NLP Tasks and Applicaitons Using Word Embeddings End-to-end problem solving

Venelin Kovatchev

Lecturer in Computer Science

v.o.kovatchev@bham.ac.uk

Outline

- NLP Applications
- Using embeddings. Compositionality.
- End to end neural models

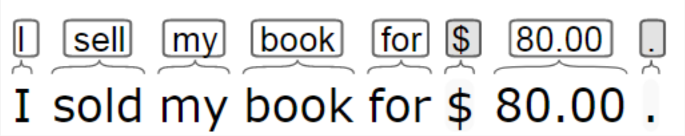
NLP Applications
Classical NLP Pipeline and Features

What we have learned so far

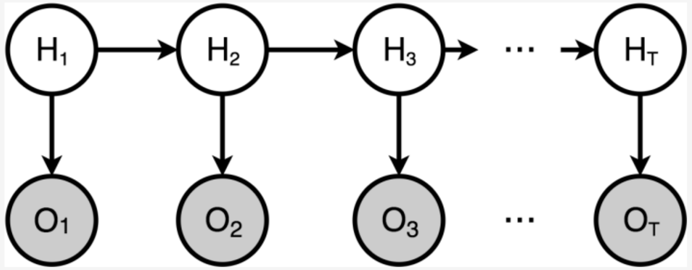
- Tokenization
- Pos tagging
- Chunking / Syntactic parsing / Dependency parsing
- Word co-occurrence and distributional semantic models
- Word embeddings

“Linguistic” tasks: Text tokenization and POS tagging

- Tokenization: output is a text segmented in tokens
  - Regular expressions, BPE



- POS tagging: output is a sequence of hidden states:
  - noun, verb, adjective
  - Hidden Markov Models (HMM



"Linguistic tasks: Chunking and constituent analysis

- Grouping words together based on their shared "function" in the text
- Find all groups that "function together" in the sentence
- I went to the movies with a friend who I know from high school.
  - **[I]** went to the movies with a friend who I know from high school. ← S
  - I **[went]** to the movies with a friend who I know from high school. ← action
  - I went **[to the movies]** with a friend who I know from high school.
  - I went to the movies **[with a friend who I know from high school]** .

word itself doesn't provide compositionality. (affect from other words)

one possible way to group them.

"Linguistic" tasks: The problem of syntax

Word order matters

"What combinations can we get with the constituents "dog", "human", and "bites"

- "Dog bites human" – (statistically) most common
- "Human bites dog" – meaningful, possible, but unlikely
- "Bites dog human", "Human dog bites", etc. – ungrammatical
- Same constituents, different rules -> different (im-)possible complex expression

Bag of words doesn't distinguish these.

“Linguistic” tasks: Full syntactic parsing

- “Colorless green ideas sleep furiously in love

Context - Free Grammar

S -> NP VP

NP -> A N

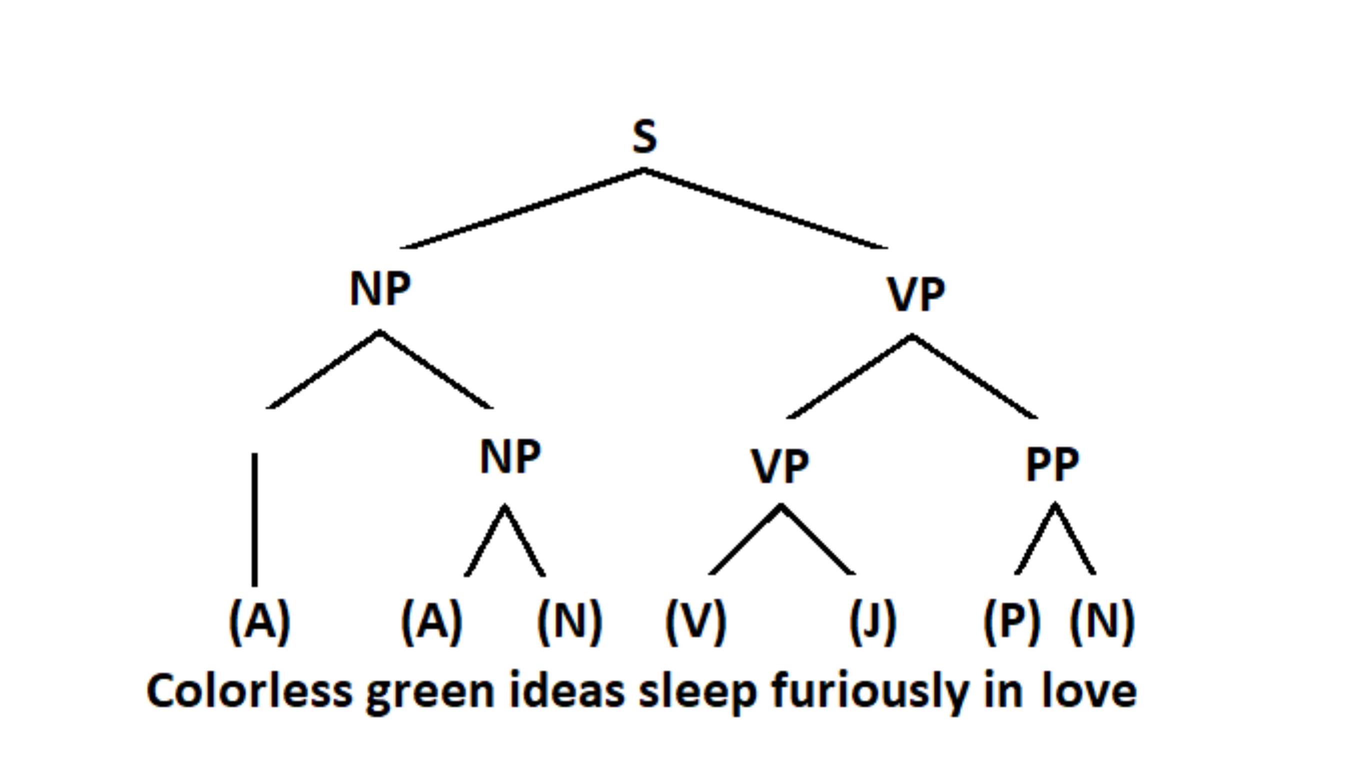
NP -> A NP

VP -> V J

VP -> VP PP

PP -> P N

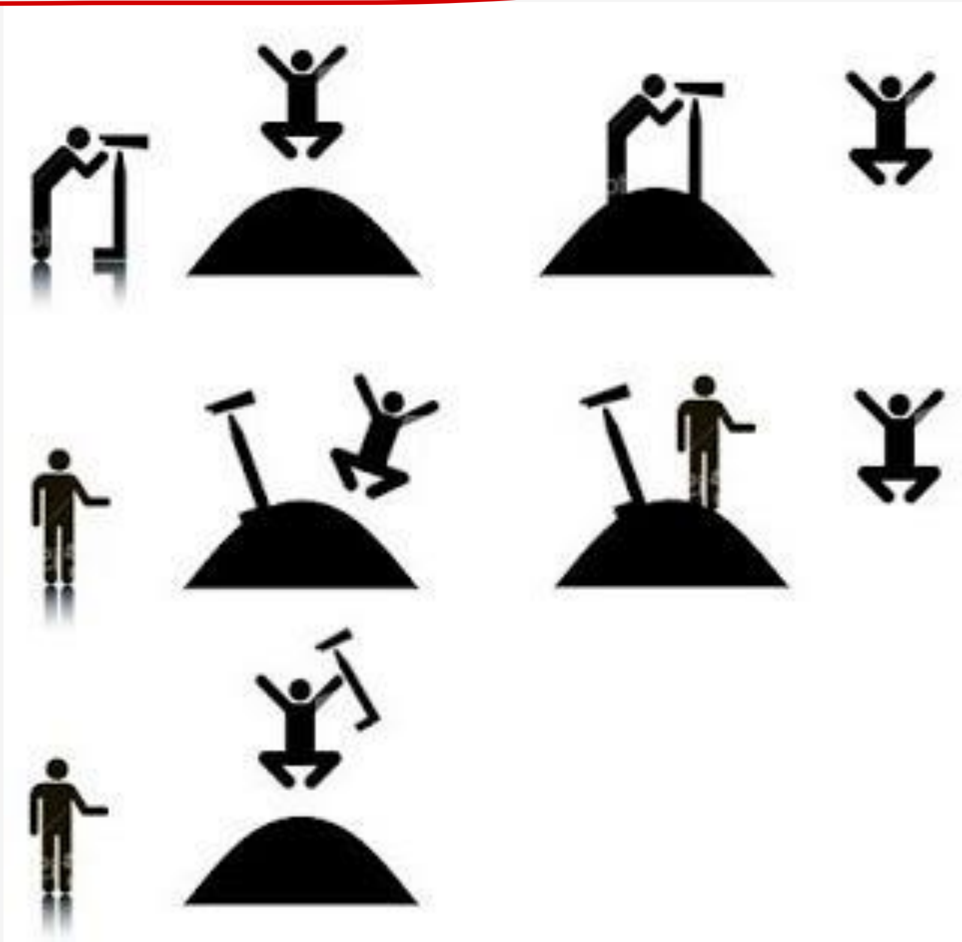
(NP -> N)



Syntactic ambiguity

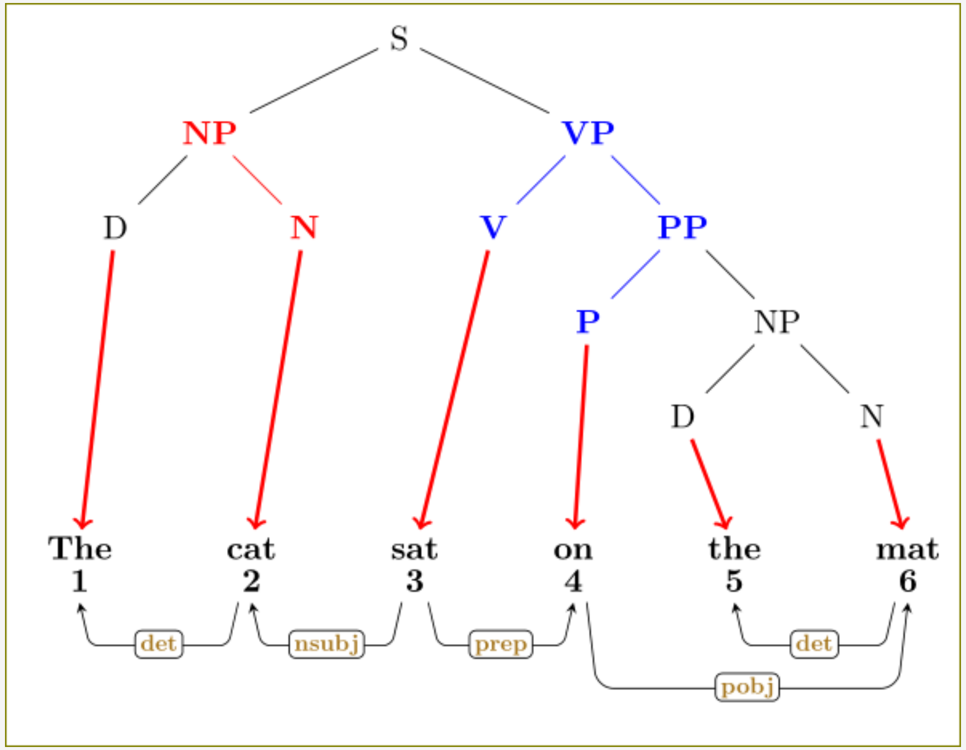
syntax, rules, word meanings all needed for NLP

- Sentences are often ambiguous
- How many interpretations for the following sentence:
  - "I saw a man in the park with the telescope"



“Linguistic” tasks: Dependency parsing

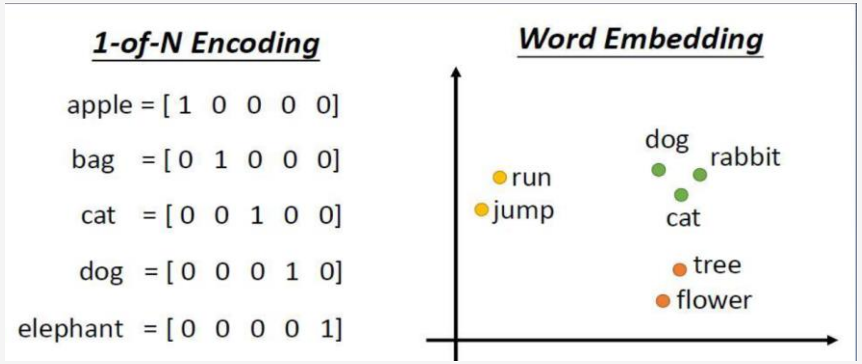
- The – (det) -> cat
- cat – (subj) -> sat
- on – (prep) -> sat
- mat – (pobj) -> on
- the – (det) mat



“Linguistic” tasks: word embeddings → training, might overfit.

but for most LM, it's still useful even with overfitting.

- What do the words “mean”?
- How can we “measure” the meaning?
- Distributional semantics:
  - the meaning is a function of the context



- Word vectors: one-hod, count based, embeddings

“Linguistic” NLP tasks

- Why do we need linguistic tasks?
  - To help computers make sense of language
    - Pre-processing and feature extraction
  - To help humans make sense of language
    - Linguistic and cognitive science experiments
    - For some problems “linguistic” NLP tasks are end goals

Linguistic tasks and machine learning

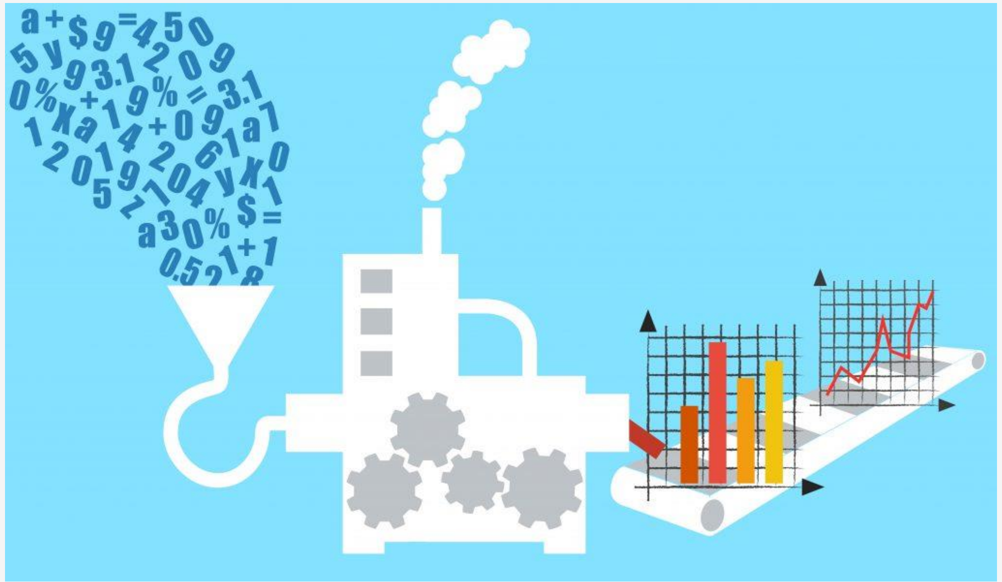
- Linguistic tasks are a goal on their own
- Linguistic tasks are tools in the (classical) NLP toolbox
- The bi-direction interaction between ML and linguistic tasks and data
  - Many linguistic tasks require ML
  - The output of linguistic analysis is used in practical applications
  - Word embeddings and transfer learning

Everyday tasks that use language / NLP

- Marketing
  - Sentiment analysis
  - Recommender systems
  - Content creation

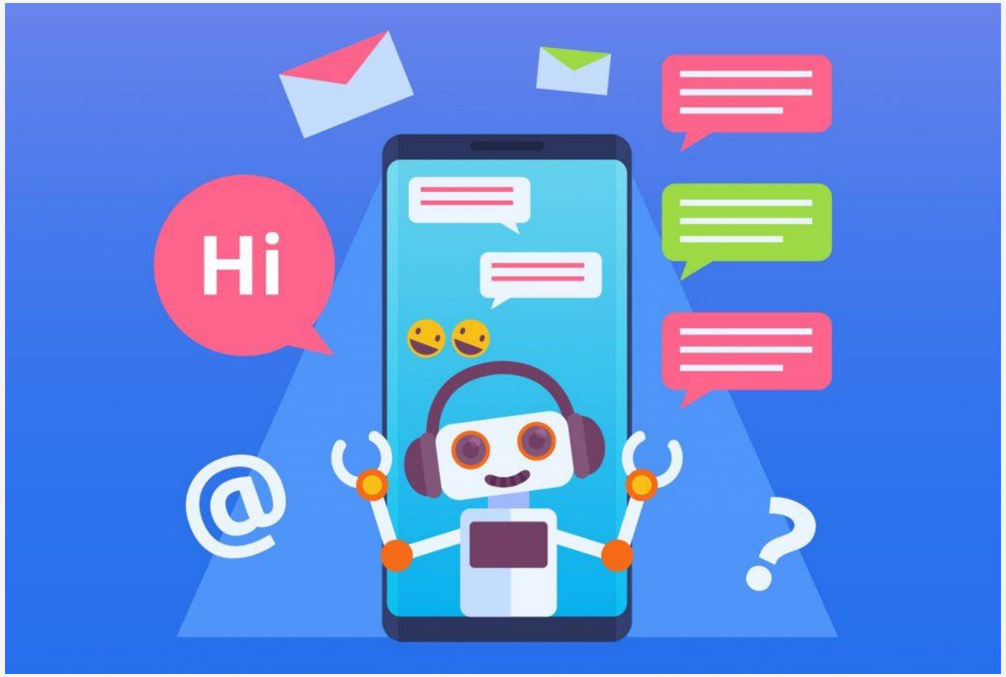
Everyday tasks that use language / NLP (2)

- News articles, social media, and search
  - Information extraction
  - Question answering
  - Inference and Fact checking
  - Content moderation



Everyday tasks that use language / NLP

- User experience and assistance
  - Conversational Agents / Chat bots
  - Machine translation
  - Personal assistants
  - Autocomplete, copilot



Extra-linguistic tasks

- Extra-linguistic tasks are not “internal” to language
  - Part-of-speech tagging vs book recommendation
- Language can be used to solve extra-linguistic tasks (in part)
- Rest of the module: taking a more practical direction
  - How do we solve problems using language
  - How do we define (and evaluate) problems and solutions
  - Feature-based solutions vs end-to-end solutions

Text classification
Feature Engineering

Machine learning and NLP – supervised and unsupervised NLP

- Types of machine learning problems: supervised, unsupervised, reinforcement
- Pop quiz: can you name ML problems of each of the three types?
- NLP problems are predominantly (represented as) supervised
  - Text classification: Sentiment analysis, Textual Inference, Fact checking, Toxic language detection
  - Text generation: Question answering, Chatbots, Machine translation

Text classification

- Observations are independent from each other (e.g. single tweet)
- Observations are preprocessed and fed into (a trained) classifier
- The classifier assigns the correct class-label to the observation
- Classes are discreet and often disjointed:
  - an email is either spam or ham, never both
- Different types of classification: binary, multi-class, multi-label

Extra-linguistic problems and feature engineering

- Historically, we approached extra-linguistic problems via feature engineering

- Feature engineering
  - Converting language data into relevant data that is easy to process for machines
  - A “faulty” translation from “human” to “computer”
  - Loses some (often a lot of) information
  - Requires a lot of human intervention

Feature engineering

- Analyze the problem, the input, and the desired outcome
- Explore existing resources and processing techniques

→ tokenisation, etc...

- Select the most relevant features and feature-extraction methods

Bag of words, N-grams etc... thinking of the word lengths.

- Empirically test what works best

Feature engineering

- Various features can be extracted from texts
- Bag-of-words (+ tf-idf)
- N-grams
- Part-of-speech tags
- Named entities
- Sentiment words
- Stop words
- Length

Feature engineering (example)

- Consider the task of sentiment analysis
- What features can you extract from the following examples?
- Are these tweets positive or negative?

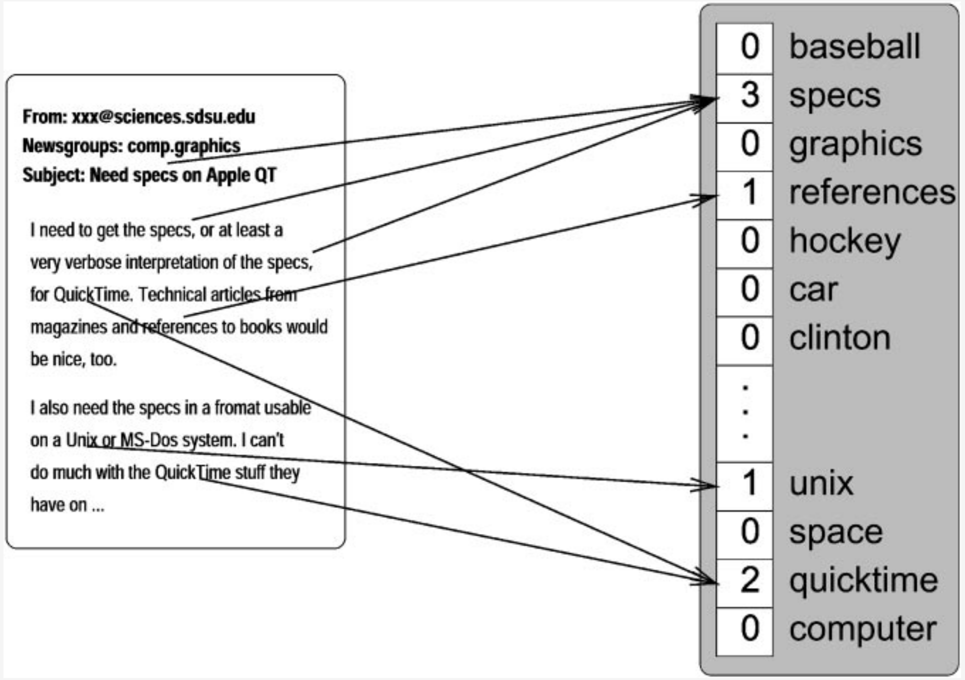
dictionary words, negative/positive.

1."🌟 Absolutely in love with my new headphones from SoundWave! The sound quality is top-notch, crystal clear, and the noise cancellation is a game-changer! 🎧 ✨ Highly recommend to all music lovers out there! #SoundWave #MusicLife 👍 😊"

2."Really disappointed with my purchase from QuickTech. The laptop crashes constantly and the battery life is a joke. 😡 💻 Worst customer service ever - they just don't care. Totally regret this buy. #QuickTechFail #Frustrated 😞 👎"

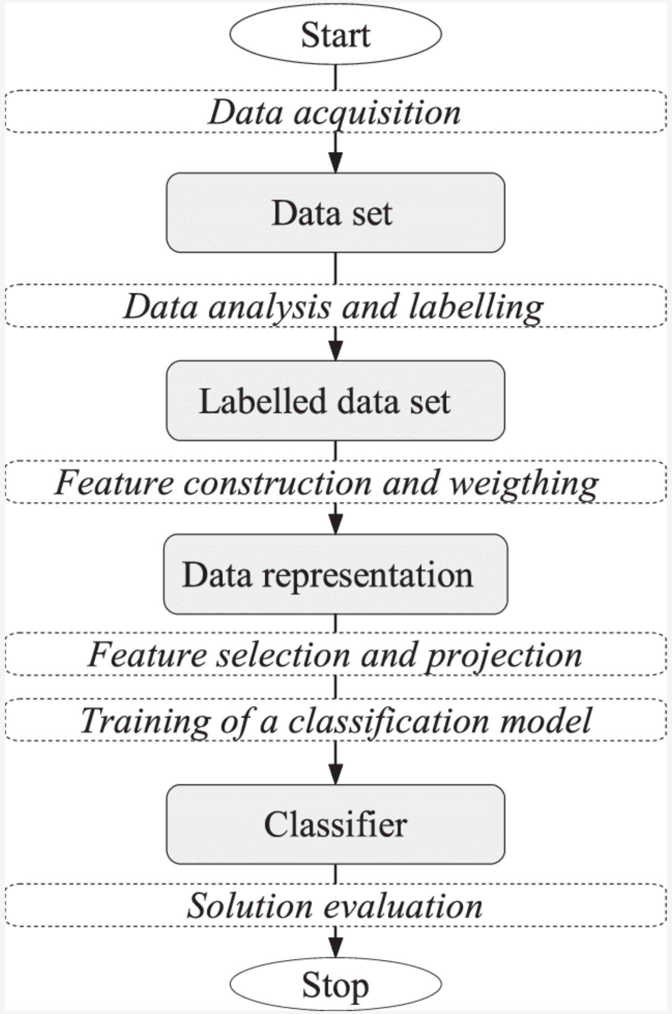
Feature extraction

- Define the set of relevant features
- Train (or program) algorithms to process the text
- Extract features
- Represent the text via the features



Text classification using features

- Step by step process
- Involves active human engagement
  - Feature selection and extraction
- Data is fed into a classifier (Logistic, NB, SVM)
- Iteratively improve feature selection and model (hyper) parameters



The shift towards data-driven approaches

- DSM and embeddings extract (relevant) features from the data
  - Minimal supervision
  - Easier maintenance
- Allow computers to “read” and use language in a more direct way
- The cost is loss of control and transparency

New NLP paradigm – the rise of end-to-end neural models

- Word embeddings mark a major shift in NLP
- What do you think is an “end-to-end” neural model?
- End-to-end neural model represent the complete target system
  - No external preprocessing
  - No explicit pipelines
  - Input-output mapping
  - Training for a specific task (with some transfer learning)
- What would be some limitations of end-to-end models?

alot of data rely on your data and can lead to racial biases ⋮

we don't know, what's inside black box.

Using embeddings
Compositionality

Use of word embeddings

- Embeddings are high dimensional vector representations of words

- All words in the vocabulary can be represented as a high dimensional vector

- What can we use embeddings for?
  - Write five different applications

Querying embeddings for lexical information

- Embeddings are originally a lexical resource
- Performing operations at word level
  - Find words that are semantically similar to a given word
    - Expand existing dictionaries by automatically searching for similar words
  - Explore relational similarity (e.g., UK – London: France – Paris)
  - Compare the meaning of a word to its context
    - Automatic error correction
-

Querying embeddings for lexical information

- Learning from embeddings
  - Learn specific semantic relations (e.g. hypernymy) (Shwartz, et al. 2016)
  - Learn compositionality rules (Baroni and Zampareli, 2010, Socher et al. 2013)
    - Learn representations for phrases and compare with words
  - Clustering (Kovatchev et al. 2016)
    - Topics
    - Part of speech

From words to text

- Embeddings represent words
- NLP is about processing text
- “The cat sat on a mat” vs “the mat sat on a cat”

Compositionality of meaning

- “The meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them”

- Two key questions:
  - How do we combine individual word meaning?
  - Does the word meaning remain static?

How to combine word meaning

- Assume that the word meaning is a vector or a tensor

- How can you calculate the meaning of a phrase?
  - Addition/aggregation
  - Complex (hierarchical) operations
  - Via a deep neural network

Vector addition

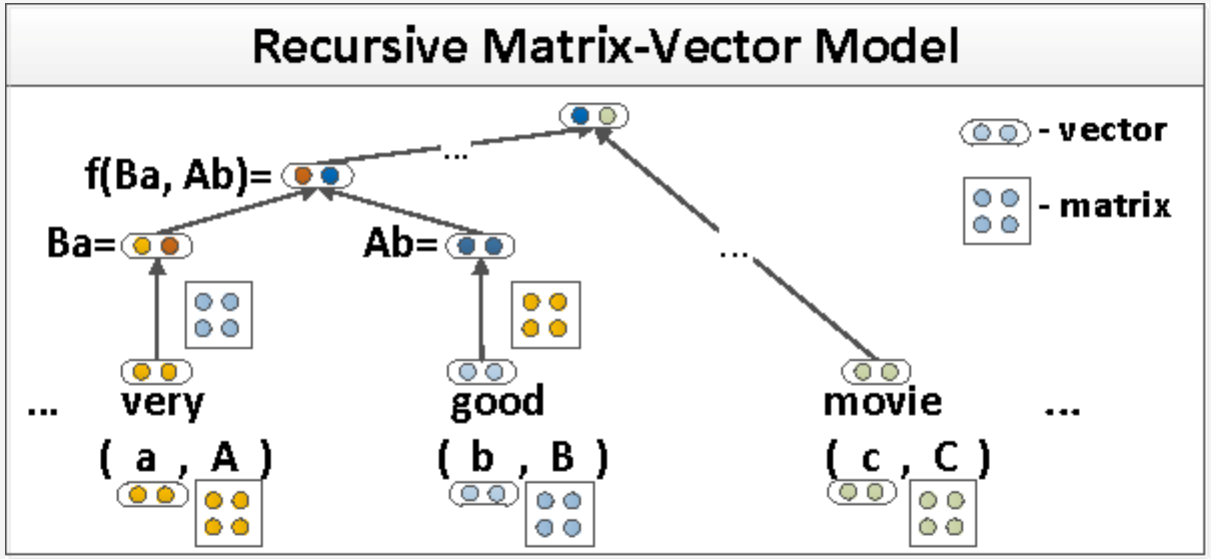
- The simplest form of compositionality
  - “The cat sat on a mat” = “The”+ “cat” + “sat” + “on” + a” + “mat”
- Advantages
  - Easy to calculate
  - Fixed vector length, regardless of text length
- Disadvantages
  - Loses word order
  - Lower impact of individual words (e.g., “not”)

Vector concatenation

- Alternative to vector addition
  - Instead of adding dimensions, we concatenate the vectors
- Can you identify advantages and disadvantages of this approach?

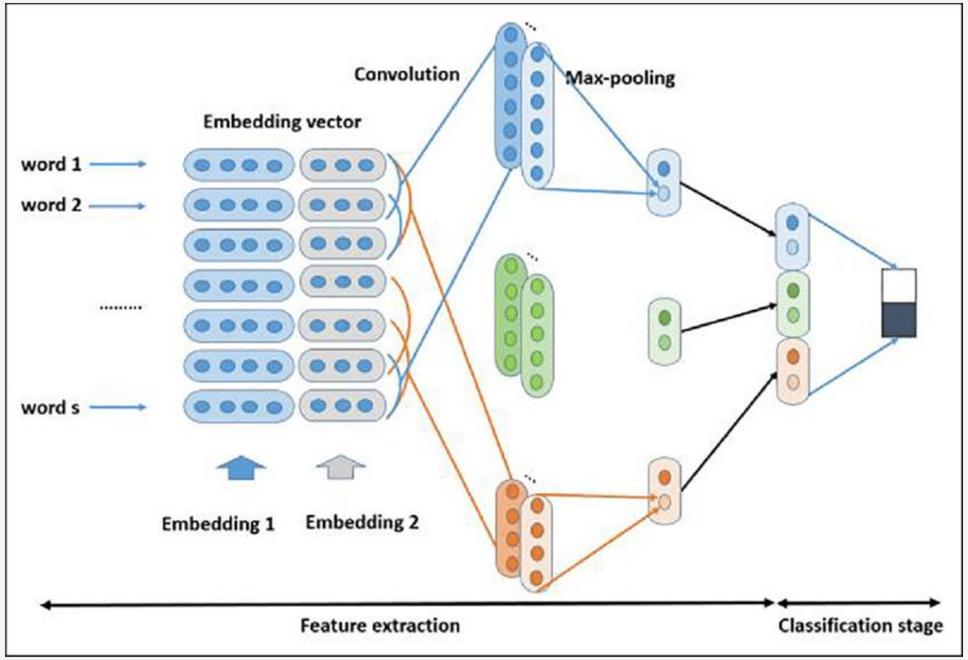
Recursive compositionality

- Baroni and Zampareli (2010), Socher et al. (2012, 2013)
- Meaning is not just a vector, but can be a vector + matrix
- Compositionality is a recursive vector-matrix operation
- Follow the syntactic structure



Compositionality via deep neural networks

- The current state-of-the-art
- Input the embeddings into a neural network
- Let the network handle the interactions
  - The network architecture determines the compositionality



Task specific embeddings

- General purpose word embeddings
- Polysemy (“blue cat”, “cat myfile.sh”, “CAT scan”)
- Retraining embeddings
    - Domain: news, medical, social media (Major et al. 2018, Soares, 2019)
    - Task: sentiment, NER (Siencnik, 2015)
    - Languages

Contextual word embeddings

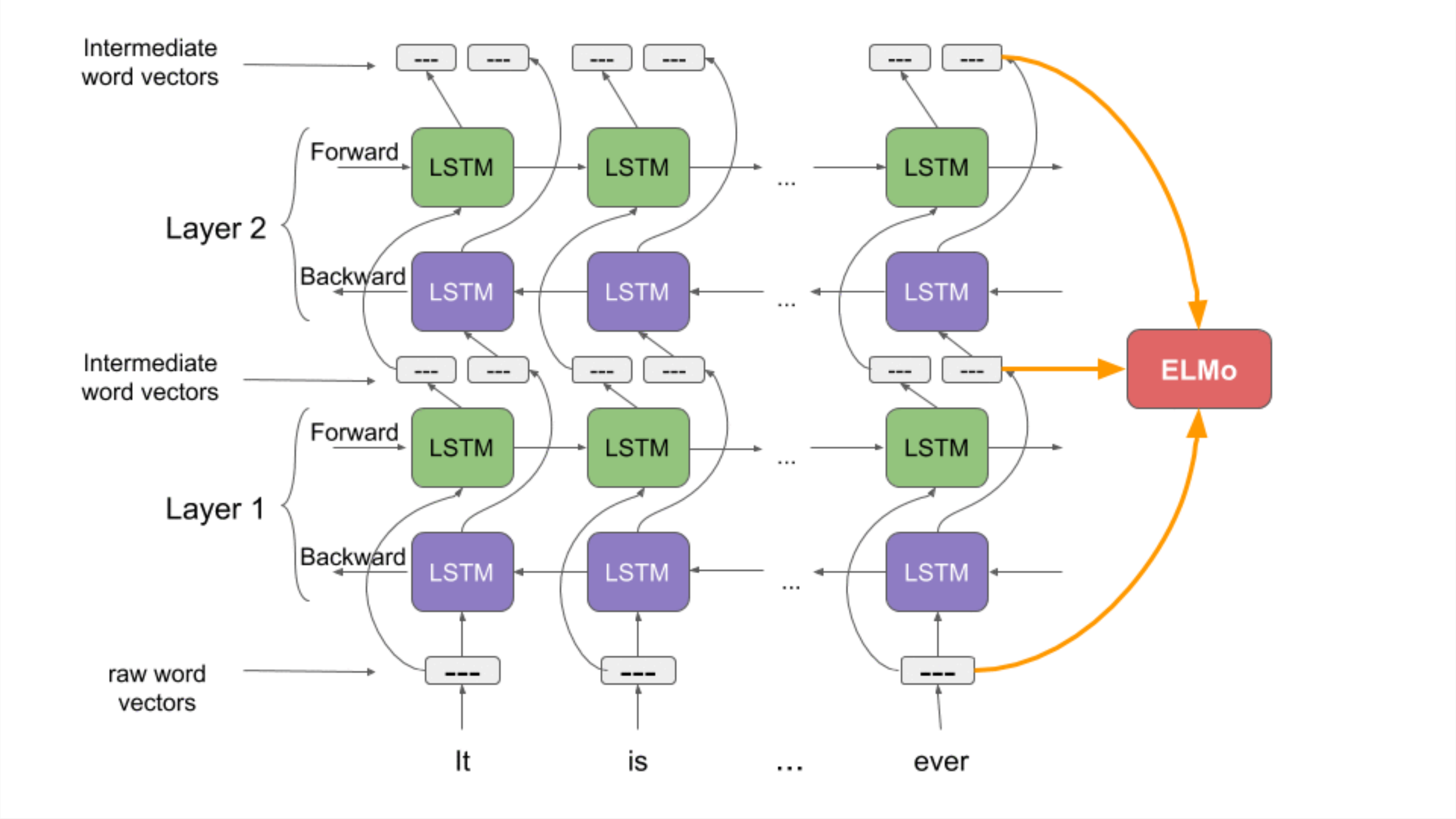
- The problem of polysemy
  - “The **cat** sat on a mat”
  - “You can **cat** this text file”
  - “I just got the results from my **cat** scanner”
- Are these the same word?
- Should they have the same embedding?
- Task specific embeddings may solve the problem, but can we do better?

ELMO – Deep contextualized word representations

- Key idea: generate a dynamic embedding, based on the context a word appears
  - “The cat sat on a mat” -> W2V -> the vector of “cat” depends only on “cat”
  - “The cat sat on a mat” -> ELMO -> the vector of “cat” depends on all words

- How? Bi-directional (LSTM) language model
- Concatenation of different layers
- Task-specific weights

Bi-directional language model



Source: https://www.analyticsvidhya.com/blog/2019/03/learn-to-use-elmo-to-extract-features-from-text/

Bi-directional language model

- Forward language model: $$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}).$$

- Backward language model: $$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N).$$

- Bi-directional LM: $$\sum_{k=1}^{N} \left(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right).$$

- Predict a word given its left and right context; keep input and output weights shared;

Deep representations

- For each token k, an L-layer bi-directional LM obtains 2L + 1 representations

$$R_k = \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} = \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}$$

- Initial (static) representation x
- For each layer – hidden representation of forward and backward
- ELMO learns a task-specific linear combination:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}. \tag{1}$$

- The representations R_k depend only on the context
- The embedding $ELMO_k$ depends on the context and the task

The impact and importance of ELMO

- ELMO (Peters et al. 2017) significantly improves NLP model performance
- Not the first contextual representation
- The first deep contextual representation (prior work only takes last layer)
- The first task-specific representation
- Short lived success due to the appearance of transformers and BERT
- Many of the concepts in ELMO are adopted in BERT

Using embeddings in downstream tasks and student projects

- What kind of vectors to use?
- Use pre-trained or re-train?
- Use as feature vectors or use to learn features?
  - E.g., can you “learn” which vectors are positive?
- Can you combine with other features?
- What would be the classifier?
- Size and scale of vectors?

Using embeddings in downstream tasks and student projects

- What kind of vectors to use?
- Use pre-trained or re-train?
- Use as feature vectors or use to learn features?
  - E.g., can you “learn” which vectors are positive?
- Can you combine with other features?
- What would be the classifier?
- Size and scale of vectors?