Natural Language Processing (CS5803)

Lecture 4
(More on word embeddings)

FastText

- Enriching Word Vectors with Subword Information
- Character n-gram based model
- Incorporates information about word structure
- Words are represented as bag of character n-grams
- Example: n=3, word = where. Character n-grams:
  - <wh, whe, her, ere, re>
  - <where>
- n can be chosen by developer/architect
- Multiple n values can be used
  - For example, $n \in \{3, 4, 5, 6\}$

FastText

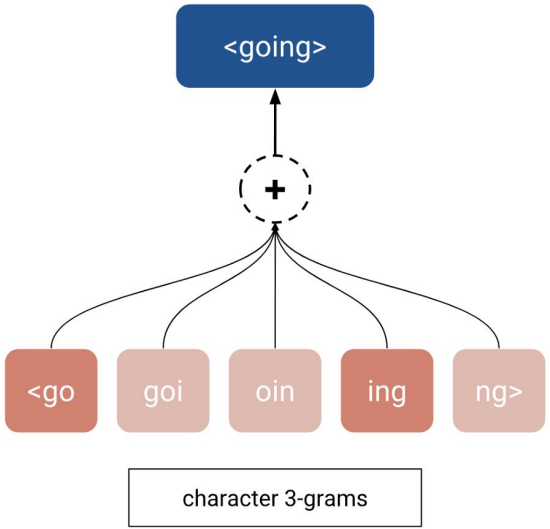
- w= ‘where’, n= 3
- G_w={'<wh', 'whe', 'her', 'ere', 're>" ,'<where>' }

$$S(w, c) = \sum_{g \in G_w} z_g^T v_c$$

- The above score function is used while training the model

$$\sum_{t=1}^{T} \left[\sum_{c \in \mathcal{C}_t} \ell(s(w_t, w_c)) + \sum_{n \in \mathcal{N}_{t,c}} \ell(-s(w_t, n)) \right]$$

Sub-word Vectors to Main Vector



Getting the vector for ‘going’

```
int32_t Dictionary::find(const std::string& w, uint32_t h) const {
  int32_t word2intsize = word2int_.size();
  int32_t id = h % word2intsize;
  while (word2int_[id] != -1 && words_[word2int_[id]].word != w) {
    id = (id + 1) % word2intsize;
  }
  return id;
}
```

Concept of hashing is used to find the embeddings of the char n-grams,

```
import fasttext

# Skipgram model :
model = fasttext.train_unsupervised('data.txt', model='skipgram')

# or, cbow model :
model = fasttext.train_unsupervised('data.txt', model='cbow')
```

Learning FastText representations