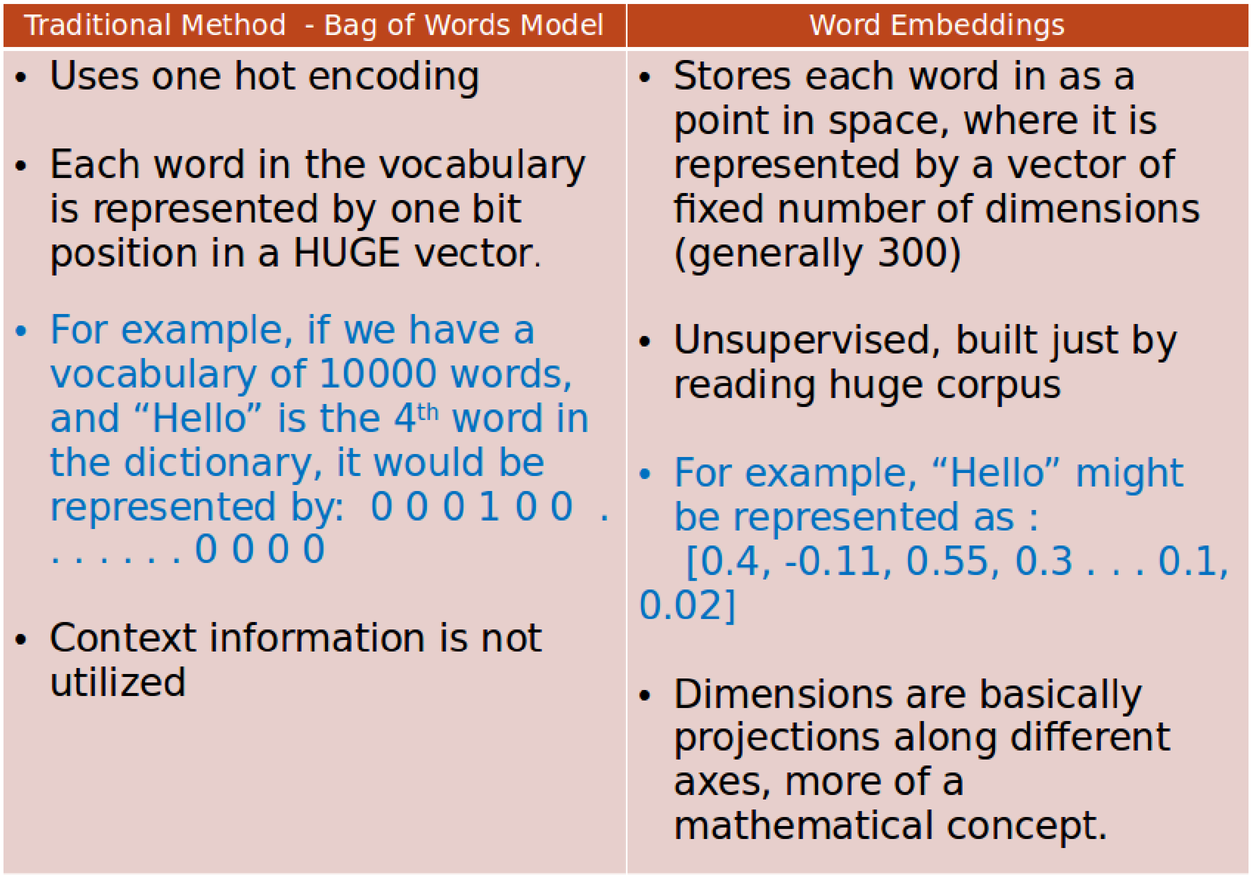
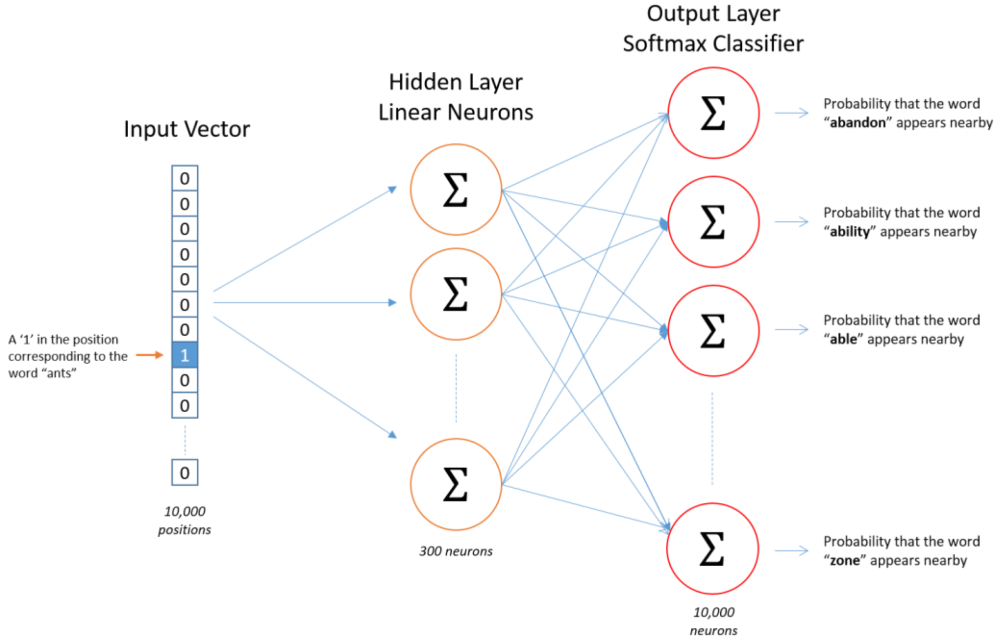
Word Representation

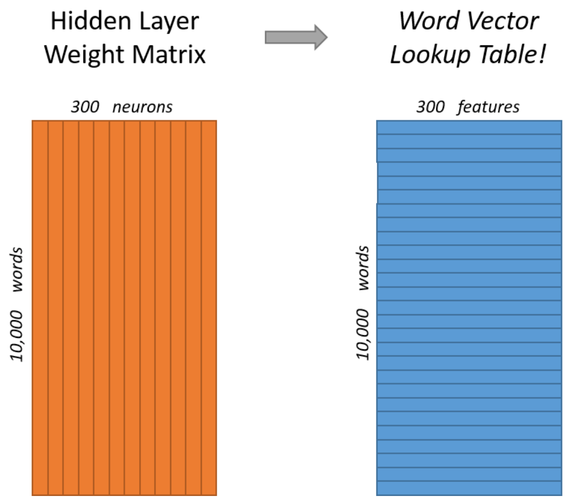


| Traditional Method - Bag of Words Model | Word Embeddings |
|---|---|
| • Uses one hot encoding<br>• Each word in the vocabulary is represented by one bit position in a HUGE vector.<br>• For example, if we have a vocabulary of 10000 words, and “Hello” is the 4th word in the dictionary, it would be represented by: 0 0 0 1 0 0 . . . . . . . 0 0 0 0<br>• Context information is not utilized | • Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300)<br>• Unsupervised, built just by reading huge corpus<br>• For example, “Hello” might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02]<br>• Dimensions are basically projections along different axes, more of a mathematical concept. |

Word2Vector(Google)

- Skip-gram Neural Network Model


Output Layer
Softmax Classifier
Hidden Layer
Linear Neurons
Input Vector
0
0
0
0
0
0
0
1
0
0
0
A ‘1’ in the position corresponding to the word “ants”
10,000 positions
Σ
300 neurons
Probability that the word “abandon” appears nearby
Probability that the word “ability” appears nearby
Probability that the word “able” appears nearby
Probability that the word “zone” appears nearby
10,000 neurons



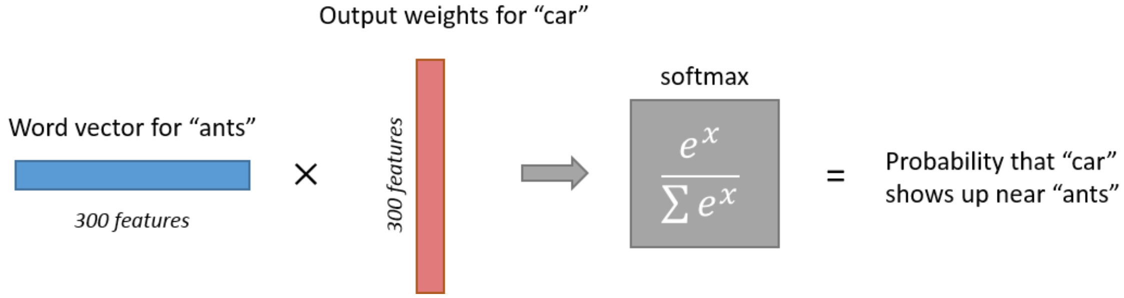
- Hidden Layer:
  - operating as a lookup table. The output of the hidden layer is just the "word vector" for the input word.

- The end goal :
  - learn this hidden layer weight matrix

- One hot vector :
  - Will effectively select the matrix row corresponding to the '1'.

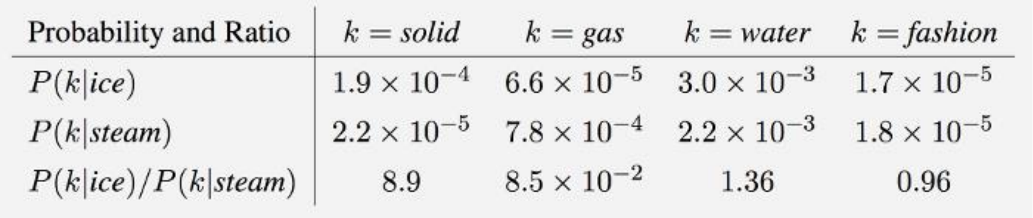
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

- Build a vocabulary of words from training documents.
- Input: a word like "ants" as a one-hot vector.
- Output layer: a softmax regression classifier.
- Output of the network: a single vector containing, for every word in our vocabulary, the probability that each word would appear near the input word.



## Global Vector Representations (GloVe) (Stanford)

- GloVe is an unsupervised learning algorithm for obtaining vector representations for words.
- Main idea: uses ratios of co-occurrence probabilities, rather than the co-occurrence probabilities themselves.



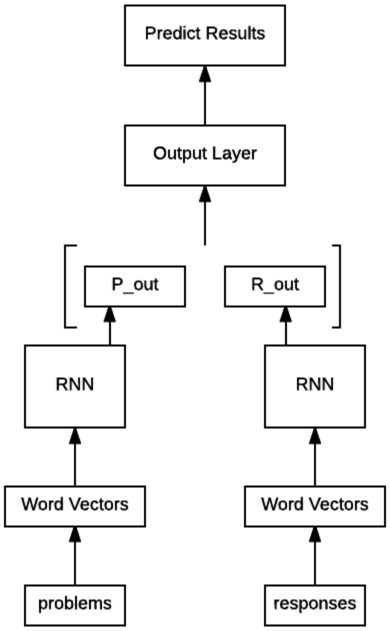
| Probability and Ratio | $k = solid$ | $k = gas$ | $k = water$ | $k = fashion$ |
|---|---|---|---|---|
| $P(k\|ice)$ | $1.9 \times 10^{-4}$ | $6.6 \times 10^{-5}$ | $3.0 \times 10^{-3}$ | $1.7 \times 10^{-5}$ |
| $P(k\|steam)$ | $2.2 \times 10^{-5}$ | $7.8 \times 10^{-4}$ | $2.2 \times 10^{-3}$ | $1.8 \times 10^{-5}$ |
| $P(k\|ice)/P(k\|steam)$ | 8.9 | $8.5 \times 10^{-2}$ | 1.36 | 0.96 |

- Because these ratios can encode some form of meaning, this information gets encoded as vector differences as well. For this reason, the resulting word vectors perform very well on word analogy task.

## Recurrent Neural Networks (RNN) Model

- Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP tasks.

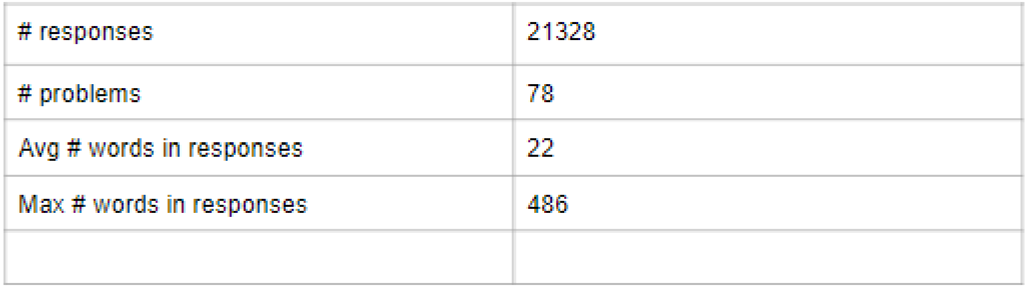
- The architecture of our model:
    - Use GloVe word embedding set to transfer “problems” and “responses” to word vectors.
    - The generated word vectors are 300 dimensional.



## Dataset from ASSISTment

- We obtain our dataset from ASSISTment System.
- Selection criteria: The number of Graded student responses is larger than 200 per problem.
- Final dataset includes:
    - Items: 21328
- The grades of all students' responses are distributed as 0, 0.25, 0.5, 0.75 and 1.
- We divide the grades into two classes:
    - 0-0.5 as class 0
    - 0.5-1 as class 1

## Dataset



| | |
|---|---|
| # responses | 21328 |
| # problems | 78 |
| Avg # words in responses | 22 |
| Max # words in responses | 486 |
| | |

## Training and Testing Results

- Training dataset:  random taking 80% from whole dataset
- Testing dataset: the remaining 20% of whole dataset
- Testing results are compared with Majority Class Model.

| Model | Test accuracy | AUC |
|---|---|---|
| Majority class | 0.63 | 0.5 |
| RNN | 0.79 | 0.85 |