Statistical Natural Language Processing course 2021 conclusion

Mikko Kurimo

Learning goals

- To learn how statistical and adaptive methods are used in information retrieval, machine translation, text mining, speech processing and related areas **to process natural language data**
- To learn how to apply the basic methods and techniques for clustering, classification, generation and recognition **by natural language modeling**

How to achieve the learning goals (and pass the course)?

- *Participate actively in each lecture, read the corresponding material and ask questions* to learn the basics, take part in discussions, complete the lecture exercises DONE
- *Participate actively in each exercise session after each lecture* to learn how to solve the problems, in practice DONE
- Complete the home exercises in time Almost DONE
- *Participate actively in project work* to learn to apply your knowledge

  Final report DL April 29
- *Prepare well for the examination*

  Course exam April 14
  (next exam September)

Course grading

- 20% of the grade comes from the **exam.** The exam will be organized at the end of the course in April. For those who can not participate in it, there will be a second exam in Autumn. Exams passed in previous years are still valid for completing the course.
- 40% of the grade is from the weekly **home exercises and lecture activities.** The lecture activities may include pen&paper tasks, quizzes, discussions. To get the points return your solutions during the lecture or on the day after, at the latest.
- 40% of the grade is from the **project work**. It depends on experiments, literature study, short (video) presentation and final report. Course projects accepted in previous years are still a valid for completing the course.

Course project grading

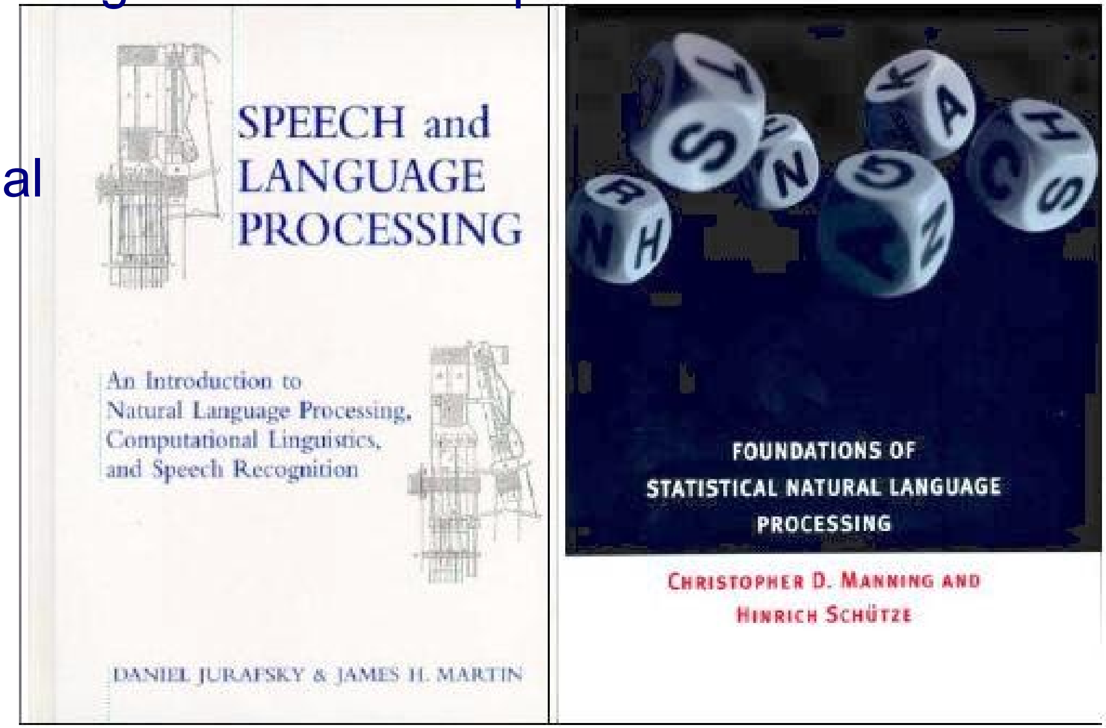
- See Mycourses for the requirements of an acceptable project report
- Peer grading will be performed for some parts to get more feedback, but that is separate from the final project grade
- Excellent projects typically include additional work such as
  - Exceptional analysis of the data
  - Application of the method to a task or several
  - Algorithm development
  - Own data set(s) (with preprocessing etc to make them usable)

Course exam grading

- The exam will be a remote digital exam
- There are 5 questions in MyCourses open in Wed 14.4. 12:00 – 15:00
- Grading 0 – 5, max 6 points per question makes total 30p. The grade limits will depend on the exam’s difficulty. Last year 11p gave the (exam) grade 1.
- You can use any books, course material, internet, calculators and toolkits
- **You must write in your own words**. If any copy-pasted texts are found, there will be an automatic reject and report to Aalto University
- **You are not allowed to communicate or collaborate** with other people during the exam. If any co-operation is found, there will be an automatic reject and report to Aalto University
- All exam questions and course exercise materials are copyrighted. You are not allowed to distribute them in any way.

Reading material

- Manning & Schütze, Foundations of Statistical Natural language processing (1999) *http://nlp.stanford.edu/fsnlp/*
- Jurafsky & Martin, Speech and Language Processing (3rd ed. Draft, 2020) *http://web.stanford.edu/~jurafsky/slp3/*
- Read the chapters corresponding to the lecture topics
- Do not forget to study the topic-specific reading material (mentioned in slides)



Lecture topics

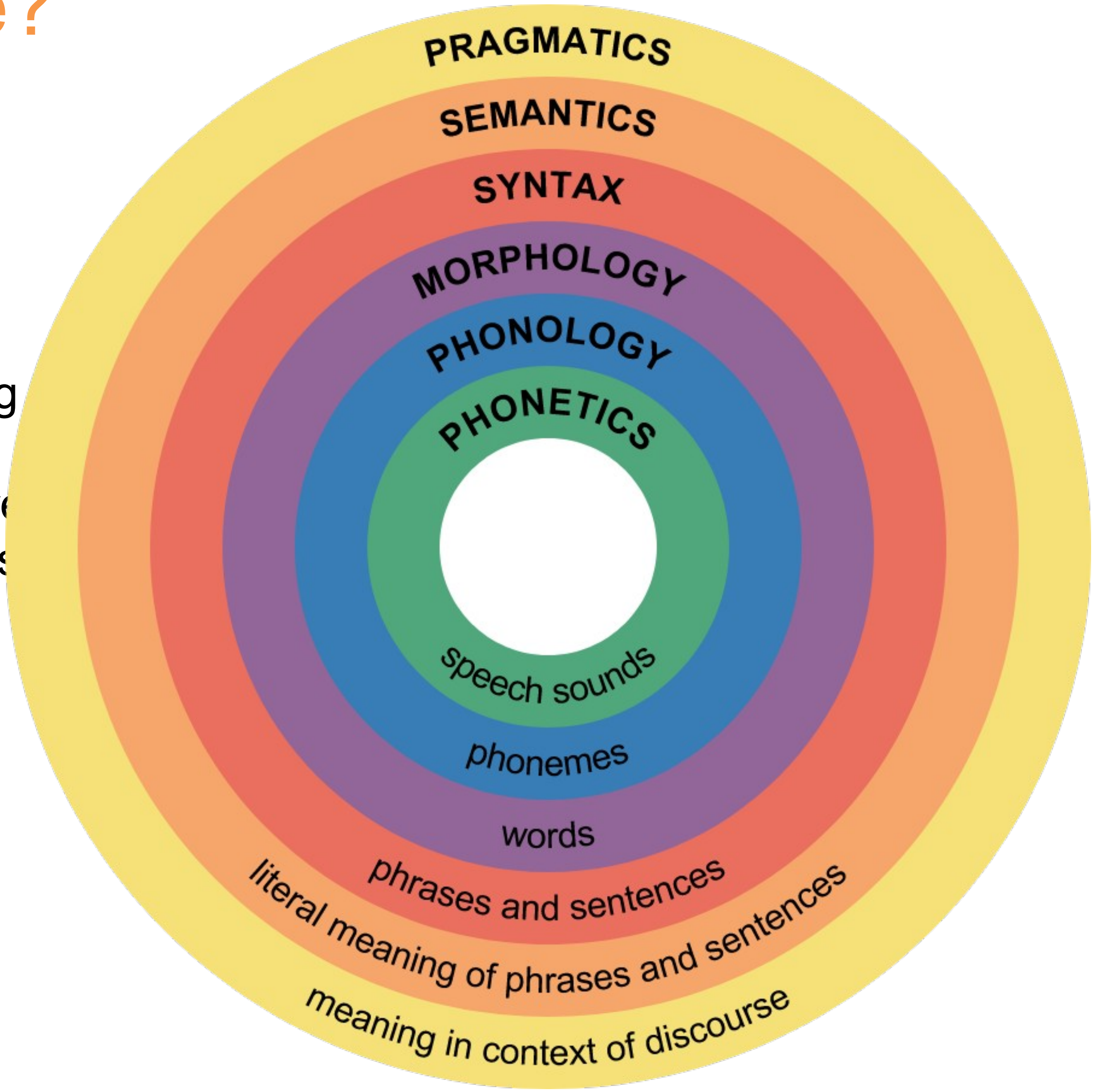
1. 12 Jan Introduction & Project groups / Mikko Kurimo
2. 19 jan Statistical language models / Mikko Kurimo
3. 26 jan Word2vec / Tiina Lindh-Knuutila
4. 02 feb Sentence level processing / Mikko Kurimo
5. 09 feb Speech recognition / Janne Pylkkönen
6. 16 feb Chatbots and dialogue agents / Mikko Kurimo
7. 02 mar Statistical machine translation / Jaakko Väyrynen
8. 09 mar Morpheme-level processing / Mathias Creutz
9. 16 mar Neural language modeling and BERT / Mittul Singh
10. 23 mar Neural machine translation / Stig-Arne Grönroos
11. 30 mar Societal impacts and course conclusion / Krista Lagus, Mikko

1. Introduction

- What does language include?
- What makes languages so complex?
- What are the applications of statistical language modeling?

What is in a language?

- Phonetics and phonology:
  - the physical sounds
  - the patterns of sounds
- Morphology: The different building blocks of words
- Syntax: The grammatical structure
- Semantics: The meaning of words
- Pragmatics, discourse, spoken interaction...



Complexity of languages

- A large proportion of modern human activity in its different forms is based on the use of language
- Large variation:
  - morphology and
  - syntactic structures
- Complexity of natural language(s)
  - More than 6000 languages, many more dialects
  - Each language a large number of different word forms
  - Each word is understood differently by each speaker of a language at least to some degree

Application areas

- Information retrieval
- Text clustering and classification
- Automatic speech recognition
- Natural language interfaces
- Statistical machine translation
- Topic detection
- Sentiment analysis
- Word sense disambiguation
- Syntactic parsing
- Text generation
- Image, audio and video description
- Text-to-speech synthesis
- ...

2. Statistical language modeling

- N-gram LMs
  - Data sparsity problem
  - Equivalence classes
  - Back-off and interpolation
  - Smoothing methods, add-one, Good Turing, Kneser-Ney
- Maximum entropy LMs
- Continuous space LMs
- Neural network LMs

Statistical Language Model

- Model of a *natural language* that predicts the *probability distribution* of words and sentences in a text
- Often used to determine which is the most probable word or sentence in given conditions or context
- Estimated by counting word frequencies and dependencies in large text corpora
- Has to deal with: *big data, noisy data, sparse data, computational efficiency*

Estimation of N-gram model

$$P(w_i \mid w_j) = \frac{c(w_j, w_i)}{c(w_j)} \qquad \frac{c(\text{“eggplant stew”})}{c(\text{“eggplant”})}$$

$$= P(w_i) b_{w_j} \qquad otherwise$$

- Bigram example:

- Start from a **maximum likelihood estimate**
- probability of *P(“stew” | “eggplant”)* is computed from **counts** of *“eggplant stew”* and *“eggplant”*
- works well for frequent bigrams
  - Why not for good rare bigrams?

Picture by B.Pellom

Zero probability problem

- If an N-gram is not seen in the corpus, it will get probability = 0
- The higher N, the sparser data, and the more zero counts there will be
- 20K words => 400M 2-grams => 8000G 3-grams, so even a gigaword corpus has MANY zero counts!
- **Equivalence classes**: Cluster several similar n-grams together to reach higher counts
- **Smoothing**: Redistribute some probability mass from seen N-grams to unseen ones

Smoothing methods

1. **Add-one**: Add 1 to each count and normalize => gives too much probability to unseen N-grams
2. **(Absolute) discounting**: Subtract a constant from all counts and redistribute this to unseen ones using N-1 gram probs and back-off (normalization) weights
3. **Witten-Bell smoothing**: Use the count of things seen once to help to estimate the count of unseen things
4. **Good Turing smoothing**: Estimate the rare n-grams based on counts of more frequent counts
5. Best: **Kneser-Ney smoothing**: Instead of the number of occurrences, weigh the back-offs by the **number of contexts** the word appears in
6. Instead of only back-off cases, **interpolate** all N-gram counts with N-1 counts

Weaknesses of N-grams

- Skips long-span dependencies:
  - “The **girl** that I met in the train **was** ...”
- Too dependent on word order:
  - “dog chased **cat**”: “koira jahtasi **kissaa**” ~ “**kissaa** koira jahtasi”
- Dependencies directly between words, instead of latent variables, e.g. word categories

Maximum entropy LMs

- Represents dependency information
- by a weighted sum of features f(x,h)
- Features can be e.g. n-gram counts

$$P(x|h) = \frac{e^{\sum_i \lambda_i f_i(x,h)}}{\sum_{x'} e^{\sum_j \lambda_j f_j(x',h)}}$$

- Alleviates the data sparsity problem by smoothing the feature weights (lambda) towards zero
- The weights can be adapted in more flexible ways than n-grams
  - Adapting only those weights that significantly differ from a large background model (1)
- Normalization is computationally hard, but can be approximated effectively

Continuous space LMs

- Alleviates the data sparsity problem by representing words in a distributed way
- Various algorithms can be used to learn the most efficient and discriminative representations and classifiers
- The most popular family of algorithm is called (Artificial) **Neural Networks (NN)**
  - can learn very complex functions by combining simple computation units in a hierarchy of non-linear layers
  - Fast in action, but training takes a lot of time and labeled training data
- Can be seen as a non-linear multilayer generalization of the maximum entropy model

3. Vector space models for words and documents

- Vector space models, distributional semantics
  - word-document and word-word matrices
  - Constructing word vectors
  - stemming, weighting, dimensionality reduction
  - similarity measures
  - Count models vs. predictive models
  - Word2vec
- Information retrieval

Vector space models

Use a high-dimensional space for documents and words

- Closeness in the vector space resembles closeness in the semantics or structure of the documents (depending on the features extracted).
- Makes the use of data mining possible

Applications:

– Document clustering and classification
  - Finding similar documents
  - Finding similar words

– Word disambiguation

– Information retrieval
  - Term discrimination: ranking keywords by their usefulness

How to build a vector space model?

1. Preprocessing
2. Defining word-document or word-word matrix
   - choosing features
3. Dimensionality reduction
   - choosing features
   - removing noise
   - easing computation
4. Weighting and normalization
   - emphasizing the features
5. Similarity / distance measures
   - comparing the vectors

To count or predict?

Count-based methods

- compute the word co-occurrence statistics with its neighbor words in a large text corpus
- followed by a mapping (through weighting and dimensionality reduction) to dense vectors

Predictive models

- try to predict a word from its neighbors by directly learning a dense representation

Statistical semantics

Statistical semantics hypothesis: Statistical patterns of human word usage can be used to figure out what people mean (Weaver, 1955; Furnas et al., 1983).

Bag of words hypothesis: The frequencies of words in a document tend to indicate the relevance of the document to a query (Salton et al., 1975).

Distributional hypothesis: Words that occur in similar contexts tend to have similar meanings (Harris, 1954; Firth, 1957; Deerwester et al., 1990).

Latent relation hypothesis: Pairs of words that co-occur in similar patterns tend to have similar semantic relations (Turney et al., 2003).

Modifying the vector spaces

The basic matrix formulation offers lots of variations:

–window sizes

–word weighting, normalization, thresholding, removing stopwords

–stemming, lemmatizing, clustering, classification, sampling

–distance measures

–dimensionality reduction methods

–neural networks

Information retrieval: **The query** is compared to **the index** and the best matching results are returned

Ranking the results

Compute a numeric score on how well each object in the database matches the query

– Distance in the vector space
– Content and structure of the document collection
– Number of hits in a document
– Number of hits in title, first paragraph, elsewhere
– Other meta information in the documents or external knowledge

The retrieved objects are ranked according to the score and only the top ranking objects shown to the user.

4. Sentence-level processing

- Tagging words in a sentence
  - Part-of-speech tagging
  - Named entity recognition
  - Solving ambiguities
  - Hidden Markov model, Viterbi, Baum-Welch, Forward-Backward
  - Recurrent neural networks
- Sentence parsing
  - Grammars, trees
  - Probabilistic context free grammar

Part of Speech (POS) tagging

Task: Assign tags for each word in a sentence

Applications: Tool for parsing the sentence

The reaction in the newsroom was emotional.

=> *DT NN IN DT NN VBD JJ*

(determiner)

(noun) (preposition)

(determiner) (noun)

(verb past tense) (adjective)

Named entity recognition

- Detect names of persons, organizations, locations
- Detect dates, addresses, phone numbers, etc
- Applications: Information retrieval, ontologies
- UN official Ekeus heads for Baghdad.

=> ORG - PER - - LOC

(organization) (person) (location)

A general approach

1. Generate candidates
2. Score the candidates
3. Select the highest scoring ones

A simple scoring method

- Count the frequency of each tagging by listing all appearances of the word in an annotated corpus
- Select the most common tag for each word
- How well would this method work?

Count transitions

- Use the Penn Treebank corpus and count how often each tag pair appears
- Prepare a tag transition matrix
- Compute transition probabilities from the counts
  - Just like bigrams for words, but now for tags
  - P(y1), P(y2|y1), P(y3|y2), P(y4|y3)

Score the tags for the sentence

- Combine the transition probabilities:

  $P(y1)\ P(y2|y1)\ P(y3|y2)\ \ldots$

- with the tag-word pair observation probabilites:

  $P(x1|y1)\ P(x2|y2)\ P(x3|y3)$

- to get the total tagging score:
- $P(y1)P(x1|y1)\ P(y2|y1)P(x2|y2)\ P(y3|y2)P(x3|y3)$
- Known as Hidden Markov Model (HMM) tagger
- Achieves about 96% accuracy

Hidden Markov model (HMM)

- Markov chain assumes that the next state (tag) depends only on the previous state (tag)
- The states (tags) are hidden, we only see the words
- The algorithm can compute the most likely state sequence given the seen words

Estimation of HMM parameters

- For corpora annotated with POS tags
  - Just count each tag observations P(y(t)|x(t)
  - And tag transitions P(y(t)|y(t-1))
- For unknown data use e.g. Viterbi to first estimate labels and then re-estimate parameters and iterate

Even better POS tags? Discriminative models

- Use previous words and tags as features
- The context is computed from a sliding window
- Train a classifier to predict the next tag
  - Jurafsky: Maximum entropy Markov model (MEMM)
  - Support vector machine (SVM)
  - Deep (feed-forward) neural network (DNN)
  - Conditional random field (CRF) is a bidirectional extension of MEMM that uses also tags on right

Recurrent neural network tagger

- No fixed-length context window
- Loop in the hidden layer adds an infinite memory
- Can provide word-level tags:
  - POS or named entity
- Or sentence-level tags:
  - Sentiment analysis
  - Topic or spam detection

5. Speech recognition

- Acoustic features
- Gaussian mixture models
- Hidden Markov models
- Deep neural networks for acoustic modeling
- Phonemes, pronunciation of words
- Decoding with language models
- End-to-end neural networks
- Encoder, Attention, Decoder

Speech Recognition Tasks

Speechly

- **Popular automatic speech recognition (ASR) application:**
  - Keyword detection
  - Command-and-control
  - Search by speech
  - Dictation
  - Conversational interaction

Easier

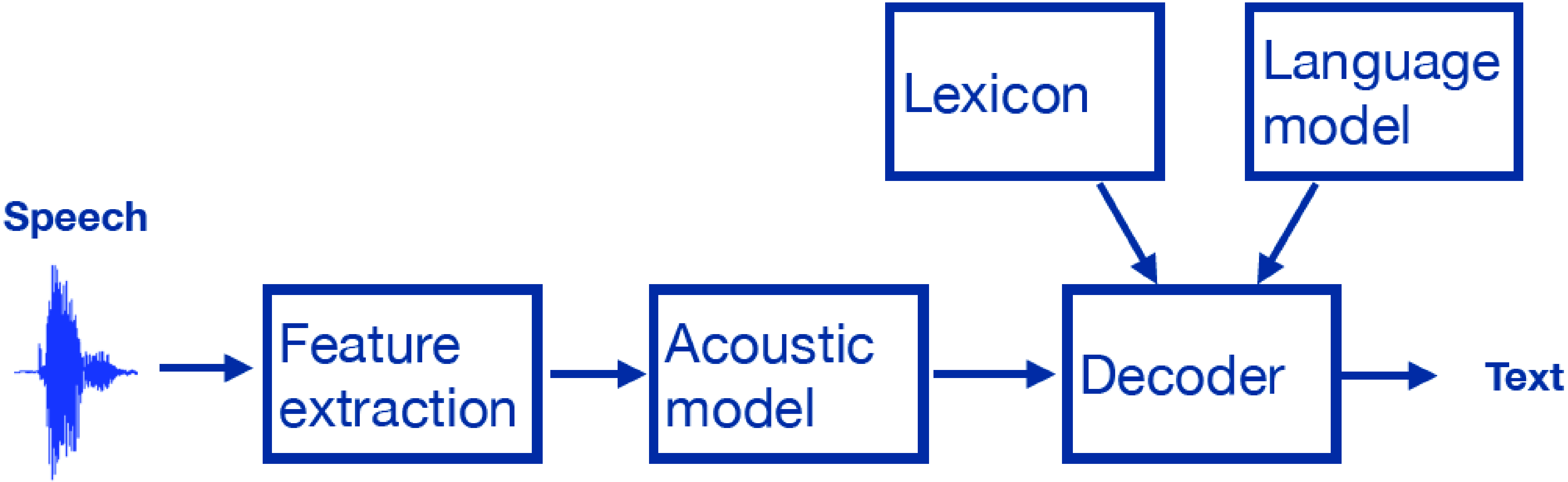
↓

Harder

- **Speech characteristics relating to the recognition task:**
  - Isolated words **vs.** continuous speech
  - Speaker dependent **vs.** independent
  - Vocabulary size
  - Read speech, planned speech, conversational speech
  - Environmental noise
  - **Space and distance to the microphone:** close-talk, near-field, far-field

- **Recognising everyday speech around us is challenging because it is speaker independent, conversational, large vocabulary, continuous speech, mixed in various environmental noises!**

Traditional Components of an ASR system



- **Task of an automatic speech recognition: Find the most likely word sequence given the observations (speech) and the models for acoustics and language**
- **Speech acoustics are matched with a statistical model**
- **Language model is also typically a statistical model (n-gram, RNN), but in simple tasks it can be a fixed grammar or just a vocabulary**

Speech recognition: large probabilistic models

Output: word sequence

Input: sound observations

$$\hat{W} = \arg\max_{W} P(W \mid O)$$

$$= \underbrace{\arg\max_{W}}_{\text{Decoding algorithm}} \underbrace{P(O \mid W)}_{\text{Acoustic model}} \underbrace{P(W)}_{\text{Language model}}$$

6. Chatbots and dialogue agents

- True chatbots vs task-oriented dialogue agents
- Text processing steps in chatbots
- Chatbot architectures
- Evaluation of chatbots

Definitions

Chatbot:
- A system that **you can chat** with
- Discussion topics can be fixed, but there is **no specific goal** except for fun and keeping company

Dialogue agent:
- A system that helps you to **reach a specific goal** by giving and collecting information by **answering and asking questions**

In popular media both are often called chatbots, but here only the first one.

Comparison of chatbots and dialogue agents: required operations

Chatbot

- Detect the discussion topic
- Ask typical questions
- React to human input, be coherent with previous turns
- World knowledge, persona

Dialogue agent

- Detect the user's intent
- Ask the required questions
- Parse and use human input

Chatbot architectures

Rule-based
- Pattern-action rules: Eliza (1966)
- Mental model: Parry (1971)

Corpus-based
- IR: Cleverbot
- DNN encoder-decoders etc

Turing's test (1950) for machine intelligence: *Can you judge between a real human and a chatbot?*

Evaluation of chatbots

Automatic evaluation

- Lack of proper evaluation data and metrics
- N-gram matching evaluations such as BLEU correlate poorly with human evaluation
  - Too many correct answers
  - Common words give a good score
- Perplexity measures predictability using a language model
  - Favours short, boring and repetitive answers
- ADEM classifier trained by human judgements
- Adversarial evaluation trained to distinguish human and machine responses

Human evaluation

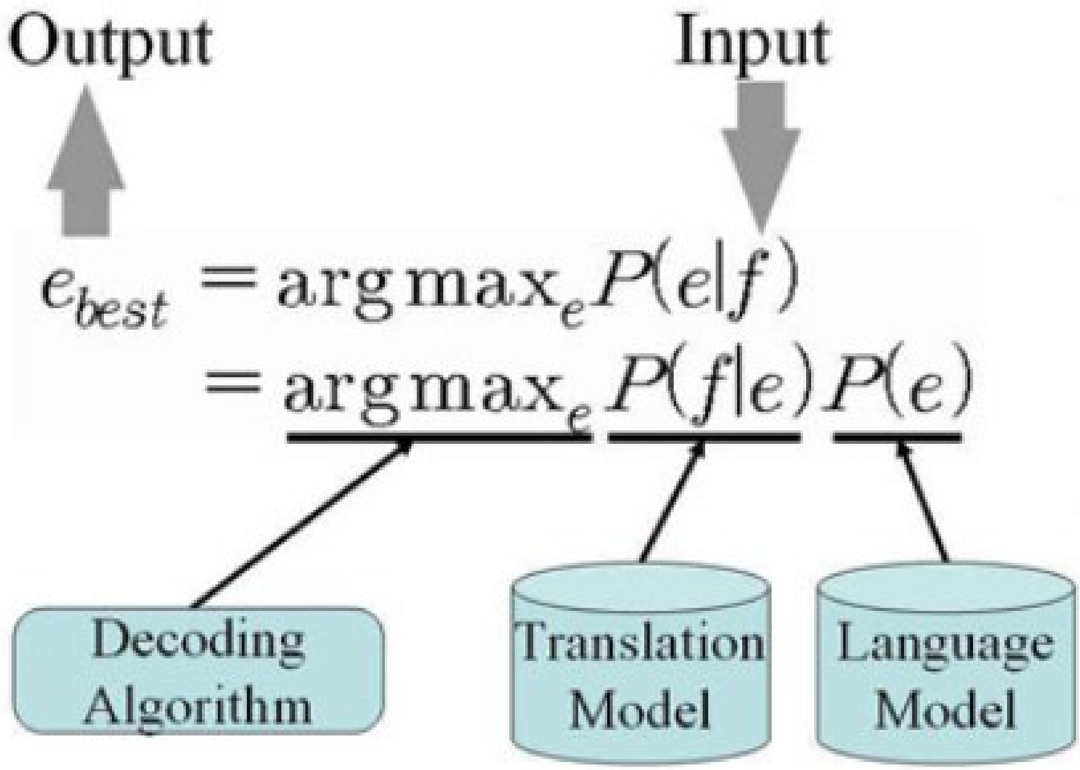
e.g. research challenges (competitions):

- ConvAI (NeurIPS)
- Dialog Systems Technology Challenge (DSTC7)
- Amazon Alexa prize
- Loebner Prize

7. Statistical machine translation

- Sentence, word and phrase alignment methods
- Re-ordering models
- Translation methods
- Full SMT systems

Machine translation: large probabilistic models



isoft.postech.ac.kr

Phrase-based SMT system

- Training data and data preprocessing
- Word aligment, phrase aligment
- Estimation of translation model scores
- Estimation of reordering model scores
- Estimation of language model scores
- Decoding algorithm and optimization of the model weights
- Translation, recasing, detokenization
- Evaluation, quality estimation
- Operational management

Sentence Alignment

- Simplifying assumptions: monotonic, break on paragraphs
- Gale&Church algorithm: model sentence lengths
- Other features: (automatic) dictionaries, cognates
- Dynamic programming (Similar to Viterbi)

Word Alignment

- The sentence alignment was the first step
- The word alignment takes into account
  - reordering (distortion)
  - fertility of the words
- Iterative Expectation-Maximization algorithm:

  1. Generate a word level alignment using estimated translation probabilities
  2. Estimate translation probabilities for word pairs from the alignment

Phrase alignment

- “Cut-and-paste" translation
- The distortion (reordering) probability typically penalizes more, if several words have to be reordered. However, usually larger multi-word chunks (subphrases) need to be moved
- Algorithms to learn a phrase translation table
- Re-ordering models

Translation methods

- Phrase-based beam-search decoder (e.g. Moses)
- Weighted finite state transducer (WFST) based translation models
- Extended word-level representations, e.g., hierarchical phrase-based models and factored translation models with words augmented with POS tags, lemmas, etc.
- Syntax-based translation models, which take syntax parse trees as input.
- Feature-based models, where translation is performed between features. E.g., discriminative training or exponential models over feature vectors

8. Morpheme-level processing

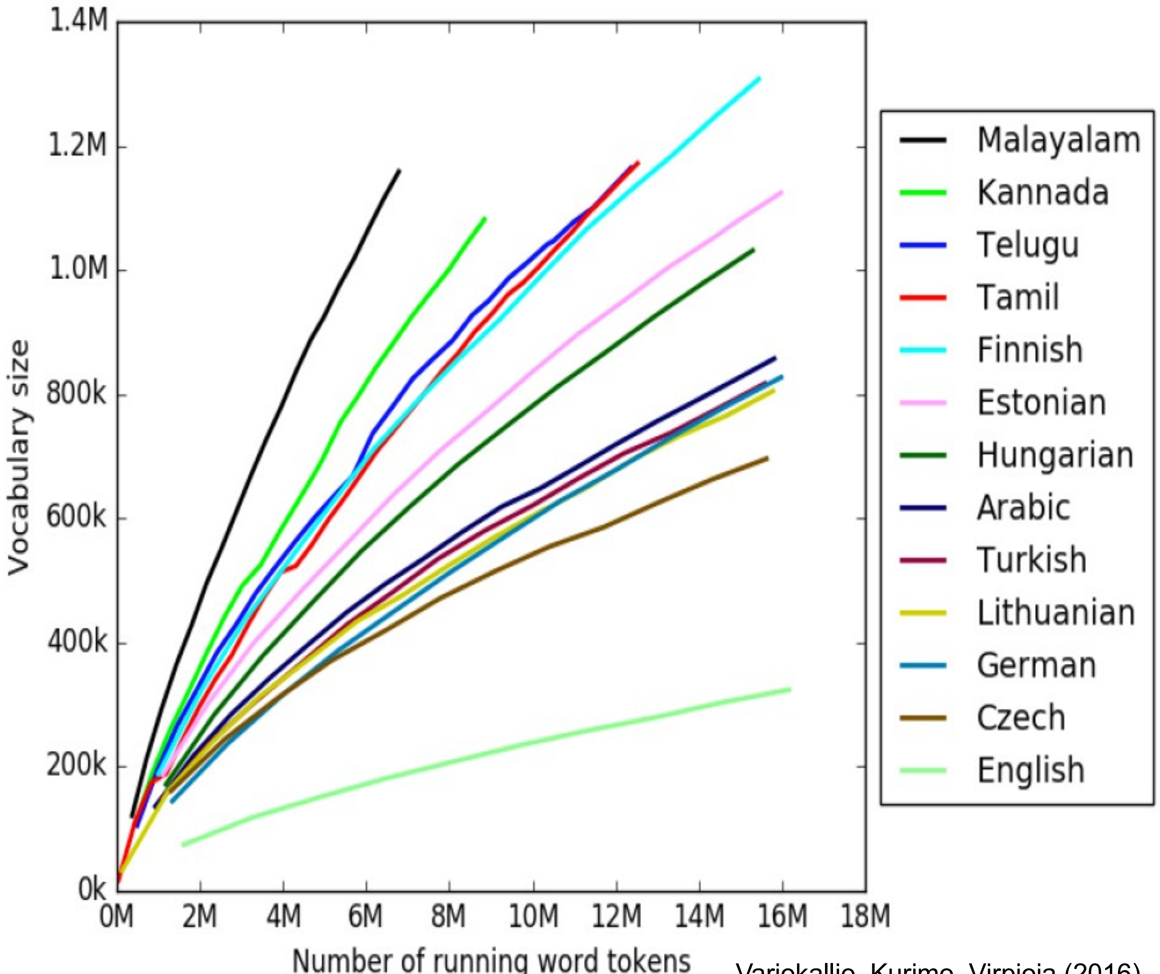
- Morphemes
  - morphological complexity, processes, models
- Morphological clustering
  - stemming, lemmatization
- Morphological analysis and generation
  - Finite-state methods, transducers
- Morphological segmentation
  - Zellig Harris's method
  - Morfessor

Types of morphemes

- **Root:** a portion of word without any affixes; carries the principle portion of meaning (buildings ℗ build)
- **Stem:** a root, or compound of roots together with derivational affixes (buildings ℗ building)
- **Affix:** a bound morpheme (does not occur by itself) that is attached before, after, or inside a root or stem
  - **Prefix** (**un**-happy)
  - **Suffix** (build-**ing**, happi-**er**)
  - **Infix** (abso-**bloody**-lutely)
  - …

Morphology affects the vocabulary size

Vocabulary size as a function of corpus size



Varjokallio, Kurimo, Virpioja (2016)

Morphological processes

Inflection:
- cat – cats
- slow – slower
- find – found

Derivation:
- build (V) – building (N)
- do (V) – doable (ADJ)
- short (ADJ) – shorten (V)
- write – rewrite
- do – undo

Compounding:
- fireman (fire + man)
- hardware (hard + ware)

3 main approaches to deal with rich morphology

1. **"Canonical" forms of a word**
   - Stemming is relatively simple and implementations are available, for English: e.g., Porter (1980), Snowball: http://snowball.tartarus.org
   - Lemmatization is more complex and needs morphological analysis
   - Applications: Information retrieval etc.
2. **Morphological analysis**
   - Hand-crafted morphological analyzers/generators exist for many languages, e.g. Tähtien => tähti N Gen Pl
   - Applications: Spell checking, syntactic parsers, machine translation, etc.
3. **Segmentation into morphs**
   - Pragmatic approaches that work well enough in practice.
   - Applications: Speech recognition, language modeling etc.

Limitations of morphological analysis

Out-of-vocabulary words

- epäjärjestelmällistyttämättömyydelläänsäkäänköhän Ⓟ epäjärjestelmällistyttämättömyydelläänsäkäänköhän+?

Ambiguous forms

- sawsee+V+PAST *or* saw+N *or* saw+V+INF ?

  "I **saw** her yesterday." Ⓟ SEE (verb)

  "The **saw** was blunt." Ⓟ SAW (noun)

  "Don't **saw** off the branch you are sitting on." Ⓟ SAW (verb)

- meetingmeet+V+PROG *or* meeting+N ?

  "We are **meeting** tomorrow." Ⓟ MEET (verb)

  "In our **meeting**, we decided not to meet again." Ⓟ MEETING (noun)

- Solutions?

Unupervised morphological segmentation

- Morphological segmentation
- Send a vocabulary of the language over a channel with limited band-width.
- compress the vocabulary.
- What regularities can we exploit?
- Use morphemes, the smallest meaning-bearing units of language?
- Morfessor method (Creutz & Lagus, 2002)

```
…
aamu
aamua
aamuaurinko
aamukahvi
aamuksi
aamulehti
aamulla
aamun
aamunaamasi
aamupalalla
aamupalan
aamupostia
aamupäivä
aamupäivällä
aamuyö
aamuyöllä
aamuyöstä
…
```

?

```
…
aamu
aamu a
aamu aurinko
aamu kahvi
aamu ksi
aamu lehti
aamu lla
aamu n
aamu naama si
aamu pala lla
aamu pala n
aamu posti a
aamu päivä
aamu päivä llä
aamu yö
aamu yö llä
aamu yö stä
…
```

Morfessor

- Instead of sending over the vocabulary as it is, we split it into two parts:
  1. a fairly compact lexicon of morphs: “aamu”, “aurinko”, “ksi”, “lla”, …
  2. the word vocabulary expressed as sequences of morphs

- Since we are doing unsupervised learning, we do not know the correct answer.

- Our target is to **minimize the combined code length** of:
  1. the code length of the morph lexicon
  2. plus the code length of the word vocabulary expressed using the morph lexicon.

Improved Morfessor model

Software: http://morpho.aalto.fi

- A later context-sensitive version of Morfessor introduces three categories: stem (STM), prefix (PRE) and suffix (SUF) that each morph must belong to.
- A word form must have the structure of the following regular expression: (PRE* STM SUF*)+
- From the updated examples below, you can see that many issues have been fixed, but the model is still fairly crude; for instance, it suggests two consecutive s-suffixes in the word “abyss”: aby s s.

```
abandon/STM ed/SUF
abandon/STM ing/SUF
abb/STM
abby/STM
abdel/STM
able/STM
ab/STM normal/STM
aboard/STM
about/STM
abroad/STM
abrupt/STM ly/SUF
absence/STM
absent/STM
absent/STM ing/SUF
```

```
absolute/STM
absolute/STM ly/SUF
absorb/STM
absorb/STM ing/SUF
absurd/STM
absurd/STM ity/SUF
abt/STM
abu/STM
abuse/STM
abuse/STM d/SUF
ab/STM users/STM
abuse/STM s/SUF
aby/STM s/SUF s/SUF
accent/STM
```

```
differ/STM
differ/STM ence/STM
differ/STM ence/STM s/SUF
different/STM
differential/STM
different/STM ly/SUF
differ/STM ing/SUF
difficult/STM
difficult/STM i/SUF es/SUF
difficult/STM y/SUF
dig/STM
digest/STM
digital/STM
diglipur/STM
```

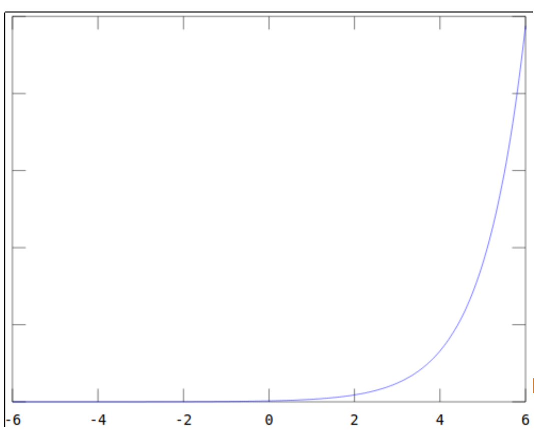
```
present/STM ed/SUF
present/STM ing/SUF
present/STM ly/SUF
present/STM s/SUF
preserv/STM e/SUF
preserv/STM e/SUF s/SUF
provide/STM s/SUF
provi/STM ding/STM
pull/STM ed/SUF
pull/STM er/SUF s/SUF
pull/STM ing/SUF
pump/STM
pump/STM ed/SUF
pump/STM ing/SUF
```

9. Neural Network LM

- Feed-Forward and Recurrent NNLM
- NNLM training
- Various architectures and modeling units
- Model combination

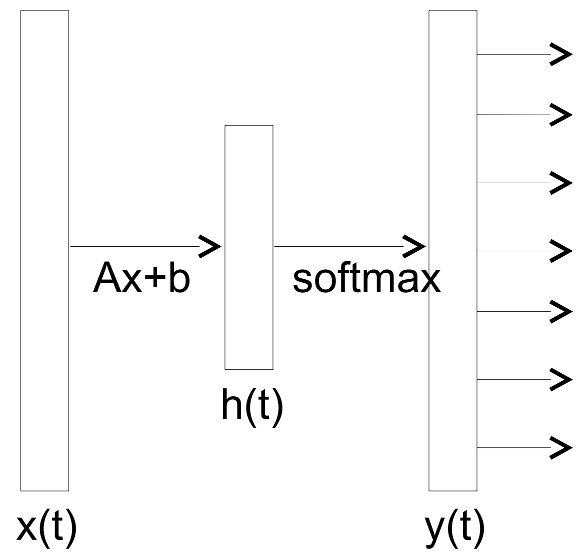
A linear bigram NN LM

- Outputs the *probability of next word* y(t) given the previous word x(t)
- **Input layer** maps the previous word as a vector x(t)
- **Hidden layer** has a linear transform h(t) = Ax(t) + b to compute a representation of *linear distributional features*
- **Output layer** maps the values by y(t) = softmax (h(t)) *to range (0,1) that add up to 1*
- Resembles a bigram Maximum entropy LM



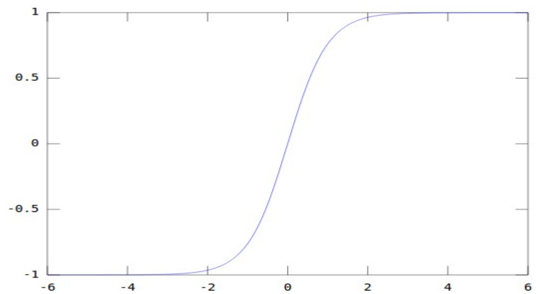
Softmax:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

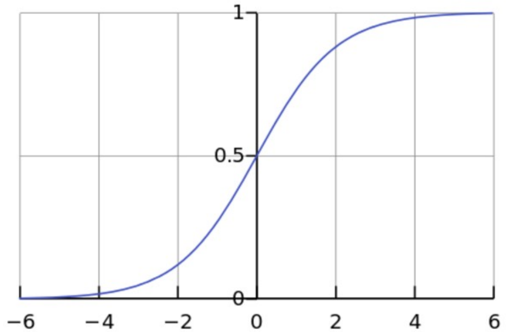


A non-linear bigram NN LM

- The hidden layer h(t) includes a non-linear function h(t) = U(Ax(t) + b)
- Can learn more complex feature representations
- Common examples of non-linear functions U:

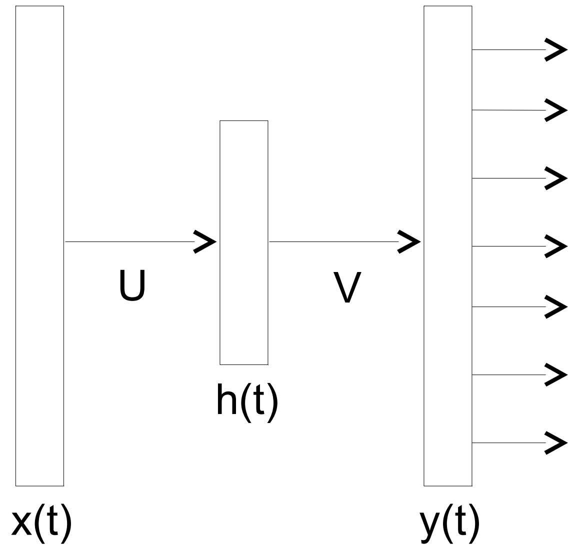


U (t) = tanh (t)



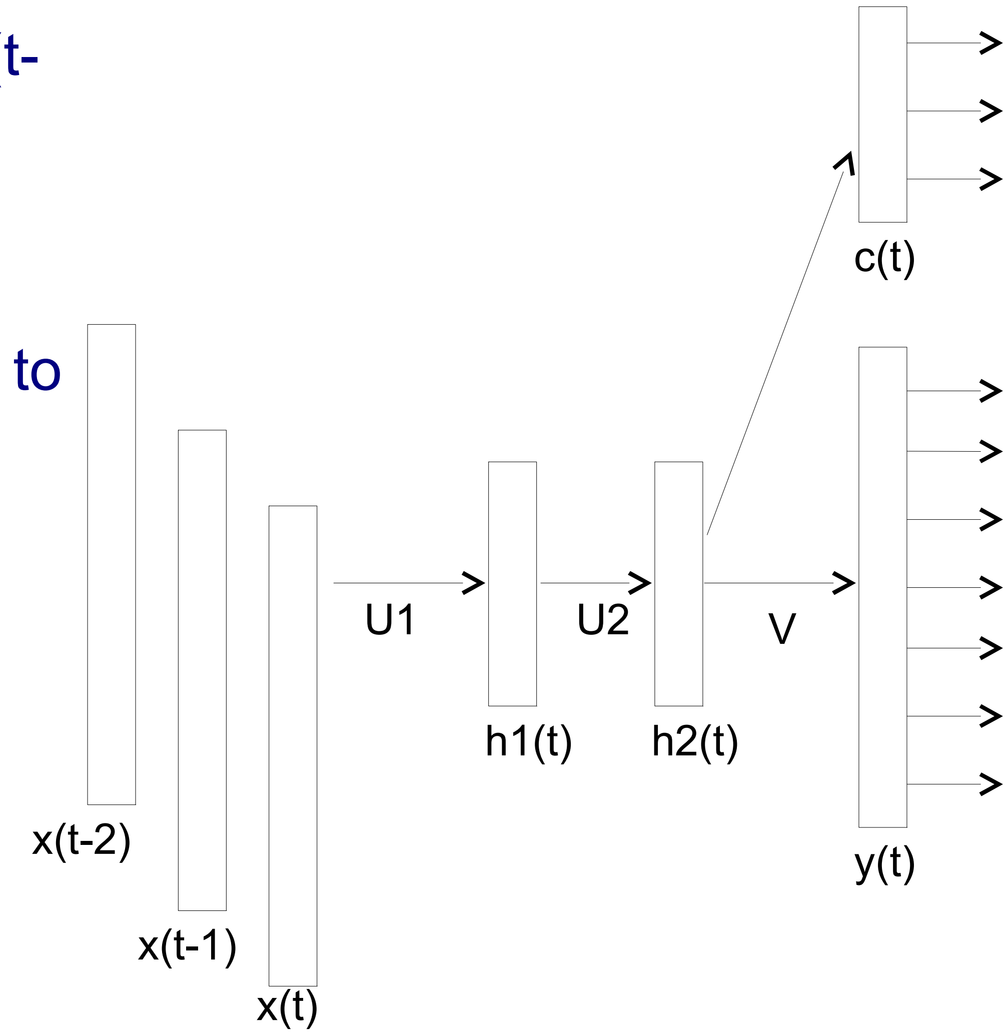
Sigmoid

$$U(t) = \frac{1}{1+e^{-t}}$$



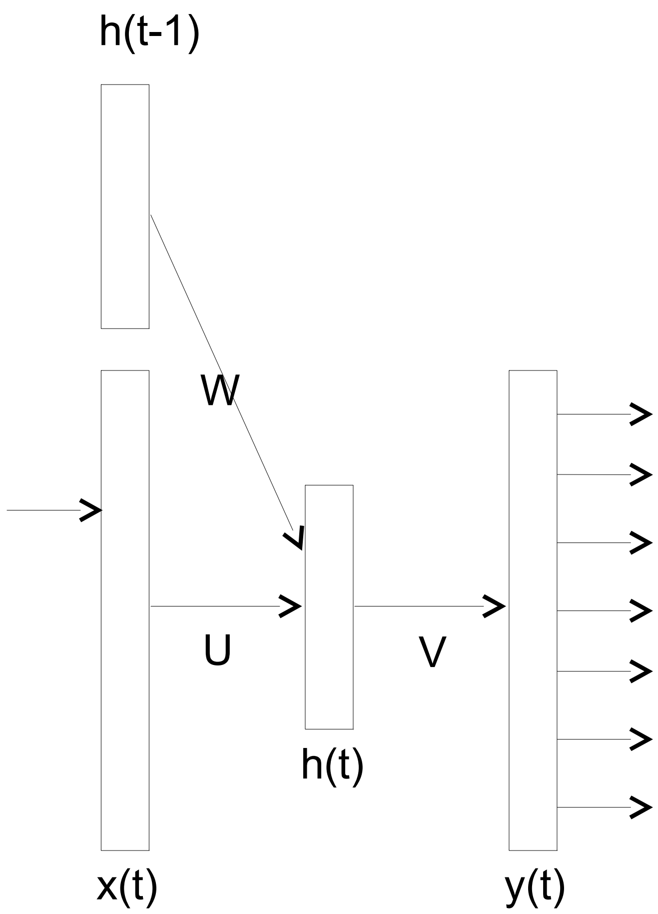
Common NN LM extensions

- ***Input layer*** is expanded *over several previous words* x(t-1), x(t-2), .. to learn richer representations
- *Deep neural networks* have *several* ***hidden layers*** h1, h2, .. to learn to represent information at several hierarchical levels
- Can be scaled to a very large vocabulary by training also *a class-based* ***output layer*** *c(t)*



Recurrent Neural Network (RNN) LM

- **Looks like** a bigram NNLM
- **But**, takes an additional input from the hidden layer of the *previous time* step
- Hidden layer becomes a compressed representation of the word history
- Can learn to represent unlimited memory, in theory
- Currently, the state-of-the-art in LMs



10. Neural machine translation

Neural machine translation

- Why NMT is the mainstream ∗ approach?
- How are the current NMT systems build?
- What are the challenges and limitations for the systems?

Evaluation of machine translation

- How machine translation systems are evaluated manually?
- How do the standard automatic metrics work and how they can be improved?
- What are the limitations of the metrics?

Why NMT?

- Generalization
- Flexibility
- Integration

Building NMT

- Encoding variable-length sequences
- Sequence decoding
- Sequence-to-sequence models
- Recurrent neural networks
- Attention model, Transformer model
- Modeling units

Evaluation of machine translation

- Human evaluation
  - Assessment, ranking, agreement, efficiency
- Automatic evaluation
  - Challenges?
  - Edit distance metrics
  - Precision & recall, BLEU
  - Beyond word-based metrics
- Meta-evaluation
  - Evaluation of evaluation metrics

Lecture topics

1. 12 Jan Introduction & Project groups / Mikko Kurimo
2. 19 jan Statistical language models / Mikko Kurimo
3. 26 jan Word2vec / Tiina Lindh-Knuutila
4. 02 feb Sentence level processing / Mikko Kurimo
5. 09 feb Speech recognition / Janne Pylkkönen
6. 16 feb Chatbots and dialogue agents / Mikko Kurimo
7. 02 mar Statistical machine translation / Jaakko Väyrynen
8. 09 mar Morpheme-level processing / Mathias Creutz
9. 16 mar Neural language modeling and BERT / Mittul Singh
10. 23 mar Neural machine translation / Stig-Arne Grönroos
11. 30 mar Societal impacts and course conclusion / Krista Lagus, Mikko