SYSC5405
Pattern Classification & Experiment Design

Introduction to Natural Language Processing

Prof James Green

jrgreen@sce.Carleton.ca

Systems and Computer Engineering, Carleton University

Topics to be Covered

- Natural Language Processing:
  - Document Representation:
    - Bag of words, tf-idf
    - Lemmatization and Stemming
    - Document embedding (doc2vec and others)
  - NLP tasks:
    - POS Tagging
    - Shallow Parsing
    - Dependency Parsing
    - Topic Modeling
    - Named Entity Extraction
    - Machine Translation
  - Learn the advanced topic of generating `Word Embeddings`

Key terms

- Word embeddings, word2vec, skip-gram, self-supervised learning, hierarchical softmax, negative sampling, NLTK (Natural Language Toolkit), bag of words (BoW), term frequency - inverse document frequency (tf-idf), lemmatization, stemming, document embedding, doc2vec, POS tagging, shallow parsing, dependency parsing, topic modeling, named entity extraction, machine translation.

NLP Bootcamp

- Intro to NLP Videos:
  - https://www.youtube.com/watch?v=f5bqPOkOJs4 (DS Dojo, 3min)
    - High-level overview of NLP goals
  - https://www.youtube.com/watch?v=d4gGtcobq8M (4:11)
    - Motivates how NLP can "unlock" unstructured data for improving safety on oil rigs
  - https://www.youtube.com/watch?v=5ctbvkAMQO4 (8:25)
    - Introduces: Tokenization, Stemming, Lemmatization, POS tagging, Named Entity Recognition, and Chunking
- NLP Tutorials:
  - Good intro tutorial:
    - https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32
  - Great walk-through of NLP topics:
    - https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72 (long)
  - Shorter article talking about NLP basics, links to courses, resources, videos, etc.:
    - https://algorithmia.com/blog/introduction-natural-language-processing-nlp

NLP Preprocessing

- **Tokenizer**
  - Split a sentence into individual words (tokens). Often splitting on spaces.
- **Remove Stop Words**
  - Words such as “a”, “the”, etc. are often removed prior to analyzing text
- **Lemmatization**
  - Replacing words with the **lemma** (i.e., root word from dictionary).
    - “saw”→ “saw” (noun) or “see” (verb)
  - vs. **Stemming**: Replacing words with the root by removing common pre-/suffixes
    - (e.g. “stopping”, “stopped” → “stop”, but “saw”→ ”s”?)
  - See https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

21c) Put the following NLP steps in the correct order:

Lemmatization/Stemming

Chunking

Tokenization

Named Entity Recognition

Remove stop words

POS Tagging

NLP: Representing an Entire Document

- **Bag of words**
  - Represent document as a (sparse) vector of word frequencies for that doc
  - Every document is represented by a vector of the same length
    - (length = # of valid words in *lexicon*)
  - Doesn’t capture order of words (context) nor relatively frequency
- Raw term frequencies: **`tf`**`(t,d)`
  - The number of times a term `t` occurs in a document `d`
- Term frequency-inverse document frequency: **`tf-idf`**
  - `tf-idf(t,d) = tf(t,d) · idf(t,d)`
  - $\mathtt{idf(t,d)=log(n_d/[1+df(t)])}$
    - `n`=total # docs; `df(t)`=#docs containing term `t`

21b) What is the tf-idf for the word "the" in the third sentence of the following corpus: 1: "The quick brown fox" 2: "Jumped over the lazy dog" 3: "Said the sleepy toad"

1

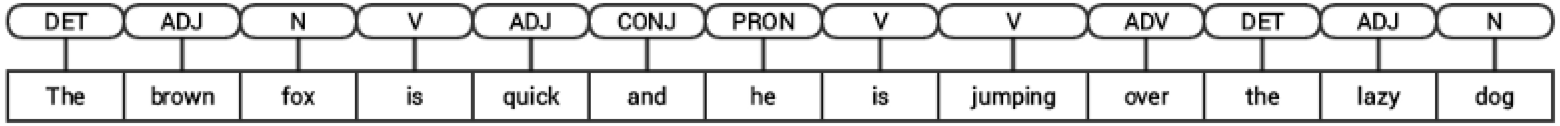
0

log(.75)

-1

NLP: Understanding statements

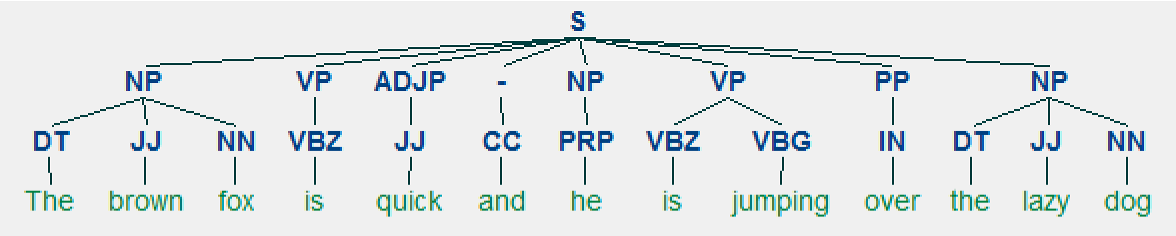
- Quick intro to various tasks:
  - https://youtu.be/fOvTtapxa9c?t=82 (until 5:24; speech rec/synthesis after that…)
    - Phrase structure rules, Parse Trees, Text generation, Chat bots
- Details of tasks:
  - https://www.kdnuggets.com/2018/08/understanding-language-syntax-and-structure-practitioners-guide-nlp-3.html
  - 1) Parts of Speech (PoS) Tagging



    - both `nltk` and `spacy` use the Penn Treebank notation for POS tagging.

NLP: Understanding statements

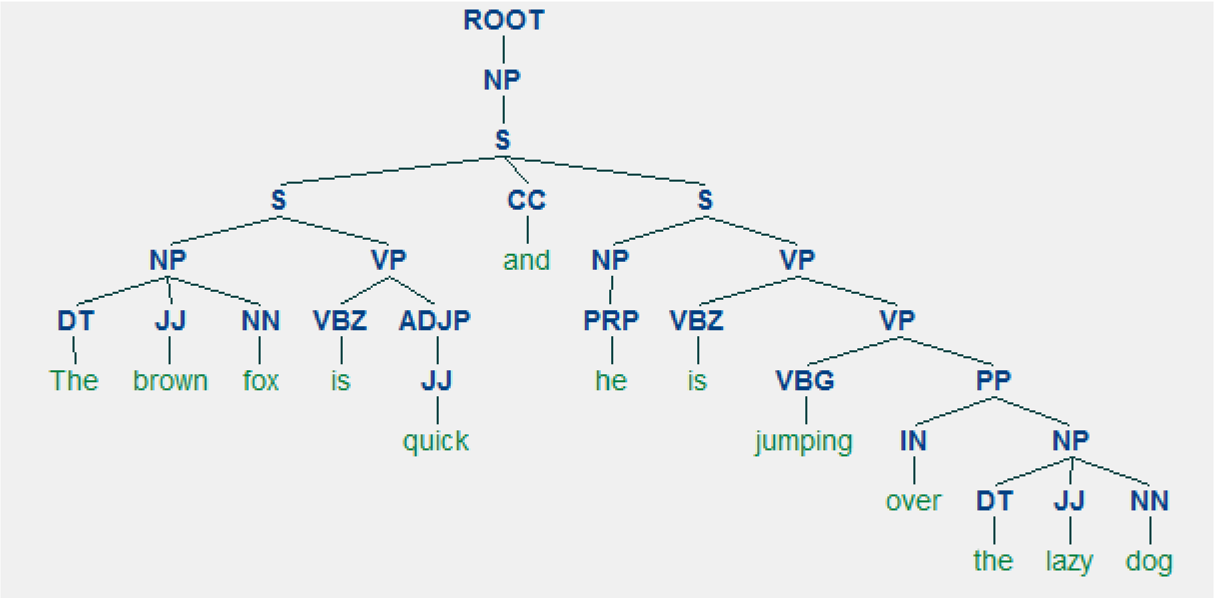
- Details of tasks: https://www.kdnuggets.com/2018/08/understanding-language-syntax-and-structure-practitioners-guide-nlp-3.html
    - 2) Shallow Parsing or Chunking



    - 5 major categories of phrases:
        - Noun phrase (NP; subject/object of verb)
        - Verb phrase (VP)
        - Adjective phrase (ADJP)
        - Adverb phrase (ADVP)
        - Prepositional phrase (PP)

NLP: Understanding statements

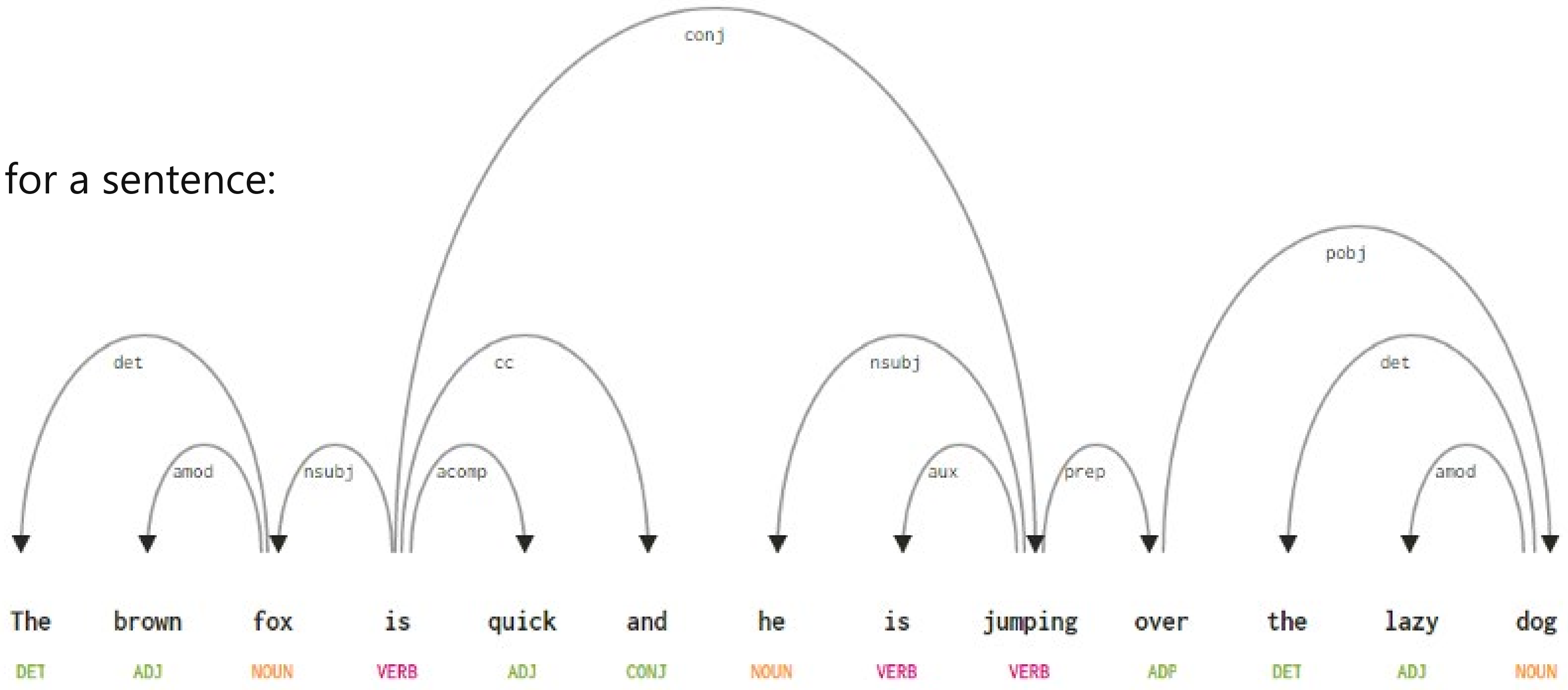
- Details of tasks: https://www.kdnuggets.com/2018/08/understanding-language-syntax-and-structure-practitioners-guide-nlp-3.html
  - 3) Constituency Parsing (often using Phrase Structure Rules)
    - Grammar governs hierarchy and ordering of the various constituents in the sentences.
      - Can use context-free grammar (CFG) or phrase-structured grammar



NLP: Understanding statements

- Details of tasks: https://www.kdnuggets.com/2018/08/understanding-language-syntax-and-structure-practitioners-guide-nlp-3.html
  - 4) Dependency Parsing
    - use dependency-based grammars to analyze and infer *both structure and semantic dependencies* and relationships between tokens in a sentence

A dependency parse tree for a sentence:



Advanced NLP Tasks

- Topic Modeling
  - https://stackabuse.com/python-for-nlp-topic-modeling/
  - Unsupervised learning. Identify "topics" from corpus of text.
    - E.g. cluster newspaper articles by "topic"
    - E.g. analyze doctor's notes from large collection of electronic medical records, identify "topics", then model each note according to which topics it discusses
  - 2 main approaches:
    - Latent Dirichlet Allocation (LDA)
      - Documents are probability distributions over latent topics
      - Topics are probability distributions over words
    - Non-Negative Matrix factorization
      - "a matrix **V** is factorized into (usually) two matrices **W** and **H**, with the property that all three matrices have no negative elements."
      - https://en.wikipedia.org/wiki/Non-negative_matrix_factorization

Advanced NLP Tasks

- Named Entity Extraction
  - Can use list of known entity names (e.g. “Google”, “Las Vegas”)
- Machine Translation
  - Automatic translation between languages.
    - Modern machine translation uses deep learning (e.g. seq2seq encoder-decoder nets)
  - Humorous incident occurred in the 1950s during the translation of some words between the English and the Russian languages.

    Here is the biblical sentence that required translation:

    “The spirit is willing, but the flesh is weak.”

    Here is the result when the sentence was translated to Russian and back to English:

    “The vodka is good, but the meat is rotten.”

    - https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32

Self-supervised Learning: Word Embeddings

- Goal: learn low-dimensional representation of "meaning" of a word
  - Word Embeddings
- `Skip-gram` Method (a type of `word2vec`):
  - `Skip-gram`: drop middle word in short string
    - Window size 7 (3 + 1 + 3). For example: "*finished reading the* **.** *on machine learning*"
    - Create infinite training samples from existing text (e.g. web)
  - Train network to predict surrounding words, based on input word
    - 1-hot encoding at input and output (~10K unique words?)
    - Have a bottleneck layer like an autoencoder
      - Learns embedding from skip-grams
  - Once trained, can apply a new word to partial network to generate embedding

[Word Embeddings: https://www.youtube.com/watch?v=gQddtTdmG_8](https://www.youtube.com/watch?v=gQddtTdmG_8)

Self-supervised Learning: Word Embeddings

Learning an embedding from skip-grams:

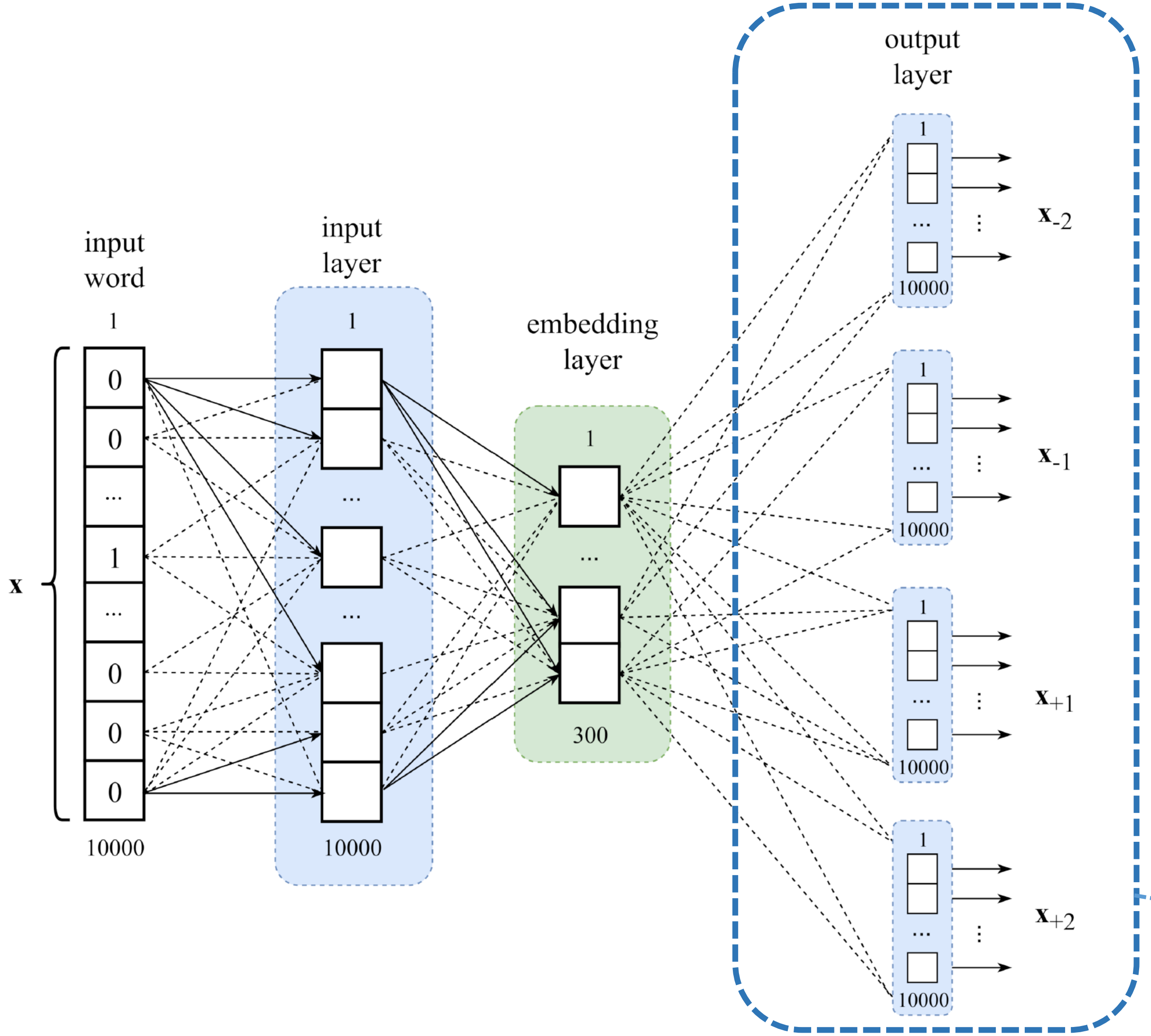


Figure 10.2: The skip-gram model with window size 5 and the embedding layer of 300 units.

Activation: softmax
Cost: neg log-likelihood

Advanced approaches:
- *Negative sampling*
- *Hierarchical softmax*

This layer removed once embedding is learned/trained

A. Burkov, The One Hundred Page Machine Learning Book

Sequence-to-sequence (*seq2seq*) learning (7.7)

- Encoder-decoder architecture → embedding

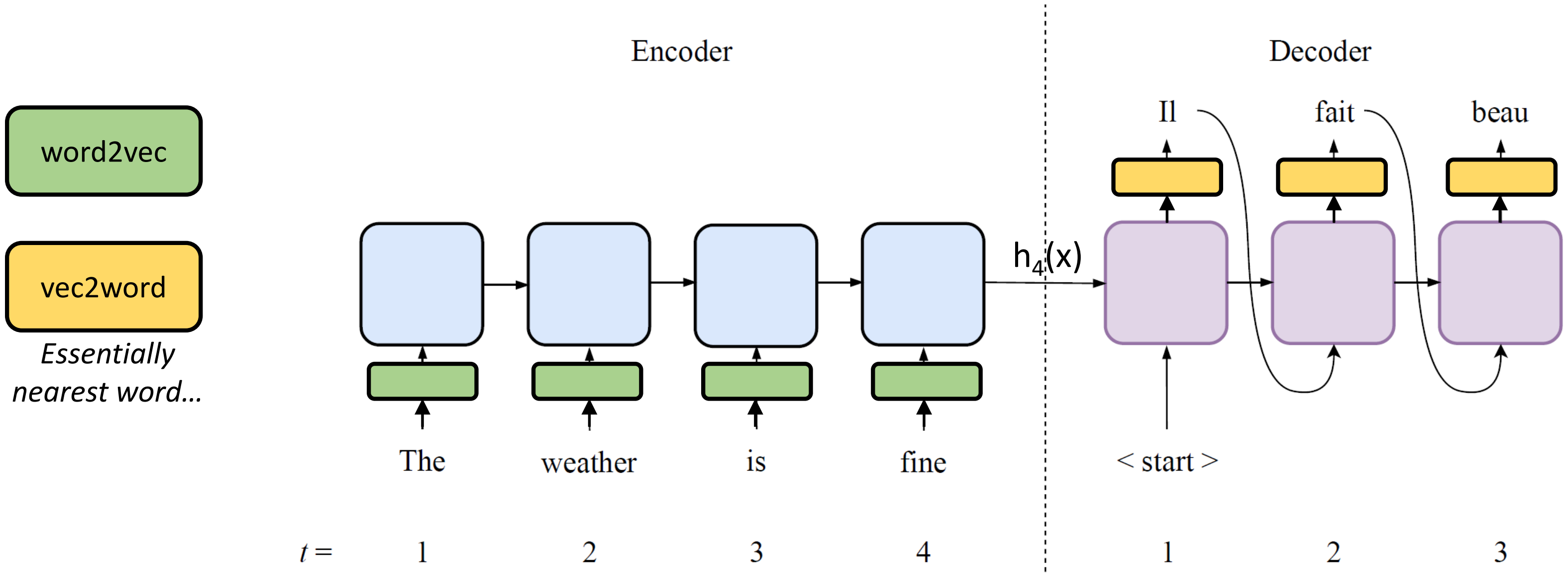


Figure 7.4: A traditional seq2seq architecture. The embedding, usually given by the state of the last layer of the encoder, is passed from the blue to the purple subnetwork.

Decoder trained to produce random (sensible) French sentences.

Decoder pre-conditioned on hidden state of encoder… Forces it to produce French sentence with same meaning as input English sentence.

A. Burkov, The One Hundred Page Machine Learning Book

21a) A skip-gram is:

A partial simile (e.g. “as red as a ___”)

A short statement with each k-th word removed

A text excerpt with the central word removed

A subsample of words from an article from the target “theme”

Now do it using Python!

- Using NLTK for sentiment analysis of Tweets:
  - https://github.com/jrgreen7/SYSC4906/blob/master/Lecture_21.ipynb
    - Siraj video: https://youtu.be/H6ii7NFdDeg?t=368

- Overview of NLP within NLTK for Python (with code):
  - https://towardsdatascience.com/introduction-to-natural-language-processing-for-text-df845750fb63