CMSC 373 Artificial Intelligence Fall 2023 19-NNsForNLP

Deepak Kumar
Bryn Mawr College

1

Language Processing & Understanding

- Building systems to use and understand language are among AI's most difficult challenges:

  Language is inherently ambiguous
  What about the bill?

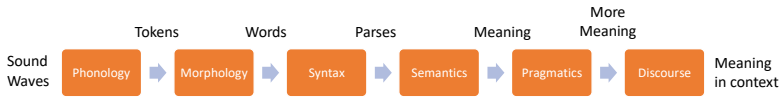
  Language understanding is deeply dependent on context
  "Fire!"

  Language understanding requires a lot of background knowledge
  The hamburger was rare.

2

2

Classic NLP/NLU Approaches



- Used linguistics knowledge:

  Syntax – the structure of sentences (grammar)

  S → NP VP
  NP → Pronoun | Proper-Noun | Det Noun
  VP → Verb | Verb NP | Verb NP PP | Verb PP
  etc.

  The cat drank the milk
  (**S** (**NP** (**Det** *The*) (**Noun** *cat*))
  (**VP** (**Verb** *drank*) (**NP** (**Det** *the*) (**Noun** *hat*))))

  Semantics – the meaning of a sentence (Logic, semantic nets, etc)
  in(cat34, hat21)

  Pragmatics – Using some kind of contextual knowledge

  It is cold in this room (i.e. Turn on the damn heat!)

  It is impossible to capture all the subtleties of language!

3

3

Statistical Approaches to NLP (1990s)

- Big data of text widely available
- Can ML/DL + big data help with NLU/NLP?

- Success in Speech Recognition (2012)

  Led to products like Google Assistant, Alexa, Siri, etc.

- Still not quite robust (90-95% accuracy in absence of noise)

  I love mousse. `I love moose.`
  The bareheaded man needed a hat. `The bear headed man needed a hat.`

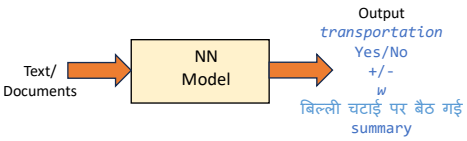
- The first 90% takes 10% of time, the last 10% takes 90% of the time!

4

4

Applications of NNs to NLP

- **Text Classification**
  What is the topic of this text?
- **Content Filtering**
  Does this text contain abuse?
- **Sentiment Analysis**
  Does this text sound positive or negative?
- **Language Modeling**
  What should be the next word in this incomplete sentence?
- **Translation**
  How would you say this in German?
- **Summarization**
  How would you summarize this text in one paragraph?



5

5

Requirements for using NNs for Text

- **Preparing/Preprocessing of text**
  Standardize (removing punctuation, convert to lowercase, etc.)
  Tokenize (splitting text into units (tokens): chars, words, groups of words, etc.
- **NNs require inputs to be numerical (a vector)**
  How to represent text as vectorized input?
- **Inputs to a NN have a fixed number of units (size of input layer)**
  How to make text/sentences conform to a fixed size input?
- **Word order in text/sentence is important**
  How to address the issue of word sequencing in NN?

6

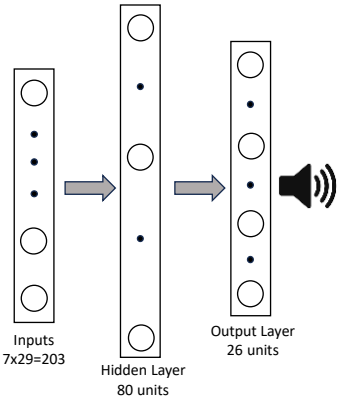
6

NETtalk (1987) – Text to Speech

- A classic backpropagation network
- **Inputs:** 7 groups of 29 units
  Each group is 1 character represented using **one-hot encoding**.
  29 chars: a..z, comma, period, space (word boundary)

```
a: [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
b: [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
...
z: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]
```

- **Outputs:** 26 units for phonemes
  The phonemes are input into a voice box
- A total of 18,629 parameters
- Used a 20,000 word vocabulary
- Demo: https://youtu.be/gakJlr3GecE
- Network generalized well to pronounce unseen words



7

7

Requirements for using NNs for Text

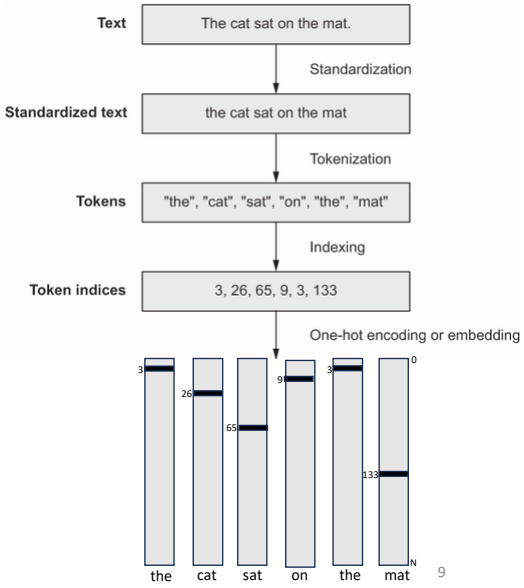
- **Preparing/Preprocessing of text**
  Standardize (removing punctuation, convert to lowercase, etc.)
  Tokenize (splitting text into units (tokens): chars, words, groups of words, etc.
- **NNs require inputs to be numerical (a vector)**
  How to represent text as vectorized input?
- **Inputs to a NN have a fixed number of units (size of input layer)**
  How to make text/sentences conform to a fixed size input?
- **Word order in text/sentence is important**
  How to address the issue of word sequencing in NN?

8

8

Text as Vectors: One Hot Encoding

- Create a fixed sized (say N = 20,000 words) vocabulary of the most frequent words in the text. Assign each word an index (1..N). Index 0 is reserved for out of vocabulary (OOV) words ("UNK").
- Vectorize: Use an input vector of size N (of 0/1s) for each word with a 1 in the index location of the word.

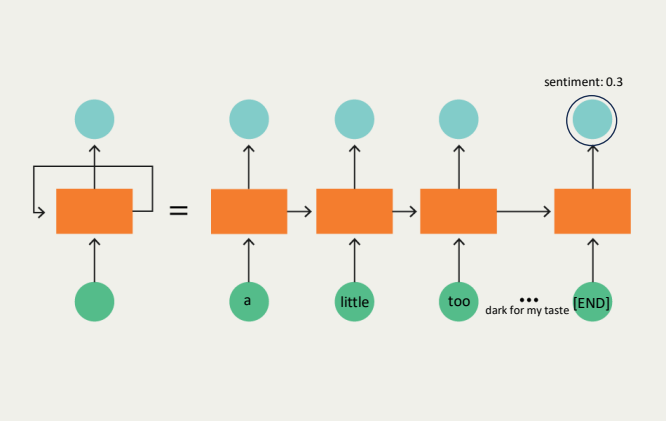


9

9

Recurrent Neural Networks

- Handle sequences of words
- Words (word vectors) are input one word at a time until the end of the sentence.
- The recurrent layer enables the network to remember the context of the sequence of words.
- The network encodes an input sentence [**Encoder Network**]



From: https://images.app.goo.gl/tVVcHFXkPpMWUnGg6

10

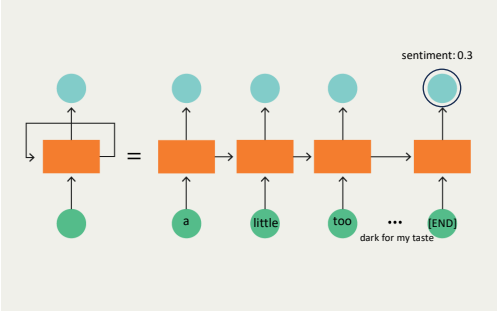
10

IMDB Movie Review Sentiment Classification

- 25,000 reviews for training, 25,000 for testing (50% positive and 50% negative)

```
The plot is heavy, and the sense of humor is largely missing.
A little too dark for my taste.
It felt as if the producers tried to make it as disturbing and horrific as they possibly could.
Temple of Doom's character development and humor is intensely subpar.
```

- Each review is truncated to 600 words Each input review is then 600 one-hot vectors (20,000 length). That is 12 million values for each review!
- Doable but very slow.
- Can we do better?



From: https://images.app.goo.gl/tVVcHFXkPpMWUnGg6

11

11

RNN Model with One-Hot Encoding

```
inputs = keras.Input(shape=(None,), dtype="int64")
embedded = tf.one_hot(inputs, depth=max_tokens)
x = layers.Bidirectional(layers.LSTM(32))(embedded)
x = layers.Dropout(0.5)(x)
outputs = layers.Dense(1, activation="sigmoid")(x)
model=keras.Model(inputs, outputs)
model.compile(optimizer="rmsprop",
              loss="binary_crossentropy",
              metrics=["accuracy"])
model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         [(None, None)]            0

tf.one_hot_6 (TFOpLambda)    (None, None, 20000)       0

bidirectional_4 (Bidirecti   (None, 64)                5128448
onal)

dropout_4 (Dropout)          (None, 64)                0

dense_2 (Dense)              (None, 1)                 65

=================================================================
Total params: 5128513 (19.56 MB)
Trainable params: 5128513 (19.56 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

After ~1.5 epochs (> 5 hours): Accuracy: 87.1%

12

12

Text/Words as Vectors
One Hot Encoding: Issues

- Early (prior to 2010) NNs used them with limited success.
- Results in too large an input vector: N-Dimensional (N=20000,30000)
- One-hot encoding results in a N orthogonal vectors in an N-dimensional **geometric space**.
- It is as if each word has no relationship with another word.
- No way to capture (**semantic**) relationships between words:

  I hated the movie.
  I abhorred the film.

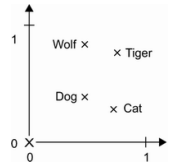
  I laughed out loud…
  I appreciated the humor…

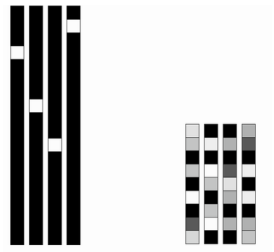
13

13

Word Embeddings – Semantic Space of Words

- **Geometric relationship** between two words (i.e the vector encoding) should reflect the **semantic relationship** between two words.
- Word embeddings are vector representations of words that map words into a structured geometric space.
- Word embeddings result in low-dimensional vectors.
- They can be learned from data!





14

14

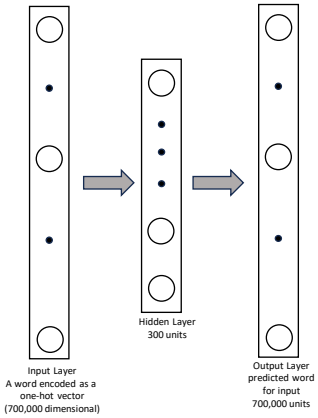
Word2Vec (2013)

- Developed at Google
- Used a NN to learn vectors for all vocabulary words (in Google News service corpus). N=700,000
- Discarded stop words (e.g. the, of, and, etc.)
- Created pairs of words:

  a man went into a restaurant and ordered a hamburger
  -> `man went into restaurant ordered hamburger`
  -> (man, went), (went, into), (into, restaurant),…
  (went, man), (into, went), (restaurant, into),…

- Train a NN to predict what words are likely to be paired with a given input word.

  Input: a word encoded as a one-hot vector (700000 dimensional).
  Output: a word prediction as a one-hot vector



15

15

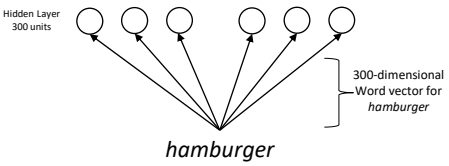
Word2Vec (2013)

- Once training is completed, extract the **learned weight vector** for any word in the vocabulary.
- The collection of word vectors for all the words is the learned **semantic space**.
- Enables NNs to do well for NLP tasks.
- Question: What do these word vectors actually capture? Does the "Semantic space" learned by the NN actually capture the semantic space?

  France: Spain, *Belgium*, *Netherlands*, *Italy*, etc.

  Hamburger: burger, *cheeseburger*, *sandwich*, *hot dog*, *taco*, and *fries*.

- See Word Embedding Demo: https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/



16

16

Word2Vec Biases

1. Man is to woman as computer programmer is to_____?
2. Woman is to man as computer programmer is to _____?
3. Man is to genius as woman is to _____?
4. Woman is to genius as man is to _____?

17

Word2Vec Biases

1. Man is to woman as computer programmer is to_home maker_?
2. Woman is to man as computer programmer is to _mechanical engineer_?
3. Man is to genius as woman is to _muse_?
4. Woman is to genius as man is to _geniuses_?

Challenge: How to de-bias word vectors??

"thought vectors": Instead of encoding words, encode entire sentences, paragraphs, documents?

18

Learning Word Embeddings

- There is no perfect word embedding that is applicable to any task
- Word embedding for different languages will be different
- Recognizing that word embedding for a specific task say, IMDB Movie Reviews, will look different from the word embedding for another task say, legal document classification.
- It is possible to learn a new embedding for every new task. Learning word embedding can be integrated into the NN's overall task.
- It is also possible to use a pre-trained word embedding. Word2Vec is one such embedding. GloVe (Global Vectors for Word Representation, Stanford University, 2014)

RNN Model with Embedding

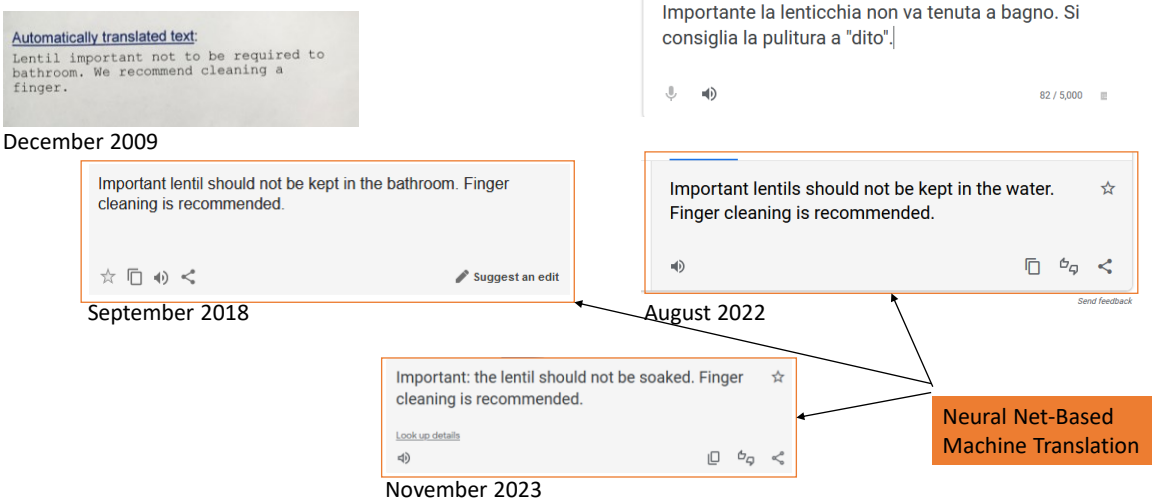
```
import tensorflow as tf

inputs = keras.Input(shape=(None,), dtype="int64")

embedded = layers.Embedding(input_dim=max_tokens,
                            output_dim=256,
                                mask_zero=True)(inputs)

x = layers.Bidirectional(layers.LSTM(32))(embedded)

x = layers.Dropout(0.5)(x)

outputs = layers.Dense(1, activation="sigmoid")(x)

model=keras.Model(inputs, outputs)

model.compile(optimizer="rmsprop",
                  loss="binary_crossentropy",
                  metrics=["accuracy"])

model.summary()
```

```
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, None)]            0

embedding_1 (Embedding)      (None, None, 256)         5120000

bidirectional_2 (Bidirecti   (None, 64)                73984
onal)

dropout_2 (Dropout)          (None, 64)                0

dense_2 (Dense)              (None, 1)                 65

=================================================================
Total params: 5194049 (19.81 MB)
Trainable params: 5194049 (19.81 MB)
Non-trainable params: 0 (0.00 Byte)
```

After 10 epochs (~2 hours): Accuracy: 98.7%, Validation Accuracy: 87%

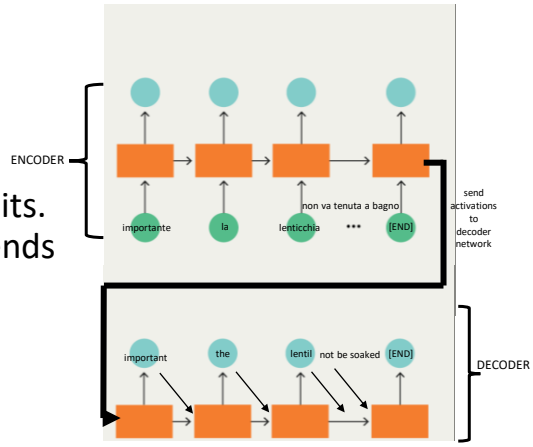
It is also possible to use pre-built word embeddings (like Word2Vec, GLoVe, etc.)

Machine Translation with NNs



Encoder & Decoder NNs

- NN models for Machine Translation
- Uses LSTM (Long Short Term Memory) units. As sentences get too long, the encoder tends to "forget"/lose memory. LSTMs fix this.
- >30 million human-translated pairs of sentences are used to train the network
- Networks also employ several improvements/tweaks.



Vocabulary

Encoding
De-Biasing
Decoding
Geometric Space
GLoVe
Language Modeling
LSTM
NETtalk
One-Hot Encoding
OOV Words
Pragmatics
RNNs
Syntax
Semantic Space
Semantics
Sentiment Analysis
Standardizing
Summarization
Text Classification
Thought Vectors
Tokenizing
Translation
Vectorizing
Word Embedding
Word2Vec

23

References

- F. Chollet: *Deep Learning with Python*, 2nd Edityion. Manning. 2021.
- M. Mitchell: *Artificial Intelligence: A Guide For Thinking Humans*, Farrar, Strouss, Giroux, 2019.
- M. Wooldridge: *A Brief History of Artificial Intelligence*. Flatiron Books, 2020.
- *Monte Carlo Tree Search*. Wikipedia. https://en.wikipedia.o rg/wiki/Monte_Carlo_tree_search (11/2023)
- *Word Embedding Demo*: https://www.cs.cmu.edu/~dst/WordEmbeddingDemo/

24