimator.set_hyperparameters(...)
estimator.fit(...)
```

Retrieves Amazon ECR image URIs for pre-built SageMaker Docker images.

Evaluate the classifier

Ingest & Analyze → Prepare & Transform → **Train & Tune** → Deploy & Manage

time	metric_name	value
00.0	train:accuracy	0.4865
10.0	train:accuracy	0.5220
20.0	validation:accuracy	0.5364

DeepLearning.AI
aws

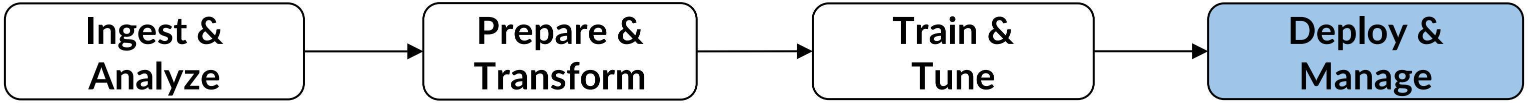
Deploy the text classifier

and make predictions

DeepLearning.AI

aws

Deploy the text classifier




```
text_classifier = estimator.deploy(
    initial_instance_count=1,
    instance_type='ml.m4.xlarge', ...)
```

Increase instance_count > 1 to easily scale out

Deploy the text classifier

Ingest & Analyze → Prepare & Transform → Train & Tune → **Deploy & Manage**

```
text_classifier = estimator.deploy(
    initial_instance_count=1,
    instance_type='ml.m4.xlarge', ...)
```

blazingtext-2020-12-07-21-45-06-296

Endpoint settings

Name	Status
blazingtext-2020-12-07-21-45-06-296	InService
ARN	Creation time
arn:aws:sagemaker:us-east-1:835319576252:endpoint/blazingtext-2020-12-07-21-45-06-296	Mon Dec 07 2020 13:45:07 GMT-0800 (Pacific Standard Time)
	Last updated
	Mon Dec 07 2020 13:51:23 GMT-0800 (Pacific Standard Time)

Deploy the text classifier

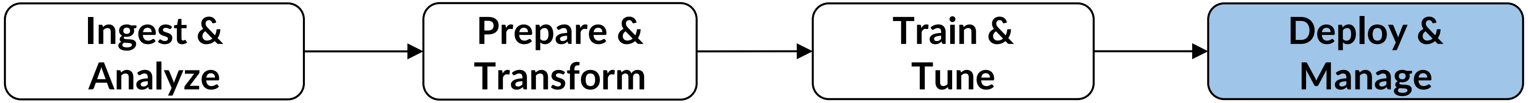
Ingest & Analyze → Prepare & Transform → Train & Tune → **Deploy & Manage**

```
text_classifier = estimator.deploy(
    initial_instance_count=1,
    instance_type='ml.m4.xlarge', ...)

payload = {'instances': ['This product is great']}
response = text_classifier.predict(...)
```

Sample prediction request

Deploy the text classifier




```
text_classifier = estimator.deploy(
    initial_instance_count=1,
    instance_type='ml.m4.xlarge', ...)

payload = {'instances': ['This product is great']}
response = text_classifier.predict(...)

## Sample response:
[{
    "label": ["__label__1"],
    "prob": [0.9506041407585144]
}]
```

Sample prediction request

Prediction response and probability score