Natural Language Processing

Lecture 10: Classification 2
Features and Embeddings
What are the Features?
Sample Representation

• List of features → Category
• Category: “small” finite discrete # of classes
  ▼ E.g. LanguageID, POS tag, Movie genre
• Features: list of real numbers
  ▼ All samples must have same # of features
How to represent words

• Samples are movie reviews:
  ∼A few sentences of text
  ∼A class: 1-5 (1 very bad, 5 very good)
• Class: simple int
• Features: ???
  ∼Encode first n words (?)
How to represent words

• # of words
• # of sentences
• # of exclamations points!!!!
• Does “good” appear?
• Does “bad” appear?
Discrete Classes

• Categories to numbers
  ∼ Business [1,0,0]
  ∼ Sports [0,1,0]
  ∼ Entertainment [0,0,1]
  ∼ “one hot” representations

• Usually better than
  ∼ Business → 1
  ∼ Sports → 2
  ∼ Entertainment → 3
How to represent words

• Decide on vocabulary size + _other_
  ~Occurrence of word
  ~Array of vocab size: set to 1 if word appears
    • (or set to # of occurrences of word)
  ~Vocab should be most frequent/relevant words in corpus
    • Including very high frequency words?
    • Only content words?
    • Only words appearing more than once?
How to represent words

• One big vector for whole movie review
  ~Lots of zeros and few ones
  ~Might be 1000, 10,000 wide (or more)

• Often called “bag of words”
  ~Not care about word order
  ~Not care about # of occurrences of word
  ~Same length vector independent of length of review
Bag of Words

• Reviews are “similar” if vectors are similar
  ~ Similar means similar word distribution
  ~ e.g. simple difference, edit difference, cos similarity

• But
  ~ “I love the film” equally different from
  ~ “I hate the film” or
  ~ “I like the film”
Bag of Words

• Word similarity (“love” vs “hate” vs “like”)
  ~ Need not just be binary representation

• Contextual effects (“good” vs “not good”)
  ~ Need longer context
  ~ Can add bi-gram feature to vector
  ~ A vector with value for each bi-gram
Word Differences

• “like” and “love” more similar than
• “like” and “hate”
• Sparse vector treat distance the same
• Word Embeddings
  ~ Dense (not sparse) representations
  ~ Distance metrics more “meaningful”
  ~ Do dimensions in word embeddings
  ~ mean something? (maybe/maybe not)
Word Embeddings

• Use existing pretrained library
  - Word2vec, GloVe, elmo/bert

• Train your own
  - Word2vec, skip-gram

• Consider:
  - Is your data like others?
  - Do you have enough examples?
  - Are there special meanings in your domain
Word Embeddings

• How long should the dense vector be?
  ~300? 768? 1000? floats/doubles

• We don’t really know
  ~It’s not the size of the space represented
  ~It’s if the dimensions found are useful

• Hard to implicitly control meaning in vectors
  ~Easy to explicitly do it,
    • concat: word, pos, dependency parent
Word Embeddings

• New embedding techniques
  ~ Word2Vec and GloVe were standard
  ~ “Everything is better with Bert”
  ~ BERT [Devlin et al 2019]
    • Contextualized word embedding with transformers
    • Give SOTA performance in 11 standard NLP tasks

• But better ones being developed (e.g. XLNet)
Sentence/Document Embeddings

• But we need a fixed sized vector for the doc
  ↘ So add up all the vectors
  ↘ So find the average of all the vectors
  ↘ So find the max of each value in vectors
  ↘ Do something else

  • Learn a representation from sequence of word embeddings (e.g. sequence model)
  • Train something on all documents
Too many words

• Contextualized word embeddings
  ~/Care about some context
• Could concat previous and next word vectors
• But it gets very big very quickly
  ~/Even with case folding
• POS is more limited size
  ~/e.g 45ish tags, smaller representation
  ~/Smaller number of contexts
Too Many Features

• If you have too many features
  ∨ Each sample has some unique combination
  ∨ Training works well, but no generalization

• How much is too much/too little?
  ∨ Depends
  ∨ Pretraining is good (usually, if in similar domain)
  ∨ Ask yourself if the system has the features you think are important for task
Summary

- Features (must) be numeric
- Convert discrete features to one-hot
- Sparse vs Dense word representations
- Bag of Words (bi-grams/tri-grams)
- Word Embeddings (dense)
  - Pretrained vs trained
- Are your features enough/not enough
- Does it work? When does it fail? Why?