Overview

23. Natural Language Processing

- **Language models**: Probabilistic distribution describing the likelihood of any string
- Grammar
- Parsing
- Augmented grammars
- Complications of real natural language

Language models (1)

- **Bag-of-words model**: Application of naive Bayes to strings of words for classifying sentences in categories
  - Each word has a probability for the categories extracted from training data
  - Deciding what a word is: **Tokenization**
  - Word standardization by **stemming** (without lexicon) and **lemmatization** (with lexicon)
- **N-gram word models**: Use context of words
  - The probability of a word is dependant on the n-1 previous words
  - **Skip-gram** model: count words that are near each other but skip a word between them
  - **Smoothing** n-gram models: Dealing with rare words in n-grams: replace them by special symboles (e.g. <UNK> or <NUM> or <Email> etc.)
- **Character-level model**: Probability of each character of a word is dependant on n-1 previous characters (helpful e.g. for dealing with unknown words or language identification)

Language models (2)

- **Word representations**
  - **Dictionary**: Atomic model explaining (nearly) all words of a language
    - **WordNet**: Open-source dictionary in machine-readable form
  - **Word Embeddings**: Factored model representing a word by a vector, thus capturing similarities among words (learned by e.g. neural nets)
- **Part-of-speech (POS) tagging**: Assigning a lexical category or tag to each word, e.g. noun, verb, adjective etc.
  - **Penn Treebank**: Corpus of ca. 3 million words annotated with 45 tags
    - Includes grammar too
  - Algorithms for POS-tagging: Viterbi algorithm, logistic regression with greedy search, …

Grammar

- **Grammar:** Set of rules that define a tree structure of allowable phrases
- **Language:** Set of sentences that follow those rules
- Natural language does not have an exact grammar, but syntactic categories (**phrase structure**) help to extract the semantics of a sentence.
- **Probabilistic context-free grammar** (PCFG):
  - Assigns a probability to each string, e.g. Adjs → Adjective [0.8] | Adjective Adjs [0.2]
- **Lexicon:** List of allowable words for each lexical category
  - Open classes (e.g. nouns, names, verbs, adjectives, adverbs): unlimited list of words
  - Closed classes (e.g. pronouns, articles, prepositions, conjunctions): enumeration of words
- Learning a grammar (manual grammar specification very complex):
  - Learning from an annoted corpus (like Penn Treebank) or semi- or unsupervised
    - e.g. **Curriculum Learning**: Start with simple sentences (e.g. „He left.“) and move stepwise to more complex sentences

Example for Simple Grammar and Lexicon

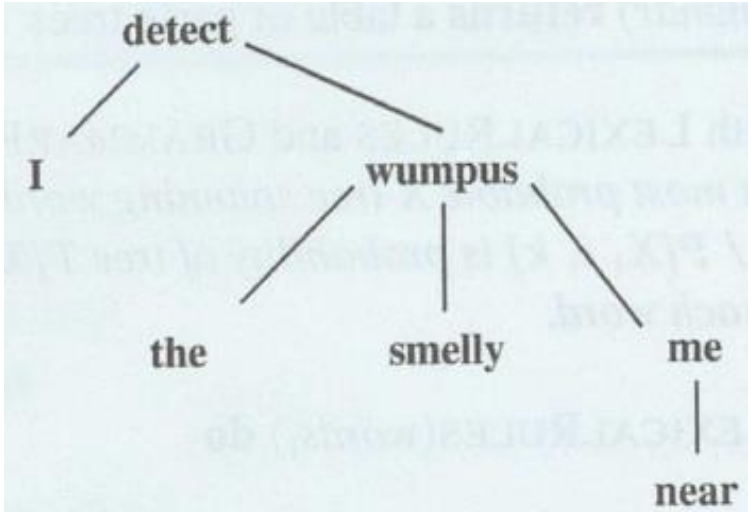
S	$\rightarrow$	*NP VP*	[0.90]	I + feel a breeze
	|	*S Conj S*	[0.10]	I feel a breeze + and + It stinks
NP	$\rightarrow$	*Pronoun*	[0.25]	I
	|	*Name*	[0.10]	Ali
	|	*Noun*	[0.10]	pits
	|	*Article Noun*	[0.25]	the + wumpus
	|	*Article Adjs Noun*	[0.05]	the + smelly dead + wumpus
	|	*Digit Digit*	[0.05]	3 4
	|	*NP PP*	[0.10]	the wumpus + in 1 3
	|	*NP RelClause*	[0.05]	the wumpus + that is smelly
	|	*NP Conj NP*	[0.05]	the wumpus + and + I
VP	$\rightarrow$	*Verb*	[0.40]	stinks
	|	*VP NP*	[0.35]	feel + a breeze
	|	*VP Adjective*	[0.05]	smells + dead
	|	*VP PP*	[0.10]	is + in 1 3
	|	*VP Adverb*	[0.10]	go + ahead
Adjs	$\rightarrow$	*Adjective*	[0.80]	smelly
	|	*Adjective Adjs*	[0.20]	smelly + dead
PP	$\rightarrow$	*Prep NP*	[1.00]	to + the east
RelClause	$\rightarrow$	*RelPro VP*	[1.00]	that + is smelly

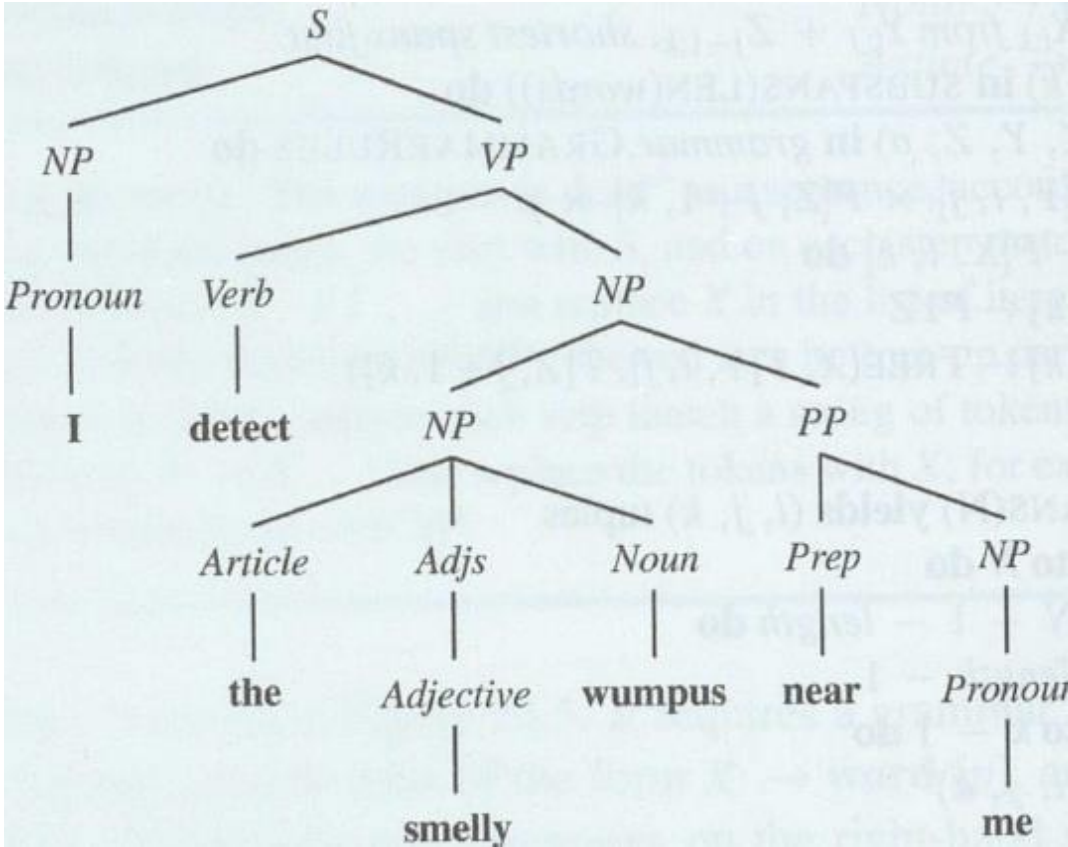
- Simple PCFG with examples from Wumpus world
- Part of the lexicon

Noun	$\rightarrow$	**stench** [0.05] | **breeze** [0.10] | **wumpus** [0.15] | **pits** [0.05] | …
Verb	$\rightarrow$	**is** [0.10] | **feel** [0.10] | **smells** [0.10] | **stinks** [0.05] | …
Adjective	$\rightarrow$	**right** [0.10] | **dead** [0.05] | **smelly** [0.02] | **breezy** [0.02] …
Adverb	$\rightarrow$	**here** [0.05] | **ahead** [0.05] | **nearby** [0.02] | …
Pronoun	$\rightarrow$	**me** [0.10] | **you** [0.03] | **I** [0.10] | **it** [0.10] | …
RelPro	$\rightarrow$	**that** [0.40] | **which** [0.15] | **who** [0.20] | **whom** [0.02] | …
Name	$\rightarrow$	**Ali** [0.01] | **Bo** [0.01] | **Boston** [0.01] | …
Article	$\rightarrow$	**the** [0.40] | **a** [0.30] | **an** [0.10] | **every** [0.05] | …
Prep	$\rightarrow$	**to** [0.20] | **in** [0.10] | **on** [0.05] | **near** [0.10] | …
Conj	$\rightarrow$	**and** [0.50] | **or** [0.10] | **but** [0.20] | **yet** [0.02] | …
Digit	$\rightarrow$	**0** [0.20] | **1** [0.20] | **2** [0.20] | **3** [0.20] | **4** [0.20] | …

Parsing

- The process of analyzing strings of words to uncover the phrase structure
  - Can be viewed as search for a valid parse tree
- Parse strategies for hierarchical parsers (right):
  - Pure top-down and bottom-up parsers quite inefficient
  - **Chart parsers** parse partial phrases in charts and reuse them in search tree (dynamic programming): Example: **CYK algorithm**
- **Dependency Parsers:** Each lexical item (word) has a binary relation to another lexical item without syntactic constituents (below)





Augmented Grammars

- Add constraints to context-free grammars
  - Example: „I ate a banana.“ vs. „Me ate a banana.“
  - Pronouns can be augmented with features like „subjective case, first person singular“
  - Called **subcategorization**
- Probabilities of parse trees may depend on more than just syntactic categories
  - e.g. „ate a banana“ versus „ate a bandanna“
  - Notion of the **head of phrase** (noun phrase, verb phrase etc.)

Semantic Interpretation

- Semantic might be expressed in predicate logic
  - e.g. the string „Ali loves Bo“ has the semantic interpretation „loves(Ali, Bo)“
- Grammar must be extended to include semantic rules
  - However, Penn Treebank does not contain semantic categories
- Alternate: Learn semantic grammar from question/answer pairs, e.g.:
  - Question: „What states border Texas?“ $\rightarrow$ Logical form: $\lambda x.\text{state}(x) \wedge \lambda x.\text{borders}(x, \text{Texas})$
  - Answer: „Louisiana, Arkansas, Oklahoma, New Mexico“

Complications of Real Natural Language

- Grammar of real languages like English or German endlessly complex
  - **Quantification:** „Every agent feels a breeze.“ → The same or a different breeze?
  - **Pragmatics:** Adding context to an interpretation
    - „I am in Berlin today“→ Resolve „I“ and „today“
    - Recognize the speakers intent in an utterance, e.g. „Go to 2,2“ → the speech act is a command (others: question, statement, promise, warning etc.)
  - **Long distance dependencies:** „Who did the agent tell you to give the gold to?“
  - **Time and tense:** „Ali loves Bo“ versus „Ali loved Bo“
  - **Ambiguity:** Nearly every sentence is ambigue with multiple possible parses
    - Lexcial ambiguity: Word with different meaning, e.g. „back“ or „Jack“
    - Syntactic ambiguity: Different parses, e.g. „I smelled a Wumpus in 2,2“
    - Semantic ambiguity: Different meaning, where the Wumpus really is
    - Metonymy: One object is used to stand for another, e.g. „VW announced a new model“
      - Methaphor: Special kind of Metonymy

Disambiguation

- Process of recovering the most probable intended meaning of an utterance
  - Probabilities associated with rules help, but are not sufficient
  - Further models necessary:
    - World model: Likelihood that a proposition occurs in the world, e.g. „I‘m dead“
    - Mental model: Likelihood that the speaker forms the intention of communicating a certain fact to the hearer, e.g. „I am not a crook“
    - Language model: Likelihood that a certain string of words will be chosen, given that the speaker has the intention of communicating a certain fact
    - Acoustic model (for spoken communication)

Natural Language Tasks

- Speech recognition (Speech-to-text)
- Text-to-speech
- Machine translation
- Information extraction
- Information retrieval
- Question Answering
- …

- Chat bots, project debater, …

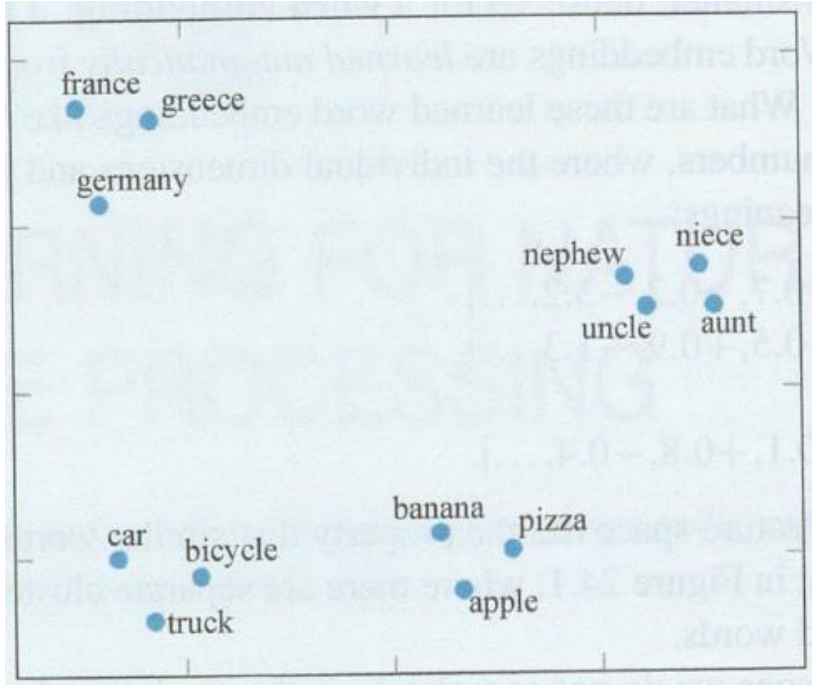
24. Deep Learning for Natural Language Processing

- Word Embeddings
- RNN for NLP
- Sequence-to-Sequence Models
- Transformer Architecture
- Pretraining and Transfer Learning
- State of the Art

Word Embeddings

„You shall know a word by the company it keeps“ (J. Firth, 1957)

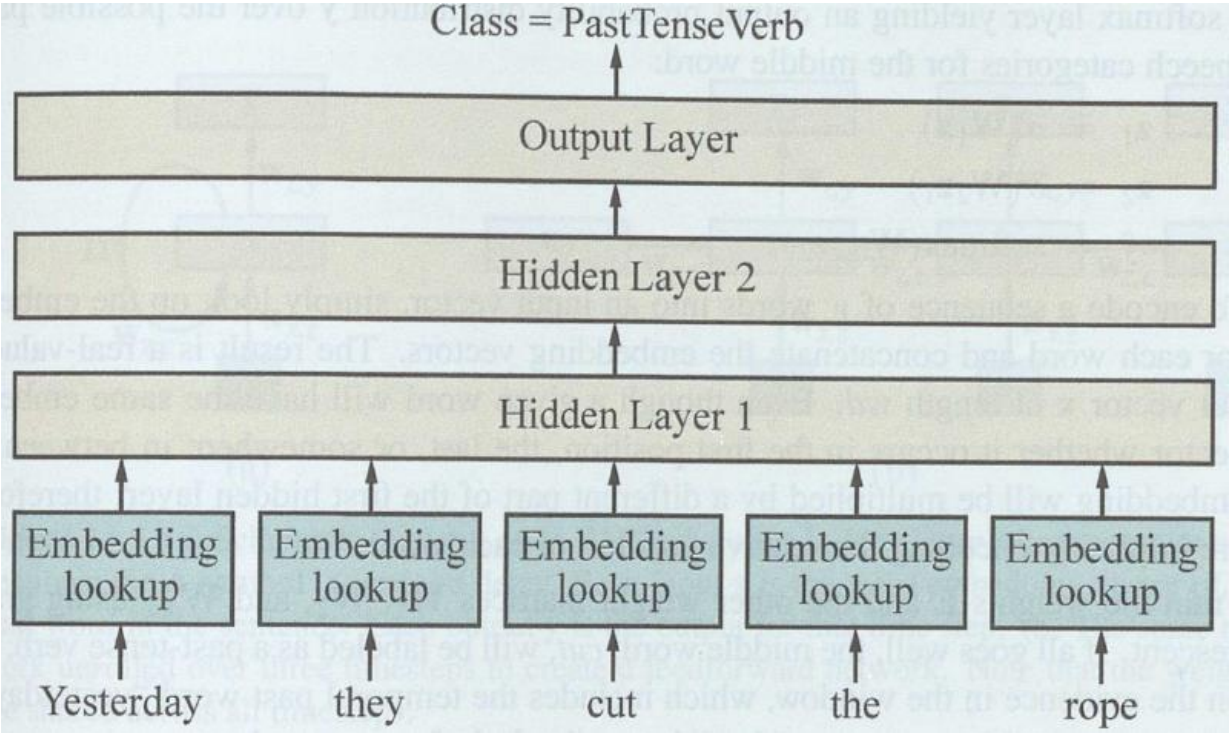
- N-gramm count of all phrases that the word appears in → large table, most entries 0
- Generalization to smaller size vector: **Word embedding**, e.g.:
  - „aardvark“ = [-0.7, +0.2, -3.2, …]
  - „abacus“ = [+0.5, +0.9, -1.3, …]
  - ….
  - „zyzzyvy“ = [-0.1, +0.8, -0.4, …]
- Similar words end up having similar vectors
- Can be used to solve word analogy problems like
  - Athens is to Greece as Oslo is to [what]?
  - In vector notation: B – A = D - C or: D = C + (A - B)



- Commonly used vector dictionaries: Word2Vec, GloVe (Global Vectors), FastText, BERT, …

Training Word Embeddings

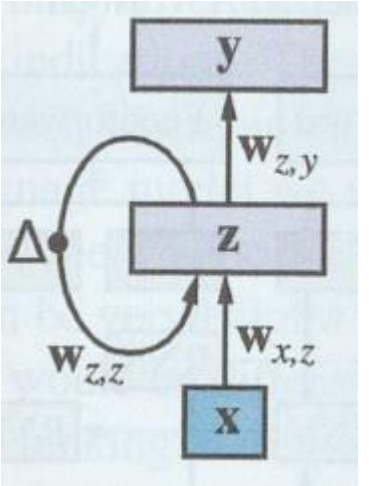
- Often starts with a pretrained (other) models and adaption to a specialized domain
- Often done as side effect to a particular task
- Example: POS-tagging with word embeddings
  - Learn embeddings and POS simultaneously



1. Choose width w (odd number of words), e.g. 5
2. Create vocabulary for all frequent word tokens (e.g. frequency > 5); let *v* = # words
3. Sort vocabulary in any arbitrary order
4. Choose value *d* of the size of each word embedding vector
6. Create new v-by-d weight matrix E (word embedding matrix) initialized randomly or pretrained
7. Set up neural network that outputs a POS tags (e.g. with 2 hidden and softmax output layers)
   - Train weights E and weight matrices W_1, W_2, W_{out} with sequences of w words as input vector

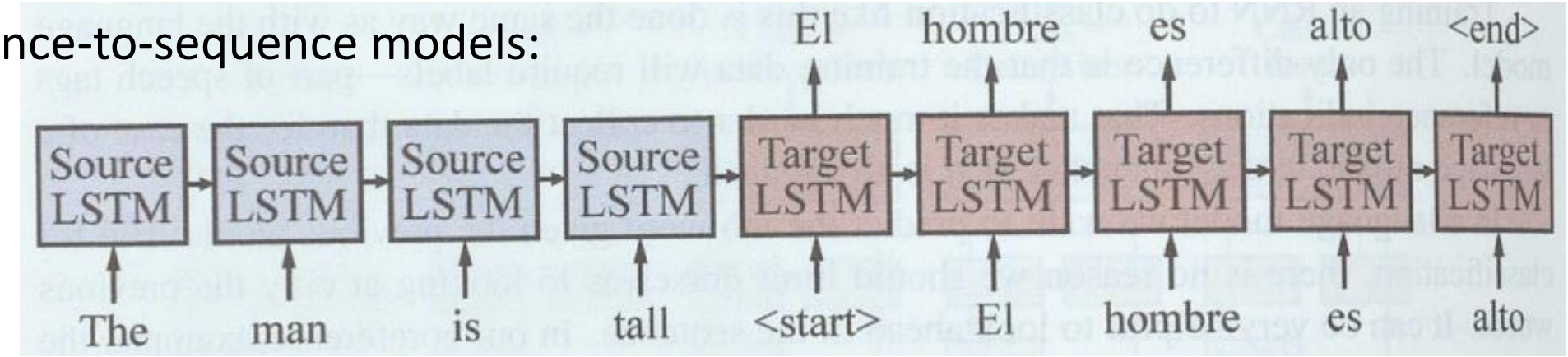
Recurrent Neural Networks for NLP

- POS-tagging can be done with a small fixed size window of perhaps 5 words
- Other NLP-tasks need a larger context, e.g. reference resolution:
  - „*Felix told me that Ben was very sick so I took **him** to the hospital*“ (him → Ben, not Felix)
  - Fixed size window unpractical (too many parameters)
  - **RNN architecture** has internal state, which can represent the relevant context
    - To capture context after the relevant word, use a bidirectional RNN
    - Can also be used for classification tasks like sentiment analysis
  - **LSTMs (Long Short Term Memory)** architecture with gating units for the memory cell (internal state) often better



Sequence-to-Sequence Models

- Example domain: Machine translation (MT)
- Word-by-word translation problematic:
  - big dog (english) → perro grande (spanish; reversed order)
  - seahorse → caballo de mar
  - Subject of sentence at the beginning (english), → at the end (Fijan language)
- Much better: Sequence-to-sequence models:



- But RNN/LSTM not sufficient:
  - **Nearby context bias:** RNN has a bias for the nearby context in its hidden state
  - **Fixed context size limit:** Hidden state has a fixed size; but the context is variable
  - **Slower sequential processing:** RNNs are rather slow (training one word at a time)

Attention

- For many tasks like translation, we need a better context than just one hidden state
  - Without a huge increase of weights
- The target RNN (LSTM) must pay attention to different parts of the source for every word
- New component **attention** for a context-based summarization of the source sentence into a fixed-dimensional representation
  - Context vector $\mathbf{c}_i$ contains most relevant information for generating the next target word
  - Will be used as additional input (to x_i) of the target RNN (h_{i-1} target vector for predicting h_i)

$$\mathbf{h}_i = \text{RNN}(\mathbf{h}_{i-1}, [\mathbf{x}_i;\mathbf{c}_i])$$

  - $[\mathbf{x}_i;\mathbf{c}_i]$ is the concatenation of the input and context vectors:

$c_i = \sum_j a_{ij} \cdot s_j$ // context vector as weighted average over source vector word s_j

$a_{ij} = e^{r_{ij}} / (\sum_k e^{r_{ik}})$ // normalized attention scores (probabilities) using softmax over all words

$r_{ij} = h_{i-1} \cdot s_j$ // raw attention score for the target state h_{i-1} and source vector word s_j

Example for Attention

$\mathbf{h}_i = \text{RNN}(\mathbf{h}_{i-1}, [\mathbf{x}_i;\mathbf{c}_i])$

where $[\mathbf{x}_i;\mathbf{c}_i]$ is the concatenation of the input and context vectors:

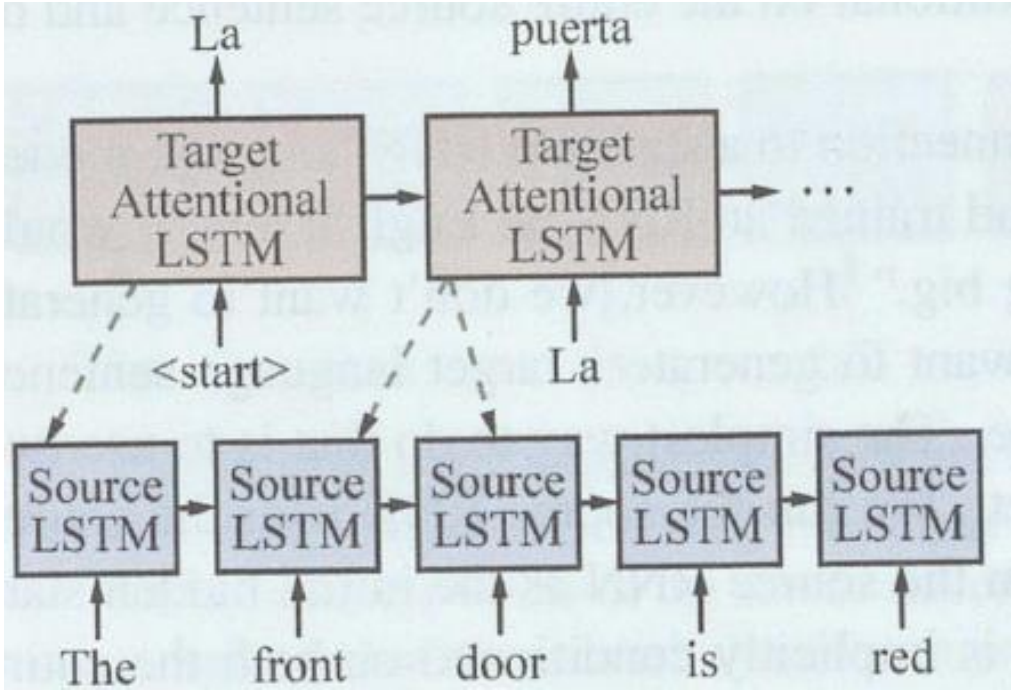
$c_i = \sum_j a_{ij} \cdot s_j$ // context vector as weighted average over source vector word s_j

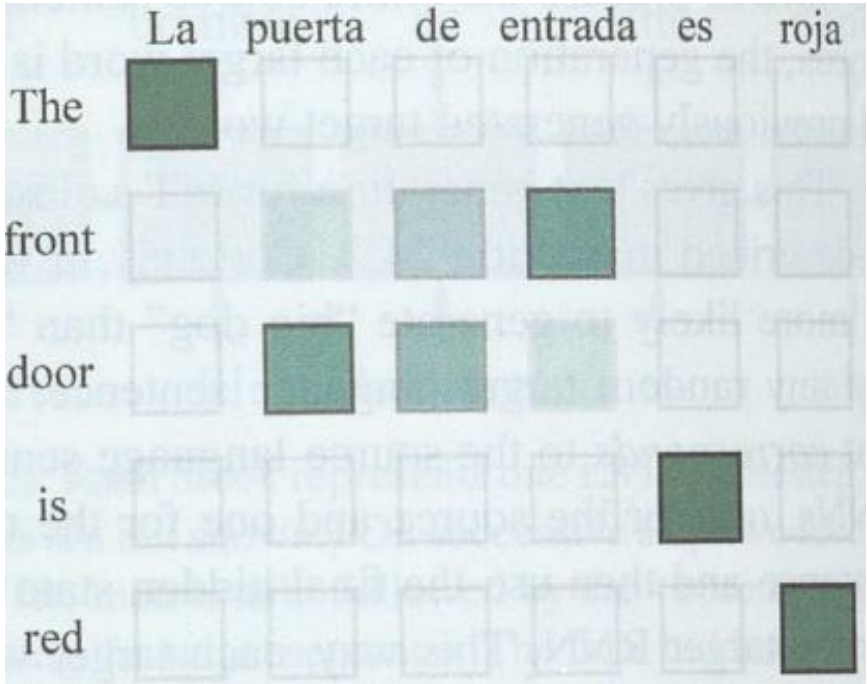
$a_{ij} = e^{r_{ij}} / (\sum_k e^{r_{ik}})$ // normalized attention scores (probabilities) using softmax over all words

$r_{ij} = h_{i-1} \cdot s_j$ // raw attention score for the target state h_{i-1} a source vector word s_j

left: dashed lines represent attention

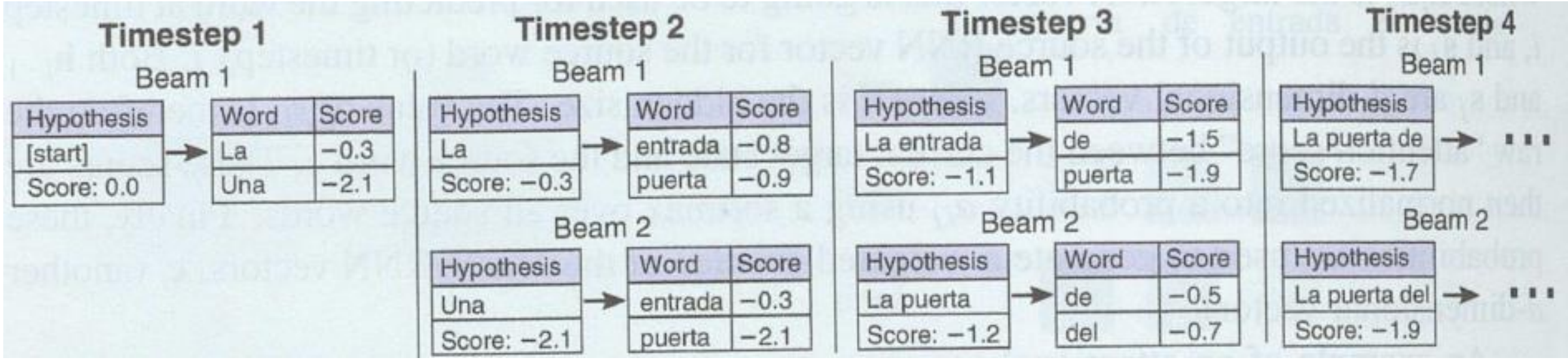
right: Example for attention probability matrix (darker boxes representing higher values for a_{ij})





Decoding

- At training time: maximize the probability of each word in the target training sequence from the source sequence and the previous target words
- **Greedy Decoding:** Generate the target sequence one word at a time and select at each timestep the highest probability word
  - Problem: Might not be optimal for the whole sequence
- **Beam Search Decoding:** Keep the top k hypotheses at each state, extending each by one word using the top k choices of words and then continue with the best k of the k^2 hypotheses

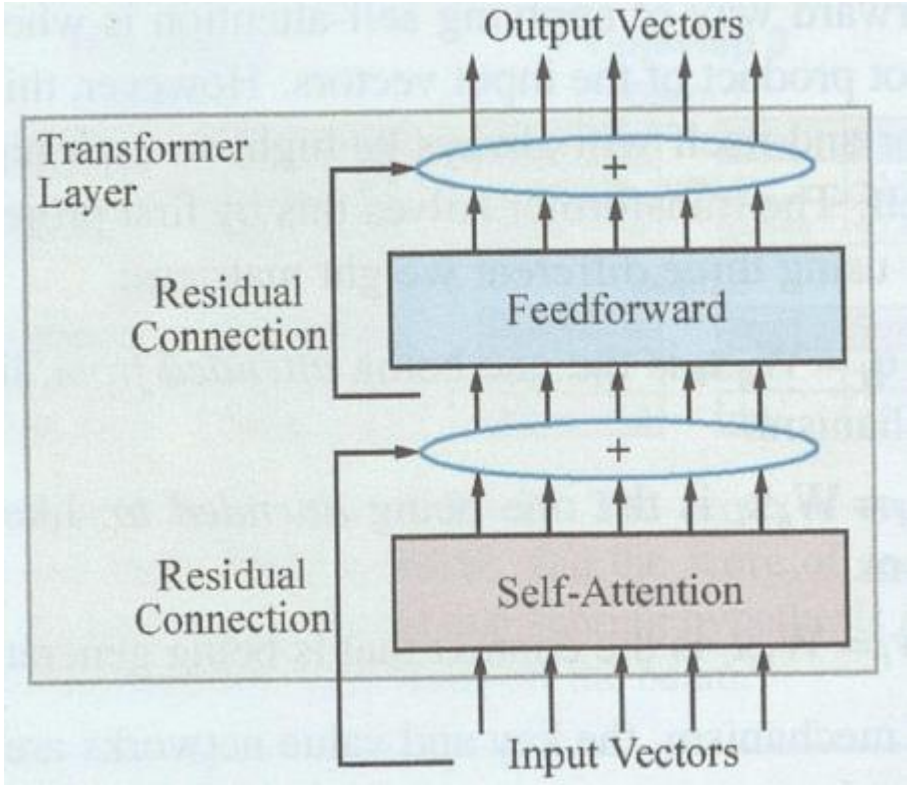


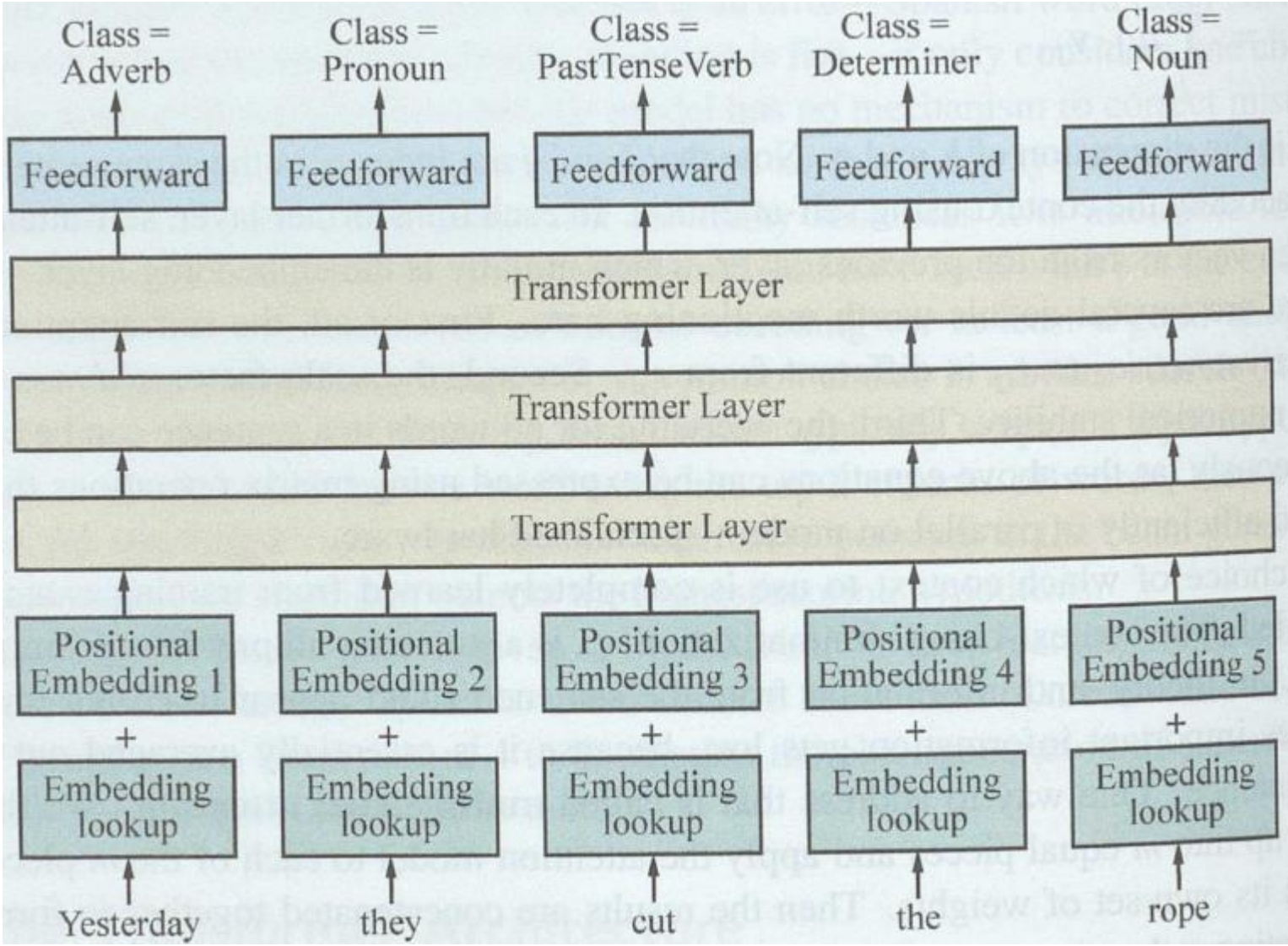
Transformer Architecture (1)

- Key concept of tranformer architecture: **Self-attention**:
  - Allows to model long-distance context with a sequential dependency
  - Extends the attention mechanism (from target RNN to source RNN) so that each sequence of hidden states also attends to itself (source to source, target to target)
  - Applying self-attention with a matrix just formed by a dot product of the input vector not optimal, since a dot product between a vector and itself is always (too) high
    - Therefore, the input is first projected into 3 different representations using three different weight matrices:
      - **Query vector** $\mathbf{q}_i = \mathbf{W}_q\mathbf{x}_i$ // attended from, like the target in standard attention
      - **Key vector** $\mathbf{k}_i = \mathbf{W}_k\mathbf{x}_i$ // attended to, like the source in standard attention
      - **Value vector** $\mathbf{v}_i = \mathbf{W}_v\mathbf{K}_i$ // is the context that is being generated
  - In long sequences, information is averaged over the whole sentence and might get lost
    - **Multiheaded attention** divides the sentence in m equal pieces to which attention is applied

Transformer Architecture (2)

- Right: Transformer for POS-Tagging
- Below: One transformer layer consists of self attention, feedforward network and residual connections



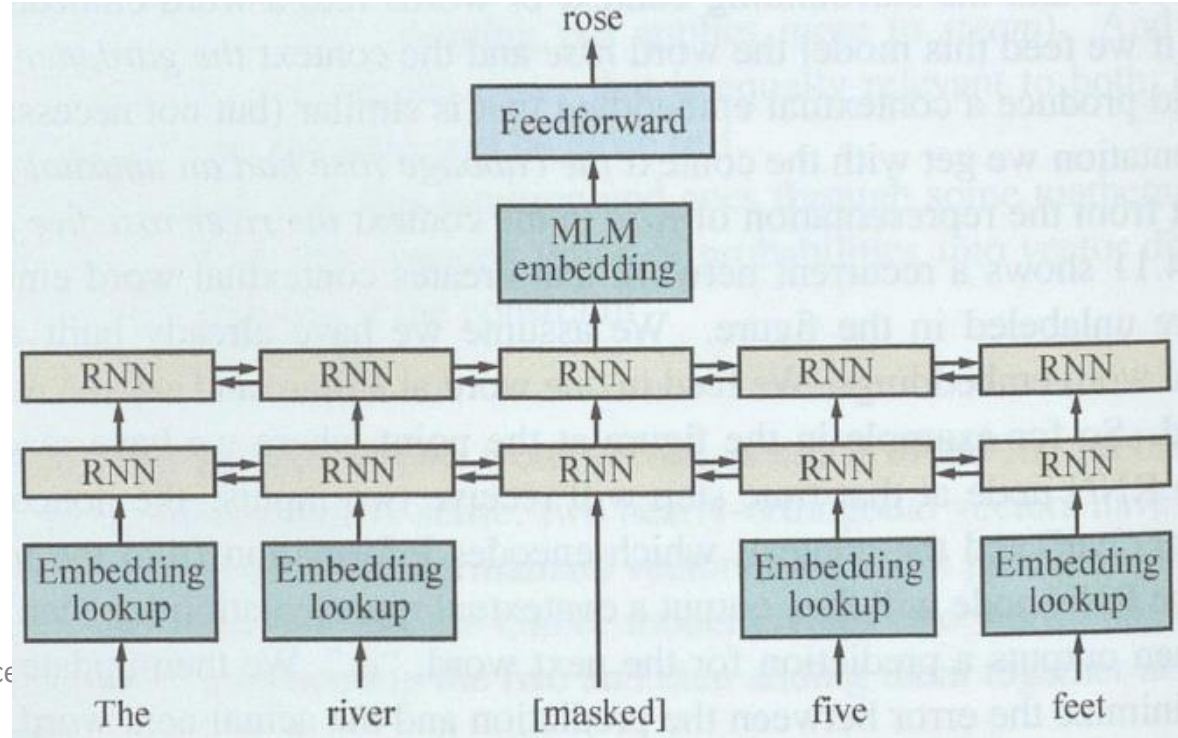


Pretraining and Transfer Learning (1)

- In image classification, lots of data (images) with labels are available (e.g. ImateNet)
- For natural language, labeled data is only partially available, e.g.
  - Any text for building n-grams or word embeddings
  - FAQs: Question-answer pairs
  - Machine translation: Translated text e.g. from international organizations
  - Reviews: Text reviews in addition to 5-star rating system
- Goal: Reduce training for specific tasks by using and adapting a general pretrained model
  - Till now: Word embeddings were created as a side effect for e.g. POS-training, but now we want to avoid any labeling for the pretrained models

Pretrained Word Embeddings

- Based on co-occurrence of words
- Based on co-occurence with third words: Two words are similar, if they both appear in the context of the same other words (**GloVe** approach: *Glo*bal *Ve*ctors)
  - e.g. *ice* and *steam*: Word *solid* co-occurs more with *ice*, *gas* more with *steam* and *water* with both
  - Compute $P_{w,ice}/P_{w,steam}$ for many third words w
  - Convert the ratio to vectors fulfilling the embedding constraint $E_i \cdot E_j = \log(P_{ij})$
- Based on **masked language models (MLM)**: pretrain a bidirectional model by masking one word input and predict the masked word from the context (right)
- Learn different embeddings for ambiguous words by including the context of the word (e.g. *rose* as flower or as verb in past tense)



State of the Art in Deep NLP

- „NLP's ImageNet moment has arrived" (Ruder 2018)
  - For computer vision 2012 with deep learning
  - Since 2018, new NLP projects typically start with a pretrained transformer model
- GPT2 can generate fairly convincing text given just a few words as prompt (successor: GPT3)

It is a truth universally acknowledged that the earth is not the center of the universe. There are those who assert there is. I do not accept them, but others I consider to be of the same opinion. The truth is, however, that if there are other than the center, and if there are any other living things in the universe and if they are not human, then we do not yet have our answers. We have to go on. This page gives a simplified, simplified answer to the problem. We don't have all the answers. The truth is, however, that the truth is out there.

- RoBERTa model with some fine tuning achieves state-of-the-art results for question answering and reading comprehension tests
- T5 (Text-to-Text Transfer Transformer) produces textual responses to various kinds of textual input including translation

Still Need for Grammars, Parsing etc.?

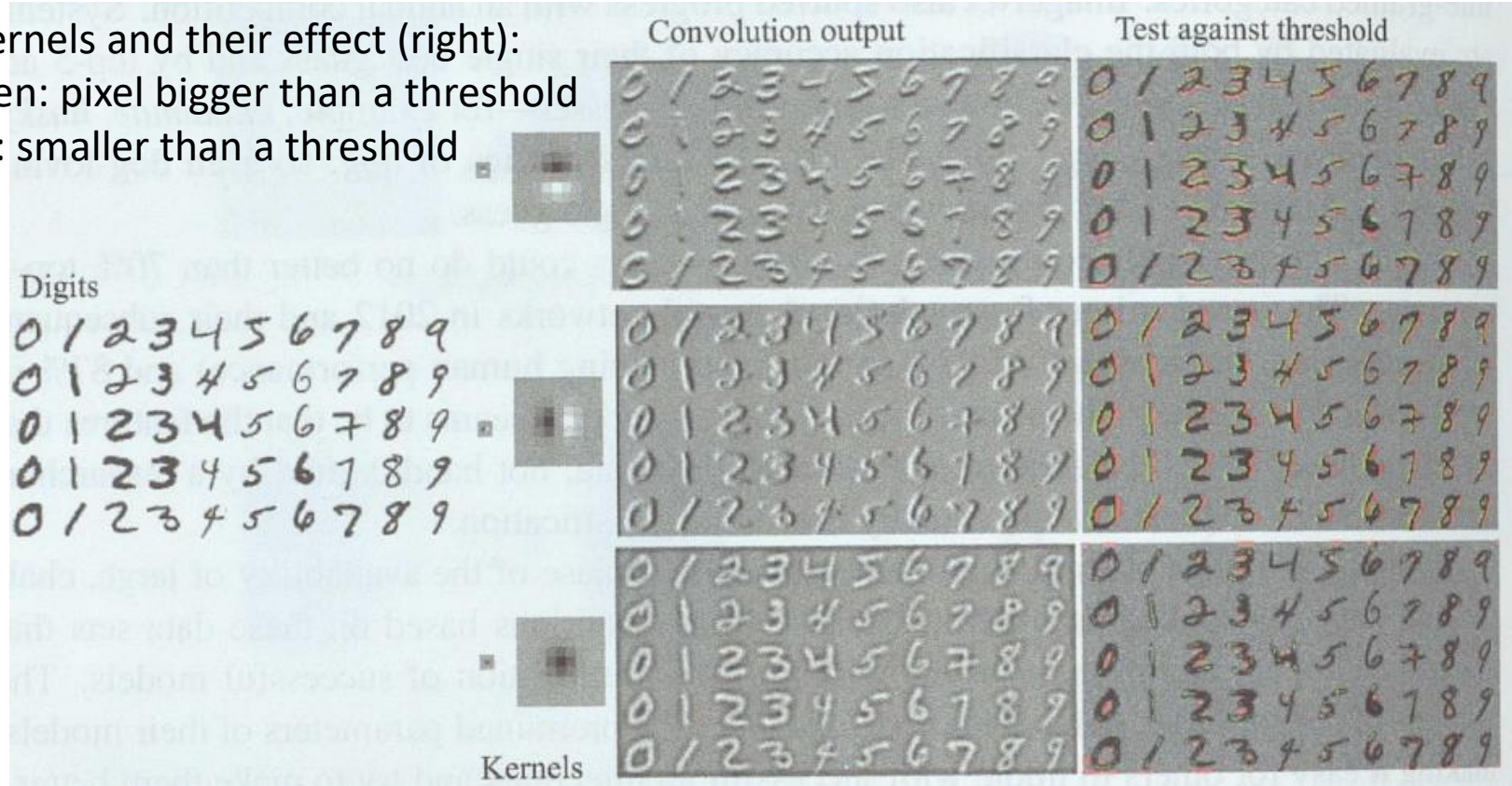
- Purely data-driven models usually outperform conventional systems based on grammars, parsing and semantic interpretation
  - Without need of hand-crafted features
- Data driven system may learn a latent representation of grammar and semantics
- Future??
  - Hybrid systems?
  - Still much room for improvement!

25. Computer Vision

- Deep computer vision applications are even more spectaluar than deep NLP applications
- For traditional computer vision there exist impressive tool boxes (like OpenCV), which are also extended for deep components
- But for many applications, it is sufficient to train neural nets with lots of labeled images and they learn both low-level and high-level features for different tasks:
  - Image classification
  - Object detection
  - Segmentation
  - Video analysis
  - 3D world
- Key technique (s. right):

Convolutional Neural Nets (CNN)

3 kernels and their effect (right):
green: pixel bigger than a threshold
red: smaller than a threshold



Frank Puppe

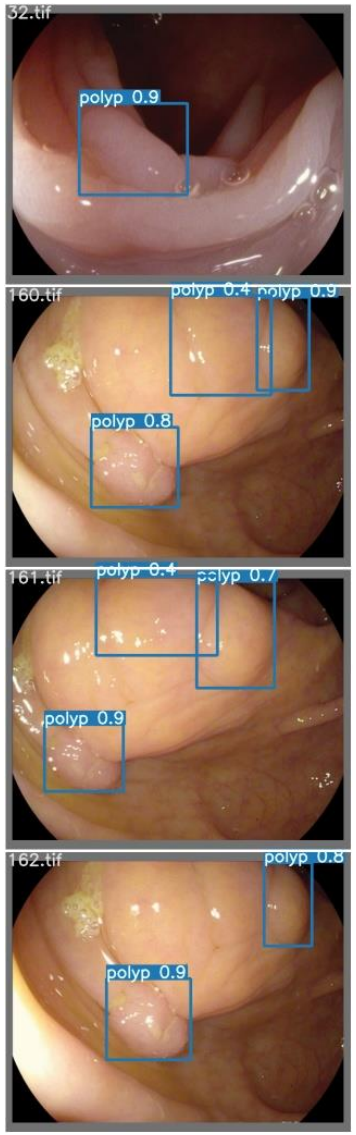
Image Classification

- Classify a whole image with a label
- Public data source: ImageNet and many others, e.g. from special competitions
- Extremely successful with CNNs
- Helpful techniques:
  - **Pretraining,** e.g. with ImageNet
  - **Data set augmentation** (add images with small modifications, e.g. shift, rotate, stretch an image by a small amount)
  - **Context** may help or hurt (e.g. wolves are often dogs with snow)

Object Detection

- Object detection: Find multiple objects in an image with a bounding box around each
- Consists of several tasks:
  - Find bounding boxes
  - Classify object within bounding box
  - If bounding boxes overlap, choose best bounding box for each object (with highest object recognition score)
- Find bounding box:
  - **Anchor-based** with predefined shapes: Needs expectations about the shape of the objects (e.g. circles, triangles and other shapes for traffic signs)
  - **Anchor-free** without predefined shapes (larger search space)
- **Two-stage approaches:** Find bounding box candidate and afterwards classify objects
- **One-stage approaches:** Use part of the neural net for both purposes (faster)

Examples from Coloscopy (Own Experiments)

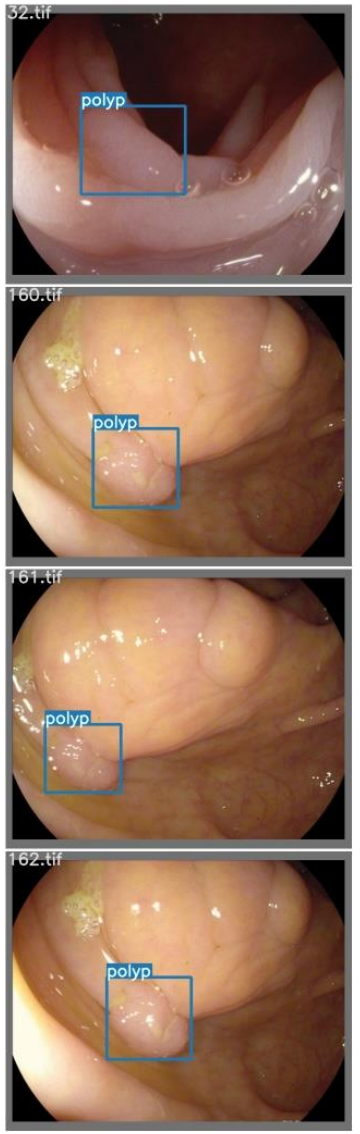


Prototyp 1 Polyp Detection:

- **96.8% F1-Score** with CVC-Clinic data
- **81.1% F1-Score** with EITS-Larib data
- Duration: 800 Milliseconds per image[1]
- Net architecture: Anchor-based two stage: Faster R-CNN

Prototyp 1 Polyp Detection:

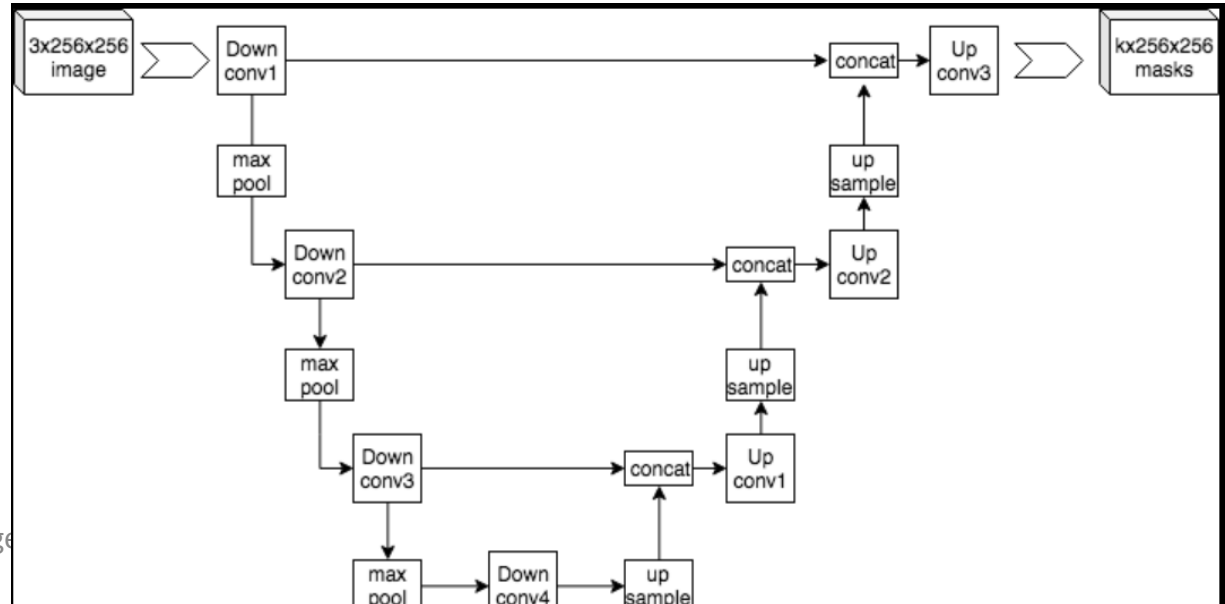
- **90.2% F1-Score** with CVC-Clinic data
- **80.0% F1-Score** with EITS-Larib data
- Duration: 20 Milliseconds per image[1] (**real time for videos**)
- Net architecture : Anchor-based one stage: YOLOv5



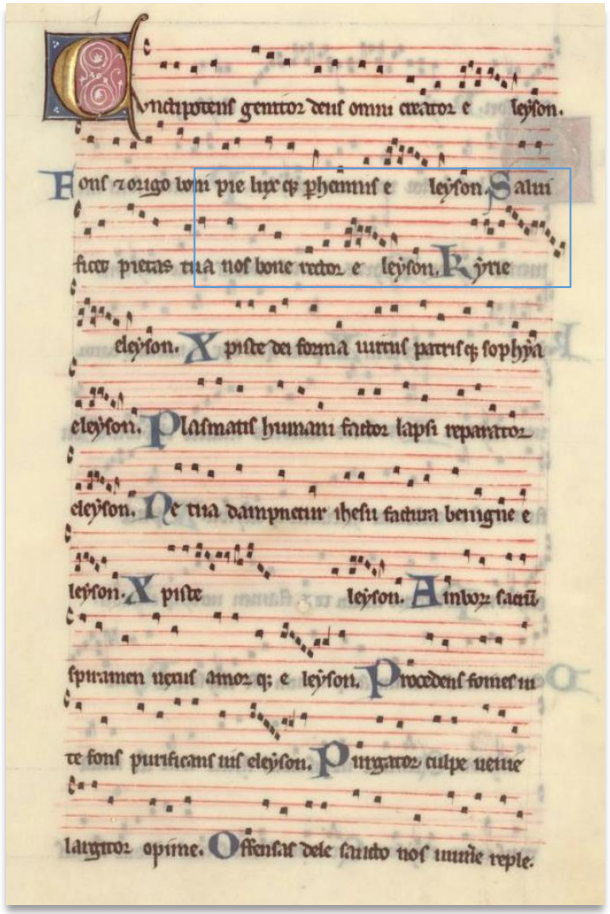
[1] Tesla P100 grafic card

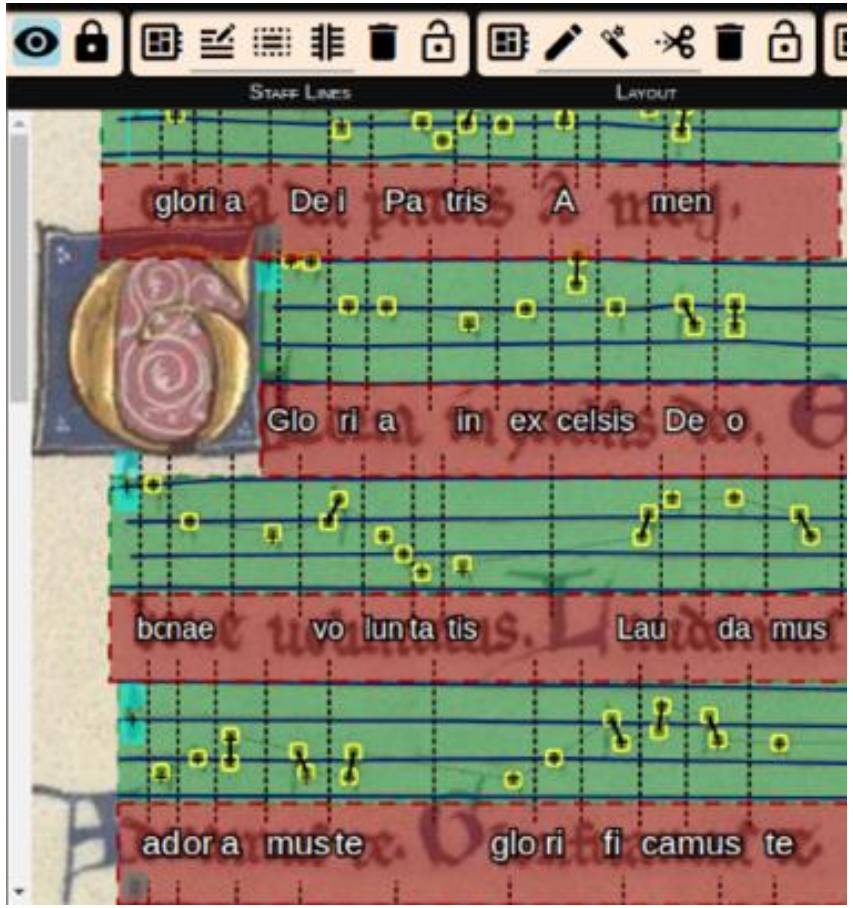
Pixel-based Segmentation

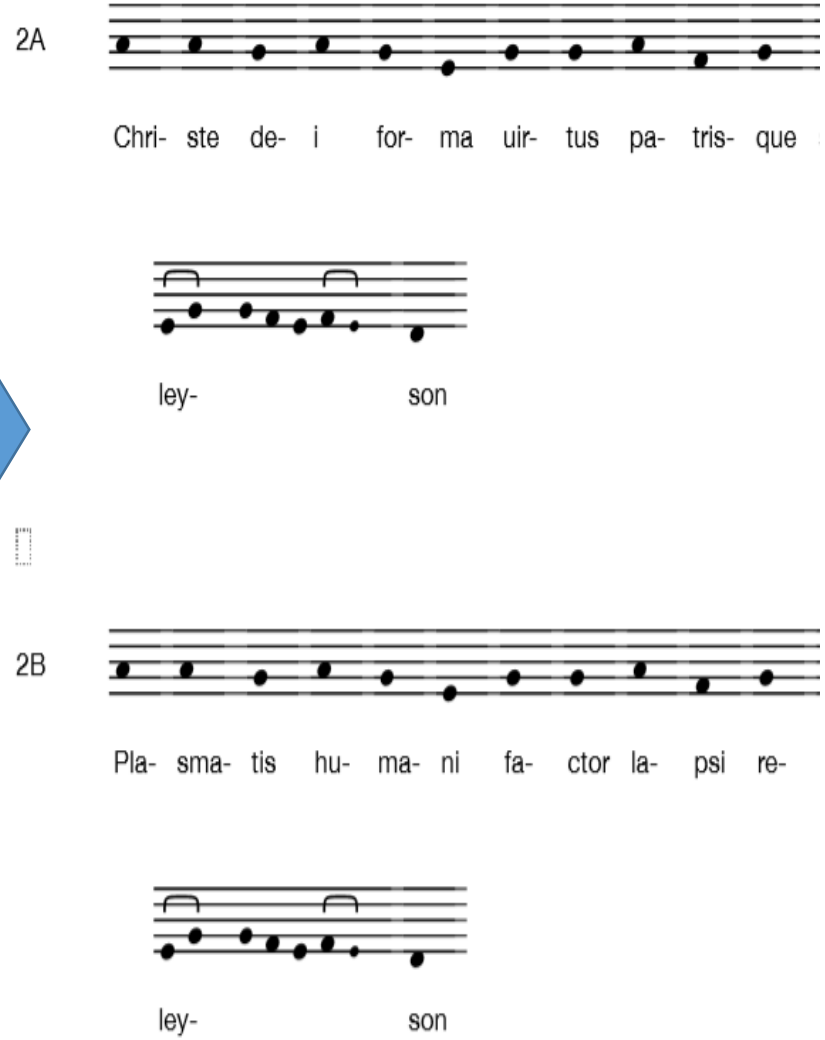
- Goal: Assign to each pixel an image a class
- Method: Train neural network, e.g. with encoder-decoder structure (similar to autoencoder)
  - Encoder consists of convolutional and max pooling layers for abstraction
  - Decoder uses upsampling operations from the abstractions and skip connection to label the pixels of the image
  - Example: U-Net architecture (right below)
  - Applications:
    - Biomedical images (e.g. brain etc.)
    - Layout recognition for OCR (right above) (Optical Character Recognition)



Pipeline for Optical Music Recognition (OMR)



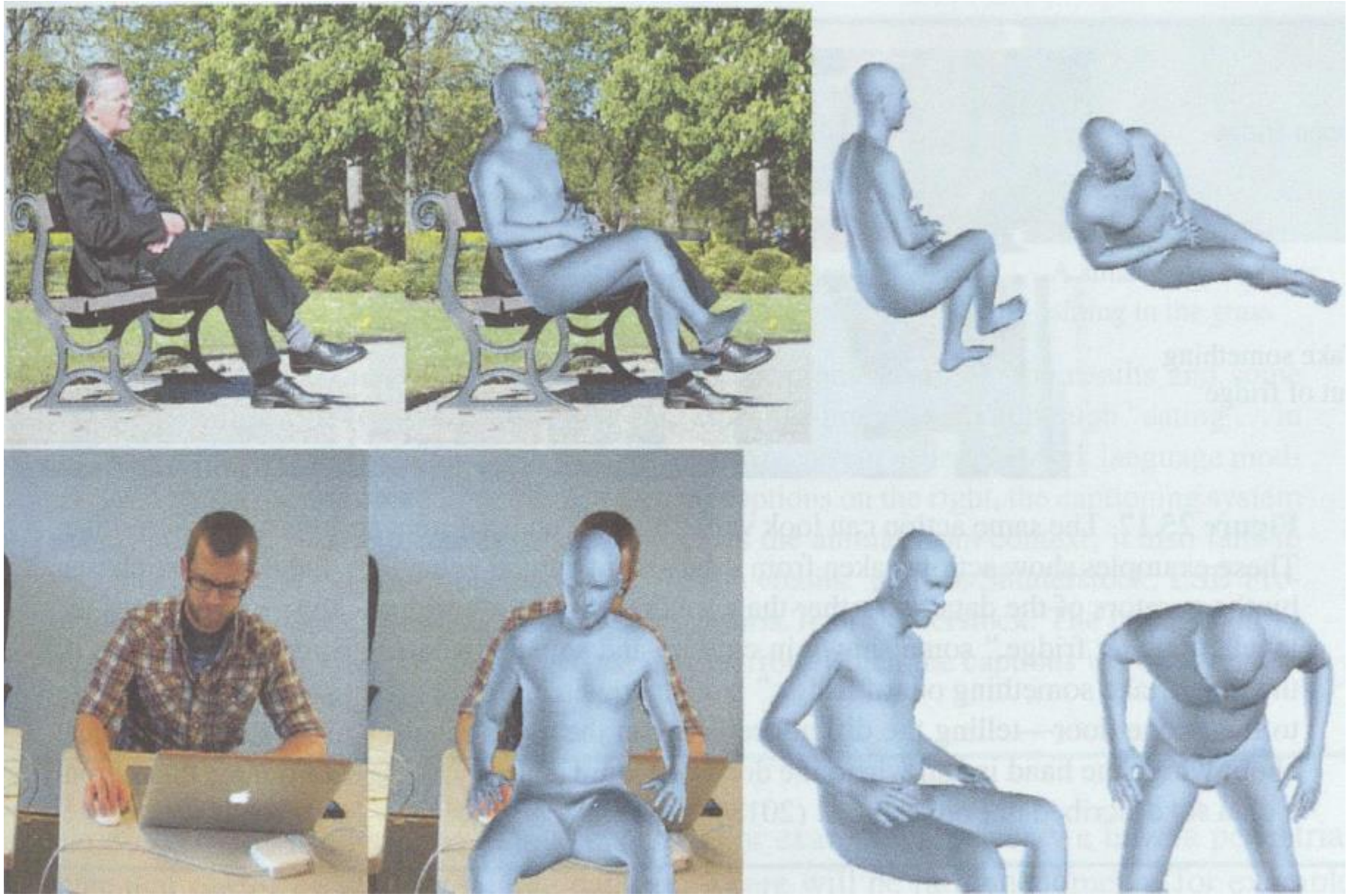




(1) Segmentation, (2) note and (3) text recognition, (4) alignment

Understanding what People are Doing

- Reconstructing humans form a single image is now practical
- Classifying what people are doing is harder
  - Possible from videos with rather structured behaviour



Many more Applications

- Linking pictures and words
- Reconstruction from many views
- Geometry from a single view (e.g. by predicting a depth map)
- Making pictures:
  - Placing objects in a scene
  - Changing pictures (e.g. a horse in a zebra)
  - Deep fakes (images or video that looks like a particular person but generated from model)
  - Style transfer, etc.
- Controlling movement with vision (e.g. autonomous cars)
- …