# Natural Language Processing

## An Overview

Brigitte Krenn, Johann Petrak




# Natural Language Processing (NLP): What is it?

- Computational methods for understanding or generating natural language
- Goal: Process/analyse human language with computers

  Analysis: build useful representations from natural language input

  Synthesis/Generation: produce natural language output from (structured) representations
- Massive Challenge: make the meaning (semantics, pragmatics) of human language accessible to computers

# NLP: Why it is difficult

- Ambiguity: same text can have very different meanings
- Different texts can have same meaning, e.g. different wordings of summaries of the same document
- No hard rules: socio-linguistic differences, colloquial and informal language (social media!)
- So many languages!
- Code and language switching: mixing standard language and dialect, mixing languages; often in emails and social media

# NLP: Why it is difficult - Examples

- "He went to the bank" (savings bank, river bank?)
- "If you love money problems show up" (love money or love money problems?); "He wrote a report on Mars" (report written being on Mars or report about Mars), "Cop kills man with knife" (who has the knife?)
- "Mark saw the men with the telescope. He had brown hair." (who had the telescope / brown hair?)
- Headline: "Teacher Strikes Idle Kids" (cred. Dan Klein)
- "This is shit" vs. "This is the shit"
- "@angry_barista I baked you a cake but I ated it", "Why won't you show my location?! http://twitpic.com/2y2es", "@markhardy1974 Me too #itm" (ex. from kaggle Sentiment140 tweets dataset)
- "he wanted to do it but he got cold feet which made him lose his face" (multi-word expressions)
- "It is cold" (statement or request for action → pragmatics)
- ⇒ Often clear from context or with background knowledge but system has to deal with it

# Linguistics View

- Speech signal
- Phonetic form: visual representation of sounds; International Phonetic Alphabet (IPA) https://www.cambridge.org/features/IPAchart/
- Phoneme – unit of sound: /p/, /t/, /k/, …
- Syllable – unit of pronunciation: /wa-ter/, /un-for-giv-ing/
- Morpheme: water, un-forgiv-ing
- Word/word form (type/token, stem/inflected form) : forgiv, forgave, forgiven, forgiving
- Phrase: an unforgiving environment (NP), is unforgiving (VP)
- Sentence: The deep sea is an unforgiving environment.
- Text: The deep sea is an unforgiving environment. This is why we should not stay in it.

# Computational / Practical View

- Text: Sequence of (Unicode) characters
- Document: a unit of text that somehow belongs together

  An article, a book, a page in a book, a Tweet, …

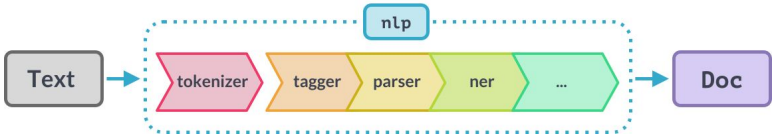
  Can have associated meta-data (author, publishing date, keywords..)
- Words/Tokens: (usually) the relevant parts of the text
- Corpus: a collection of documents that belong together

# Example NLP Applications

- Analysis: Text classification (e.g. spam detection)
- Analysis: Information Extraction (IE, extract structured information from text). Find mentions of names, organizations, relationships, events …
- Synthesis: Content Generation from structured information (e.g. weather forecast writing)
- Both: Question Answering
- Both: Machine Translation

# Approaches to NLP

- Classical divide and conquer:
  - break up into smaller tasks
  - develop methods to solve those tasks
  - combine tasks into a "pipeline" to solve overall task
  - for each task, there can be many different possible approaches
  - some or all of the sub-tasks may be based on some form of machine learning




- "End-to-end":
  - Single (Deep-Learning) model that does it all in one go
  - But, some NLP tasks (Tokenization) and ML tasks (data-loaders, batching..) are still needed

# Example NLP Tasks

- Tokenization, Sentence splitting
- POS tagging, Lemmatization
- Parsing
- Entity recognition and disambiguation/linking
- Reference resolution
- Relation extraction
- Text classification
- Topic detection, Text Clustering
- Keyword extraction

Let's look at some of those in more detail!

# Task: Tokenization

- We are interested in words and their properties
- What is a word? "It's", "don't", "data-mining", what about punctuation?
- Divide into units to process and call it a "Token":

  e.g. split on white space or punctuation
- White space does not work for each language (e.g. Chinese, Turkish) → sub-word units
- Can have multi-word tokens, e.g. "it's", suntan lotion, “New York Times”, “kick the bucket”
- Can have punctuation tokens
- Special tokens, e.g. URLs, @names, #multiwordhashtags
- Result: text is represented as a sequence of Tokens instead of a string

# Task: Sentence Splitting (1)

Sentences are a good unit of analysis:

- a sentence is a self-contained unit of meaning relations,
- it is syntactically complete, and
- typically contains subject, predicate, object(s), modifiers

The house is green.
The green house belongs to my cousin.

- Sometimes good for splitting text into 1 or more sentences for technical reasons (limited input size, parallelization etc.)

# Task: Sentence Splitting (2)

- E.g.: split on punctuation (full stop, exclamation mark), double new lines
  - but not e.g. after abbreviation with dot!
- There are also more sophisticated methods which make use of some of the subtasks and approaches we discuss later.
- After sentence splitting, the text may get represented as a sequence of sentences. Each sentence being a sequence of tokens.

  (((After) (sentence) (splitting) (,) (the) (text) (may) (get) (represented) (as) (a) (sequence) (of) (sentences) (.))
  ((Each) (sentence) (being) (a) (sequence) (of) (tokens) (.)))

- Alternative: Stand-off Annotations

# Stand-off Annotations

- Clumsy to use nested lists of lists of tuples … (e.g. Python NLTK)
- Instead, have sets / lists of stand-off annotations:
  - Start, end offset in the original text/document: annotation links to original text (some variations use token index instead of char offsets: no sub-token anns)
  - Type: e.g. "Token", "Person", "NP"
  - Features: information about that text, e.g. word of type noun, noun phrase, person name
  - Arbitrary many, arbitrarily overlapping, grouped into "sets"
  - Can be linked to form a tree or graph
  - Select the ones useful for processing
  - Details differ between implementations: SpacY, GateNLP, BRAT, …

# Stand-off Annotations: Example

Example annotated document with several types of stand-off annotations from the GateNLP framework:

https://tinyurl.com/stoffann

# Task: Part of Speech (POS) Tagging

- We need more information about each word/token: is it verb, noun?
- Ideally also additional info (gender, time, singular/plural, case, …)
- Lookup in a dictionary does not work well:

  "duck" vs. "duck", "lie" vs. "lie" …
- Use machine learning! For this we need:
  - a pre-annotated corpus
  - features for each word to use for learning
  - machine learning algorithm

# POS Tagging: Corpus Examples

- Several Treebanks for different languages, many different tag-sets
- E.g. Penn Treebank (Marcus1993, Taylor2003)
- Universal Dependencies Corpus:
  - Harmonized POS tags
  - Corpora for many languages
  - https://universaldependencies.org/
  - CoNLL-U Annotation Format
  - https://universaldependencies.org/format.html

```
1     Seriously    seriously    ADV
2     :            :            PUNCT
3     do           do           AUX
4     not          not          PART
5     waste        waste        VERB
6     your         you          PRON
7     time         time         NOUN
8     .            .            PUNCT
```

# POS Tagging: Tagsets Universal & Penn

- Universal PoS tags: core part-of-speech categories
- https://universaldependencies.org/u/pos/

| Open class words | Closed class words | Other |
|---|---|---|
| ADJ | ADP | PUNCT |
| ADV | AUX | SYM |
| INTJ | CCONJ | X |
| NOUN | DET | |
| PROPN | NUM | |
| VERB | PART | |
| | PRON | |
| | SCONJ | |

- In addition: universal features to distinguish additional lexical and grammatical properties of words

- Penn Treebank PoS tagset
- Taylor2003

*Table 1.1.* The Penn Treebank POS tagset

| | | | |
|---|---|---|---|
| CC | Coordinating conj. | TO | infinitival *to* |
| CD | Cardinal number | UH | Interjection |
| DT | Determiner | VB | Verb, base form |
| EX | Existential there | VBD | Verb, past tense |
| FW | Foreign word | VBG | Verb, gerund/present pple |
| IN | Preposition | VBN | Verb, past participle |
| JJ | Adjective | VBP | Verb, non-3rd ps. sg. present |
| JJR | Adjective, comparative | VBZ | Verb, 3rd ps. sg. present |
| JJS | Adjective, superlative | WDT | Wh-determiner |
| LS | List item marker | WP | Wh-pronoun |
| MD | Modal | WP$ | Possessive *wh*-pronoun |
| NN | Noun, singular or mass | WRB | Wh-adverb |
| NNS | Noun, plural | # | Pound sign |
| NNP | Proper noun, singular | $ | Dollar sign |
| NNPS | Proper noun, plural | . | Sentence-final punctuation |
| PDT | Predeterminer | , | Comma |
| POS | Possessive ending | : | Colon, semi-colon |
| PRP | Personal pronoun | ( | Left bracket character |
| PP$ | Possessive pronoun | ) | Right bracket character |
| RB | Adverb | " | Straight double quote |
| RBR | Adverb, comparative | ‘ | Left open single quote |
| RBS | Adverb, superlative | “ | Left open double quote |
| RP | Particle | ’ | Right close single quote |
| SYM | Symbol | ” | Right close double quote |

# POS Tagging: Features (1)

- Suffixes of lengths 1..k, prefixes of length 1..l
- Sentence start / end
- Punctuation, contains number, starts with upper-case, all-upper, contains hyphen, ..
- Add features from previous and succeeding n tokens (context window)
- Add POS-tag prediction of previous word(s) as features
- (Toutanova2000)

# POS Tagging: Features (2)

- But knowing which word exactly may also be useful
- How to add this information as a feature?
- Word number: tens of thousands of numbers
- One-hot features: tens of thousands of features
- Idea: group words into k clusters
- Use only k numbers or k one-hot features

# Brown Clustering

- Hierarchical, agglomerative algorithm (Brown1992)
- Maximise the mutual information of cluster bigrams:

$$\sum_{c_1=1}^{k}\sum_{c_2=1}^{k} p(c_1c_2)\log\left(\frac{p(c_1c_2)}{p(c_1)p(c_2)}\right)$$

- Words that appear in the same context tend to get clustered together:

  Friday, Monday, Thursday, weekends, Sundays …

  People, guys, folks, chaps, doubters, blokes …

  Down, backwards, ashore, sideways, ….
- The contexts of words give important information about the word!
- Each cluster represents a "word type/class" or "context class"
- Use the cluster number as a feature (represented as number or one-hot vector)

# POS Tagging: Machine Learning (1)

- Use the best ML Algorithm of the time!
- E.g.: Logistic regression; Decisions trees; Random Forests; Support vector machines and other large margin algorithms
- Differences: handling of numerical vs. categorical features, handling of very many/sparse features, model complexity, …
- Creates a model: mapping from the features of the current, preceding, following token to POS tag, each token gets classified separately

# POS Tagging: Machine Learning (2)

- Dependencies in how likely each POS tag follows another!

  E.g.: DET most often before NOUN or PRON
- Sequence labelling / sequence tagging algorithm:

  (linear-chain) Conditional Random Fields (Lafferty2001)
- Discriminative model based on graphical models.
- Find $\text{argmax}_{\mathbf{y}}\ p(\mathbf{y}|\mathbf{x})$ (using Viterbi algorithm):

  Whole sequence gets labeled in one go!

# Word representations

- Brown clusters are helpful but can we represent words better?
- Inspiration from Information Retrieval (IR)
- Represent a document by a vector with n elements,
  n = number of different words (word forms) in the corpus
- The ith element contains some numerical information about that word:
  indicator (0, 1), term frequency (nr occurrences in the text), …
- Bag of Words (BoW) representation of a text / document
- Note: BoW vector of a single word = one-hot vector

# Text Representation: Bag of Words (BoW)

- Problem 1: tens of thousands of elements per vector
- Problem 2: no information about how the words occur in sequence:
  "bad, not good at all" = "good, not bad at all"
- Bag of n-grams: one feature per e.g. bigram "good not", "not bad" …,
  increases Problem 1, decreases Problem 2
- Can we do better?

# Word Representation: Inspiration from IR (1)

- Information Retrieval: Initially represent each document of a corpus by BoW vector
- The corpus can be represented by a matrix with words in one dim, documents in the other dim, where the value shows how often word i occurs in document j
- Each document is represented by a vector of words (BoW),
  each word is represented by a vector of documents
- Documents can be compared by finding the similarity between their word vectors
- Words can be compared by finding the similarity of their document vectors!
- Usually using cosine similarity: $\frac{v_1 v_2^T}{\|v_1\|\|v_2\|}$

# Word Representation: Inspiration from IR (2)

- These term/document matrices are huge, mostly sparse & very redundant
- Try rank reduction: use Singular Value Decomposition (SVD) and choose top k singular values:
- Latent Semantic Indexing (LSI): documents k-dim vectors of "concepts", words represented by k-dim vectors
- Now have vectors of dim in the 100s instead of 100k s to represent words
- "dense" word vectors, "word embeddings"
- Cosine similarity still works well on those!

# Word Embeddings (1)

- Problem: embedding from occurrences over a whole (possibly large document) But local context is important!
- Can we create word embeddings based on the co-occurrence of words the local context within a corpus?
- Engineering approach: use a neural network: Word2Vec (Mikolov2013):
  - Continuous Bag of Words (CBOW): predict the center word of a context window from its neighboring words
  - Skip-Gram: predict the neighboring words from the center word
  - For both approaches, we can choose the dimensionality we want, e.g. 200
  - Simple 1 hidden layer network, symmetric windows (e.g. 5), negative samples 2-15
  - Use learned network parameters as embeddings

# Word Embeddings (2)

- Mathematical approach: generate a matrix and do dimensionality reduction
- Many different approaches, most used: Glove (Pennington2014)
- Define context window size and symmetric (left+right) or asymmetric (left only)
- This gives us a n * n matrix X where n is the size of the vocabulary
  Value in the matrix: co-occurrence count
- What we want: embeddings w such that $F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$
  Where the $P_{ik}$ are co-occurrence probabilities
- Simplification and conversion to least squares problem: $\sum_{i,j=1}^{n} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log X_{ij})^2$
- Use AdaGrad and stochastic sampling of $X_{ij}$ to train
  Use $W + \tilde{W}$ as final embeddings

# Word Embeddings (3)

- Many other approaches, e.g. FastText (Bojanowski2017):

  Make use of internal structure, use subword information (char n-gram)

- Evaluating Word Embeddings:
  - Usefulness: for solving down-stream tasks (e.g. POS tagging)
  - Similarity: does it follow human intuition?
  - Analogy: v("king") - v("man") + v("woman") = closest to v("queen")
    But also reflects bias: v("woman") + v("doctor") - v("man") = closest to v("nurse")

# Task: Named Entity Recognition (NER)

- Find and classify names in text: person, location, organization, gene name, species name, or other "chunks" of text like movie names, book titles
  Task "Chunking"
- Important for Information extraction but also for subsequently finding relationships between such entities
- Can consist of 1 to k tokens: "Vienna", "People's Republic of China"
- How to find anywhere in text: need to convert to a manageable classification task
- Nadeau2007, Li2020 (surveys)

# NER: BIO Coding

- For each token, assign a label that shows if the token is outside (O) a NE, is the beginning (B), inside (I), the end word of an NE (E), a single word NE (S),
  e.g. IOB2 format:

```
Alex  is going to Los   Angeles in California
B-PER O  O     O  B-LOC I-LOC   O  B-LOC
```

- Many similar formats, many names: BIO, BIOES=BILOU=BMEWO …
- Can use per token classification of the code similar to POS tagging
- Derive chunk boundaries from assigned codes, need to correct invalid sequences
- As with POS tagging, CRF can be used to consider the dependencies between labels

# Other Chunking tasks

- Find short token sequences of some interesting type
- Noun Phrases (NP), Verb Phrases (VP), Prepositional Phrases (PP)

  *This sentence contains two noun phrases.*
- Useful for more complex tasks

  e.g. Biological Event Extraction, Fine-Grained Opinion Analysis
- Which word is the most important (head) and which other words are attached to it?
- → (Dependency-)Parsing

# Task: Parsing (1)

- Natural language
  - has has structure (*syntax*),
  - follows construction principles (*grammar*)
  - regulating how tokens (words from the lexicon) can be combined
- Parsing
  - analyse a sequence of tokens according to the rules of a formal grammar
  - natural languages, formal languages, data structures follow different formal grammars
- Use the tree to
  - derive structure and meaningful relations (who is doing what to whom … ?)
  - disambiguate word categories (parts-of-speech, e.g., noun or verb),
  - disambiguate the nesting of constituents, e.g., PP-attachment

# Task: Parsing (2)

- Natural Language Parsing:
  - Use linguistic knowledge for structural analysis,
  - Parse a sentence into a tree

Phrase Structure

- Hierarchical structure of phrasal constituents and words
- Express linear order of a sentence

Dependency Structure

- Word pairs linked by grammatical relations
- Express syntactic word-to-word relations

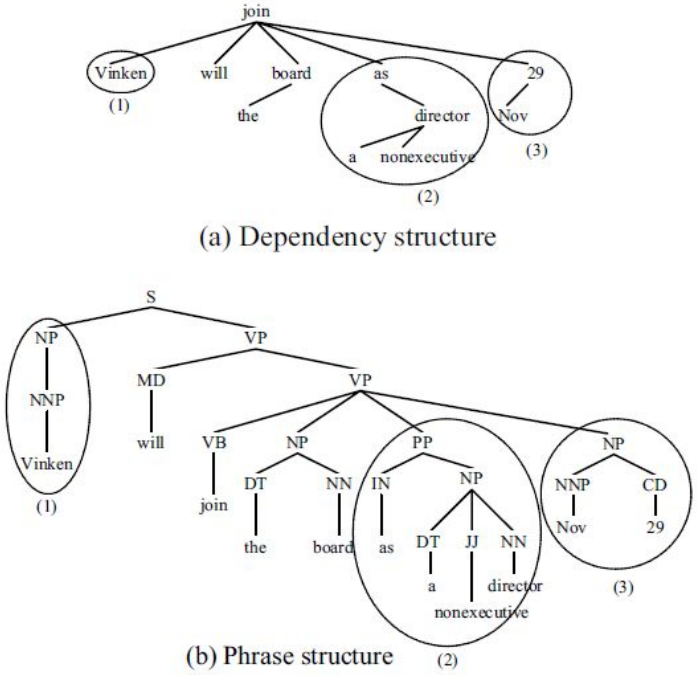



Figure 1. The consistency between phrase structure and dependency structure (Wang2010)

# Dependency Parsing (1)

- Identify related words and type of their relation
- Head word – relation type – dependent word (modifies head)
- Dependency Treebanks to train parsers
  - Universal Dependency (UD) Treebanks
  - Text annotated with Universal Dependency Relations https://universaldependencies.org/u/dep/ and PoS tags from the UD Tagset
  - E.g., English UD Treebanks https://universaldependencies.org/treebanks/en-comparison.html
- Tools and models for parsing, e.g., spaCy, NLTK, Stanza, CoreNLP, SyntaxNet (Andor2016, Parsey Mc Parseface)

# Dependency Parsing (2)

- Annotation formats
  - E.g., CoNLL-U https://universaldependencies.org/format.html
  - Each word is represented by 10 fields:
    ID, FORM, LEMMA, UPOS (universal PoS), lang.specific PoS (XPOS), morphological features (FEATS), the HEAD word, the dependency relation (DEPREL), head-dependency pairs (DEPS), any other annotation (MISC)

```
# sent_id = 1
# text = They buy and sell books.
1   They     they     PRON    PRP    Case=Nom|Number=Plur                2   nsubj   2:nsubj|4:nsubj   _
2   buy      buy      VERB    VBP    Number=Plur|Person=3|Tense=Pres     0   root    0:root            _
3   and      and      CONJ    CC     _                                   4   cc      4:cc              _
4   sell     sell     VERB    VBP    Number=Plur|Person=3|Tense=Pres     2   conj    0:root|2:conj     _
5   books    book     NOUN    NNS    Number=Plur                         2   obj     2:obj|4:obj       SpaceAfter=No
6   .        .        PUNCT   .      _                                   2   punct   2:punct           _
```

# Task: Keyword/Key Phrase Extraction

- Subtasks:
  - Candidate identification
  - Candidate ranking
- Approaches
  - Graph-based approaches:  TextRank (Mihalcea2004), Topicrank
  - Heuristic-/statistics-based models, e.g., YAKE! (computes combined score based on various statistics on each word, combines high-score adjacent words to phrases)

# Task: Named Entity Disambiguation (NERD)

- Finding NEs is not enough: is it Vienna (Austria), Vienna (Ontario), Vienna (Alabama), Vienna (the river running though Vienna) …
- NERD, Entity Linking
- Distinguish by linking to some knowledge base, e.g. Wikimedia, Wikidata
- Some NEs may refer to an entity not in the KB, several NEs may refer to different entities not in the KB: clustering of new unlinked concepts

# Named Entity Disambiguation: Approaches

- Many different approaches, but frequent commonalities:
- Usually Entity Recognition (ER) is followed by Entity Disambiguation (ED), a few methods of doing it jointly (Kolitsas2018)
- Find semantic similarity between information about the entity (e.g. Wikipedia article text) and the context of the mention
- Also consider a-priory likelihood, popularity, … of each link
- Multiple entities in proximity should be "compatible"

# NER Evaluation (1)

Exact-match evaluation:

- Consider NE boundary and type together; both must be correct
- Precision P: how many of the found NEs are correct? TP/(TP+FP)
- Recall R: how many of the actual NEs have been found? TP/(TP+FN)
- F1-score: 2(PR)/(P+R)
- Micro-averaged: if there are several NE classes, average using TP/FN/FP over all
- Macro-averaged: Calculate per class, then average

# NER Evaluation (2)

Relaxed/lenient-match evaluation

- Consider type and (token-/character-) boundary separately, e.g.
- MUC-6 (Grishman1996): type correct if overlaps with target, boundary correct if matches target boundaries, independent of type
- ACE (Doddington2004): complex scheme that is more flexible and powerful, but less intuitive and more difficult to use for error analysis.

Area under curve (AUC): evaluate TP vs FP (AUC-ROC) or P vs R (AUC-PR) for different probability thresholds and calculate the area under that curve: depending on threshold more correct examples may appear but also more wrong ones.

# ML/NLP Evaluation

- What we really want to know is: "if we train a method to solve this task (e.g. NER) on this training set, how well will it do on new data? (generalization performance)
- A single evaluation on some train/test split does not tell very much: maybe we were lucky, maybe we used the right random seed?
- Instead use e.g. (class-stratified) k-fold cross-validation to get average and stdev
- Compare different approaches / settings by checking stdev or doing some significance test
- However, many papers, leaderboards still neglect this and just show bare numbers from a single train/test split
- Error analysis: which errors have been made? How "severe"/"surprising"?

# Task: Text Classification

- E.g.: sentiment/polarity classification: pos/neg, pos/neutral/neg
- E.g.: movie reviews: IMDB Movie Review Dataset: 50K texts
- Need: features, ML algorithm
- Features: how to represent the whole text for the ML algorithm?
  - BoW vector, LSI document vector
  - Sum word vectors (all; remove stop words) ⇒ DAN
  - Use list of word vectors (how? ⇒ CNN, RNN, LSTM)

# DAN: Deep Averaging Network

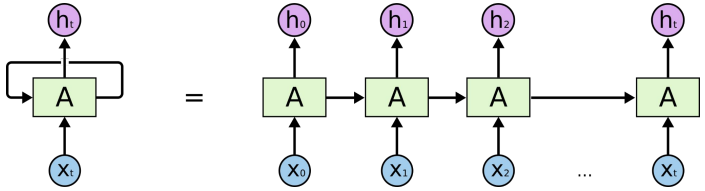
- Simple, but worked surprisingly well at the time (Iyyer2015)
- Input: averaged sum of embeddings, with optional word dropout
- Model: k hidden layers with dropout and non-linearity
- Output: softmax from final layer
- Obvious disadvantage: word order, syntax does not influence the input, e.g. negation cannot be handled!

# CNN (Convolutional Neural Network)

- CNN for sentence classification: Kim2014
- Sentence: limited number of words / embeddings
- Input n embeddings of dim k (padded if text is shorter than n)

  n depends on the max size of the training / expected test set
- Convolutional layer with multiples filters, followed by max-pooling
- Fully connected layer with dropout and softmax output

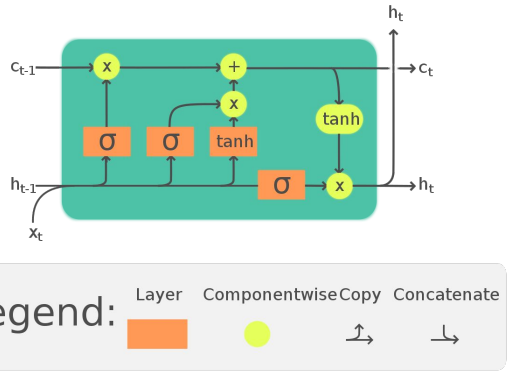
# RNN: Recurrent Neural Network



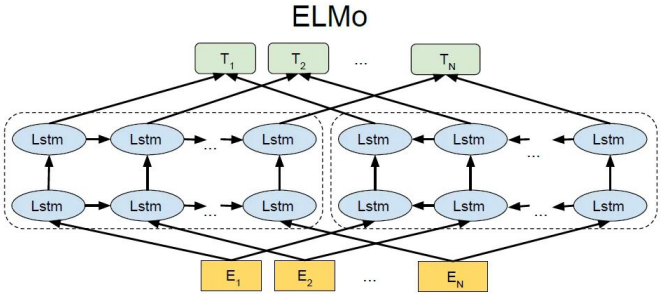

- Apply a (multi-layer) NN to each input of a sequence, e.g. from left to right
- In addition to the input, at each time step, we get the hidden layer activations from the previous step and pass them on to the next step
- Apply back-propagation "through time"
- Advantages: no fixed sequence length, can do text both:
  Classification: use last output activations
  Sequence labeling: use output activations for each input
- Disadvantages: local optima, exploding/vanishing gradients

# LSTMs: Long short-term memory

- Adds Input/Forget/Ouput Gates to control flow (Hochreiter1997)
- Advantages: learn more complex and long-range patterns, less overfitting
- Disadvantages: more complex, slower to train
- Comparable Alternative: GRU (Gated Recurrent Units)
  - Fewer gates and fewer parameters, but still often performance equal to LSTMs



# ELMO

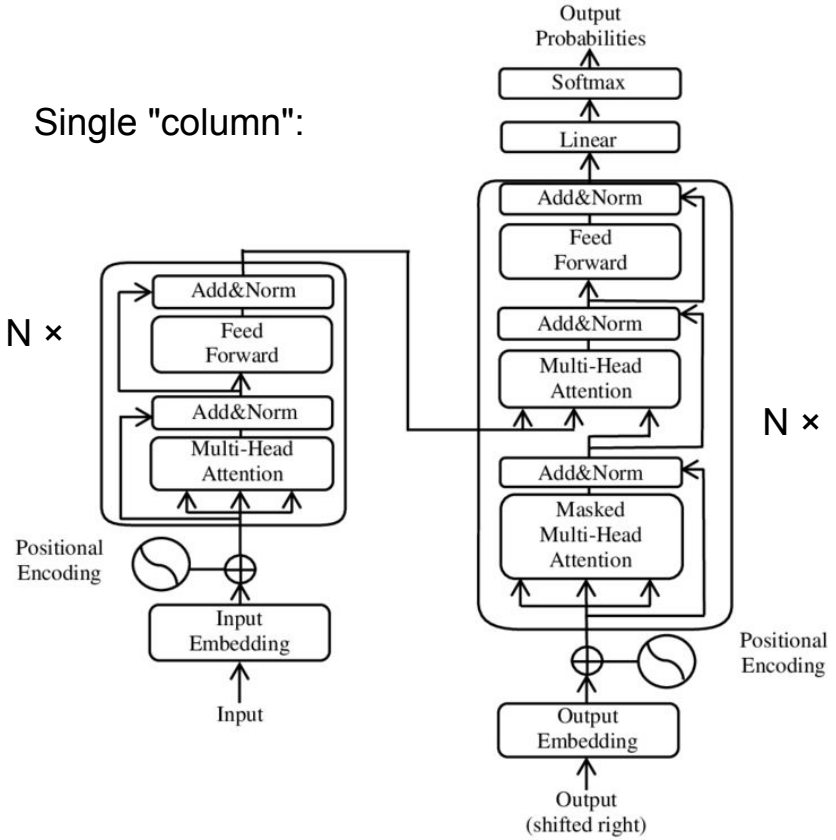



- Embeddings from Language Models (Peters2018)
- Idea: embedding should depend on context
  ⇒ different embeddings for same word, depending on context
- Use forward+backward multi-layer LSTM to predict token from left and right context simultaneously: "Neural Language Model"
- Language Model: model $p(w_k|w_{k-n}w_{k-n+1}...w_{k-2}w_{k-1})$ (from left)
- Lower layers capture more syntactic, upper more semantic properties:
  ⇒ Combine (possibly learn task-specific weights)
- Contextualized word embeddings!

# Transformers

- (Vasvani2017)
- Encoder-Decoder sequence-to-sequence model
- Instead of RNN/CNN use a feed-forward architecture with "multi-head self attention"
- One feed-forward "column" per token, horizontally interconnected via self-attention

Single "column":



# Transformers: Tokenization / Inputs

- Per-wordform tokens: large number, some wordforms very rare, internal word structure is ignored
- Per-byte tokens: small number, but hard for model to learn meaningful representations for the token
- Compromise: sub-word tokenization

# Subword Tokenization

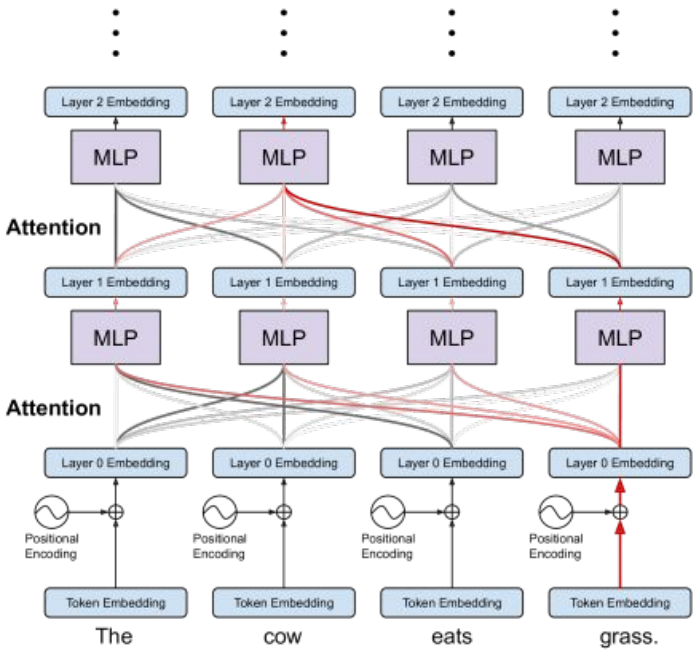
- Byte-Pair Encoding BPE (Sennrich2016):
  - Start with base vocabulary (unicode, byte) from a list of unique words
  - Merge tokens according to frequency to create new tokens from base tokens
  - Add new tokens and repeat, learning merge rules for splitting unseen words (tokenizer must be trained!)
- WordPiece (Schuster2012): similar to BPE, use different merge criterion
- Unigram (Kudo2018a): initialize with large vocabulary of words, substrings, characters, progressively trim to reduce vocabulary size: used for SentencePiece
- SentencePiece (Kudo2018b): better suited for languages that do not separate words by spaces, treat input as stream, treat whitespace as ordinary char
- Detokenization: reconstruct the original string including whitespace from subwords

# Transformers: Self-Attention (1)

- Attention was used before Transformers e.g. in seq2seq LSTMs: depending on input, use representation from different states differently by learning weights for combining them.
- Transformer: attention is all you need!
- Transformer: in each layer the representation is a weighted sum of all value vectors in all columns. The weight is calculated from the similarity between query vector of the current column, and the key vectors of all columns.
- Query, key, value vectors are calculated from the current layer input via trainable matrices.

# Transformers: Self-Attention (2)

$$q_i = Qx_i$$
$$k_j = Kx_j$$
$$v_j = Vx_j$$
$$h_i = \sum_{j=1}^{n} w_{ij} vj$$
$$w_{ij} = \frac{\exp q_i^T k_j}{\sum_{j'=1}^{n} exp(q_i^T kj')}$$




(Brunner2019)

# Transformers: Multi-head Self Attention (3)

- In order to allow different kinds of influences, have several parallel attention blocks per column
- Technical trick: each head uses reduced dimension d/k (k=number of heads) so all k attentions can be calculated in parallel
- Dimensional scaling: dot products scaled by square-root of dimension of vectors

# Transformers: Positional Encoding

- Since the result of self-attention is just a weighted sum, there is no way to find out where each value comes from
- So "position embeddings" PE are added to the input embeddings, where each dimension of the position embedding contains the value of a sine / cosine function that depends on position and dimension
- For offset k, $PE_{pos+k}$ can be calculated as a linear function of $PE_{pos}$

# Transformer: full Encoder

- Each column has input embeddings+positional embeddings
- Followed by multiple layers, of which each has:
- Multi-head attention
- Layer normalization  per column (norm)
- Residual connection adding the input of the layer to the output (add)
- Feed-forward dense layer with add+norm
- Per-column nonlinearity (ReLU) and dropout per sub-layer

# Encoder only transformer: BERT (Devlin2019)

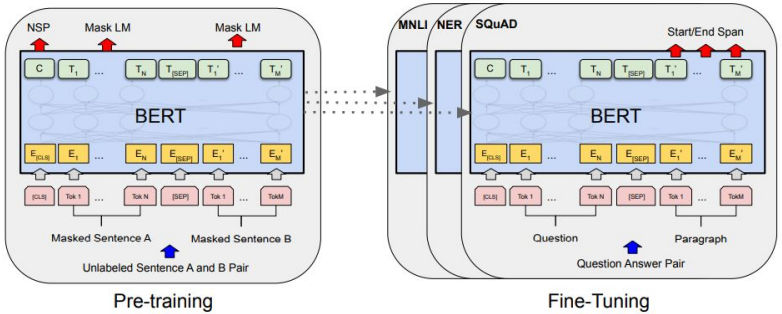
- Use Transformer encoder only, designed to solve many different NLP tasks
  - Base cased/uncased: 12 layers, 768 hidden, 12 attention heads, 110M parms
  - Large cased/uncased: 24 layers, 1024 hidden, 12 attention heads, 340M parms
- First token is always a special classification token CLS, first column specially for classification
- Additional "segment embeddings" are added in addition to positional embeddings: allow to deal with several segments/sentences of text
- The model gets pre-trained on large amounts of text

# BERT: Pre-training

- Masked Language Model (MLM): mask a word/token in the input and try to predict from context (actually use MASK 80%, random token 10%, unchanged token 10%), only use masked tokens for loss function.
  For this a classification head with a dense layer and softmax is added on top of all columns except CLS
- Next Sentence Prediction: Input are two sentences separated by a special SEP token. Additional Segment embeddings encode which tokens belong to which sentence.
  Binary Classification: 2nd sentence would follow 1st or not.
  For this a classification head with a dense layer and softmax is added on top of the CLS column

# BERT: Fine-tuning / Downstream tasks (1)

- Fine-tuning: load the pre-trained model, add appropriate head(s) and train on new data. The LM layers can be "frozen", have adapted learning rate or just get fully trained.



# BERT: Fine-tuning / Downstream tasks (2)

- Sequence tagging: per-column classification head over label set (POS tags, IOB codes), optionally followed by a CRF layer
- Classification: add a single dense layer after the output of the CLS column, followed by softmax
- Question answer identification: find start/end token of answer for a question in context: softmax over all context token column outputs after dot product with start / end vectors
- Get contextual word vectors: just forward the text and use the column outputs
- Many subtasks can be based on these

# BERT: Zero Shot Classification

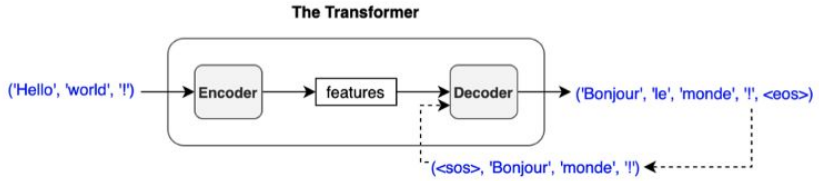
- Given: some text to classify and a number of candidate labels
- A template for creating the hypothesis sentence from labels
- Use Natural Language Inference NLI: get probability of text-to-classify implying the filled template:
  "That movie was great" implies "This is about movies[music|books]"?
- Alternative: Embed both the text and the (contextualized) label/template and use similarity

# Transformer Encoder-Decoder (1)

- Original Transformer has both encoder and decoder, intended for Machine Translation
- Seq2seq model: given an input sequence of tokens, generate an output sequence of tokens
- This has been done before using e.g. LSTMs
- Decoder is similar to encoder:
- Adds another masked multi-head attention where only previous positions get attended to

# Transformer Encoder-Decoder (2)

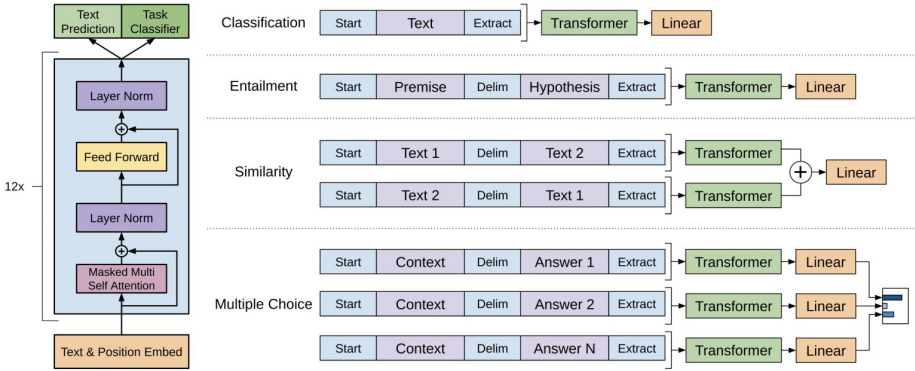
- Decoder outputs one token at at time: to generate the next output token, the token sequence generated so far (or the special start of sentence token SOS) gets fed to the input right shifted: causal autoregressive text generation.
- Output embedding goes through softmax to generate token probabilities
- Rather than always using the most probable token, use beam search to find better (higher overall probability) combinations of tokens
- Stop when the end of sentence (EOS) token is the most probable output




Cred: https://tinyurl.com/wm3chcaw

# Decoder only transformer: GPT-x (1)

- GPT (Radford2018): Use transformer decoder to pre-train language model then use fine-tuning on textual entailment, similarity, question answering, commonsense reasoning.
- 12 layers, 768-dim, 12 attention heads, 3072 hidden nodes for feed forward context size 512
- BPE tokenization



# Decoder only transformer: GPT-x (2)

- GPT-2 (Radford2019): like GPT with small modifications (layer norm at beginning of sub-block, final normalization after attention, context size is now 1024.
- GTP-3 (Brown2020): like GPT-2, but alternating dense and sparse attention patterns in the layers.
  - 96 layers, 12288-dim, 96 attention heads, context window of 2048 tokens 3.2M batch size, 175B parameters
  - No fine-tuning of the base model
  - Pre-trained on ~500B tokens of text (93% English)
  - Shows remarkable zero-shot capabilities
  - Large Language Model (LLM): huge model, trained on huge dataset

# Decoder only transformer: GPT-x (3)

- GPT-3.5: no scientific papers, fewer information from OpenAI.
  - Based on GPT-3, fine-tuned and improved using RLHF (Reinforcement Learning from Human Feedback)
  - Several versions with different properties (context size 4k-16k, 1.3B-175B parms(?))
- GPT-4 (OpenAI2023a):
  - Multimodal (text and image inputs)
  - Trained on both publicly available and licensed data
  - Context size 32k tokens
  - Fine-tuned on multiple task and using RLHF
  - "No further details about the architecture (including model size)"
  - Parameters: unknown, estimated to be around 1T

# Reinforcement Learning from Human Feedback

- Start with pre-trained language model
- Optionally fine-tune on a variety of tasks
- Gather many prompts (e.g. from deployed LM users) and generate responses from current and possibly also other LMs
- Human annotators order responses in order of preference, generate a score using an Elo(chess tournament scoring)-like method
- Train a scoring (e.g. Elo-rating) model on the prompt/response + score data
- Use e.g. Proximal Policy Optimization (PPO) (or NLPO, A2C, ..) with a copy of the LM to tune some or all of its parameters (see https://huggingface.co/blog/deep-rl-ppo)

# Other LLMs (1)

- Number of pre-trained LLMs, both with and without fine-tuning (FT) or RLHF grows rapidly
- While recent models from OpenAI, Google are closed and restricted (CS/R), open source (OS) models start to appear on the scene (what does "open source" mean in that context though?)
- PaLM/PaLM2 (Chowdhery2022, Google2023a): (8B-)540B parms, 118 Layers, 48 heads, 18432-dim, trained on 780B token corpus (123 languages, 78% en, 24 proglangs)
- LLaMA (Touvron2023) and many others derived from it, OS, 7B-65B parms, 64 heads, 8192-dim, 80 layers, trained on 1.4T tokens (unknown language distr.)
- BLOOM (Scao2022): 176B parms, OS, Responsible AI License (RAIL), ROOTS corpus (1.6TB, 46 languages, 13 proglangs, ~70% non-English)

# Other LLMs (2)

- GALACTICA (Taylor2022) (FT): Intended for scientific knowledge. Public demo showed "hallucinations" (inventing competent sounding stuff) got pulled off. Trained on papers, code, KBs, .. 106B tokens. 125M-120B parms, 96 layers, 10240-dim, 80 heads
- Falcon (Penedo2023): improve pre-training data, filter, deduplicate, 600B token subset of 5000B token set is open-sourced. Model: 40B,7B parms, FT versions available, OS
- Sparrow (Glaese2022): RLHF for different aspects of for "good dialogue". 70B parms, can lookup information, CS/R
- LaMDA (Thoppilan2022): up to 137B, 64 layers, trained on documents, dialogs, 1.5T tokens (> 90% English). Allow to query knowledge sources, translator, calculator
- … and many, many others

# LLMs and LLM-based NLP: (some) Advantages & Issues

- (+) Huge pre-training improves overall performance on many tasks
- (+) Capable of solving many tasks with the same or just slightly modified model (zero/few-shot)
- (+) capable of resolving common ambiguities, e.g. Winograd Schema Challenge (Kocijan2022)
- (-) "Hallucinations": competent-sounding nonsense. Variation of often-seen (catastrophic) failure modes with ML, esp. DNNs
- (-) Hard to control bias, performance and bias depends on selection of TB of data
- (-) As with most ML-based approaches: does not know what it doesn't know
- (-) As with most DNN-based ML approaches: missing model explainability
- (-) Currently: not easily accessible (effort in hardware, data, know-how)
  ⇒ dependency on large-player models, APIs and solutions.

# Pre-training Corpora Issues

- Mostly "accessible" data is used (web crawls, wikipedia, fan-fiction ..)
- But most good data is restricted: copyrighted, licensed, hidden, not digital …
- Huge imbalance on data in non-English languages availability and usage

  Even worse for low-resource languages, local idioms and dialects
- Cleaning / filtering data already improves LLM (cf. Falcon)
- Imbalance on topics (freely) available: computer science versus social sciences, biology, …
- Problem of opinionated texts, propaganda, religions, ideologies
- Problem of time: facts often relate to a time span (who is the US president), new facts need to get constantly added: "life-long learning".

  Data may become obsolete, out-dated, wrong over time.

# Corpus Annotation

- Almost all the tasks we discussed need training material
- For supervised learning, human annotation is needed:
  - What is the POS tag of a word
  - What is a Named Entity in a sentence and which KB entry does it refer to?
  - Does Sentence A entail sentence B?
  - Is that text misogynist?  Does this tweet contain hate speech, directed at whom?
  - Is this a good and factually correct answer to a chat question?
  - ….
- For unsupervised pre-training we need large amount of quality documents across many different languages, domains and writing styles

# Quality Evaluation of Human Annotations (1)

- Quality annotations are created by at least 2 independent (expert) annotators who are trained on the task-specific annotation guidelines.
- Cases of disagreement are resolved by mutual agreement, a third annotator, or majority voting if there are more than 2 annotators.
- This requires a single truth which, however, does not exist in many real-world cases, e.g., what is considered misogynist, in which context, by whom, to which extent.

# Measuring Inter-annotator Agreement/Reliability (1)

Inter-annotator agreement (IAA) assesses

- The agreement among individual annotators on an annotation task
- Quality of annotation guidelines has an effect
- IAA reflects how likely the annotators are to achieve the same annotation results using the same guidelines.
- There is a number of measures ranging from simple (e.g., percent-agreement) to rather complex measures (e.g., Krippendorff' Alpha)

# Measuring Inter-annotator Agreement/Reliability (2)

- Percent agreement:
  - #annotations agreed upon/# annotations in total
  - 2 annotators annotating all data of  the same dataset
  - No account for chance agreement → may overestimate true agreement
  - No agreement on the significance of the numeric result
- Krippendorff's Alpha
  - Complex measure
  - Any number of annotators, no need to annotate all data by each annotator
  - Minimizes the effect of chance
  - Better general interpretability of numeric results, e.g.,
    > 0.9 generally acceptable, > 0.8 fair reliability, > 0.7 tolerable (cf. Nili2020);

# Quality of Human Annotations (2)

- Is the model noisy because of bad data quality?
  - E.g., low-paid workers annotating huge amounts of data
  - Non-expert annotators
- Do the data reflect a bandwidth of (task-relevant) views, opinions, conceptions?
  - Different groups of annotators may introduce very different biases, e.g. with culturally differing concepts as obscenity, offensiveness, sexism; Problem of diverse groups of annotators (different opinions, low IAA) vs. more homogenous groups
  - Ethically: Expected model output defines data collection and annotation; transparency is required, e.g., datasheets for datasets, model cards.

# Model Cards and Datasheets for Datasets

- Model Card

| | |
|---|---|
| Model Details: | basic information about the model, model version, type, training algorithms, developers |
| Intended Use: | use cases, users, out of scope uses |
| Data: | Evaluation Data, Training Data |
| Model evaluation: | Factors, Metrics, Quantitative Analyses |
| Ethical Considerations: | if, how and to which extent the use of the specific model may affect human life, what would be likely harms due to model usage and which user groups might be potentially affected, which risk mitigation strategies were used during model development |
| Caveats and Recommendations: | was the model development reviewed by an external board, testing with a specific user community represented in the evaluation dataset, additional recommendations for model use, ideal characteristics of an evaluation dataset |

Table 1. High-level overview of the structuring of model cards

- Datasheet for Dataset

| | |
|---|---|
| Motivation: | purpose of dataset, creators, customers, funding body |
| Composition: | detailed description of the composition of the dataset, including information on completeness, annotation, recommended splits in training/development/test data etc. |
| Collection process: | detailed data acquisition process |
| Preprocessing/cleaning/ labelling: | how was the data cleaned and preprocessed, is the raw data also available, are the data automatically labelled, is the used software available |
| Uses: | uses of the dataset, potential future uses, unintended uses |
| Distribution: | availability of dataset, license, IP-based restrictions of use |
| Maintenance: | who is supporting/hosting/maintaining the dataset, are previous versions available, can third parties augment/extend/contribute to the dataset, are respective processes in place |

AIML Center for Artificial Intell and Machine Learning

# Quality of Annotated Datasets

- For which languages is effort invested?
- For which topics/tasks are datasets created?
- Life after annotation: on the one hand want static corpora for ML evaluation and comparison. On the other hand want to correct mistakes, improve. Also: obligation to change, e.g. remove deleted tweets

# Dealing with annotator disagreement

- Usual view: one true label, annotator disagreement a sign of error (noise)
- Usual strategy: use majority label, adjudicate, get additional annotations
- But for many complex concepts there is no definite single true label
- Get a label distribution p(y|x) which the model should learn: soft-labels
- Learning with Disagreements (see Uma2021, Leonardelli2023)
- Perform model calibration (Minderer2021) to human uncertainty (Baan2022)
- Also needs different evaluation scores: cross-entropy, correlation, KL-divergence, …
- Similar issues with machine translation: which of many translations for a text are acceptable/correct, and what is a good amount of variation between them?
- One major problem: low number of annotator labels may not reflect population distribution

# Opinionated Short List of Related NLP Tools

- Huggingface (P): transformer-related code, model-zoo, data
  Pytorch (P): Deep Learning Library
- Spacy(P), Stanza(P), CoreNLP (J): pipeline-based NLP, models for several languages
- Online services: GoogleNLP, Perspective, ELG
- GateNLP (P): integrates most of the above, good abstractions
- Gensim (P), fastText (C+P), AllenNLP (P), FLAIR (P)
- Label-studio, Teamware, Amazon Mechanical Turk: human annotation
- Scipy, Scikit-Learn: lower level Python packages

P=Python, J=Java, C=C/C++

# Outlook

- Main progress: highly contextualized representations through massive pretraining, leading to large improvements in performance on traditional tasks and impressive new systems like conversational AI
- But also makes understanding and researching these more important:
  - Knowing not to know
  - Explanation of results on both the domain and model level
  - How to evaluate models trained on a huge area of topics, facts, knowledge?
  - How to make models and research more democratic: who can afford to train a 1T parameters model?
  - How to deal with opinions, ideologies, disagreement, emotions represented in those models?
  - Representation grounding: link concepts to images, videos, sensory data, robot-actions
- NLP/ML research has become more interesting than ever!

# THE END

Thank you!

# References (1)


Andor2016: Andor, Daniel, et al. "Globally Normalized Transition-Based Neural Networks." *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 2016.

Baan2022: Baan, Joris, et al. "Stop Measuring Calibration When Humans Disagree." Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. 2022.

Bojanowski2017: Bojanowski, Piotr, et al. "Enriching word vectors with subword information." *Transactions of the association for computational linguistics* 5 (2017): 135-146.

Brown1992: Brown, Peter F., et al. "Class-based n-gram models of natural language." *Computational linguistics* 18.4 (1992): 467-480.

Brown2020: Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.

Brunner2019: Brunner, Gino, et al. "On the validity of self-attention as explanation in transformer models." arXiv preprint arXiv:1908.04211 40 (2019).

# References (2)


Chowdhery2022: Chowdhery, Aakanksha, et al. "Palm: Scaling language modeling with pathways." arXiv preprint arXiv:2204.02311. 2022.

Devlin2019: Devlin, J.; Chang, M.-W.; Lee, K. & Toutanova, K. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language (NAACL2019).* 2019.

Doddington2004: Doddington, George R., et al. "The automatic content extraction (ace) program-tasks, data, and evaluation." *Lrec*. Vol. 2. No. 1. 2004.

Galstyan2007: Galstyan, Aram, and Paul R. Cohen. "Empirical comparison of “hard” and “soft” label propagation for relational classification." *International Conference on Inductive Logic Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

Glaese2022: Glaese, Amelia, et al. "Improving alignment of dialogue agents via targeted human judgements." *arXiv preprint arXiv:2209.14375* (2022).

# References (3)


Google2023a: PaLM 2 Technical Report. 2023.

Grishman1996: Grishman, Ralph, and Beth M. Sundheim. "Message understanding conference-6: A brief history." COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics. 1996.

Hochreiter1997: Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Iyyer2015: Iyyer, Mohit, et al. "Deep unordered composition rivals syntactic methods for text classification." *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. 2015.

Kiim2014: Yoon Kim. "Convolutional Neural Networks for Sentence Classification". *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. 2014.

Kocijan2022: Kocijan, Vid, et al. "The defeat of the winograd schema challenge." arXiv preprint arXiv:2201.02387 (2022).

# References (4)


Kolitsas2018: Kolitsas, Nikolaos, Octavian-Eugen Ganea, and Thomas Hofmann. "End-to-End Neural Entity Linking." *Proceedings of the 22nd Conference on Computational Natural Language Learning*. 2018.

Krippendorff2011: Krippendorff, Klaus. "Agreement and information in the reliability of coding." *Communication methods and measures* 5.2 (2011): 93-112.

Kudo2018a: Kudo, Taku. "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates." *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018.

Kudo2018b: Kudo, Taku, and John Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing." *EMNLP 2018* (2018): 66.

Lafferty1992: Lafferty, John, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." (2001).

Leonardelli2023: Leonardelli, Elisa, et al. "SemEval-2023 Task 11: Learning With Disagreements (LeWiDi)." arXiv preprint arXiv:2304.14803 (2023).

Li2020: Li2020: Li, Jing, et al. "A survey on deep learning for named entity recognition." *IEEE Transactions on Knowledge and Data Engineering* 34.1 (2020): 50-70.

# References (5)


Marcus1993: Marcus, Mitchell, Beatrice Santorini, and Mary Ann Marcinkiewicz. "Building a large annotated corpus of English: The Penn Treebank." (1993).

Mihalcea2004: Mihalcea, Rada, and Paul Tarau. "Textrank: Bringing order into text." Proceedings of the 2004 conference on empirical methods in natural language processing. 2004.

Mikolov2013: Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems* 26 (2013).

Minderer2021: Minderer, Matthias, et al. "Revisiting the calibration of modern neural networks." Advances in Neural Information Processing Systems 34 (2021): 15682-15694.

Nadeau2007: Nadeau, David, and Satoshi Sekine. "A survey of named entity recognition and classification." Lingvisticae Investigationes 30.1 (2007): 3-26.

Nili2020: Nili, Alireza, et al. "An approach for selecting and using a method of inter-coder reliability in information management research." *International Journal of Information Management* 54 (2020): 102154.

# References (6)


OpenAI2023a: OpenAI. “GPT-4 Technical Report.” *ArXiv* abs/2303.08774. 2023.

Penedo2023: Penedo, Guilherme, et al. "The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only." *arXiv preprint arXiv:2306.01116* (2023).

Pennington2014: Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.

Peters2018: Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations." *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237. 2018.

Radford2018: Radford, Alec, and Narasimhan, Karthik, and Salimans, Tim, and Sutskever, Ilya. "Improving language understanding by generative pre-training." 2018.

# References (7)


Radford2019: Radford, Alec, and Wu, Jeffrey, and Child Rewon, and Luan, David, and Amodei, Dario, and Sutskever, Ilya. "Language models are unsupervised multitask learners." *OpenAI blog* 1.8. 2019

Scao2022: Scao, Teven Le, et al. "Bloom: A 176b-parameter open-access multilingual language model." *arXiv preprint* *arXiv:2211.05100*. 2022.

Schuster2012: Schuster, Mike, and Kaisuke Nakajima. "Japanese and korean voice search." *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2012.
Sevgili2022: Sevgili, Özge, et al. "Neural entity linking: A survey of models based on deep learning." *Semantic Web* 13.3 (2022): 527-570.

Sennrich2016: Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units." *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016.
Schmidhuber1997: Schmidhuber, Jürgen, and Sepp Hochreiter. "Long short-term memory." *Neural Comput* 9.8 (1997): 1735-1780.

Taylor2022: Taylor, Ross, et al. "Galactica: A large language model for science." *arXiv preprint* *arXiv:2211.09085*. 2022.

# References (8)


Thoppilan2022: Thoppilan, Romal, et al. "Lamda: Language models for dialog applications." arXiv preprint arXiv:2201.08239 (2022).

Toutanova2000: Toutanova, K. & Manning, C. D. "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger." *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*. 2000.

Touvron2023: Touvron, Hugo, et al. "Llama: Open and efficient foundation language models." *arXiv preprint arXiv:2302.13971*. 2023.

Uma2021: Uma, Alexandra, et al. "Semeval-2021 task 12: Learning with disagreements." Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021). 2021.

Vasvani2017: Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Ye2023: Ye, Junjie, et al. "A comprehensive capability analysis of gpt-3 and gpt-3.5 series models." arXiv preprint arXiv:2303.10420 (2023).

Yu2023: Yu, Lili, et al. "Megabyte: Predicting million-byte sequences with multiscale transformers." *arXiv preprint arXiv:2305.07185* (2023).