NLP and Word Embeddings

FPT Education

FPT UNIVERSITY

Learning Objectives:

- Explain how word embeddings capture relationships between words
- Load pre-trained word vectors
- Measure similarity between word vectors using cosine similarity
- Use word embeddings to solve word analogy problems such as Man is to Woman as King is to ______.
- Reduce bias in word embeddings
- Create an embedding layer in Keras with pre-trained word vectors

NLP and Word Embeddings

Learning Objectives:

- Describe how negative sampling learns word vectors more efficiently than other methods
- Explain the advantages and disadvantages of the GloVe algorithm
- Build a sentiment classifier using word embeddings
- Build and train a more sophisticated classifier using an LSTM

NLP and Word Embeddings

FPT Education

FPT UNIVERSITY

NLP and Word Embeddings

Word representation

Word representation

- In contrast to traditional one-hot vector representation, word embeddings are a technique to represent words in a higher dimensional space where each word is represented by a set of features and values.
- While one-hot vectors treat each word separately and make it difficult for an algorithm to generalize across words, word embeddings allow an algorithm to recognize that words with similar meanings or connotations have similar feature values, even if they are not the same word.

Word representation

V = [a, aaron, ..., zulu, <UNK>]

1-hot representation

$$\begin{array}{cccccc}
\text{Man} & \text{Woman} & \text{King} & \text{Queen} & \text{Apple} & \text{Orange} \\
(5391) & (9853) & (4914) & (456) & (7157) & (6257) \\
\begin{bmatrix}0\\0\\0\\0\\\vdots\\1\\\vdots\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\\vdots\\1\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\\vdots\\1\\\vdots\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\\vdots\\1\\\vdots\\0\end{bmatrix} &
\begin{bmatrix}0\\\vdots\\1\\\vdots\\0\\0\\0\\0\\0\end{bmatrix} &
\begin{bmatrix}0\\0\\0\\0\\0\\\vdots\\1\\\vdots\\0\end{bmatrix}
\end{array}$$

I want a glass of orange ______

I want a glass of an apple ____

Featurized representation: word embedding

- Consider an example of how word embeddings could be used to recognize that "orange juice" is a popular phrase and to generalize to other fruit juices, such as apple juice.
- With word embeddings, words like man and woman or king and queen would be represented as vectors that are closer to each other in the feature space, while words like apple and orange would be represented as vectors that are also closer to each other.

Featurized representation: word embedding

Analogies

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

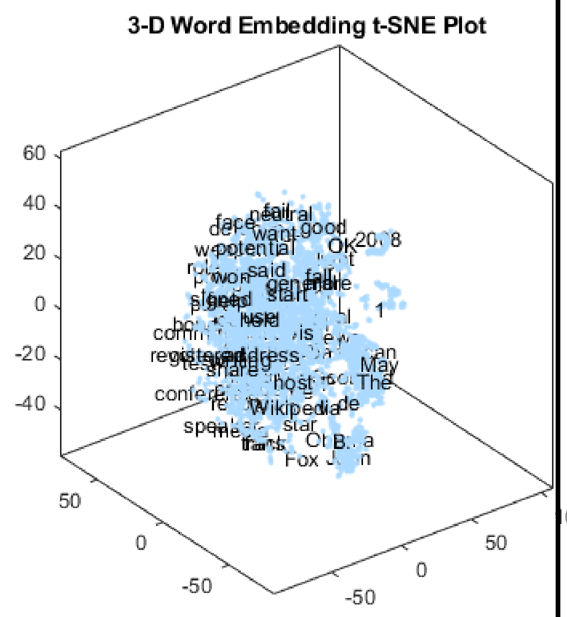
I want a glass of orange juice

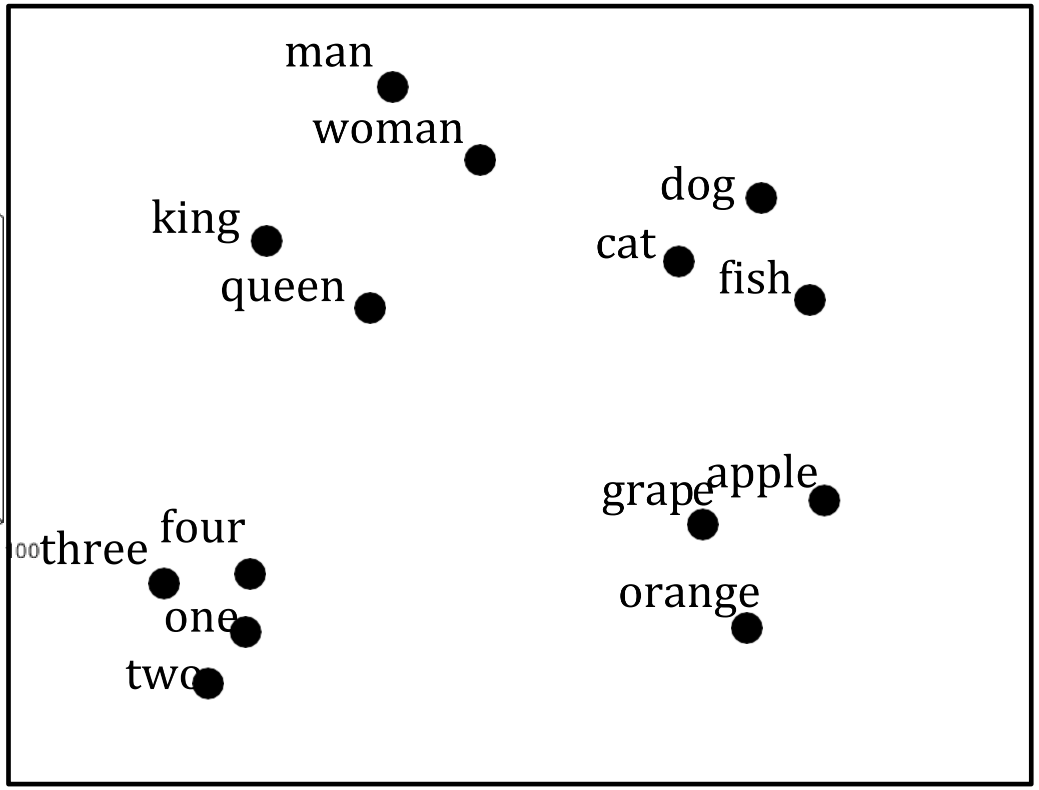
_______. I want a glass of apple juice.

Visualizing word embeddings

- The features learned in word embeddings may not have an easy-to-interpret interpretation, but they will allow an algorithm to recognize the relationships between different words more easily.
- The embeddings can also be visualized in lower dimensions to see how words cluster together based on their feature values.
- Word embeddings have been one of the most important ideas in NLP and have allowed for significant improvements in tasks such as language translation, sentiment analysis, and text classification.

Visualizing word embeddings





FPT Education

FPT UNIVERSITY

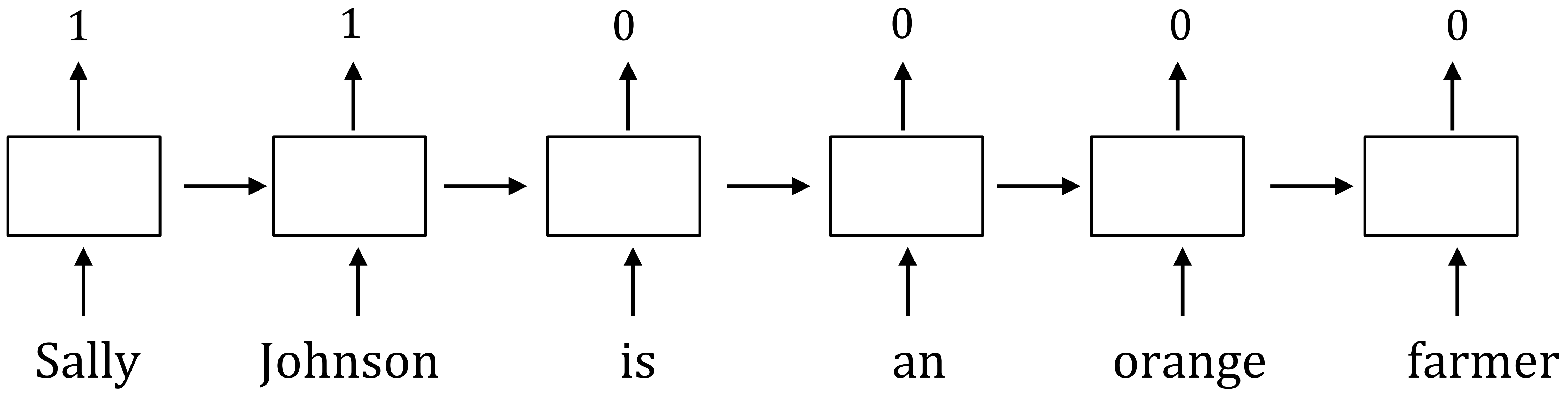
NLP and Word Embeddings

Using word embeddings

Named entity recognition example

- How can featurized representations of words, known as word embeddings, be used in NLP applications, specifically named entity recognition?
- Word embeddings can be learned from large text corpuses, which can be used to identify relationships between words, such as the fact that "orange" and "durian" are both fruits. This information can then be used to improve the accuracy of named entity recognition, even when the task includes words that were not present in the original training set.

Named entity recognition example



Robert Lin is an apple farmer

Transfer learning and word embeddings

- The benefits of using word embeddings for transfer learning, where knowledge learned from a larger dataset can be applied to a smaller, related task. Word embeddings can also be used to represent words using lower-dimensional feature vectors, which can be faster and more efficient than traditional one-hot vectors.
- Word embeddings have been useful for many NLP tasks, including named entity recognition, text summarization, co-reference resolution, and parsing. However, they may be less useful for language modeling and machine translation when a large dataset is available.

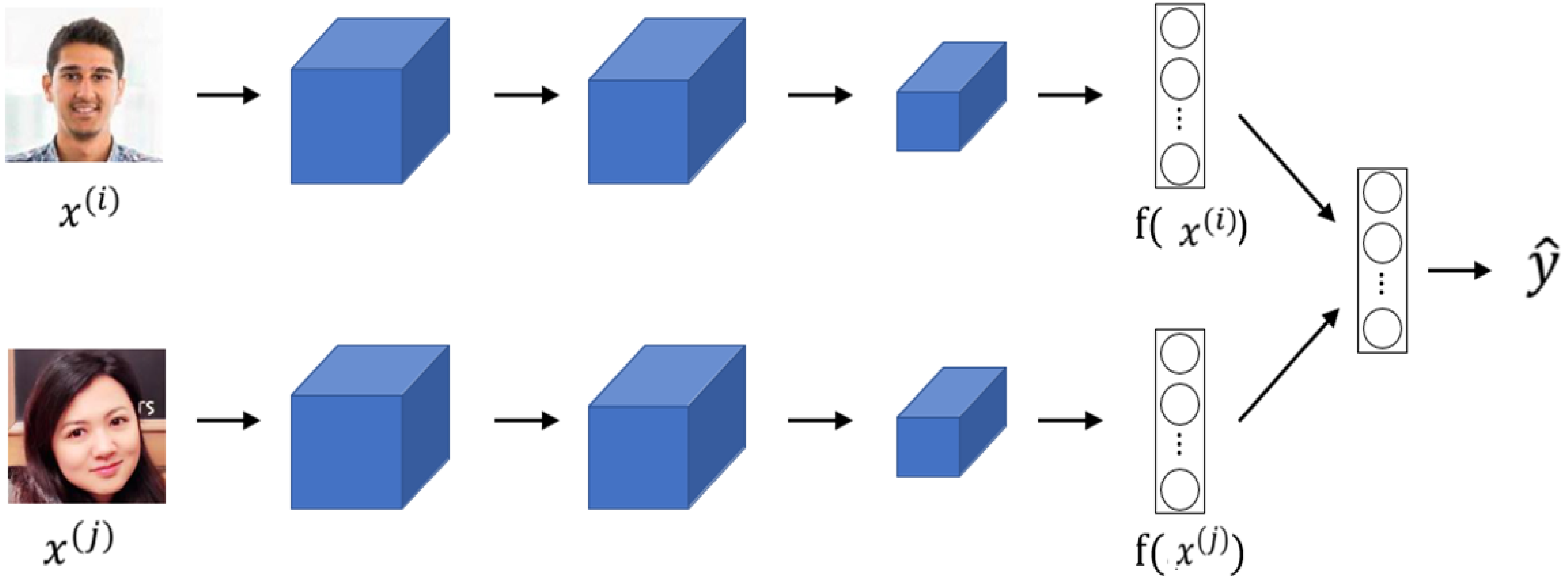
Transfer learning and word embeddings

- Learn word embeddings from large text corpus. (1-100B words) (Or download pre-trained embedding online.)
- Transfer embedding to new task with smaller training set. (say, 100k words)
- Optional: Continue to finetune the word embeddings with new data.

Relation to face embeddings

- While the concepts are similar, there are differences between word embeddings and face encodings in how the algorithms are applied due to the fixed vocabulary in NLP.
- The terms "encoding" and "embedding" are often used interchangeably in these contexts.

Relation to face embeddings



FPT Education

FPT UNIVERSITY

NLP and Word Embeddings

Properties of word embeddings

Analogies

- Word embeddings in NLP capture semantic relationships in a high-dimensional space, enabling analogy reasoning.
- For instance, by manipulating vector representations of "man," "woman," "king," and "queen," we can automatically solve analogies like "man is to woman as king is to what?"
- This involves subtracting the gender difference vector from "king" and finding the word closest to the result, which, in this case, is "queen."

Analogies

	Man (5391)	Woman (9853)	King (4914)	Queen (7157)	Apple (456)	Orange (6257)
Gender	−1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

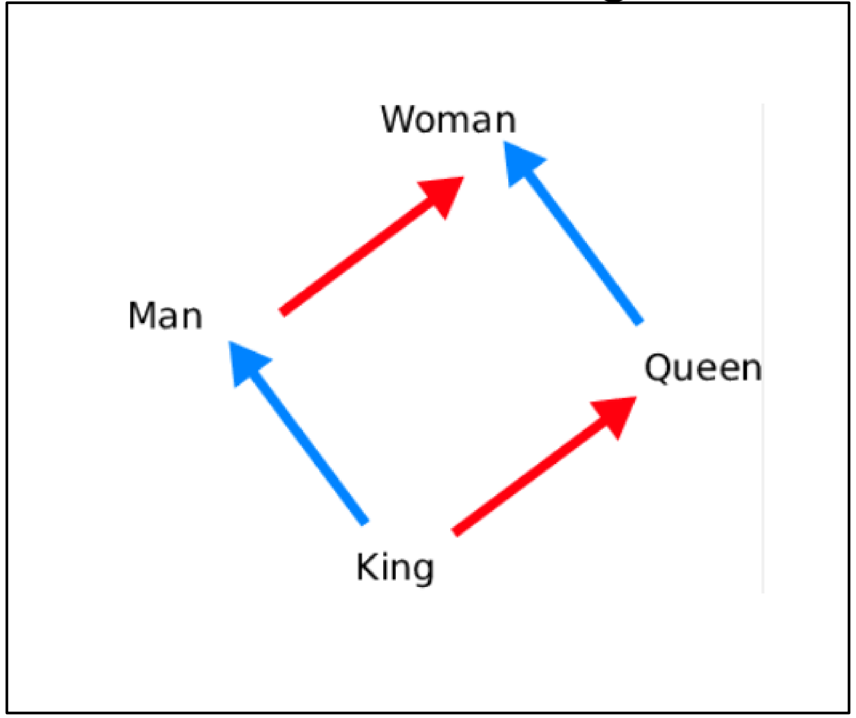
$$e_{man} - e_{woman} \approx e_{king} - e_{w}$$

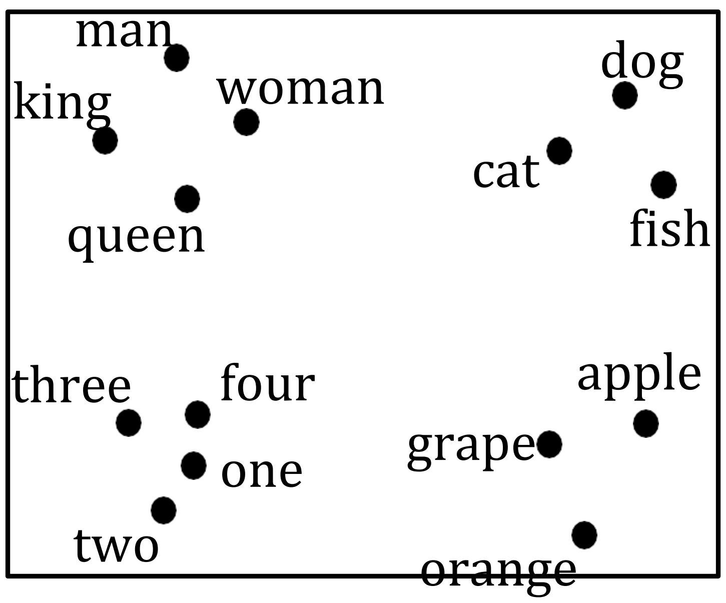
Analogies using word vectors

- One of the remarkable results of word embeddings is the generality of the analogy relationships they can learn.
- For example, they can learn analogies such as "man is to woman as boy is to girl," "Ottawa is to Canada as Nairobi is to Kenya," "big is to bigger as tall is to taller," and "yen is to Japan as ruble is to Russia."
- These relationships can be learned by running a word embedding learning algorithm on a large text corpus.

Analogies using word vectors

- Find word w : argmax





$$e_{man} - e_{woman} \approx e_{king} - e_w$$

$$sim(e_w, e_{king} - e_{man} + e_{woman})$$

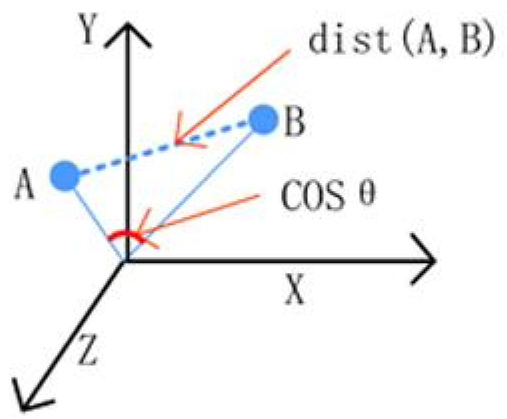
Cosine similarity

- The most commonly used similarity function for measuring the similarity between two vectors in the high-dimensional space is cosine similarity.
- Cosine similarity measures the cosine of the angle between the two vectors and ranges from -1 to 1, where 1 indicates that the two vectors are identical, 0 indicates that the two vectors are orthogonal, and -1 indicates that the two vectors are pointing in opposite directions.

Cosine similarity

$$sim(e_w, e_{king} - e_{man} + e_{woman})$$

Cosine Similarity & Cosine Distance



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

$$\text{Distance} = \quad D_C(A, B) = 1 - S_C(A, B)$$

Man:Woman as Boy:Girl

Ottawa:Canada as Nairobi:Kenya

Big:Bigger as Tall:Taller Yen:Japan

as Ruble:Russia

Properties of word embeddings

- Overall, word embeddings provide a way to capture the semantic relationships between words in a high-dimensional space, and analogy reasoning is just one of the many applications that can be built on top of them.

FPT Education

FPT UNIVERSITY

NLP and Word Embeddings

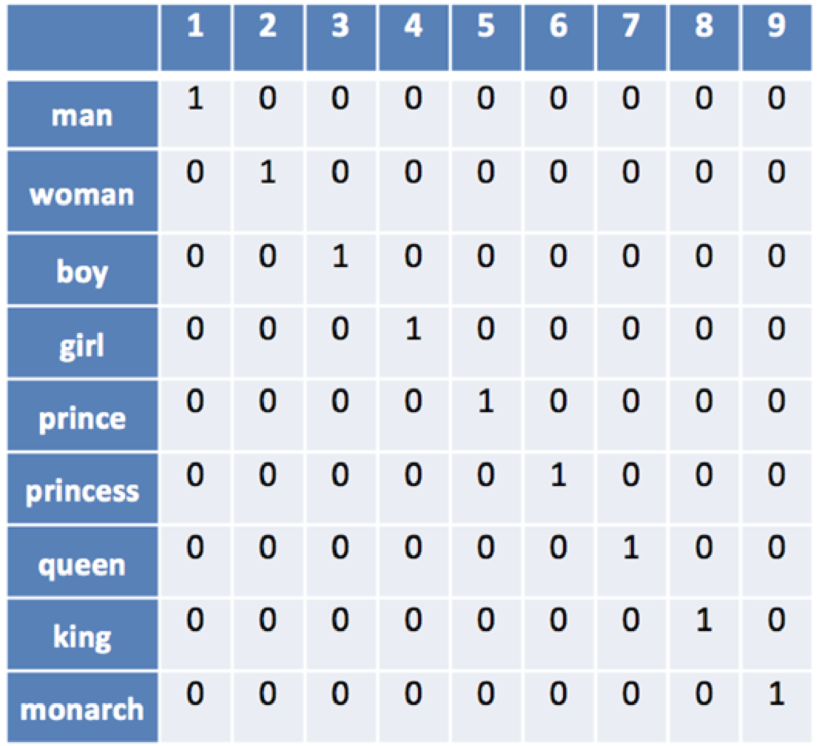
Embedding matrix

Embedding matrix

- Learning a word embedding involves an embedding matrix, E, for a vocabulary of 10,000 words. Each column corresponds to an embedding for a word.
- One-hot encoding, a 10,000-dimensional vector, represents each word.
- To get the embedding vector for a word, multiply E by its one-hot vector, resulting in a 300-dimensional vector.
- The notation E 6257 represents the embedding vector for the word at index 6257 in the vocabulary.

Embedding matrix

Vocabulary:
Man, woman, boy, girl, prince, princess, queen, king, monarch



	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets a 1x9 vector representation

In practice, use specialized function to look up an embedding.

FPT Education

FPT UNIVERSITY

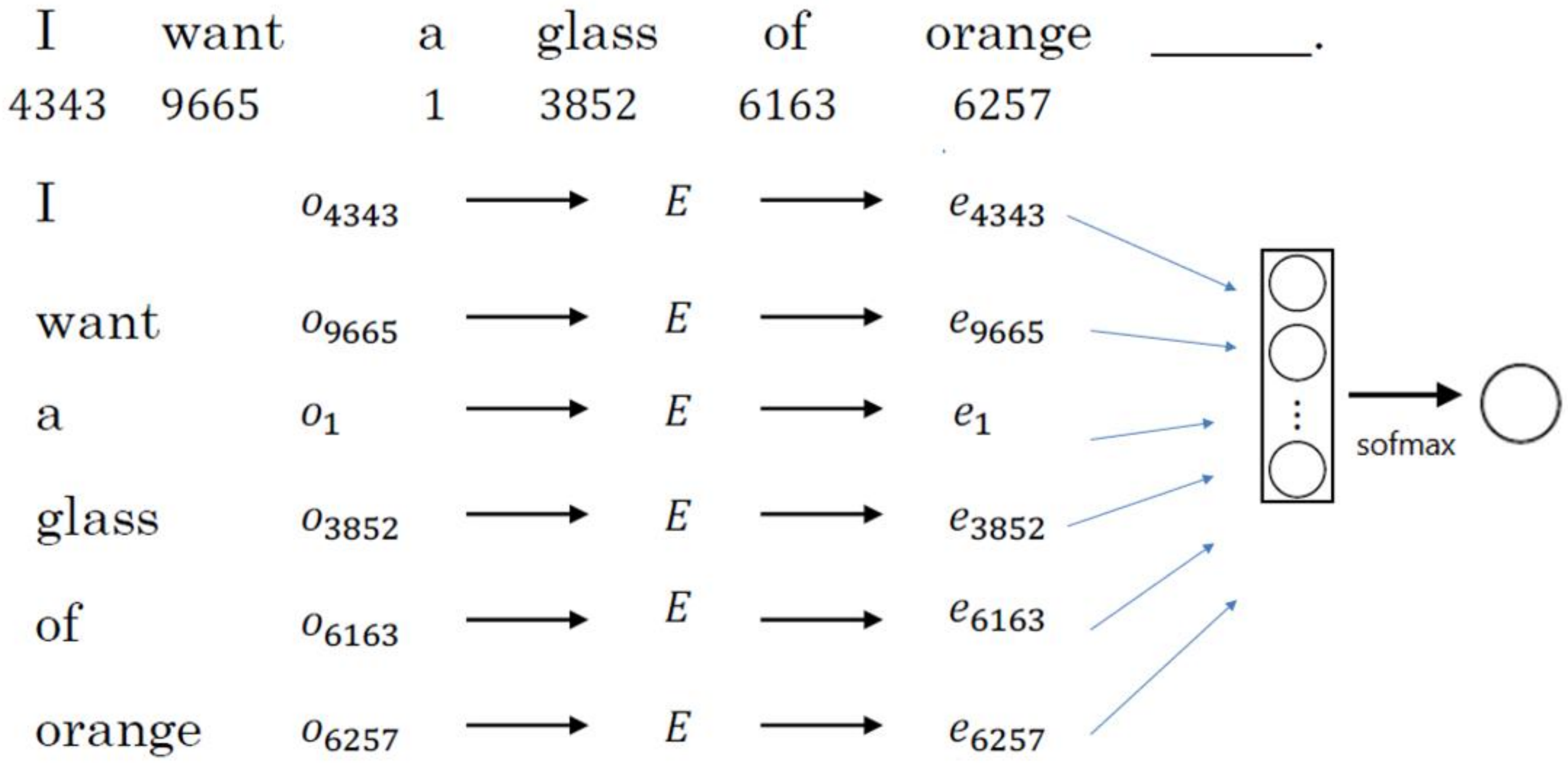
NLP and Word Embeddings

Learning word embeddings

Neural language model

- Word embeddings involve a 300-dimensional by 10,000-dimensional (or 10,001) embedding matrix. Columns represent word embeddings.
- In a neural language model, the network predicts the next word in a sequence. Input words are represented by embedding vectors, obtained by multiplying a one-hot vector with the embedding matrix. The neural network's softmax layer classifies among 10,000 possible vocabulary outputs for the final word.

Neural language model



I want a glass of orange ______.
4343 9665 1 3852 6163 6257
I o_{4343} → E → e_{4343}
want o_{9665} → E → e_{9665}
a o_1 → E → e_1
glass o_{3852} → E → e_{3852}
of o_{6163} → E → e_{6163}
orange o_{6257} → E → e_{6257}
sofmax

Other context/target pairs

- This algorithm, successful for learning word embeddings, has simpler alternatives.
- Researchers explored various contexts, like words before or around the target word, constructing learning problems for neural networks to predict the target word and learn meaningful embeddings. The next section delves into the Word2Vec model.

Other context/target pairs

FPT Education
FPT UNIVERSITY

I want a glass of orange juice to go along with my cereal.

Context: Last 4 words.

a glass of orange ? to go along with

4 words on left & right

orange ?

Last 1 word

glass ?

Nearby 1 word

FPT Education

FPT UNIVERSITY

NLP and Word Embeddings

Word2Vec

Word2Vec

- The Word2Vec algorithm is a simpler and more efficient way of learning these types of embeddings.
- The skip-gram model is used in Word2Vec, and it works by creating supervised learning problems by selecting a context word and a target word within a certain window.
- The goal of this is not to do well on the supervised learning problem but to learn good word embeddings.

Skip-grams

- Skip-gram uses a softmax unit to predict a target word within a window using the input context word's embedding vector.
- The softmax unit, with parameters for each target word, employs negative log likelihood as the loss function.
- The computational speed issue arises due to the need for a sum over the entire vocabulary during probability evaluation.
- A solution is a hierarchical softmax classifier, organizing classifiers in a tree to scale logarithmically with vocabulary size rather than linearly.

Skip-grams

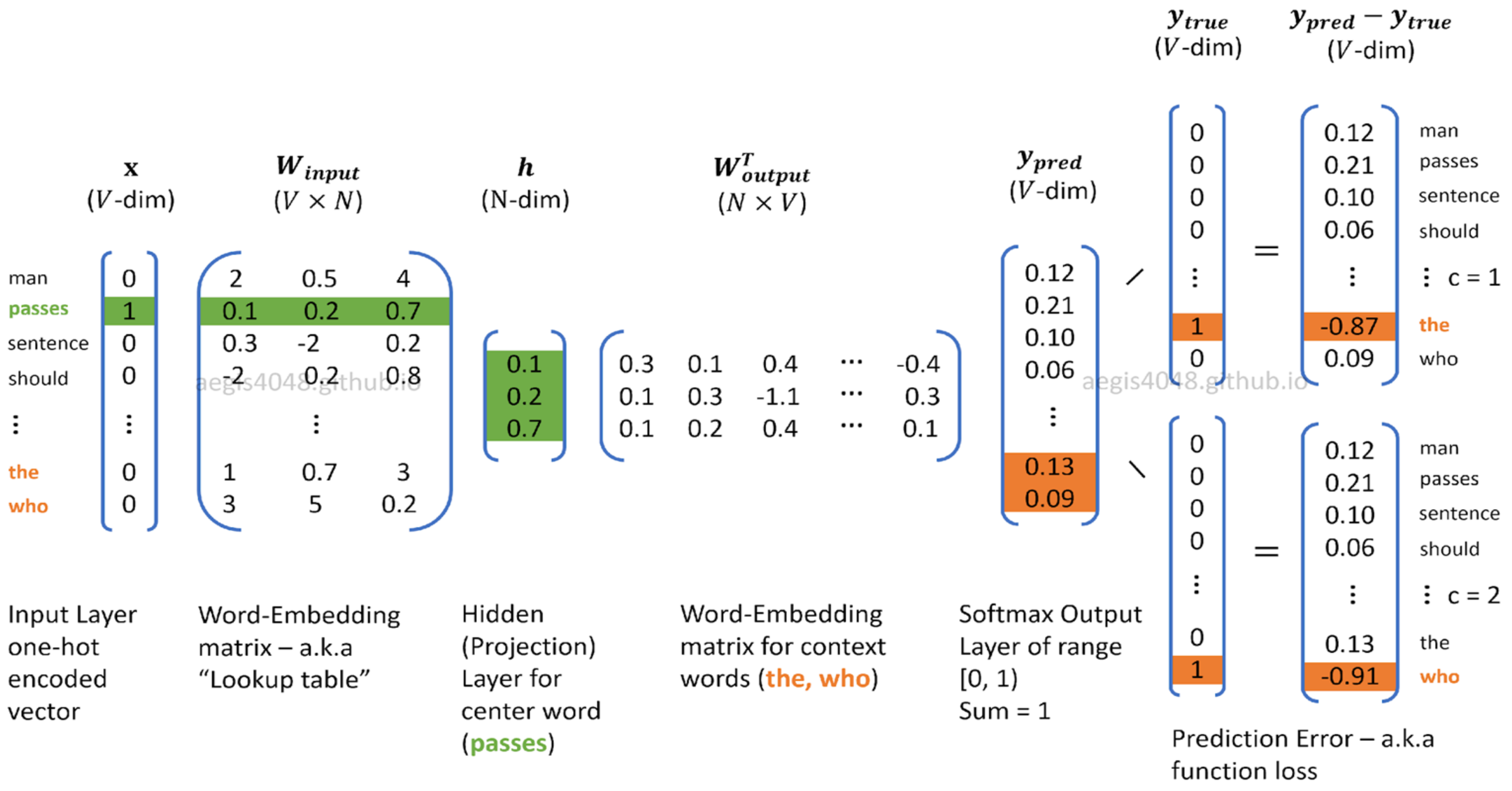
I want a glass of orange juice to go along with my cereal.

Context	Targer
orange	juice
orange	glass
orange	my

Model

- The skip-gram model is one of two versions of the Word2Vec model, with the other being the continuous backward model (CBow), which uses the surrounding words to predict the middle word.
- The key problem with the skip-gram model is that the softmax step is very expensive to calculate, an algorithm that modifies the training objective is presented to make it run much more efficiently and learn much better word embeddings.

Model


y_{true} (V-dim)
$y_{pred} - y_{true}$ (V-dim)
$\mathbf{x}$ (V-dim)
W_{input} ($V \times N$)
$\boldsymbol{h}$ (N-dim)
W^T_{output} ($N \times V$)
y_{pred} (V-dim)
man
passes
sentence
should
the
who
c = 1
c = 2
aegis4048.github.io
Input Layer one-hot encoded vector
Word-Embedding matrix – a.k.a "Lookup table"
Hidden (Projection) Layer for center word (passes)
Word-Embedding matrix for context words (the, who)
Softmax Output Layer of range [0, 1) Sum = 1
Prediction Error – a.k.a function loss

$$\mathbf{x} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad W_{input} = \begin{bmatrix} 2 & 0.5 & 4 \\ 0.1 & 0.2 & 0.7 \\ 0.3 & -2 & 0.2 \\ -2 & 0.2 & 0.8 \\ & \vdots & \\ 1 & 0.7 & 3 \\ 3 & 5 & 0.2 \end{bmatrix}, \quad \boldsymbol{h} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.7 \end{bmatrix}, \quad W^T_{output} = \begin{bmatrix} 0.3 & 0.1 & 0.4 & \cdots & -0.4 \\ 0.1 & 0.3 & -1.1 & \cdots & 0.3 \\ 0.1 & 0.2 & 0.4 & \cdots & 0.1 \end{bmatrix}$$

$$y_{pred} = \begin{bmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ 0.13 \\ 0.09 \end{bmatrix}$$

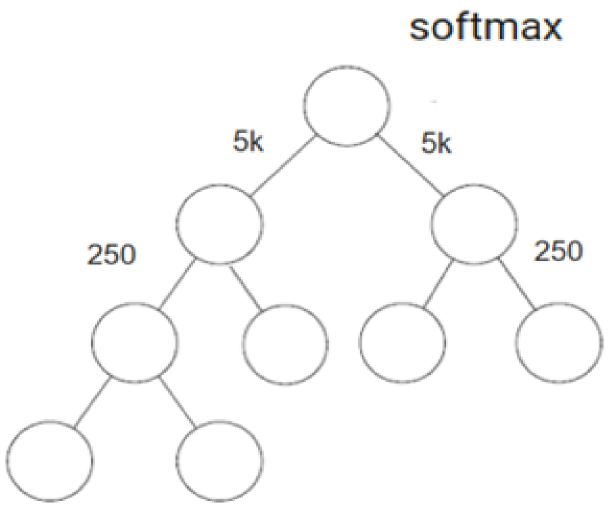
$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ -0.87 \\ 0.09 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ 0.13 \\ -0.91 \end{bmatrix}$$

Problems with softmax classification

- Another problem is how to sample the context C.
- Sampling uniformly at random results in some words appearing extremely frequently and dominating the training set, while others appear less often.
- The distribution of words pc is not taken entirely uniformly at random for the training set, and instead, different heuristics can be used to balance out common words and less common words.

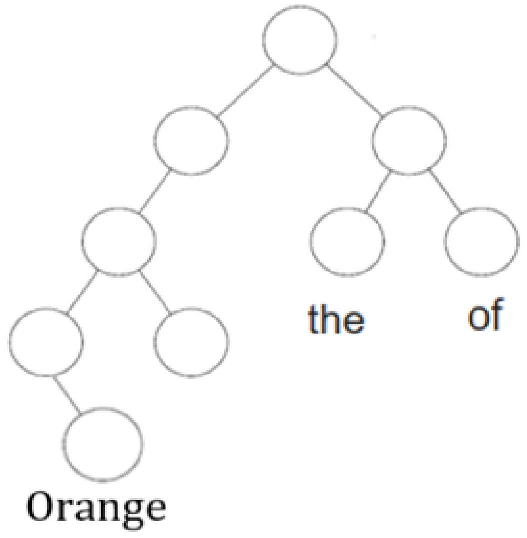
Problem with softmax classification

Softmax: $$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e}}$$



How to sample the context c ?

- The , of , a , and , to ....
- Orange , apple , banana ....



FPT Education

FPT UNIVERSITY

NLP and Word Embeddings

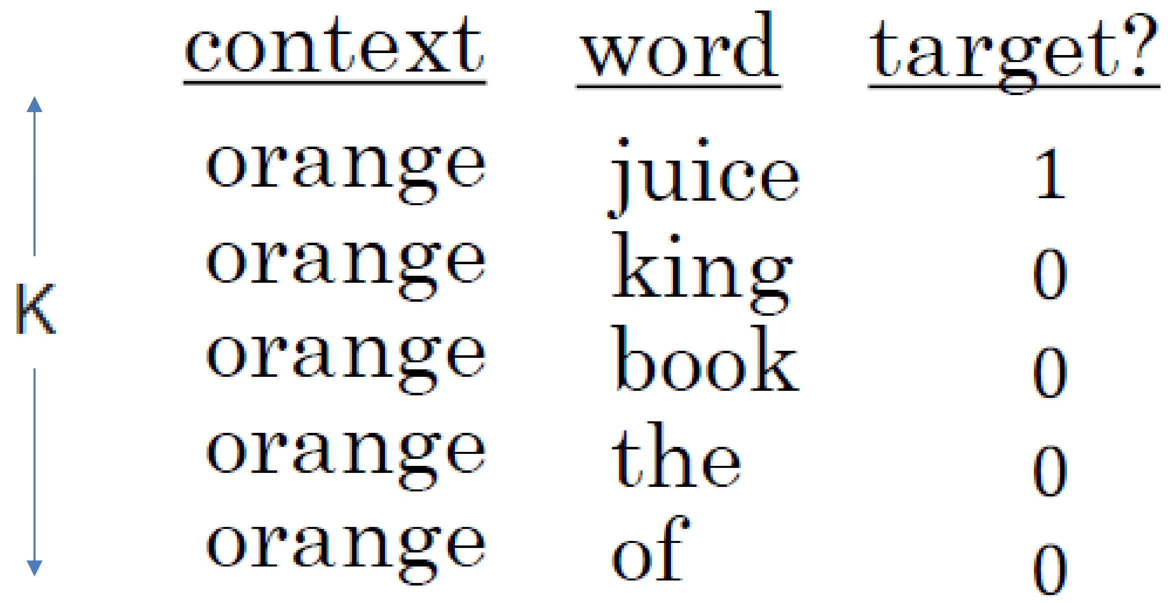
Negative sampling

Defining a new learning problem

- How can the negative sampling technique be used to learn word embeddings efficiently?
- Negative sampling creates a supervised learning problem where the task is to predict if a pair of words is a context-target pair or not.
- For instance, given a pair of words like orange and juice, the model has to predict if they appear together or not.
- If they appear together, the label is 1, and if they don't, the label is 0. To generate negative examples, the same context word is used, and a random word is chosen from the dictionary.
- The negative examples are labeled 0.

Defining a new learning problem

- K = 5 – 20   Small datasets
- I want a glass of orange juice to go along with my cereal
- K = 2 – 5   Large datasets



context	word	target?
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

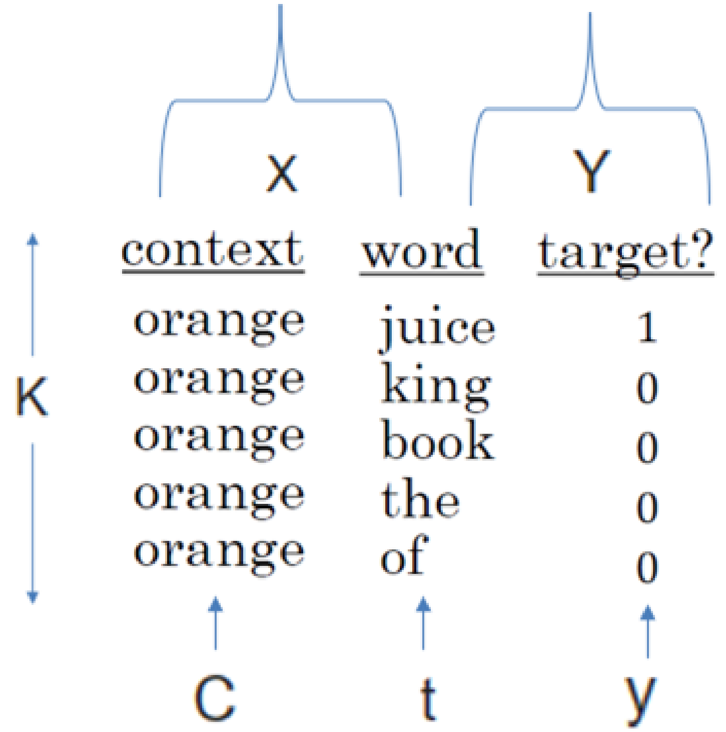
Model

- To define the model, a logistic regression model is used. The chance of the label being 1 is modeled using a regression model. The specific formula used is sigmoid applied to the dot product of two vectors, one vector for the context word and another for the target word. The parameters are learned during training.
- The computation cost of the negative sampling algorithm is much lower than that of the Softmax model because instead of updating a giant Softmax classifier, only k + 1 binary classification problems are updated on each iteration. K represents the number of negative examples. It is suggested that k should be 5 to 20 for smaller data sets and 2 to 5 for larger data sets.

Model

Softmax: $$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

$$p(y = 1 \mid c, t) = \sigma(\theta_t^T e_c)$$

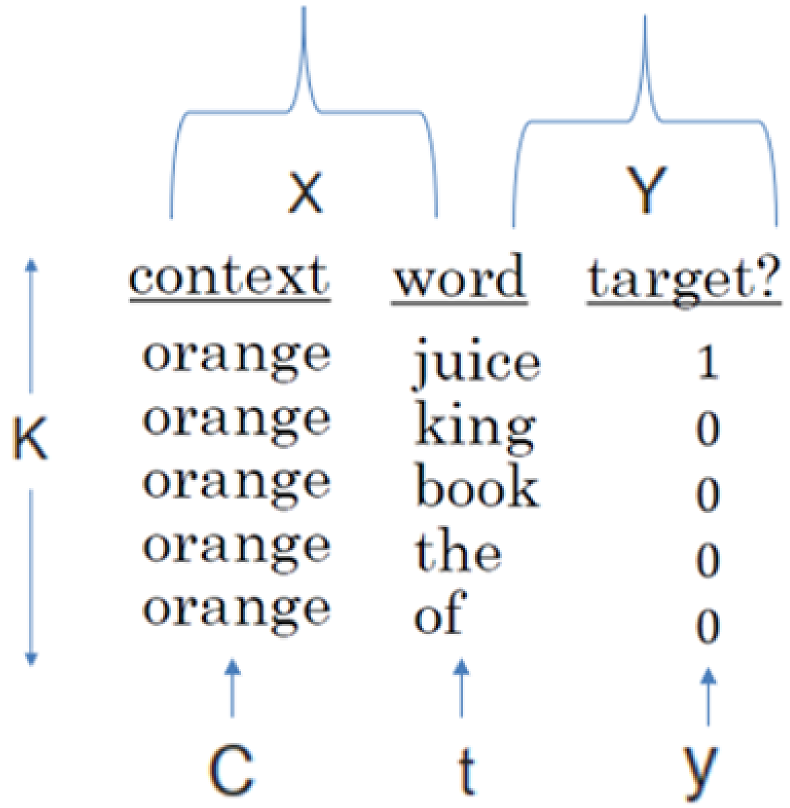


context	word	target?
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

Selecting negative examples

- How can negative examples be sampled to train the model?
  - Sampling from the empirical frequencies or uniformly at random from the dictionary is not recommended. Instead, a heuristic value is suggested, which is a little bit in between the two extremes. Negative examples are sampled proportional to the frequency of a word to the power of three-fourths.
  - There are open source implementations and pre-trained word vectors available online that can be used as a starting point for NLP problems. The next section will introduce the Glove algorithm, which is another word embedding learning algorithm.

Selecting negative examples



context	word	target?
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

The , of , a .....

$$P(w\,i) = \frac{f(wi)^{3/4}}{\sum_{j=1}^{10,000} f(wi)^{3/4}} \qquad \frac{1}{|V|}$$

FPT Education

FPT UNIVERSITY

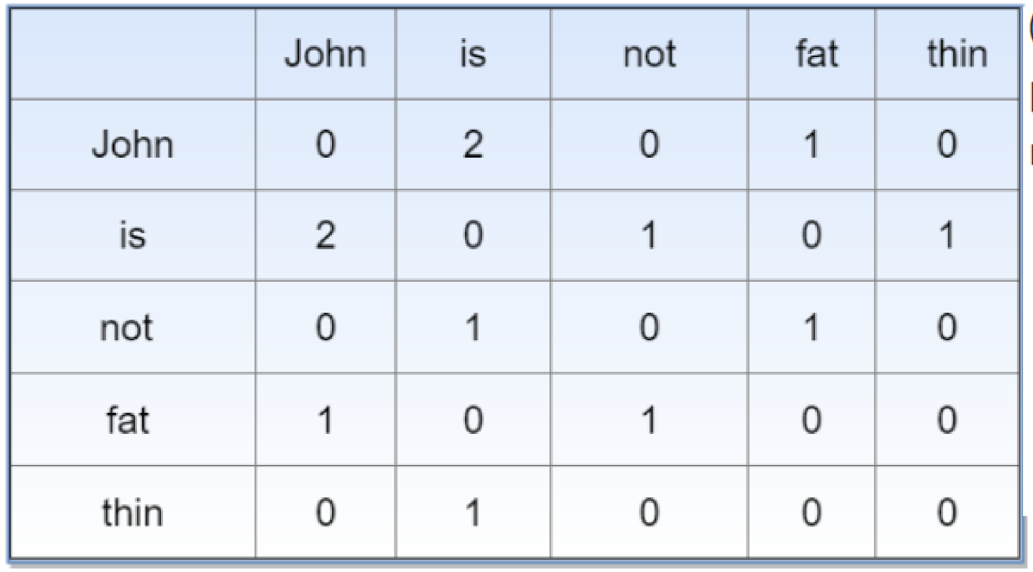
NLP and Word Embeddings

GloVe word vectors

GloVe (global vectors forword representation)

- The GloVe algorithm is another algorithm for computing word embeddings that was created by Jeffrey Pennington, Richard Socher, and Chris Manning.
- GloVe stands for "global vectors for word representation."
- The algorithm starts by making explicit the pairs of words that appear in close proximity to each other in a text corpus by counting how many times a word i appears in the context of a different word j. The count X_ij captures how often words i and j appear with each other or close to each other. The GloVe model optimizes the difference between theta_i transpose e_j and log of X_ij squared using gradient descent.

GloVe (global vectors for word representation)



	John	is	not	fat	thin
John	0	2	0	1	0
is	2	0	1	0	1
not	0	1	0	1	0
fat	1	0	1	0	0
thin	0	1	0	0	0

Figure 2-Ví dụ về Co-occurence Matrix

(introduced in 2014), it is actually built on the ***Co-occurrence Matrix***. ***GloVe*** is probabilistic in nature, the idea of building this method comes from the following ratio:

$$\frac{P(k|i)}{P(k|j)} \qquad (1)$$

GloVe (global vectors for word representation)

Inside:

P(k|i) is the probability of the occurrence of ***the word k*** in the context of the word ***i***, similar to ***P(k|j).***

The formula of ***P(k|i)*** :

$$P(k|i) = \frac{X_{ik}}{X_i} = \frac{X_{ik}}{\sum_m X_{im}}$$

Inside:

Xik : the number of occurrences of the ***k-word*** in the context of the ***i-word*** (or vice versa).

Xi : the number of occurrences of the word i in the context of all the remaining words except ***i***.

Model

- The goal is to learn vectors for effective word co-occurrence prediction. If $X_{ij} = 0$, the log of 0 is undefined, so an extra weighting term is added to sum only over co-occurred word pairs.
- Various heuristics for the weighting function *F* balance meaningful computation for infrequent words without excessive weight. In the GloVe algorithm, roles of θ and e are symmetric. Training involves initializing θ and e uniformly, minimizing the objective with gradient descent, and averaging for a word afterward.
- Despite potentially arbitrary linear transformations, the parallelogram map still works for figure analogies.

Model

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{ij}))^2$$

Minimize: $\Sigma^{10000}_{i=1}\Sigma^{10000}_{j=1} f(X_{ij})(\theta^T_i e_j + b_i + b_{j'} - \log X_{ij})^2$

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases}$$

$$e^{(found)}_{\omega} = (e_{\omega} + \theta_{\omega})/2$$

wi,wj are magnetic vectors.

bi, ***bj*** are the corresponding ***biases*** (added at simplified and optimal steps).

Xij: the entry corresponds to the pair from ***i,j*** in ***the Co-occurrence Matrix***.

The ***function f*** is called the ***weighting*** function, which is added to reduce the influence of pairs of words that appear too often, this function satisfies 3 properties:

A note on the featurization view of word embeddings

- Man Woman King Queen (5391) (9853) (4914) (7157)
- Gender Royal Age Food

-1	1	-0.95	0.97
0.01	0.02	0.93	0.95
0.03	0.02	0.70	0.69
0.09	0.01	0.02	0.01

$$\text{minimize} \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})\left(\theta_i^T e_j + b_i - b_j' - \log X_{ij}\right)^2$$

FPT Education

FPT UNIVERSITY

NLP and Word Embeddings

Sentiment classification

Sentiment classification problem

- Sentiment classification in NLP analyzes text to determine positive or negative opinions. Limited labeled training data is a challenge, but word embeddings mitigate this by enabling learning from a larger text corpus.
- A common approach uses one-hot encoding for words, multiplied by an embedding matrix to get word embeddings. These are then averaged or summed to create a fixed-length feature vector for softmax sentiment prediction.

Sentiment classification problem

x	Y
The dessert is excellent.	★★★★☆
Service was quite slow.	★★☆☆☆
Good for a quick meal, but nothing special.	★★★☆☆
Completely lacking in good taste, good service, and good ambience.	★☆☆☆☆

Simple sentiment classification model

The	dessert	is	excellent
8928	2468	4694	3180

★★★★☆

The $o_{8928} \longrightarrow E \longrightarrow e_{8928}$

desert $o_{2468} \longrightarrow E \longrightarrow e_{2468}$

is $o_{4694} \longrightarrow E \longrightarrow e_{4694}$

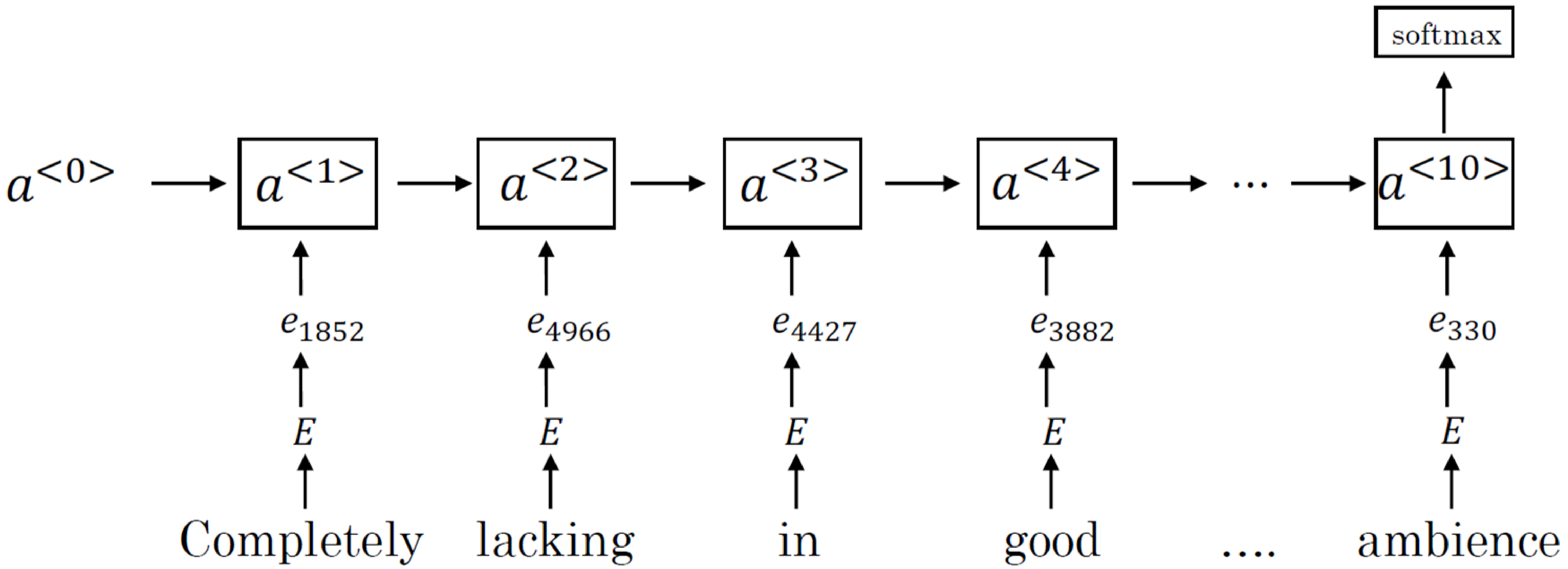
excellent $o_{3180} \longrightarrow E \longrightarrow e_{3180}$

"Completely lacking in taste, good service, and ambience."

RNN for sentiment classification

- While the one-hot encoding and embedding matrix approach for sentiment classification lacks consideration of word order, a more effective solution involves using a recurrent neural network (RNN).
- RNNs process text sequentially, capturing temporal dependencies and improving accuracy by considering word order.
- In summary, word embeddings enhance sentiment classification, and RNNs enable better modeling of language nuances for more accurate predictions.

RNN for sentiment classification



FPT Education

FPT UNIVERSITY

NLP and Word Embeddings

Debiasing word embeddings

The problem of bias in word embeddings

- Word embeddings represent words as vectors in high-dimensional space, applied in NLP tasks like sentiment analysis and translation.
- Despite their utility, word embeddings may inherit biases from the training text, exemplified by gender stereotypes. To mitigate bias, ideas for reduction or elimination are proposed in word embeddings.

The problem of bias in word embeddings

- Man:Woman as King:Queen
- Man:Computer_Programmer as Woman: Homemaker
- Father: Doctor as Mother: Nurse

- Word embeddings can reflect gender, ethnicity, age, sexual orientation, and other biases of the text used to train the model.

Addressing bias in word embeddings

- To mitigate bias in word embeddings, the first step is identifying the bias direction, such as the gender difference between "he" and "she."
- Next, non-definitional words like "grandmother" are projected onto this bias direction to reduce their gender component.
- Finally, pairs like "grandmother" and "grandfather" are equalized to be equidistant from the bias direction.
- A classifier is used to identify definitional words and neutralize others, hand-picking pairs for equalization. The full algorithm, more intricate than described, is detailed in the original paper.

Addressing bias in word embeddings

1. Identify bias direction.
2. Neutralize: For every word that is not definitional, project to get rid of bias.
3. Equalize pairs.

Debiasing word embeddings

- Overall, reducing bias play a important role in learning algorithms, given their increasing use in making important decisions in society, such as college admissions, job searches, loan applications, and criminal justice.

Summarization

- Word representation involves converting words into numerical vectors for computational processing.
- Word embeddings, like those from Word2Vec or GloVe, capture semantic relationships and exhibit properties like semantic similarity.
- An embedding matrix maps words to vectors, crucial in NLP models. Word2Vec uses neural networks to learn embeddings, employing methods like Skip-gram.
- GloVe focuses on global corpus statistics for embeddings.
- Sentiment classification, determining text sentiment, benefits from word embeddings.
- Techniques like debiasing address biases inherited by word embeddings from training data.