Chapter 4
Word Embeddings and Recurrent NNs

4.1 Word Embeddings for Language Models

- Variable $E_{1,n} = (E_1 \dots E_n)$
  - E_i = a word
- Value $e_{1,n}$
  - E.g. $e_{1,6}$ = (We live in a small world)
    e_1 = We, e_2 = live, …
- Probability Model

$$p(\text{We live in a small World}) = p(\text{We})p(\text{live}|\text{We})p(\text{in}|\text{We live})\cdots$$

  - Generally

$$p(E_{1,n} = e_{1,n}) = \prod_{j=1}^{j=n} p(E_j = e_j | E_{1,j-1} = e_{1,j-1})$$

- Tokenization
  - Breaking the strings into a sequence of words
  - Vocabulary set V
    - Punctuation(final period .) is a word
    - *UNK* is unknown words
      - E.g. “132,423” in the sentence “The population of Providence is 132,423.”

- Penn Treebank(PTB)
  - news articles from the Wall Street Journal.
  - Files
    - ptb.train.txt, ptb.test.txt, ptb.valid.txt
  - Train data
    - 929589 words
  - Test data
    - 82430 words
  - Vocabulary set V
    - 10000 words

aer banknote berlitz calloway centrust cluett fromstein gitano guterman hydro-quebec ipo kia memotec mlx nahb punts rake regatta rubens sim snack-food ssangyong swapo wachter
pierre <unk> N years old will join the board as a nonexecutive director nov. N
mr. <unk> is chairman of <unk> n.v. the dutch publishing group

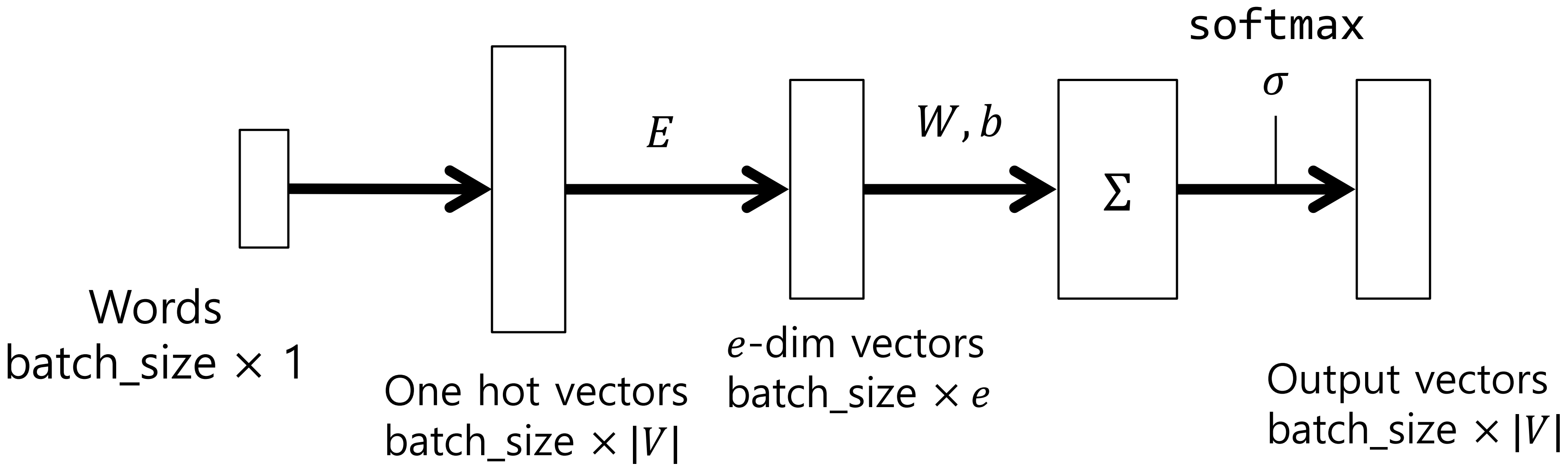
- Probability Model
  - Bigram model

$$p(E_{1,n} = e_{1,n}) = p(E_1 = e_1) \prod_{j=2}^{j=n} p(E_j = e_j | E_{j-1} = e_{j-1})$$

  - Sentence padding
    - Put a imaginary word “STOP” at the beginning of every sentence

$$p(E_{1,n} = e_{1,n}) = \prod_{j=1}^{j=n} p(E_j = e_j | E_{j-1} = e_{j-1})$$

- Neural Network
    - Word embedding
        - word → index → one hot vector → e-dimensional vector
        - $e = 20,\ 100$ or 1000



- Words embedding
  - Cosine similarity
    - $\cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|}$
    - Cosine of angle between $\boldsymbol{x}$ and $\boldsymbol{y}$

Word Numbers	Word	Largest Cosine Similarity	Most Similar
0	under		
1	above	0.362	0
2	the	–0.160	0
3	a	0.127	2
4	recalls	0.479	1
5	says	0.553	4
6	rules	–0.066	4
7	laws	0.523	6
8	computer	0.249	2
9	machine	0.333	8

4.2 Building Feed-Forward Language Models

```
inpt = tf.placeholder(tf.int32, shape=[batchSz])
answr = tf.placeholder(tf.int32, shape=[batchSz])
E = tf.Variable(tf.random_normal([vocabSz, embedSz], stddev = 0.1))
embed = tf.nn.embedding_lookup(E, inpt)
```

```
xEnt = tf.nn.sparse_softmax_cross_entropy_with_logits(logits=logits,labels=answr)
loss = tf.reduce_sum(xEnt)
```

4.3 Improving Feed-Forward Language Models

1. Adding layer

2. Trigram model

```
embed2 = tf.nn.embedding_lookup(E, inpt2)
both = tf.concat([embed,embed2], 1)
```

4.4 Overfitting

- Regularization
  - Early stopping
  - Dropout
  - L2 regularization

Epoch	1	2	3	4	5	6	7	10	15	20	30
Train	197	122	100	87	78	72	67	56	45	41	35
Dev	172	152	145	143	143	143	145	149	159	169	182

Figure 4.3: Overfitting in a language model

Epoch	1	2	3	4	5	6	7	10	15	20	30
Dropout	213	182	166	155	150	144	139	131	122	118	114
L2 Reg	180	163	155	148	144	140	137	130	123	118	112

Figure 4.4: Language-model perplexity when using regularization

4.5 Recurrent Networks