Text Mining and Language Analytics

Lecture 6

Word embeddings II

Dr Stamos Katsigiannis

2023-24

Durham University

Embeddings: TF / TF-IDF

- TF/TF-IDF
  - Common baseline model
  - The standard solution in Information Retrieval!
  - Words represented by a simple function of their frequency and the frequency of other words
  - Similar words expected to have similar vectors
  - High dimensionality (| V | dimensions)
    - Millions of dimensions for large corpora
  - Very **sparse** vectors (most elements are zero)

Similar words keep the same company

- **How to increase density and reduce dimensionality?**

- **Solution** → Learn shorter and denser vectors

Durham University

Embeddings: Sparse vs. Dense vectors

Why dense vectors?

- Dense vectors are typically shorter → Easier to use as features in machine learning
- Dense vectors may generalise better than explicit word frequencies
- Dense vectors may do better at capturing synonymy
- **In practice, they work better!**

Durham University

Embeddings: Dimensionality reduction

- **Given a Word-Word matrix, how to reduce dimensions?**
  - Filtering
  - Feature selection methods
  - Project data on different space
    - Rotate the axes into a new space where the highest order dimension captures the most variance in the data
    - The next dimension captures the next most variance
    - And so on …

- Many methods exist, e.g.:
  - **Principal Component Analysis (PCA)**
  - **Singular Value Decomposition (SVD)**

- How to select the number of dimensions?
  - **Empirically decided!**
  - Typically between 50 and 500 dimensions
  - 300 dimensions often used in the literature

Durham University

Singular Value Decomposition (SVD)

- Consider a Word-Word matrix M
- SVD of M: $\mathrm{M} = \mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}}$

U: Rows correspond to original. Mapped in a different space. Columns ordered by descending variance they account for
Σ: Diagonal matrix of singular values expressing importance of each dimension

General SVD:

$$\underset{m\times n}{\begin{bmatrix} \mathrm{M} \end{bmatrix}} = \underset{m\times m}{\begin{bmatrix} \mathrm{U} \end{bmatrix}} \underset{m\times n}{\begin{bmatrix} \Sigma \end{bmatrix}} \underset{n\times n}{\begin{bmatrix} \mathrm{V}^{\mathrm{T}} \end{bmatrix}}$$

Each row ***i*** of **U** is an embedding of the ***i***th word

SVD on Word-Word matrix:

$$\underset{|\mathrm{V}|\times|\mathrm{V}|}{\begin{bmatrix} \mathrm{M} \end{bmatrix}} = \underset{|\mathrm{V}|\times|\mathrm{V}|}{\begin{bmatrix} \mathrm{U} \end{bmatrix}} \underset{|\mathrm{V}|\times|\mathrm{V}|}{\begin{bmatrix} \Sigma \end{bmatrix}} \underset{|\mathrm{V}|\times|\mathrm{V}|}{\begin{bmatrix} \mathrm{V}^{\mathrm{T}} \end{bmatrix}}$$

Keep top ***k*** dimensions

SVD on Word-Word matrix:

$$\underset{|\mathrm{V}|\times|\mathrm{V}|}{\begin{bmatrix} \mathrm{M} \end{bmatrix}} = \underset{|\mathrm{V}|\times|\mathrm{V}|}{\begin{bmatrix} \mathrm{U} \end{bmatrix}} \underset{k\times k}{\begin{bmatrix} \Sigma \end{bmatrix}} \underset{k\times|\mathrm{V}|}{\begin{bmatrix} \mathrm{V}^{\mathrm{T}} \end{bmatrix}}$$

Approximation of **M**

Use **U** as the embeddings

Usually $k \in [50,500]$. Also $k = 300$ very common!

Embeddings: word2vec (I)

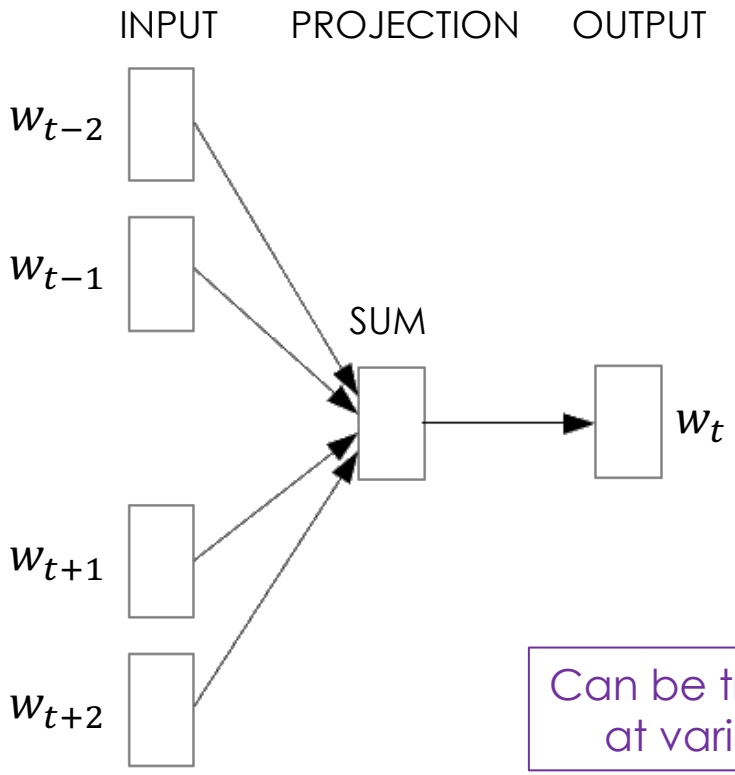
- **Idea** → **Predict than count!**
- Mikolov et al., https://code.google.com/archive/p/word2vec/
- Popular embedding method
- Very fast to train
- Simple neural network with one hidden layer
- word2vec is **not a single algorithm**

Embeddings: word2vec (II)

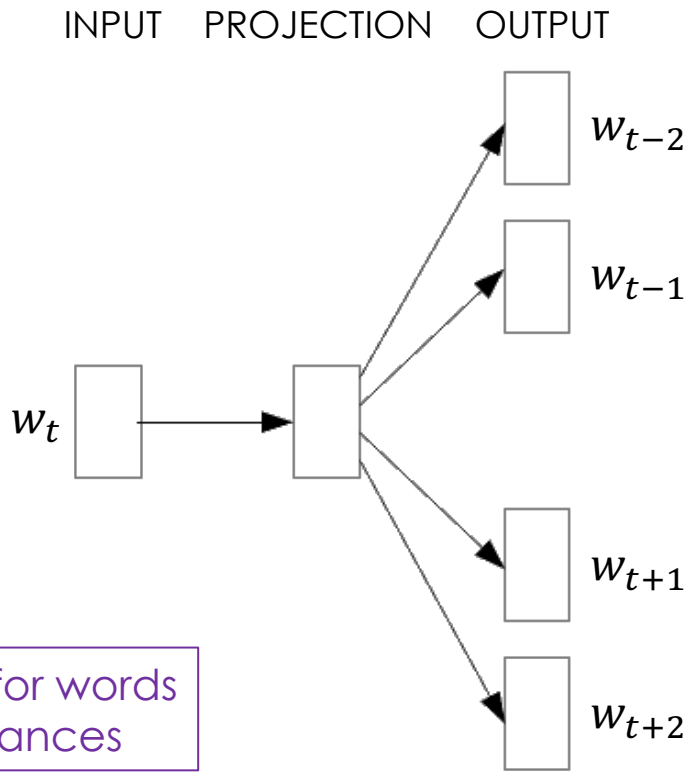
- word2vec **is a software package** for representing words as vectors:
  - Two distinct models
    - Continuous Bag of Words (CBoW)
    - Skip-gram
  - Various training methods
    - Negative sampling
    - Hierarchical Softmax
  - Pre-processing pipeline
    - Dynamic Context Windows
    - Subsampling
    - Deleting rare words

Embeddings: word2vec (III)

CBOW
Predict word from surrounding words



Can be trained for words at various distances



Skip-gram
Use word to predict surrounding words

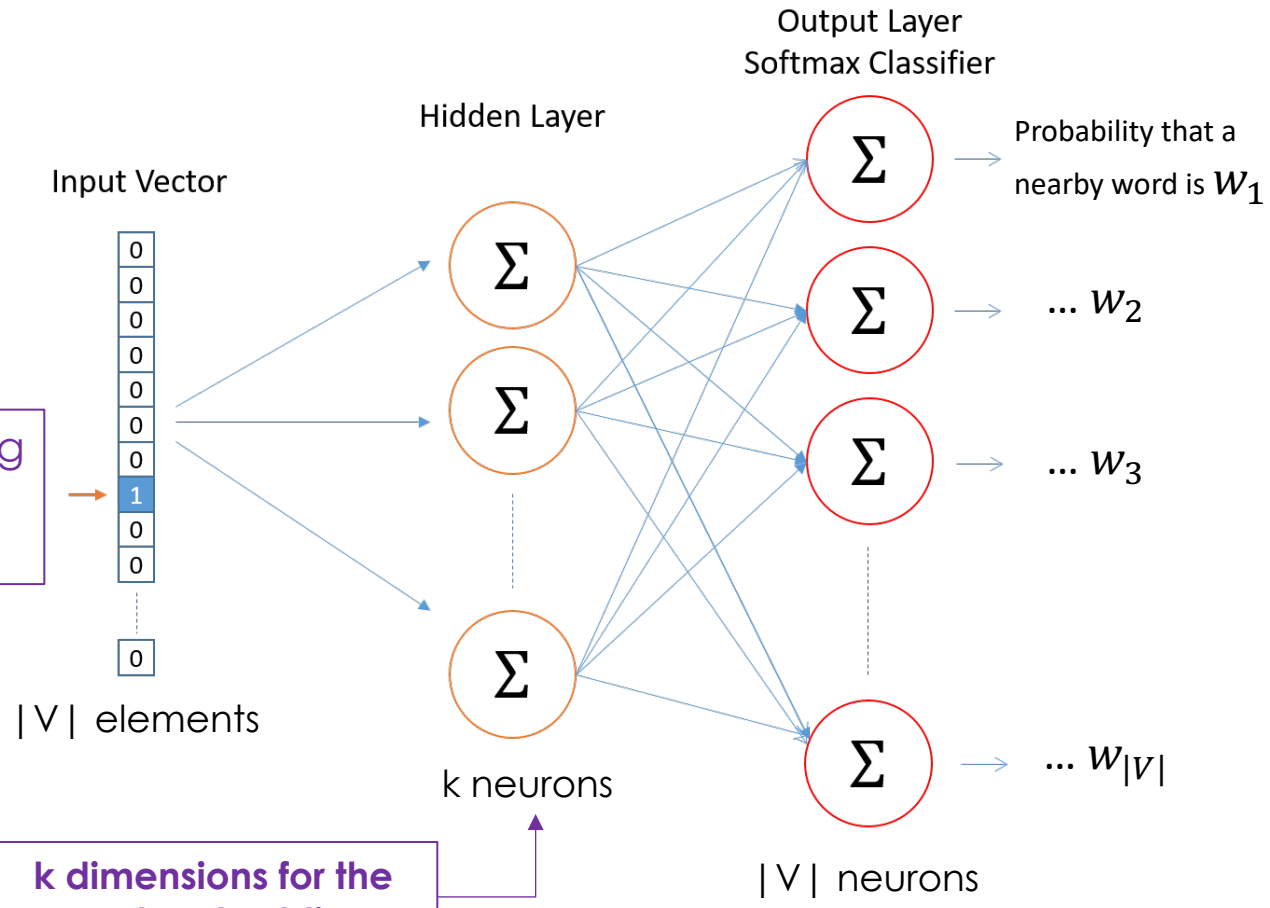
CBoW **Skip-gram**

Durham University

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

Embeddings: word2vec (IV)

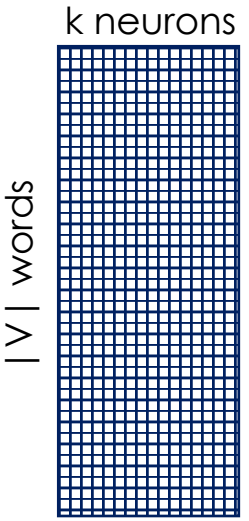
Skip-gram



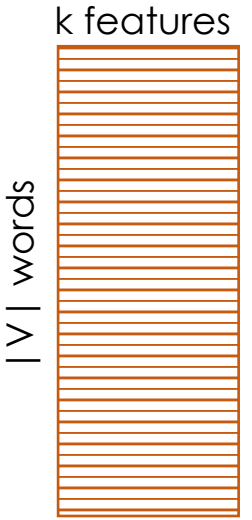
One-hot encoding of word w_i.
i^{th} element = 1

k dimensions for the word embedding

Hidden layer weight matrix



Word vector lookup table



Multiplying a one-hot vector with the weight matrix returns the i^{th} row of the matrix

Training objective → Maximise $\frac{1}{|V|}\sum_{t=1}^{|V|}\sum_{-c\leq j\leq c, j\neq 0}\log P(w_{t+j}|w_t)$

Durham University

Embeddings: word2vec (V)

Skip-gram training

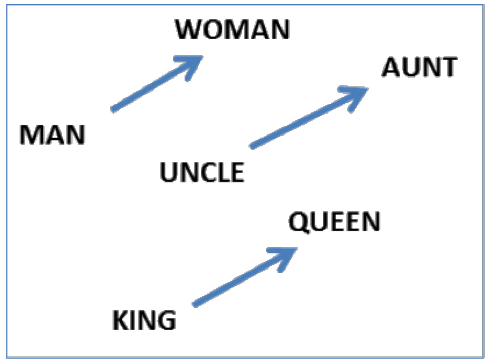
1. Convert words to vectors using one-hot encoding → Vector size 1x|V|
2. Use input layer of |V| neurons to pass word w_t to hidden layer
3. Hidden layer of k neurons performs dot product between weight matrix $W_{|V|\times k}$ and the input vector w_t → The t^{th} row of $W_{|V|\times k}$ will be the output $H_{1\times k}$
4. No activation function used → $H_{1\times k}$ passes directly to the output layer
5. Output layer applies dot product between $H_{1\times k}$ and $W'_{k\times|V|}$ → Output $U_{1\times|V|}$
6. Use softmax to compute probability of each word. The word with the highest probability is the result.
7. If the predicted word for a given context position is wrong, then use backpropagation to modify the weight matrices W and W'

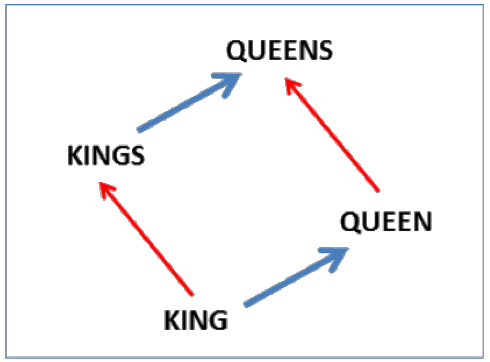
- Execute these steps for each word w_t in vocabulary and each context word
- Each word w_t will be passed 2 x c times, with c being the size of context
- Forward propagation will be processed |V| x 2 x c times per epoch

Embeddings: Properties (I)

- Similarity depends on context size c
  - e.g., Nearest words to Hogwarts
    - c=±2: Sunnydale, Evernight
    - c=±5: Dumbledore, Malfoy, halfblood

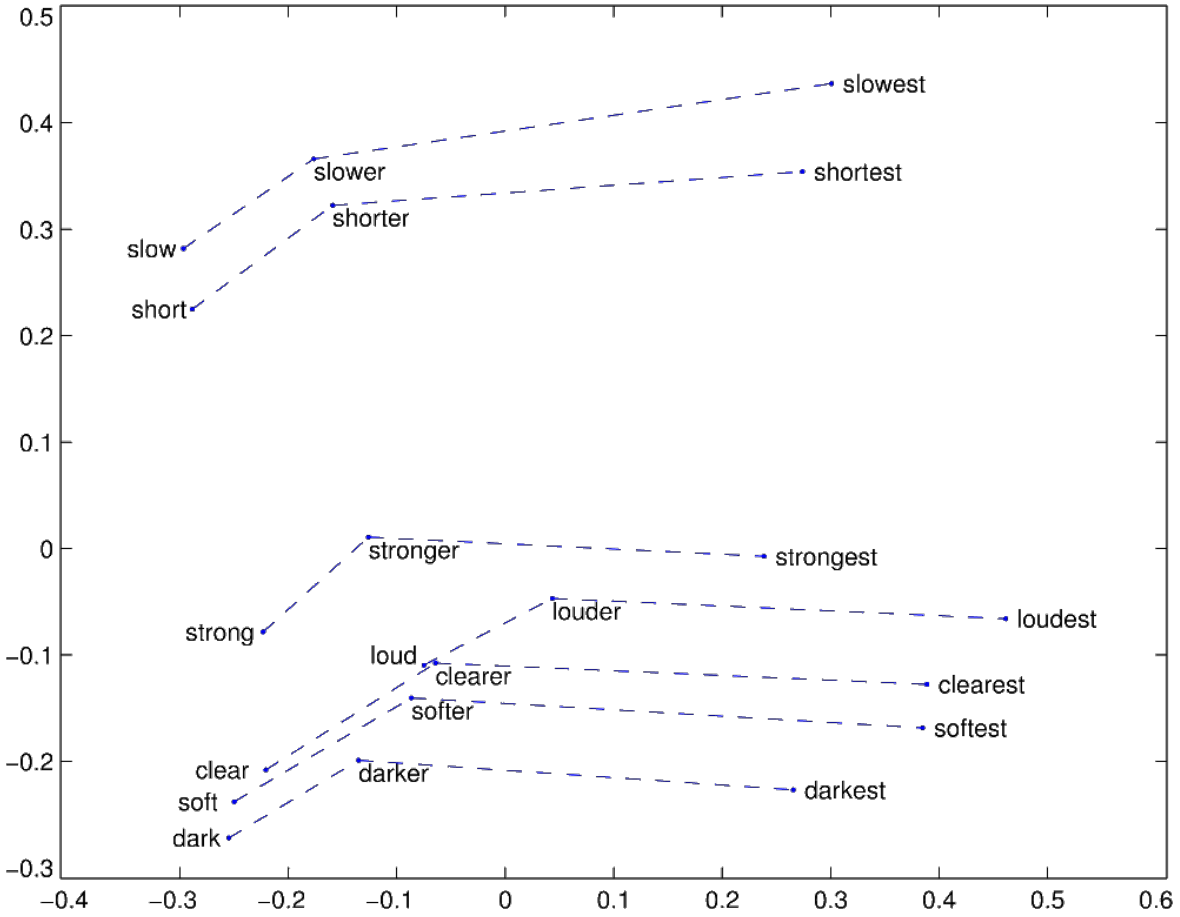
- Embeddings capture relational meaning
  - e.g., vector(king) – vector(man) + vector(woman) ≈ vector(queen)
  - e.g., vector(Paris) – vector(France) + vector(Italy) ≈ vector(Rome)





Durham University

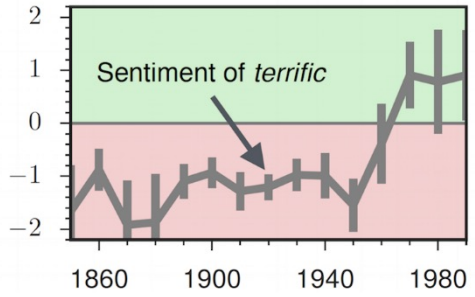
Embeddings: Properties (II)



Durham University

Embeddings: Properties (III)

- Embeddings can help study word history
  - Train embeddings on old books to study changes in word meaning
  - Remember the N-gram example?



- **Embeddings reflect cultural bias!**
  - vector(man) – vector(woman) ≈ vector(computer programmer) - vector(homemaker)
  - Male names associated more with math, female names with arts
  - Words for specific ethnicities associated with stereotypes
  - Embeddings for competence adjectives are biased toward men → Bias slowly decreasing

- Embeddings can be used as historical tool to study bias
  - e.g., The cosine similarity of embeddings for decade X for occupations to male vs female names is correlated with the actual percentage of females and males in that occupation in decade X

- William L. Hamilton, Kevin Clark, Jure Leskovec, Dan Jurafsky, SocialSent: Domain-Specific Sentiment Lexicons for Computational Social Science, https://nlp.stanford.edu/projects/socialsent/
- Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In Advances in Neural Information Processing Systems, pp. 4349-4357. 2016.
- Garg, Nikhil, Schiebinger, Londa, Jurafsky, Dan, and Zou, James (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. Proceedings of the National Academy of Sciences, 115(16), E3635–E3644
- Caliskan, Aylin, Joanna J. Bruson and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. Science 356:6334, 183-186.

Embeddings: Summary

Embeddings → **Vector representations of words that encode meaning**

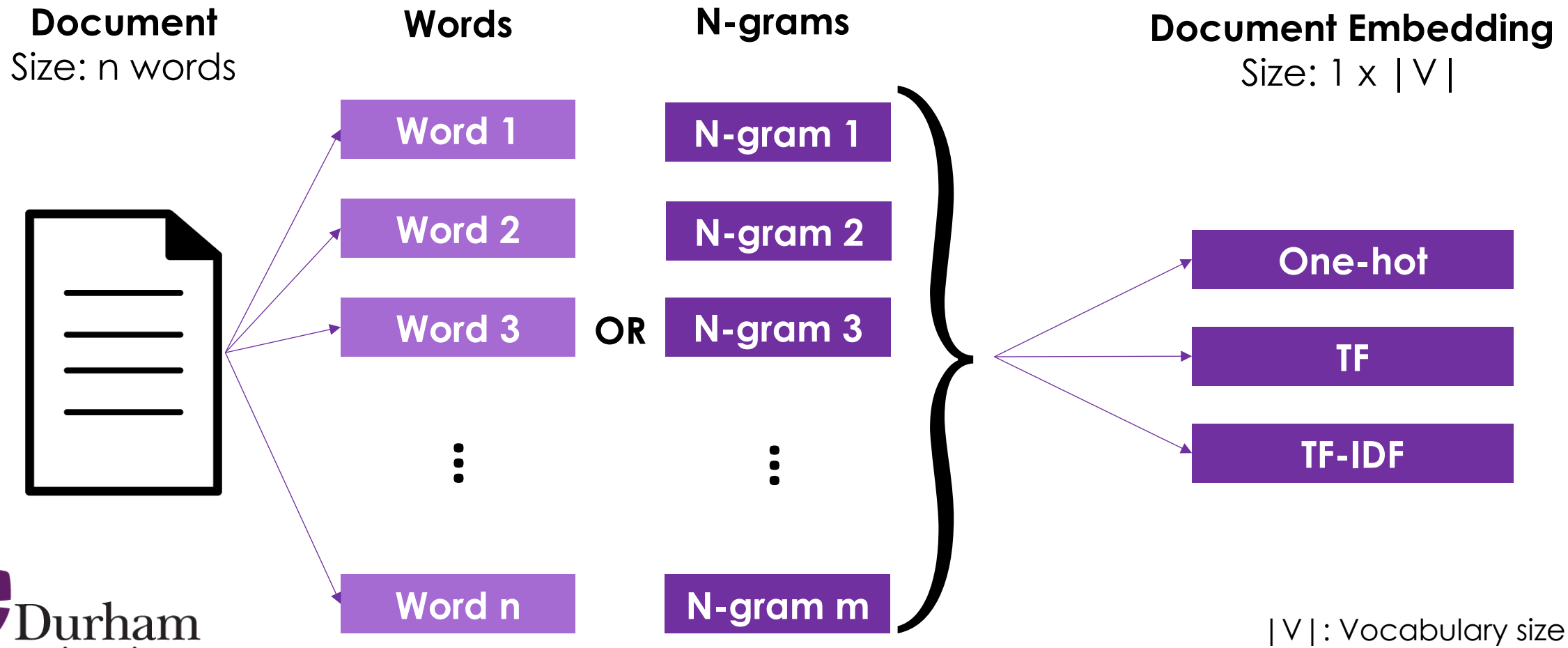
- More refined representation than a sequence of characters
- Very effective in modelling similarity/analogy → Use cosine distance
- Sparse models (One-hot, TF, TF-IDF, etc.)
- Dense models (word2vec, GloVe, etc.)
- **Work well in practice!**
- Remember that they can encode cultural/dataset bias

Document embeddings

- **What if we wanted to represent whole documents as vectors for machine learning applications?**
  - Topic modelling
  - Sentiment analysis
  - Spam detection
  - Text classification
  - Social media post analysis
  - Product review analysis
  - …
- **Challenges**
  - Representative information
  - Support high classification accuracy
  - Maintain accuracy across all lengths of documents

Durham University

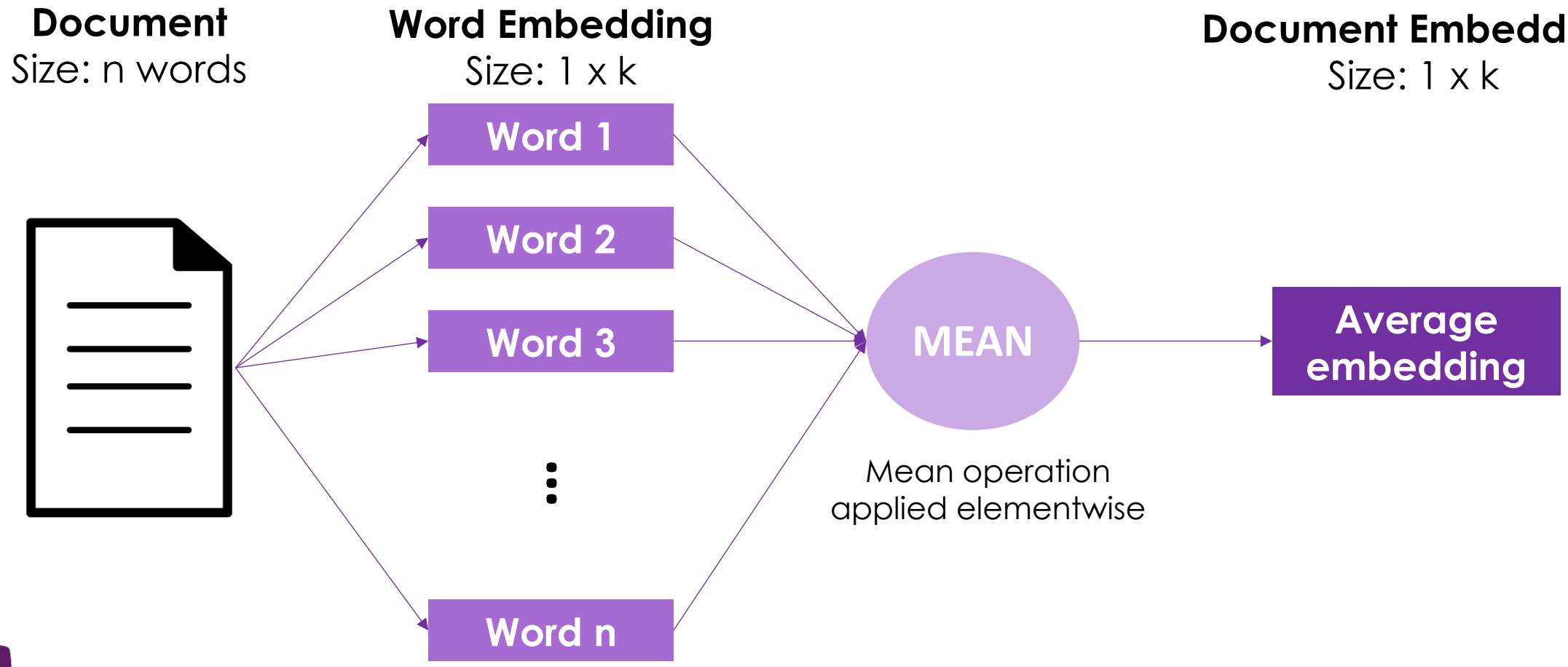
Document embeddings: One-hot / TF / TF-IDF / N-gram



|V|: Vocabulary size

Durham University

Document embeddings: Averaging



Durham University

Document embeddings: Fixed length (I)

Document
Size: n words

Word Embedding
Size: 1 x k

Word 1
Word 2
Word 3
⋮
Word n

n vectors → Size: n x k

Word 1 | Word 2 | Word 3 | ... | Word n

3 vectors → Size: 3 x k

Word 1 | Word 2 | Word 3 | ... | Word n

n+1 vectors → Size: (n+1) x k

Word 1 | Word 2 | Word 3 | ... | Word n | Padding

n+m vectors → Size: (n+m) x k

Word 1 | Word 2 | Word 3 | ... | Word n | Padding | ... | Padding

Concatenate word embeddings!

Durham University

Document embedding: Fixed length (II)

- Fixed length is experimental
- Very short length → Poor performance (discards valuable information)
- Very long length → More computationally expensive
- **Balance between the two needed!**
- Look at the statistics of the training data:
  - **Average** number of words per document in training dataset
  - **Maximum** number of words per document in training dataset
  - **Median** number of words per document in training dataset

Document embeddings: Summary

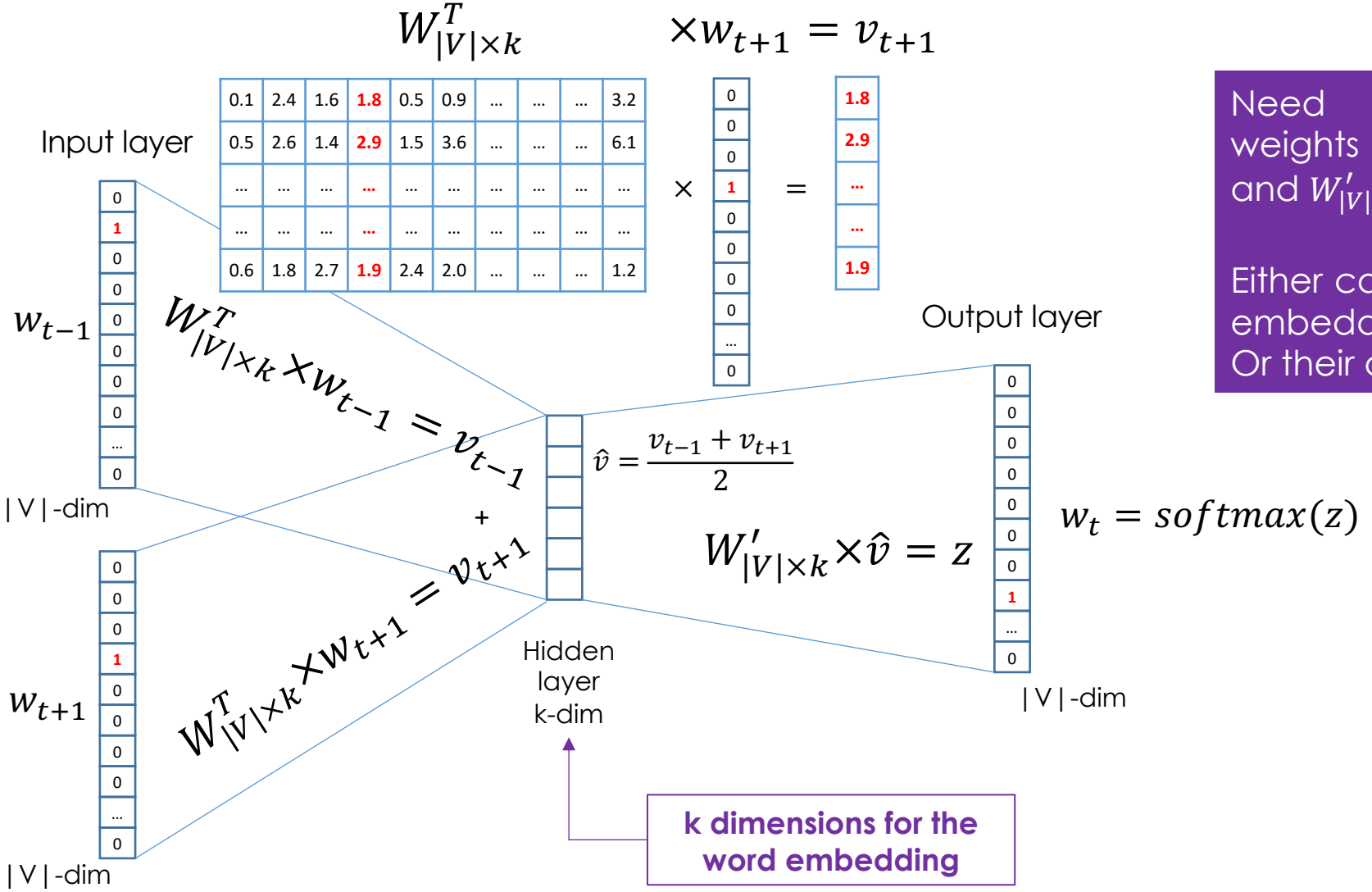
- One-hot / TF / TF-IDF –based
  - Sparse vectors
  - High dimensionality
  - Fixed size (typically $|V|$) regardless of document length

- Word embeddings-based
  - Dense vectors
  - High dimensionality
  - Fixed size (k) for averaging approach
  - Fixed size (m) for fixed length approach but may require padding and lead to information loss depending on m and document size

Questions?

Embeddings: word2vec (VI)

CBoW



One-hot encoding of words w_{t-1} and w_{t+1}

Need to compute weights matrices $W_{|V|\times k}$ and $W'_{|V|\times k}$

Either can be used as the embedding.
Or their average!

Durham University