Advanced Methods in Natural Language Processing

Session 3: Word Embeddings

Arnault Gombert

May 2026

Barcelona School of Economics

Introduction

Introduction to Word Embeddings

Today's Focus: Unveiling the Power of Word Embeddings

- **Sparse Vectors and Ontologies**: Evolution of word representation.
- **Embedding quality**: Evaluating embeddings methods.

Old Fashion Embedding Techniques

- **Word2Vec Insights**: Understanding the mechanics and impact of Word2Vec and the Skip-Gram model.
- **Exploring Static Word Embeddings**: Other famous static wordembedding methods: GloVe and FastText.

Advancing to Sophisticated Embedding Techniques

- **Contextual Word Embeddings**: Delving into ELMo.
- **Future of Embeddings**: Advanced models like BERT and GPT-x.

Word Representation

Introduction & Motivations

Traditional word representation methods (cf. Session 1):

- **One-Hot Vectors**: Each token is associated with a unique index.
- **Token Counts** (Hans P. Luhn, 1957): Token frequencies in the text.
- **TF-IDF** (Spärck Jones, K., 1972): Token's importance relative to its frequency across documents.

Limitations of Traditional Techniques:

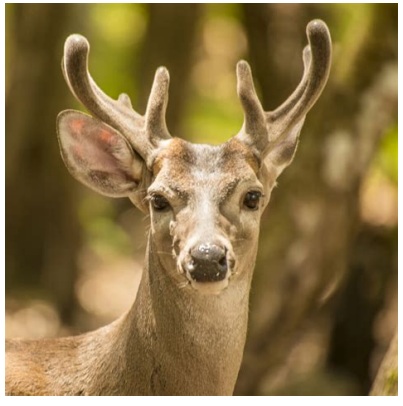
- **Sparsity**: Dimensions depends on vocabulary length, sparse vectors, inefficient for computation.
- **Lack of Context**: Do not capture the context or semantics of words, limiting the representation's expressiveness.
- **Synonymy and Polysemy**: Struggle with words that have multiple meanings or similar meanings, leading to ambiguity in representation.

Introduction & Motivations: Capturing Semantic Similarity

Semantic Similarity - A Human Concept:

- Traditional models often fail to capture the inherent semantic similarity between concepts that are intuitively understood by humans.
- Examples of semantically similar concepts:
  - *'stag'* and *'deer'*
  - *'osteopath'* and *'physiotherapist'*
  - *'prince'* and *'king'*
- This gap highlights the need for models that can effectively *transfer* human-like understanding of language and semantics.

Stag



Deer

Credits: DuckDuckGo Research

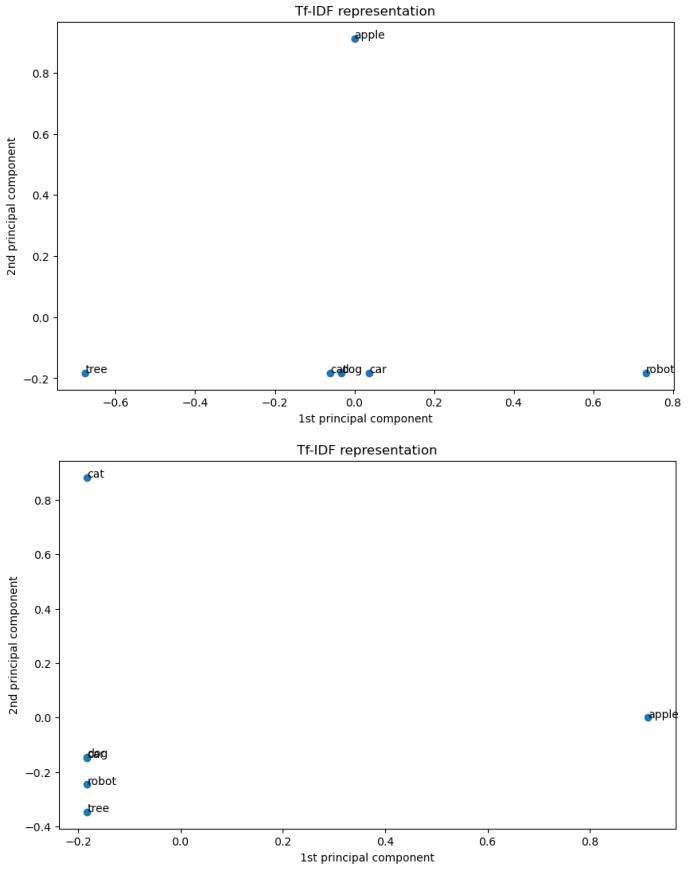
Semantic Similarity with TF-IDF

TF-IDF Representation:

- Vocabulary: {'cat', 'dog', 'apple', 'car', 'tree', 'robot'}
- $w_i : x_j = 1$ if $j = i$ else 0

Limitation - Words Distance

- In this high-dimensional space, distance between words means nothing.
- Fails to capture any semantic or contextual relationships between words.



High-dimensional space of Tf-IDF Vectorizer

Word Embeddings

Concept of Word Embeddings

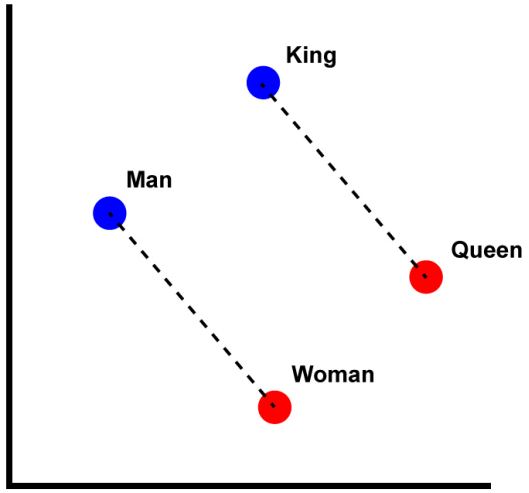
Ideal representations: Vector representations of words in a vector space where semantically similar words are mapped to nearby points.

Concept of Word Embeddings

Ideal representations: Vector representations of words in a vector space where semantically similar words are mapped to nearby points.

Key Properties:

- **Semantic Relationships**: Words with similar meanings are located in close proximity. The closer the words, the more similar.



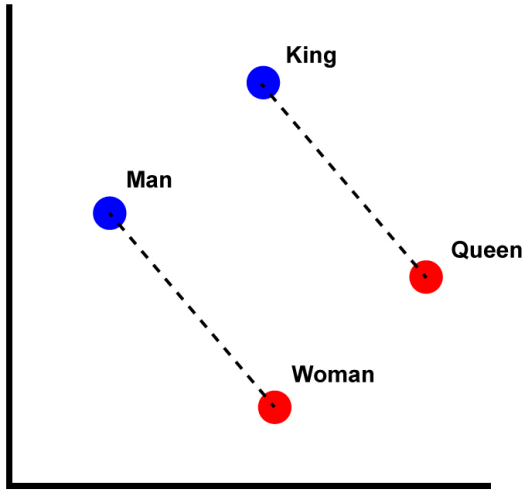
Ideal Vector Space Credits: Wikipedia

Concept of Word Embeddings

Ideal representations: Vector representations of words in a vector space where semantically similar words are mapped to nearby points.

Key Properties:

- **Semantic Relationships**: Words with similar meanings are located in close proximity. The closer the words, the more similar.
- **Syntactic Relationships**: Capturing patterns in word use based on the context, and certain algebraic "operations" to produce meaningful relationships (e.g., "king" - "man" + "woman" "queen").



Ideal Vector Space Credits: Wikipedia

Evaluations of the Embeddings

Assessing the Quality of Word Embeddings

Evaluating Word Representation Word representation effectiveness can be assessed through intrinsic and extrinsic evaluations.

Intrinsic Evaluation: Measures how well the embedding captures linguistic properties:

- **Word Similarities**: Evaluating the closeness of words in the embedding space

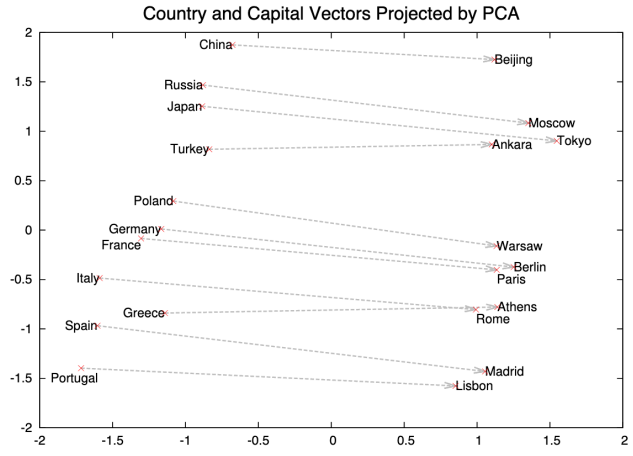


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Illustration of Word Analogies

Credit: Mikolov et al., 2013

Assessing the Quality of Word Embeddings

Evaluating Word Representation Word representation effectiveness can be assessed through intrinsic and extrinsic evaluations.

Intrinsic Evaluation: Measures how well the embedding captures linguistic properties:

- **Word Similarities**: Evaluating the closeness of words in the embedding space
- **Word Analogies**: Testing the embedding's ability to deduce relationships.

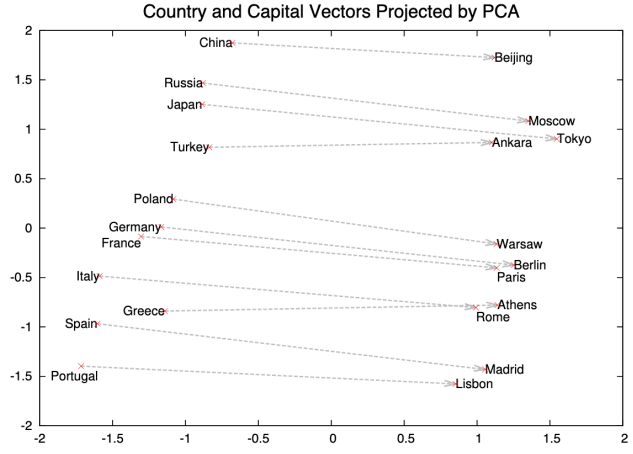


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Illustration of Word Analogies

Credit: Mikolov et al., 2013

Assessing the Quality of Word Embeddings

Evaluating Word Representation Word representation effectiveness can be assessed through intrinsic and extrinsic evaluations.

Intrinsic Evaluation: Measures how well the embedding captures linguistic properties:

- **Word Similarities**: Evaluating the closeness of words in the embedding space
- **Word Analogies**: Testing the embedding's ability to deduce relationships.
- **Synonym Detection**: Identifying words with similar meanings.

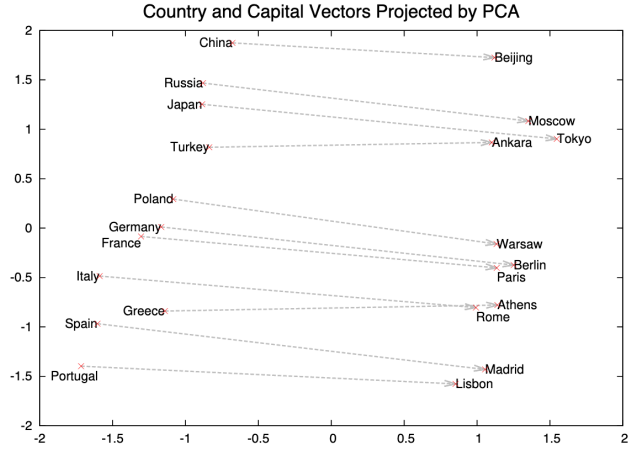


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Illustration of Word Analogies

Credit: Mikolov et al., 2013

Assessing the Quality of Word Embeddings

Evaluating Word Representation Word representation effectiveness can be assessed through intrinsic and extrinsic evaluations.

Intrinsic Evaluation: Measures how well the embedding captures linguistic properties:

- **Word Similarities**: Evaluating the closeness of words in the embedding space
- **Word Analogies**: Testing the embedding's ability to deduce relationships.
- **Synonym Detection**: Identifying words with similar meanings.
- **Word Clustering**: Grouping semantically similar words.

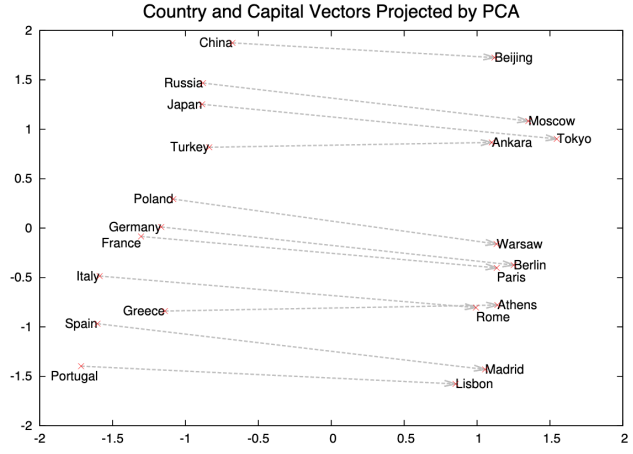


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Illustration of Word Analogies

Credit: Mikolov et al., 2013

Intrinsic Evaluation Example: Word Similarities

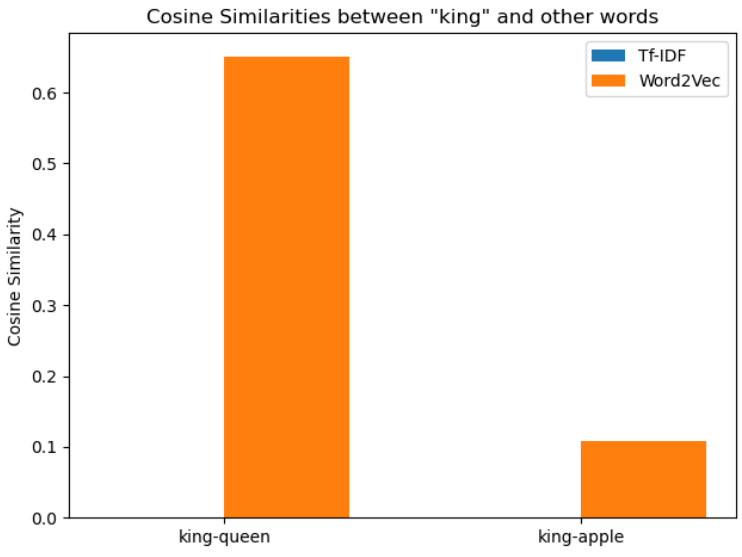
Assessing Word Similarities:

- Evaluates how well the embedding captures semantic similarities between words.
- Measures the **cosine similarity** between vectors representing different words.

Example:

- Comparing words such as 'king' and 'queen' versus 'king' and 'apple'.
- Expect 'king' and 'queen' to have a higher similarity score than 'king' and 'apple'.

Importance: Understanding word similarities allows for a nuanced understanding of the embedding space, reflecting how well the model captures semantic relationships.



Word similarities

Assessing the Quality of Word Embeddings

Extrinsic Evaluation: Assesses the performance of embeddings in downstream tasks:

- **Named Entity Recognition (NER)**: Identifying named entities in text.

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP
1	T5 Team - Google	T5		89.7	70.8	97.1	91.9/89.2	92.5/92.1	74.6/90.4
2	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5
3	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.0	69.2	97.1	93.6/91.5	92.7/92.3	74.4/90.7
4	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8	68.0	96.8	93.1/90.8	92.4/92.2	74.8/90.3
5	Facebook AI	RoBERTa		88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2
6	XLNet Team	XLNet-Large (ensemble)		88.4	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3
7	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9
8	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4
9	Stanford Hazy Research	Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9
10	XLM Systems	XLM (English only)		83.1	62.9	95.6	90.7/87.1	88.8/88.2	73.2/89.8

GLUE Benchmark

Assessing the Quality of Word Embeddings

Extrinsic Evaluation: Assesses the performance of embeddings in downstream tasks:

- **Named Entity Recognition (NER)**: Identifying named entities in text.
- **Text Classification**: Categorizing text into predefined classes.

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP
1	T5 Team - Google	T5		89.7	70.8	97.1	91.9/89.2	92.5/92.1	74.6/90.4
2	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5
3	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.0	69.2	97.1	93.6/91.5	92.7/92.3	74.4/90.7
4	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8	68.0	96.8	93.1/90.8	92.4/92.2	74.8/90.3
5	Facebook AI	RoBERTa		88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2
6	XLNet Team	XLNet-Large (ensemble)		88.4	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3
7	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9
8	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4
9	Stanford Hazy Research	Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9
10	XLM Systems	XLM (English only)		83.1	62.9	95.6	90.7/87.1	88.8/88.2	73.2/89.8

GLUE Benchmark

Assessing the Quality of Word Embeddings

Extrinsic Evaluation: Assesses the performance of embeddings in downstream tasks:

- **Named Entity Recognition (NER)**: Identifying named entities in text.
- **Text Classification**: Categorizing text into predefined classes.
- **Part-of-Speech (POS) Tagging**: Labeling words with their corresponding part of speech.

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP
1	T5 Team - Google	T5		89.7	70.8	97.1	91.9/89.2	92.5/92.1	74.6/90.4
2	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5
3	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.0	69.2	97.1	93.6/91.5	92.7/92.3	74.4/90.7
4	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8	68.0	96.8	93.1/90.8	92.4/92.2	74.8/90.3
5	Facebook AI	RoBERTa		88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2
6	XLNet Team	XLNet-Large (ensemble)		88.4	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3
7	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9
8	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4
9	Stanford Hazy Research	Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9
10	XLM Systems	XLM (English only)		83.1	62.9	95.6	90.7/87.1	88.8/88.2	73.2/89.8

GLUE Benchmark

Assessing the Quality of Word Embeddings

Extrinsic Evaluation: Assesses the performance of embeddings in downstream tasks:

- **Named Entity Recognition (NER)**: Identifying named entities in text.
- **Text Classification**: Categorizing text into predefined classes.
- **Part-of-Speech (POS) Tagging**: Labeling words with their corresponding part of speech.
- ...and other NLP tasks.

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP
1	T5 Team - Google	T5		89.7	70.8	97.1	91.9/89.2	92.5/92.1	74.6/90.4
2	ALBERT-Team Google Language	ALBERT (Ensemble)		89.4	69.1	97.1	93.4/91.2	92.5/92.0	74.2/90.5
3	王玮	ALICE v2 large ensemble (Alibaba DAMO NLP)		89.0	69.2	97.1	93.6/91.5	92.7/92.3	74.4/90.7
4	Microsoft D365 AI & UMD	FreeLB-RoBERTa (ensemble)		88.8	68.0	96.8	93.1/90.8	92.4/92.2	74.8/90.3
5	Facebook AI	RoBERTa		88.5	67.8	96.7	92.3/89.8	92.2/91.9	74.3/90.2
6	XLNet Team	XLNet-Large (ensemble)		88.4	67.8	96.8	93.0/90.7	91.6/91.1	74.2/90.3
7	Microsoft D365 AI & MSR AI	MT-DNN-ensemble		87.6	68.4	96.5	92.7/90.3	91.1/90.7	73.7/89.9
8	GLUE Human Baselines	GLUE Human Baselines		87.1	66.4	97.8	86.3/80.8	92.7/92.6	59.5/80.4
9	Stanford Hazy Research	Snorkel MeTaL		83.2	63.8	96.2	91.5/88.5	90.1/89.7	73.1/89.9
10	XLM Systems	XLM (English only)		83.1	62.9	95.6	90.7/87.1	88.8/88.2	73.2/89.8

GLUE Benchmark

Extrinsic Evaluation Example: Text Classification

Text Classification with Embeddings:

- Evaluates the effectiveness of word embeddings in text classification.
- Measures the improvement in classification accuracy when using embeddings.

Example:

- Using word embeddings as features for a sentiment analysis model.
- Comparing model performance with and without the use of embeddings.

Architecture	Accuracy [%]
4-gram [32]	39
Average LSA similarity [32]	49
Log-bilinear model [24]	54.8
RNNLMs [19]	55.4
Skip-gram	48.0
Skip-gram + RNNLMs	**58.9**

Microsfot Sentence Completion Challenge

Credit: Mikolov et al. 2013

Importance: Demonstrates the practical utility of word embeddings in real-world applications, highlighting their contribution to model performance.

One proposal

Ontologies

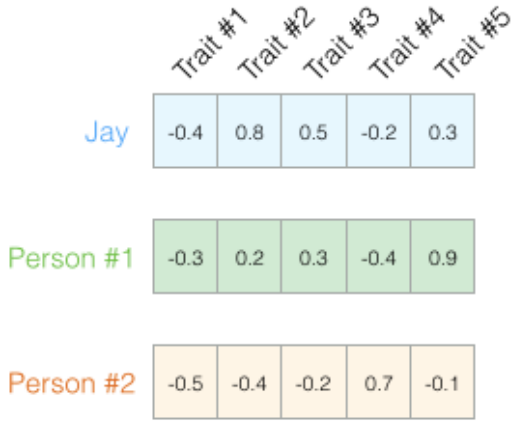
Structured framework to define and categorize entities within a domain.

- **Level**: Distinguishing between instances.
- **Class**: Grouping entities into collections or concepts.
- **Attribute**: Describing entities through characteristics.
- ...and more domain-specific categorizations.

Domain-Specific Ontologies:

- **Medicine**: ICD-9 or ICD-10 for diseases.
- **Computer Science**: Structured codes.

Limitations: Ontologies are not scalable: establishing comprehensive links between all entities can be labor-intensive and complex.



Word Representation through Ontology

Credits: Jay Alammar

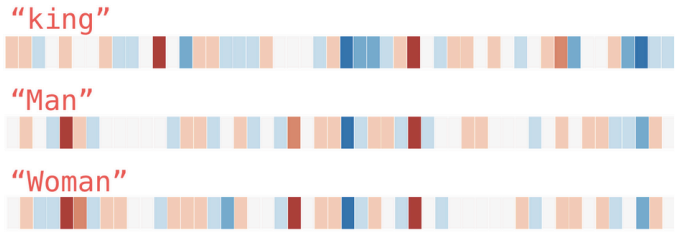
Continuous Vector representation

Advantages of Continuous Dense Embeddings

Dense embeddings provide compact, rich representations of words, capturing semantic and syntactic nuances effectively.

Key Benefits:

- **Semantic Richness**: Encapsulate meanings.



Word Representation

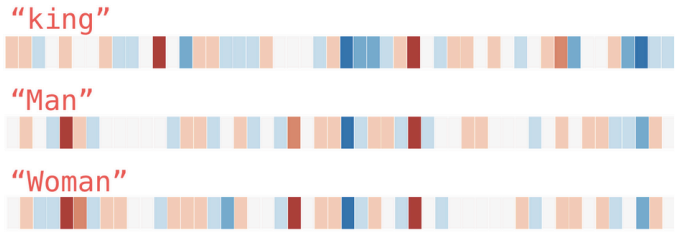
Credits: Jay Alammar

Advantages of Continuous Dense Embeddings

Dense embeddings provide compact, rich representations of words, capturing semantic and syntactic nuances effectively.

Key Benefits:

- **Semantic Richness**: Encapsulate meanings.
- **Reduced Dimensionality**: Lower-dimensional space.



Word Representation

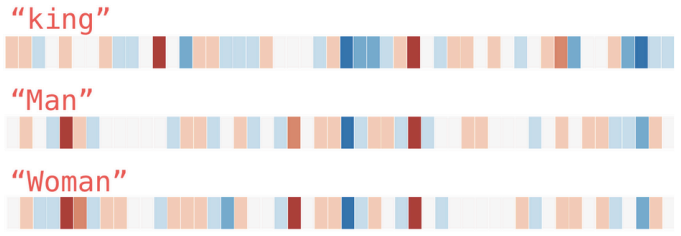
Credits: Jay Alammar

Advantages of Continuous Dense Embeddings

Dense embeddings provide compact, rich representations of words, capturing semantic and syntactic nuances effectively.

Key Benefits:

- **Semantic Richness**: Encapsulate meanings.
- **Reduced Dimensionality**: Lower-dimensional space.
- **Mathematical Operability**: Enable operations like analogy solving.



Word Representation

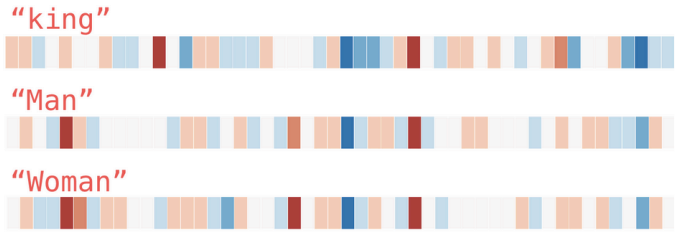
Credits: Jay Alammar

Advantages of Continuous Dense Embeddings

Dense embeddings provide compact, rich representations of words, capturing semantic and syntactic nuances effectively.

Key Benefits:

- **Semantic Richness**: Encapsulate meanings.
- **Reduced Dimensionality**: Lower-dimensional space.
- **Mathematical Operability**: Enable operations like analogy solving.
- **Enhanced Performance**: Boost NLP tasks with contextually-aware representations.



Word Representation

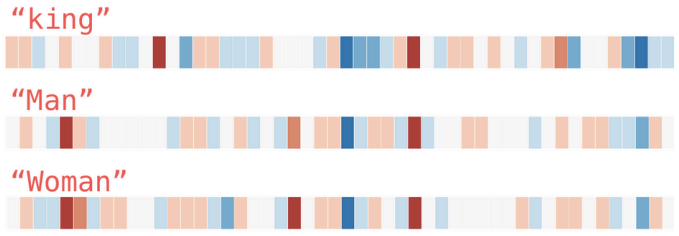
Credits: Jay Alammar

Advantages of Continuous Dense Embeddings

Dense embeddings provide compact, rich representations of words, capturing semantic and syntactic nuances effectively.

Key Benefits:

- **Semantic Richness**: Encapsulate meanings.
- **Reduced Dimensionality**: Lower-dimensional space.
- **Mathematical Operability**: Enable operations like analogy solving.
- **Enhanced Performance**: Boost NLP tasks with contextually-aware representations.



Word Representation

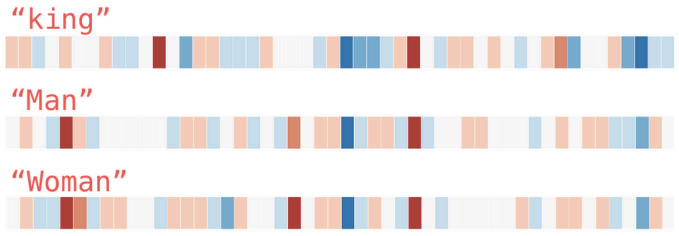
Credits: Jay Alammar

Advantages of Continuous Dense Embeddings

Dense embeddings provide compact, rich representations of words, capturing semantic and syntactic nuances effectively.

Key Benefits:

- **Semantic Richness**: Encapsulate meanings.
- **Reduced Dimensionality**: Lower-dimensional space.
- **Mathematical Operability**: Enable operations like analogy solving.
- **Enhanced Performance**: Boost NLP tasks with contextually-aware representations.



Word Representation

Credits: Jay Alammar

Conclusion: Dense embeddings represent a leap in language modeling, facilitating advanced architectures and deeper language understanding.

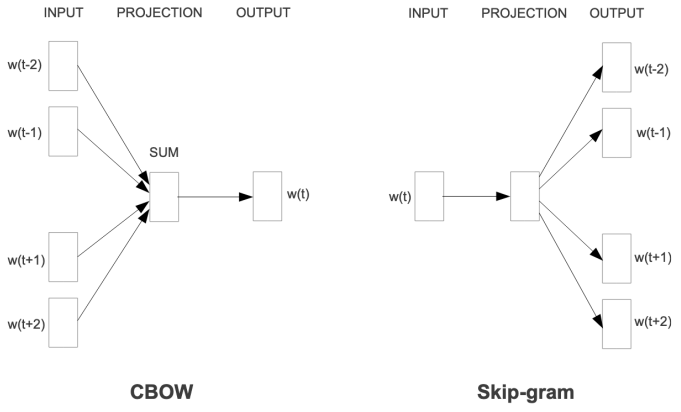
Continuous Vector Representation: Word2Vec (W2V)

Word2Vec - Mikolov et al. (2013): Word2Vec offers two architecture choices for generating dense word embeddings, inspired by language modeling:

Continuous Bag-Of-Words (CBOW):

- Represents words through n-gram context.

Continuous Skip-gram:



Illustrations of Skip-gram and CBOW Architectures Credit: Mikolov et al. (2013)

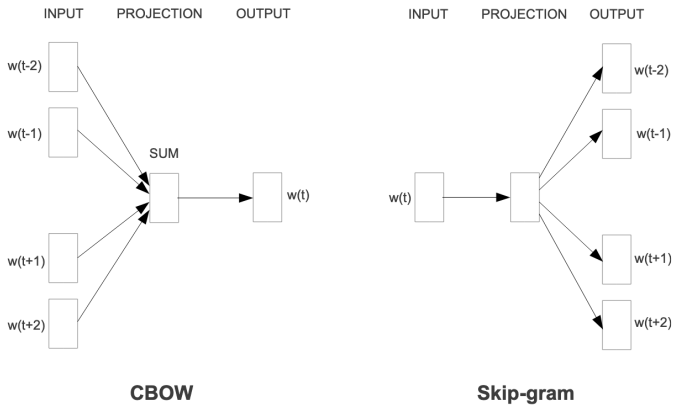
Continuous Vector Representation: Word2Vec (W2V)

Word2Vec - Mikolov et al. (2013): Word2Vec offers two architecture choices for generating dense word embeddings, inspired by language modeling:

Continuous Bag-Of-Words (CBOW):

- Represents words through n-gram context.
- Uses projection matrices (embedding layers) to capture word features.

Continuous Skip-gram:



Illustrations of Skip-gram and CBOW Architectures Credit: Mikolov et al. (2013)

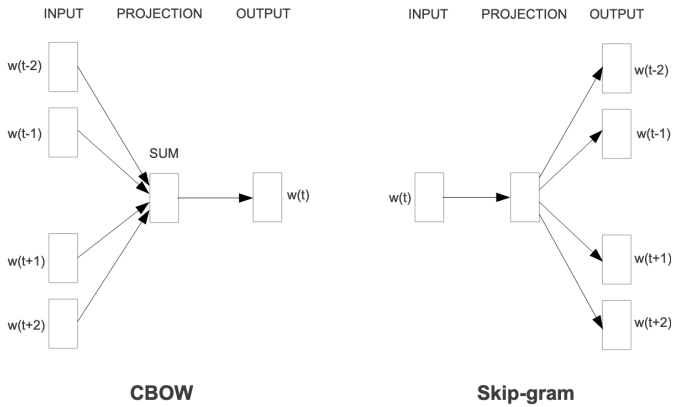
Continuous Vector Representation: Word2Vec (W2V)

Word2Vec - Mikolov et al. (2013): Word2Vec offers two architecture choices for generating dense word embeddings, inspired by language modeling:

Continuous Bag-Of-Words (CBOW):

- Represents words through n-gram context.
- Uses projection matrices (embedding layers) to capture word features.
- Aims to predict a target word based on surrounding context words.

Continuous Skip-gram:



Illustrations of Skip-gram and CBOW Architectures Credit: Mikolov et al. (2013)

Continuous Vector Representation: Word2Vec (W2V)

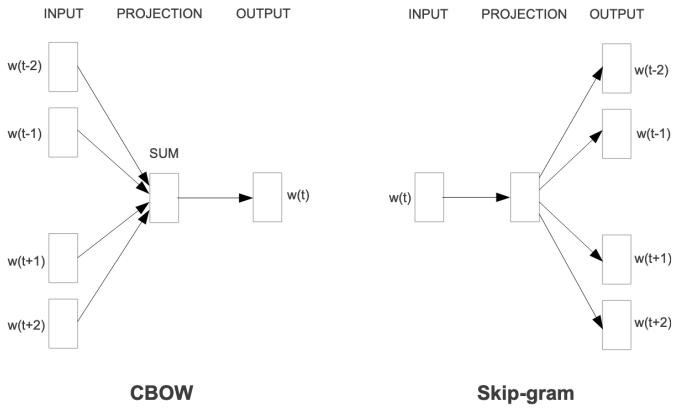
Word2Vec - Mikolov et al. (2013): Word2Vec offers two architecture choices for generating dense word embeddings, inspired by language modeling:

Continuous Bag-Of-Words (CBOW):

- Represents words through n-gram context.
- Uses projection matrices (embedding layers) to capture word features.
- Aims to predict a target word based on surrounding context words.

Continuous Skip-gram:

- Also uses n-gram representation and projection matrices.



Illustrations of Skip-gram and CBOW Architectures Credit: Mikolov et al. (2013)

Continuous Vector Representation: Word2Vec (W2V)

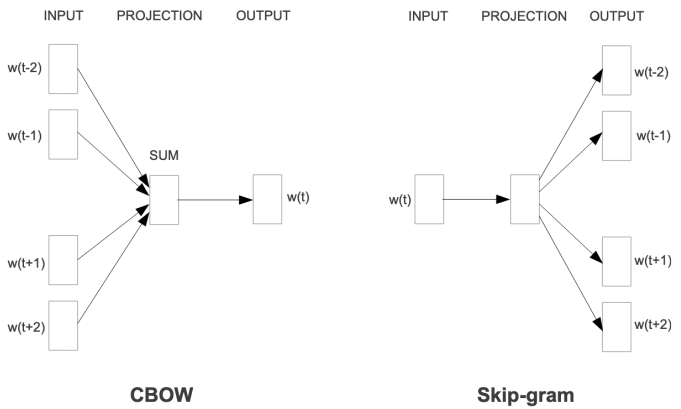
Word2Vec - Mikolov et al. (2013): Word2Vec offers two architecture choices for generating dense word embeddings, inspired by language modeling:

Continuous Bag-Of-Words (CBOW):

- Represents words through n-gram context.
- Uses projection matrices (embedding layers) to capture word features.
- Aims to predict a target word based on surrounding context words.

Continuous Skip-gram:

- Also uses n-gram representation and projection matrices.
- In contrast to CBOW, predicts surrounding context words from a target word, offering quality embeddings for even infrequent words.



Illustrations of Skip-gram and CBOW Architectures Credit: Mikolov et al. (2013)

Training the Skip-Gram Model: Data Collection

Objective of Skip-Gram: The Skip-Gram model aims to predict context words given a target word, strengthening the word associations within a specified window size.

Training the Skip-Gram Model: Data Collection

Objective of Skip-Gram: The Skip-Gram model aims to predict context words given a target word, strengthening the word associations within a specified window size.

Training Data Preparation:

- **Features (Input)**: Target words.
- **Labels (Output)**: Context words within a defined window size around the target word.

Training the Skip-Gram Model: Data Collection

Objective of Skip-Gram: The Skip-Gram model aims to predict context words given a target word, strengthening the word associations within a specified window size.

Training Data Preparation:

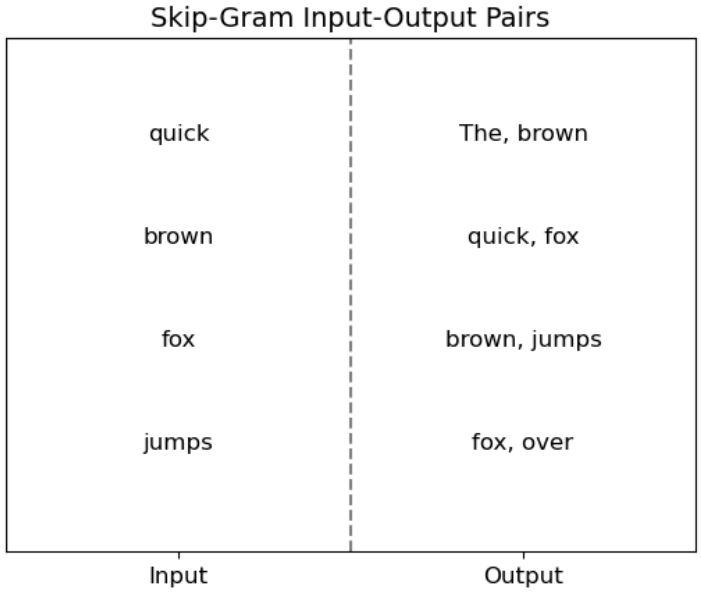
- **Features (Input)**: Target words.
- **Labels (Output)**: Context words within a defined window size around the target word.

We don't need labelled data: we can collect texts from Wikipedia, books, internet... and create the training set from it !

Training the Skip-Gram Model: Data Preparation

Example: Given the sentence "The quick brown fox jumps over", with a window size of 1, the training pairs are:

- Input: "quick", Output: ["The", "brown"]
- Input: "brown", Output: ["quick", "fox"]
- ... and so on.



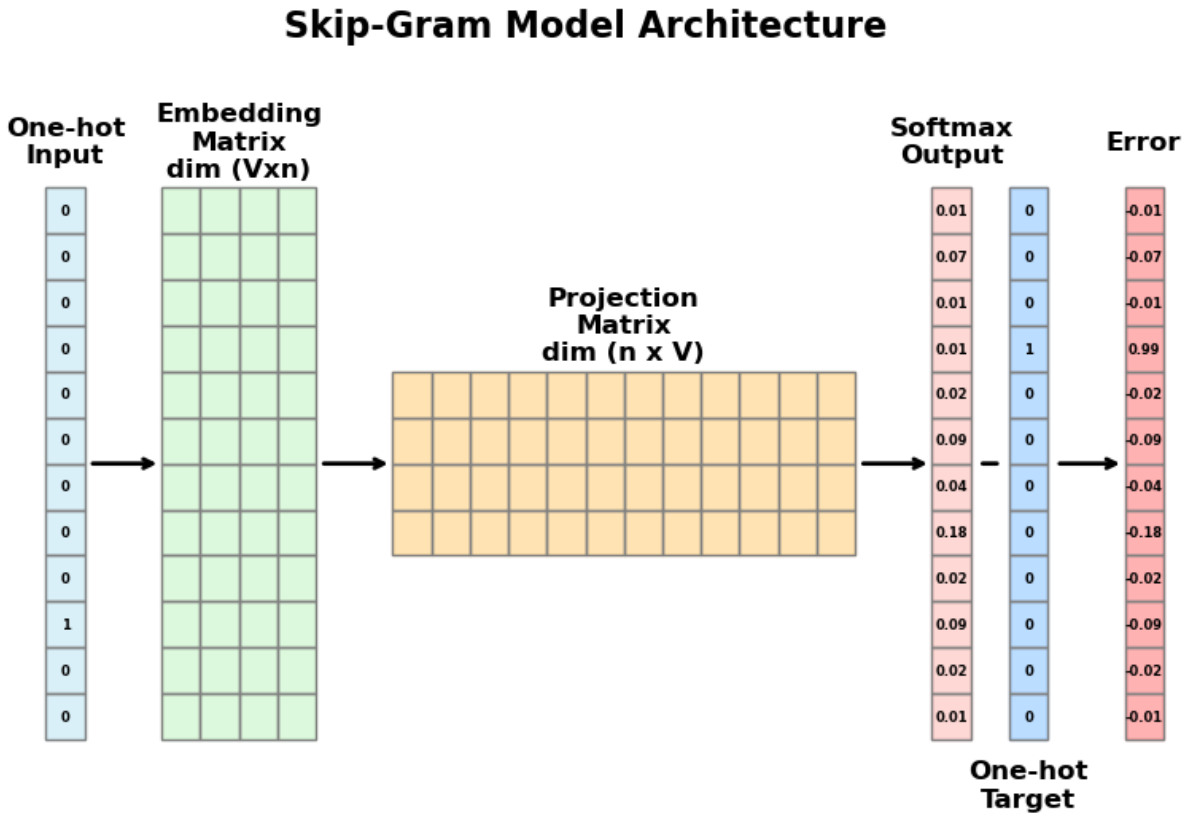
Training pairs generation in Skip-Gram Model

Training Skip-Gram Model: Model Architecture (1/2)

Model Architecture:

- **Input Layer**: One-hot encoded vectors of the target words.
- **Embedding layer**: A fully connected layer of dimension |Vocabulary| x Embedding size n
- **Hidden Layer**: A fully connected layer without activation, to project to the output vocabulary.
- **Output Layer**: Predicts the probability distribution (softmax) of context words for the given input.

Training Skip-Gram Model: Model Architecture (2/2)



Skip-Gram Model Architecture; predict context from one word

Main Limitation: each time we're computing softmax for V words (could be 10e6!) the process is computationally expensive..

Negative Sampling in Skip-Gram Model

Challenge in Training: The standard Skip-Gram model with softmax can be computationally expensive due to the large vocabulary size.

Negative Sampling in Skip-Gram Model

Challenge in Training: The standard Skip-Gram model with softmax can be computationally expensive due to the large vocabulary size.

Solution - Negative Sampling:

- **Concept**: Instead of predicting the probability for all words in the vocabulary, negative sampling trains the model to distinguish a target word from a small set of random 'noise words'.

Negative Sampling in Skip-Gram Model

Challenge in Training: The standard Skip-Gram model with softmax can be computationally expensive due to the large vocabulary size.
Solution - Negative Sampling:

- **Concept**: Instead of predicting the probability for all words in the vocabulary, negative sampling trains the model to distinguish a target word from a small set of random 'noise words'.
- **Benefits**: Simplifies the computation and accelerates the training process, especially for large corpora.

Negative Sampling in Skip-Gram Model

Challenge in Training: The standard Skip-Gram model with softmax can be computationally expensive due to the large vocabulary size.
Solution - Negative Sampling:

- **Concept**: Instead of predicting the probability for all words in the vocabulary, negative sampling trains the model to distinguish a target word from a small set of random 'noise words'.
- **Benefits**: Simplifies the computation and accelerates the training process, especially for large corpora.

Negative Sampling in Skip-Gram Model

Challenge in Training: The standard Skip-Gram model with softmax can be computationally expensive due to the large vocabulary size.
Solution - Negative Sampling:

- **Concept**: Instead of predicting the probability for all words in the vocabulary, negative sampling trains the model to distinguish a target word from a small set of random 'noise words'.
- **Benefits**: Simplifies the computation and accelerates the training process, especially for large corpora.

Implementation: In each training step:

- Select a small number of negative samples (words not in the context).

Negative Sampling in Skip-Gram Model

Challenge in Training: The standard Skip-Gram model with softmax can be computationally expensive due to the large vocabulary size.
Solution - Negative Sampling:

- **Concept**: Instead of predicting the probability for all words in the vocabulary, negative sampling trains the model to distinguish a target word from a small set of random 'noise words'.
- **Benefits**: Simplifies the computation and accelerates the training process, especially for large corpora.

Implementation: In each training step:

- Select a small number of negative samples (words not in the context).
- Update weights based on the target word and the sampled negative words.

Training Skip-Gram Model with NS: Data Preparation

Example: Given the sentence "The quick brown fox jumps over", with a window size of 1, the training pairs are:

- Input: ["quick", "brown"] Output: 1
- Input: ["quick", "yellow"] Output 0
- Input: ["brown", "fox"] Output: 1
- Input: ["brown", "fly"] Output: 0
- ... and so on.

Skip-Gram Input-Output Pairs

Input	Output	Value
quick	The	1
quick	brown	1
quick	bow	0
quick	yellow	0
brown	quick	1
brown	fox	1
brown	wolf	0
brown	maths	0
fox	brown	1
fox	jumps	1
fox	bark	0
fox	fly	0
jumps	fox	1
jumps	over	1
jumps	dig	0
jumps	special	0

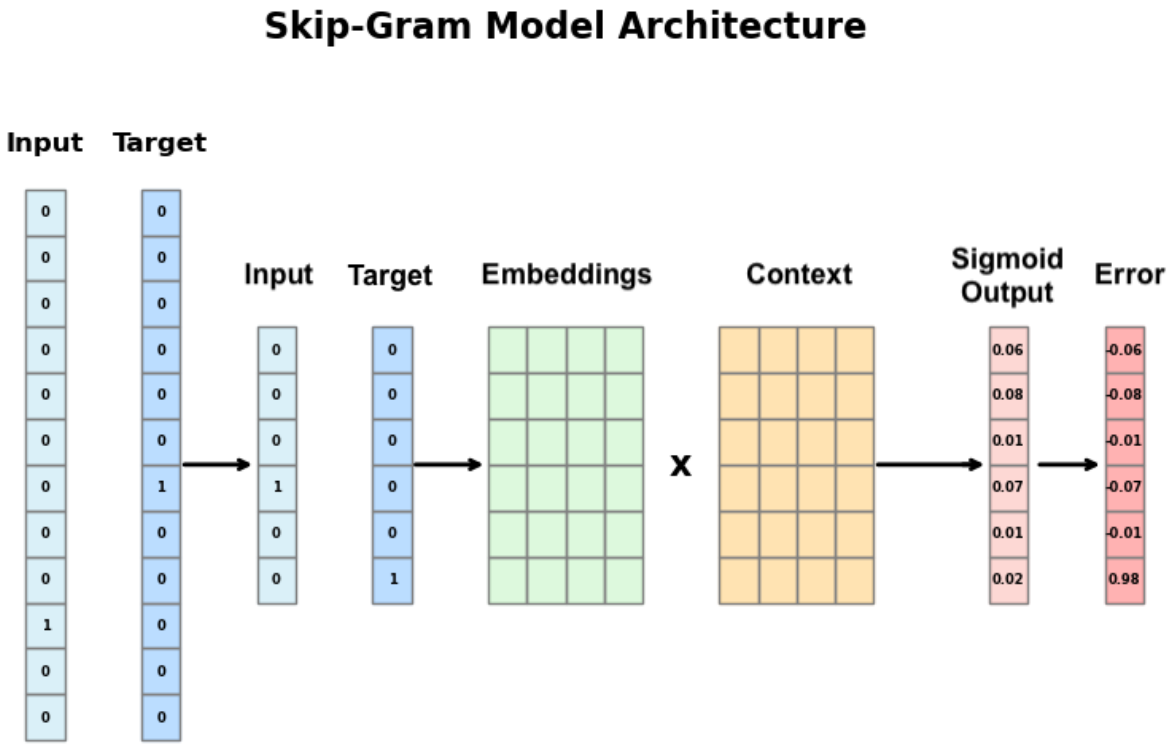
Training pairs generation in Skip-Gram Model with Negative Sampling

Training Skip-Gram Model with NS: Model Architecture (1/2)

Model Architecture:

- **Input Layer**: One-hot encoded vectors of the target words. One-hot Encoded vectors of the context words.
- **Embedding layer**: A fully connected layer of dimension |Vocabulary| x Embedding size n
- **Context Layer**: A fully connected layer of dimension |Vocabulary| x Embedding size n
- **Output Layer**: Computes the dot product between the target and context embeddings, then applies a **sigmoid** to output the probability that the pair co-occurs (binary classification, one sigmoid per pair).

Training Skip-Gram Model with NS: Model Architecture (2/2)



Skip-Gram Model Architecture; predict context from one word

Main Difference: instead of one softmax over the full vocabulary V, we compute $k+1$ independent sigmoids (one for the true pair, k for the negative samples) — no normalization across the vocabulary.

Word2Vec Application: Analyzing Word Pair Relationships

- **Cosine Similarity**: Measures the cosine of the angle between two word vectors, indicating how similar they are in the embedding space.
- Word pair relationship analysis has practical uses in fields like semantic search, automated text analysis, and even in creative domains like generating novel content based on identified patterns.
- A deeper understanding of word relationships can enhance language models, making them more robust and contextually aware.

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Word Pairs Relationships

Credit: Mikolov et al. (2013)

Introduction to GloVe and FastText

GloVe - Global Vectors for Word Representation:

- **Co-occurrence Matrix**: GloVe is built on word-word co-occurrence statistics from a corpus, capturing global statistical information.
- **Applications**: Widely used in applications requiring an understanding of word similarity and analogy based on global corpus statistics.

FastText - Advanced Word Representation:

- **Subword Information**: Extends the Word2Vec model by representing each word as a bag of character n-grams, capturing morphological information.
- **Handling Rare Words**: Particularly effective in understanding and representing rare words or misspellings.
- **Applications**: Useful in tasks where morphological information is crucial, like language modeling and text classification in morphologically rich languages.

Limitations of Static Word Embeddings

Key Limitations:

- **Word Sense Ambiguity**: only one vector per word, ignoring the polysemy where words have multiple meanings based on context.

Limitations of Static Word Embeddings

Key Limitations:

- **Word Sense Ambiguity**: only one vector per word, ignoring the polysemy where words have multiple meanings based on context.
- **Context Ignorance**: Unable to capture the meaning of a word in different contexts. The same word in different sentences will have the same representation.

Limitations of Static Word Embeddings

Key Limitations:

- **Word Sense Ambiguity**: only one vector per word, ignoring the polysemy where words have multiple meanings based on context.
- **Context Ignorance**: Unable to capture the meaning of a word in different contexts. The same word in different sentences will have the same representation.
- **Out-of-Vocabulary (OOV) Words**: Challenges in handling words not present in the training corpus. FastText addresses this partially with subword information.

Limitations of Static Word Embeddings

Key Limitations:

- **Word Sense Ambiguity**: only one vector per word, ignoring the polysemy where words have multiple meanings based on context.
- **Context Ignorance**: Unable to capture the meaning of a word in different contexts. The same word in different sentences will have the same representation.
- **Out-of-Vocabulary (OOV) Words**: Challenges in handling words not present in the training corpus. FastText addresses this partially with subword information.
- **Fixed Representations**: Fixed after training, not allowing the model to adapt to evolving language use or domain-specific jargon.

Limitations of Static Word Embeddings

Key Limitations:

- **Word Sense Ambiguity**: only one vector per word, ignoring the polysemy where words have multiple meanings based on context.
- **Context Ignorance**: Unable to capture the meaning of a word in different contexts. The same word in different sentences will have the same representation.
- **Out-of-Vocabulary (OOV) Words**: Challenges in handling words not present in the training corpus. FastText addresses this partially with subword information.
- **Fixed Representations**: Fixed after training, not allowing the model to adapt to evolving language use or domain-specific jargon.
- **Resource Intensive**: Requires substantial computational resources and time to train on large corpora, making it less feasible for resource-constrained scenarios.

Contextual Embeddings

Introduction to Contextual Embeddings

Overcoming the Limitations: Contextual embeddings represent the next evolution in word representations, addressing the inherent constraints of static word embeddings.

What are Contextual Embeddings?

- **Dynamic Word Representations**: Unlike static embeddings, contextual embeddings provide representations that change based on the word's context.
- **Deep Contextualization**: These models consider the entire sentence or even larger contexts to understand the meaning of each word.

Introduction to Contextual Embeddings

Pioneering Models:

- **ELMo (Embeddings from Language Models), Peters et al. (2018)**: Utilizes bidirectional LSTM trained on a specific task to generate embeddings.
- **BERT (Bidirectional Encoder Representations from Transformers), Delvin et al. (2019)**: Transforms the landscape with a transformer-based model, pre-trained on vast amounts of text and fine-tuned for specific tasks.
- **GPT-2 (Generative Pre-trained Transformer), Radford et al. (2019)**: Emphasizes generative capabilities and large-scale pre-training for versatile language understanding.

Introduction to Contextual Embeddings

Advantages Over Static Embeddings:

- Captures polysemy by providing context-specific word meanings.
- Adapts to different domains and evolving language use without retraining from scratch.
- Enhances performance across a wide array of NLP tasks.

Conclusion: Contextual embeddings mark a significant milestone in NLP, offering nuanced and adaptable understanding of language, far surpassing the capabilities of static embeddings.

Introduction to ELMo (Embeddings from Language Models)

What is ELMo?

- ELMo, Peters et al. (2018) was developed by Allen Institute for AI.
- It stands for Embeddings from Language Models.

Key Features of ELMo:

- **Deep Contextualization**: ELMo considers the entire context of a word by using the internal states of a bidirectional LSTM.
- **Dynamic Word Representation**: Each word's representation is a function of the entire sentence.

Architecture Overview: ELMo combines the representations of a two-layer pretrained bidirectional LSTM internal states into downstream tasks to outperfom current models.

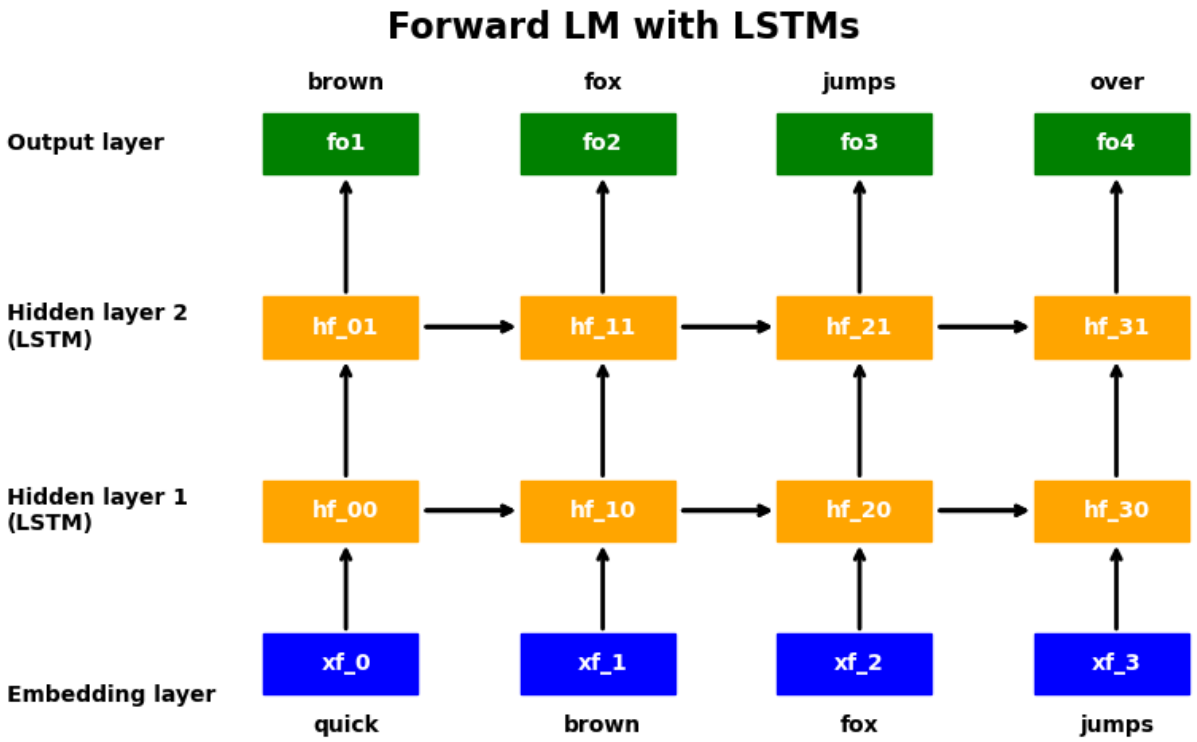
Pre-training Steps of ELMo

Pre-training Procedure:

- **Bidirectional Language Modeling**:
  - *Forward LSTM*: Predicts the next word from the past context.
  - *Backward LSTM*: Predicts the previous word from the future context.
- **Character Embeddings**: Captures word morphology and manages out-of-vocabulary words.
- **Training on Large Corpus**: Enhances understanding of language structure by predicting surrounding words.

Objective: ELMo's pre-training on a large corpus with bidirectional context and character embeddings lays a robust foundation for nuanced language understanding.

Forward LM LSTM training

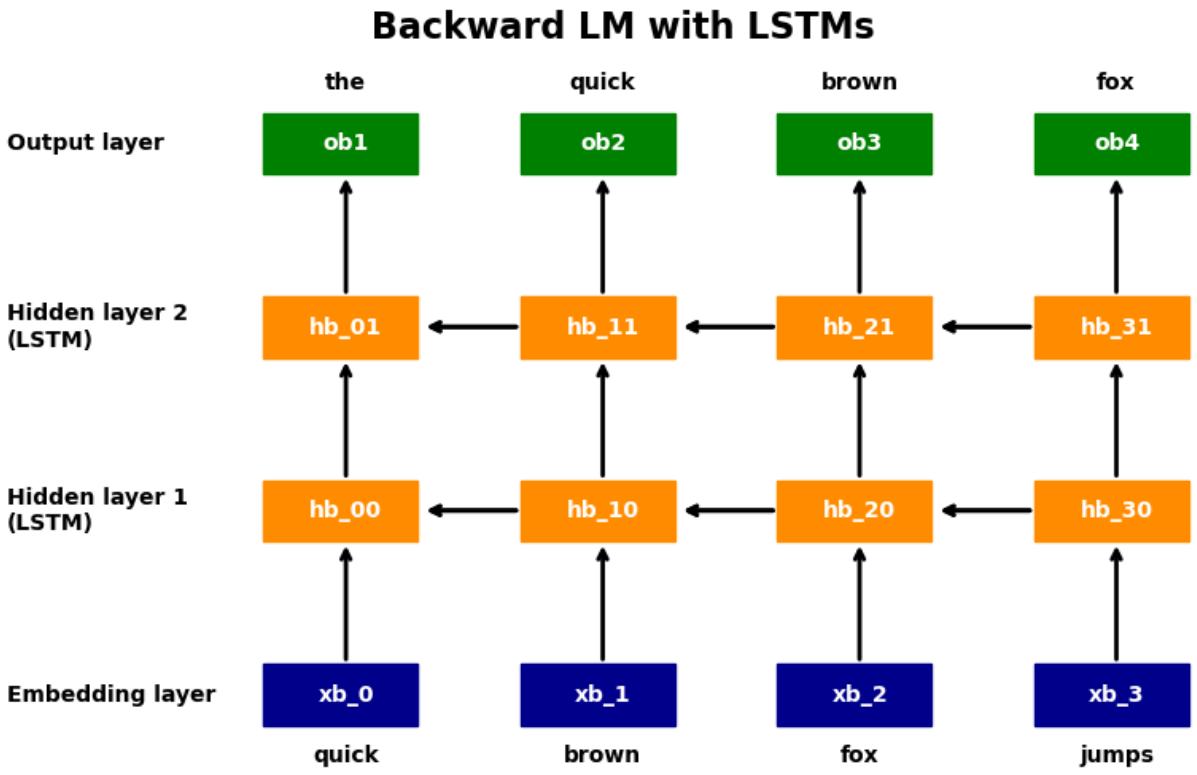


Forward LSTM

Architecture:

- One embedding layer.
- Two hidden layers.
- One softmax layer to predict next token.

Backward LM LSTM training

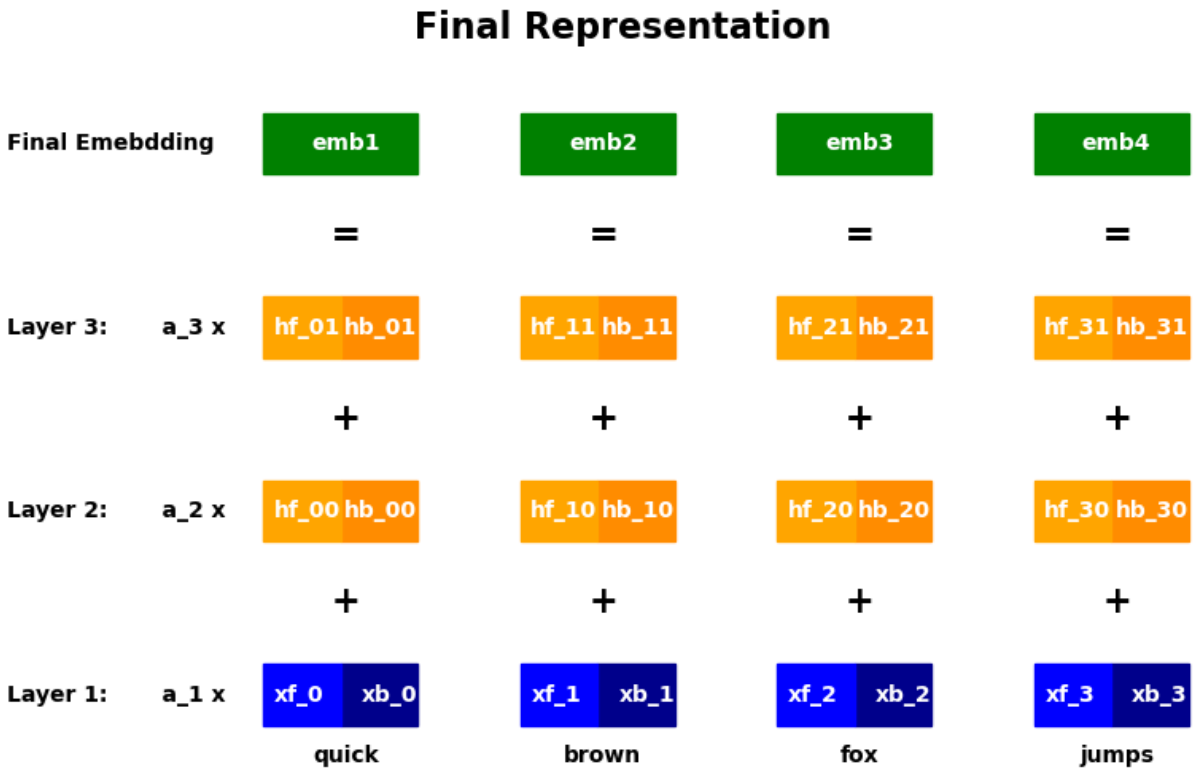


Backward LSTM

Architecture:

- One embedding layer.
- Two hidden layers.
- One softmax layer to predict previous token.

Final Embeddings



Final Embeddings

Architecture:

- We concatenate each layer together.
- We weight each of the layer to get the final embedding.
- The weights depend of the downstream task.

Utilizing Pre-trained LSTM in Downstream Tasks

Integration into Downstream Tasks:

- **Feature Extraction**: Use the output of the pre-trained LSTM layers as features in a new model tailored to a specific task, such as sentiment analysis or named entity recognition.

Utilizing Pre-trained LSTM in Downstream Tasks

Integration into Downstream Tasks:

- **Feature Extraction**: Use the output of the pre-trained LSTM layers as features in a new model tailored to a specific task, such as sentiment analysis or named entity recognition.
- **Fine-tuning**: Adjust the pre-trained LSTM weights slightly during the training of the downstream task to better adapt the model to the specific requirements of the task.

Utilizing Pre-trained LSTM in Downstream Tasks

Integration into Downstream Tasks:

- **Feature Extraction**: Use the output of the pre-trained LSTM layers as features in a new model tailored to a specific task, such as sentiment analysis or named entity recognition.
- **Fine-tuning**: Adjust the pre-trained LSTM weights slightly during the training of the downstream task to better adapt the model to the specific requirements of the task.

Utilizing Pre-trained LSTM in Downstream Tasks

Integration into Downstream Tasks:

- **Feature Extraction**: Use the output of the pre-trained LSTM layers as features in a new model tailored to a specific task, such as sentiment analysis or named entity recognition.
- **Fine-tuning**: Adjust the pre-trained LSTM weights slightly during the training of the downstream task to better adapt the model to the specific requirements of the task.

Downstream Task Architecture:

- Start with the pre-trained ELMo embeddings as the input layer.

Utilizing Pre-trained LSTM in Downstream Tasks

Integration into Downstream Tasks:

- **Feature Extraction**: Use the output of the pre-trained LSTM layers as features in a new model tailored to a specific task, such as sentiment analysis or named entity recognition.
- **Fine-tuning**: Adjust the pre-trained LSTM weights slightly during the training of the downstream task to better adapt the model to the specific requirements of the task.

Downstream Task Architecture:

- Start with the pre-trained ELMo embeddings as the input layer.
- Add task-specific layers on top of the ELMo layers (e.g., additional LSTM layers, dense layers...).

Utilizing Pre-trained LSTM in Downstream Tasks

Integration into Downstream Tasks:

- **Feature Extraction**: Use the output of the pre-trained LSTM layers as features in a new model tailored to a specific task, such as sentiment analysis or named entity recognition.
- **Fine-tuning**: Adjust the pre-trained LSTM weights slightly during the training of the downstream task to better adapt the model to the specific requirements of the task.

Downstream Task Architecture:

- Start with the pre-trained ELMo embeddings as the input layer.
- Add task-specific layers on top of the ELMo layers (e.g., additional LSTM layers, dense layers...).
- The final output layer is designed according to the downstream task (e.g., softmax for classification).

Example: ELMo in Sentiment Analysis

ELMo embeddings can enhance the sentiment analysis models by providing deep contextualized word representations.

Incorporation into Model (example):

- **Input Layer**: Start with ELMo embeddings for each token in the input text.

Example: ELMo in Sentiment Analysis

ELMo embeddings can enhance the sentiment analysis models by providing deep contextualized word representations.

Incorporation into Model (example):

- **Input Layer**: Start with ELMo embeddings for each token in the input text.
- **Additional Layers**: Add a bi-directional LSTM layer to further capture context not encapsulated by ELMo.

Example: ELMo in Sentiment Analysis

ELMo embeddings can enhance the sentiment analysis models by providing deep contextualized word representations.

Incorporation into Model (example):

- **Input Layer**: Start with ELMo embeddings for each token in the input text.
- **Additional Layers**: Add a bi-directional LSTM layer to further capture context not encapsulated by ELMo.
- **Output Layer**: A dense layer with softmax activation to classify the sentiment as positive or negative.

Example: ELMo in Sentiment Analysis

ELMo embeddings can enhance the sentiment analysis models by providing deep contextualized word representations.

Incorporation into Model (example):

- **Input Layer**: Start with ELMo embeddings for each token in the input text.
- **Additional Layers**: Add a bi-directional LSTM layer to further capture context not encapsulated by ELMo.
- **Output Layer**: A dense layer with softmax activation to classify the sentiment as positive or negative.
- **Train your model**: The whole model with ELMo frozen or not !

Example: ELMo in Sentiment Analysis

ELMo embeddings can enhance the sentiment analysis models by providing deep contextualized word representations.

Incorporation into Model (example):

- **Input Layer**: Start with ELMo embeddings for each token in the input text.
- **Additional Layers**: Add a bi-directional LSTM layer to further capture context not encapsulated by ELMo.
- **Output Layer**: A dense layer with softmax activation to classify the sentiment as positive or negative.
- **Train your model**: The whole model with ELMo frozen or not !

Example: ELMo in Sentiment Analysis

ELMo embeddings can enhance the sentiment analysis models by providing deep contextualized word representations.

Incorporation into Model (example):

- **Input Layer**: Start with ELMo embeddings for each token in the input text.
- **Additional Layers**: Add a bi-directional LSTM layer to further capture context not encapsulated by ELMo.
- **Output Layer**: A dense layer with softmax activation to classify the sentiment as positive or negative.
- **Train your model**: The whole model with ELMo frozen or not !

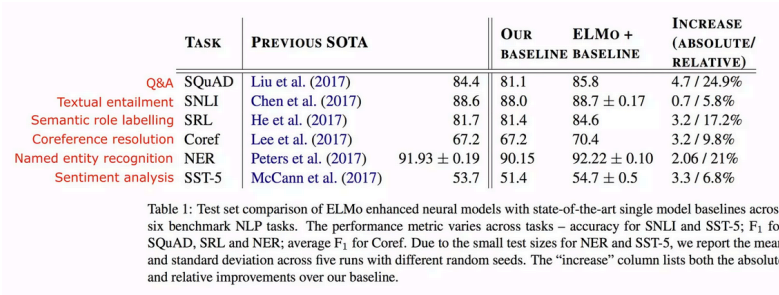
Advantage: The use of ELMo *transfers* the nuances of language, leading to a more accurate sentiment classification compared to models without contextualized embeddings: **3.3% in absolute improvement**.

ELMo's Impact on NLP Tasks

ELMo's introduction marked a new era in NLP by setting state-of-the-art (SOA) benchmarks across multiple tasks simultaneously.

Remarkable Achievements:

- **SOA in Six Benchmarks**: ELMo established new records in a range of NLP tasks.
- **Double-Digit Improvements**: Notably increased performance by over 10% in four of those tasks.



	Task	Previous SOTA		Our baseline	ELMo + baseline	Increase (absolute/ relative)
Q&A	SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
Textual entailment	SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
Semantic role labelling	SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coreference resolution	Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
Named entity recognition	NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
Sentiment analysis	SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines acros six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 fo SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mea and standard deviation across five runs with different random seeds. The "increase" column lists both the absolut and relative improvements over our baseline.

Visualization of ELMo generating embeddings

Credit: Pieters

Transfer Learning with ELMo:

- **Knowledge Acquisition**: Gained from pre-training on extensive datasets.
- **Knowledge Transfer**: Applied to enhance task-specific models, demonstrating the power of transfer learning in NLP.

Conclusions on ELMO

Benefits of Using ELMo's LSTM:

- **Contextualized Understanding**: Models become more aware of the context within sentences, leading to better understanding and predictions.
- **Transfer Learning**: Leveraging pre-trained models significantly reduces the need for large labeled datasets for the target task.
- **State-of-the-Art Results**: Many tasks see improved performance metrics when integrating pre-trained LSTMs from ELMo.

Related Work in Contextual Embeddings

Several works have built upon and extended the concepts introduced by ELMo, pushing the boundaries in various NLP tasks.

- **ULM-Fit** (Howard and Ruder, 2018): A model that employs transfer learning specifically for text classification, setting new benchmarks on six different tasks.
- **Semi-supervised Sequence Tagging** (Peters et al., 2017): Improved named entity recognition (NER) performance on multiple datasets by using bidirectional language models.
- **Learned in Translation: Contextualized Word Vectors** (McCann et al., 2017): Achieved state-of-the-art results in translation tasks by leveraging contextualized word vectors derived from an LSTM model with an attention mechanism.

These works collectively demonstrate the growing impact of transfer learning and contextual embeddings across a range of NLP applications.

Limitations of those Advanced NLP Models

Challenges in Cutting-Edge NLP: Models like ELMo and ULM-Fit push the boundaries but encounter challenges:

- **Data and Resources**: Dependence on extensive datasets and substantial computational power limits accessibility and applicability, especially for under-resourced languages and domains.
- **Model Complexity**: The intricate nature of these models can obscure interpretability, leading to the 'black-box' issue and challenges in model trust and fine-tuning.
- **Real-World Application**: Mastery over benchmarks doesn't always translate to real-world scenarios, where data can be noisy and unpredictable.

Conclusion: The future of NLP lies in overcoming these hurdles, striving for models that balance performance with efficiency, interpretability, and broader applicability.

QA

Open Discussion

- Feel free to ask questions or share your thoughts about today's topics.
- Any insights, experiences, or perspectives you'd like to discuss are welcome.

Summary of Key Takeaways

- We explored **static embeddings** like Word2Vec, GloVe, and FastText, and their role in establishing the foundation for current NLP advancements.
- Static embeddings, while transformative, have **limitations** in handling polysemy, dynamic context, and out-of-vocabulary words.
- Advanced NLP models like ELMo and ULM-Fit build upon static embeddings, offering **context-aware representations** that better capture the nuances of language.
- These advanced models **set new benchmarks** but face challenges regarding data dependency, computational demands, and generalization.
- Addressing the limitations of both static and advanced models is crucial for the development of more efficient, **interpretable, and generalizable NLP solutions**.