C2: Regex, Text Normalization, Edit Distance

- If the caret ^ is the first symbol after the open square brace [,the resulting pattern is negated. /[^A-Z]/ -> not an upper case letter
- * -> zero or more
- + -> at least one
- Wildcard -> Any str /beg.n/
- ^ start of line
- $ end of line

-The operator *? is a Kleene star that matches as little text as +? Possible
- ? exactly zero or one occurrence of the previous char or expression
-{n,m} from n to m repeat of the previous char or expression
- \b word boundary
- \B non-word boundary

Heaps Law - How many words are there in English?

$|V| = kN^{\beta}$ len(Vocab) = V, N tokens we seen so far.

Byte Pair Encoding (BPE)

1. start with symbol-vocabulary of characters + end-of-word-character.
2. for most frequent character n-gram pair in words: create new n-gram.
3. Iterate -> word segmentation into character n-grams

Wordpiece algorithm: Fixes rare word problem

Only difference using probability p

C3 : N-gram Language Models

The assumption that the probability of a word depends only on the previous word is called a Markov assumption.

$$P(w_1, w_2, \ldots, w_n) = \prod_{k=1}^{n} P(w_k|w_{1:k-1})$$

N= N-gram size n-> current index

$$P(w_n|w_{n-N+1:n-1}) = \frac{C(w_{n-N+1:n-1}\ w_n)}{C(w_{n-N+1:n-1})}$$

Perplexity PP of TeSet: (lower PP = better model):

General Formula

$$PP(W) = P(w_1 w_2 \ldots w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 \ldots w_N)}}$$

Bigram

$$\underset{\text{(for a bigram model)}}{=} \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}}$$

Laplace Smoothing

Unigram

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

Bigram

$$\frac{C(w_{n-1}w_n) + 1}{C(w_{n-1}) + V}$$

Adjusted ("discounted") count

Unigram

$$c_i^* = (c_i + 1)\frac{N}{N+V}$$

Bigram

$$c^*(w_{n-1}w_n) = \frac{[C(w_{n-1}w_n) + 1] \times C(w_{n-1})}{C(w_{n-1}) + V}$$

Backoff : use trigram if evidence sufficient, if not use bigram, if bigram evidence too low use unigram

$$P_{BO}(w_n|w_{n-N+1:n-1}) = \begin{cases} P^*(w_n|w_{n-N+1:n-1}), & \text{if } C(w_{n-N+1:n}) > 0 \\ \alpha(w_{n-N+1:n-1})P_{BO}(w_n|w_{n-N+2:n-1}), & \text{otherwise.} \end{cases}$$

Interpolation

$$\hat{P}(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

$$\sum_i \lambda_i = 1$$

Absolute Discounting: Divide corpora into 2 part and count bigrams of first part, we will see that there are 449k bigram that seen 2 times in the first part. and if we count same bigrams in second corpora it seen 564153. So we can add this probability to unseen example instead of adding 1 (laplace). They are distinct bigram counts

$$P_{\text{AbsoluteDiscounting}}(w_i|w_{i-1}) = \underbrace{\frac{C(w_{i-1}w_i) - d}{\sum_v C(w_{i-1}v)}}_{\text{discounted bigram}} + \underbrace{\lambda(w_{i-1})}_{\text{Interpolation weight}}\underbrace{P(w_i)}_{\text{unigram}}$$

$d \approx 0.75$

Kneser Ney Smoothing : Pcont: "How likely is w to appear as a novel continuation?". Pcont proportional to number of different (bigram) contexts that w has appeared in TrSet

$$P_{\text{CONTINUATION}}(w) = \frac{|\{v : C(vw) > 0\}|}{\sum_{w'} |\{v : C(vw') > 0\}|}$$

$$\lambda(w_{i-1}) = \underbrace{\frac{d}{\sum_v C(w_{i-1}v)}}_{\text{the normalized discount}} \underbrace{|\{w : C(w_{i-1}w) > 0\}|}_{\text{The number of word types that can follow } w_{i-1}}$$

= # of word types we discounted
= # of times we applied normalized discount

$= d / C(w_{i-1})$

$$P_{KN}(w_i|w_{i-1}) = \frac{\max(C(w_{i-1}w_i) - d, 0)}{C(w_{i-1})} + \lambda(w_{i-1})P_{\text{CONTINUATION}}(w_i)$$

C4: Naive Bayes And Sentiment Classification

$$\hat{c} = \underset{c \in C}{\text{argmax}}\, P(c|d) = \underset{c \in C}{\text{argmax}}\, \frac{P(d|c)P(c)}{P(d)}$$

c is class d is input

Bayes theorem:

- **P(d) is constant** we delete it
- Naive Bayes is **generative** since stating implicit assumption about how a document is generated
- **Why Naive**: assumption that the probabilities P(fi|c) are independent given the class c and hence can be 'naively' multiplied

Assumption2: Position doesn't matter.

$$\hat{P}(c) = \frac{N_c}{N_{doc}} \quad \hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V}(count(w, c) + 1)}$$

$$\theta_{MLE} = \frac{N_{vc} + 1}{\sum_{v=1}^{V} N_{vc} + |V|}$$

- We add Totally V(Union of all word types) to denominator since and 1 to nominator for smooting. (prevent If one term is 0, all equation is 0)

C5: Logistic Regression

Naïve Bayes	Logistic Regression
Good for small documents independence assumption unrealistic in most cases cannot deal well with correlated features fast training	can robustly deal well with correlated features may work better on large documents

C6: Vector Semantics and Embeddings

Word senses (bank1, bank2) (mouse1, mouse2)

$\text{tf}_{t,d} = \text{count}(t, d)$ the frequency of the word t in the document d

$\text{idf}_t = \log_{10}\left(\frac{N}{\text{df}_t}\right)$ N is the total number of documents, and dft is the number of documents in which term t occurs.

Tf-Idf -> $w_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$ **-> Sparse**

At the same time, they encode very little information making it hard to compare documents or singular sentences efficiently.

PPMI(Cooccurrence of 2 words) =

$$\text{PPMI}(w, c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

Skip-Gram:

$$-\left[\log P(+|w, c_{pos}) + \sum_{i=1}^{k} \log P(-|w, c_{neg_i})\right]$$

substract negative samples(Hinge loss)

Negative sampling is less computationally intensive than calculating the full probabilities over all possible tokens in the vocabulary

C8: Sequence Labeling for Parts of Speech and Named Entities

- Hidden Markov Model (HMM) – Generative

A Markov chain makes a assumption that if we want to predict the future in the sequence, all that matters is the current state. All the states before the current state have no impact on the future except via the current state.

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})} \qquad P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

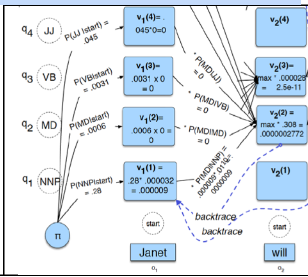
First one for transition from states second one is emission prob of word w in that state

Viterbi Algorithm : decoding algorithm for HMMs

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

V11 den v2 2 ye geciste o2 nin will olma olasiligi .308 * onceki statelerin maxi v11 olan yani 0.00009



- No of states N large -> Viterbi O((N^2)*T), inefficient
- Beam search keep most probable B
- Problems of HMM: unknown words

CRF discriminative:

$$p(Y|X) = \frac{\exp\left(\sum_{k=1}^{K} w_k F_k(X, Y)\right)}{\sum_{Y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} w_k F_k(X, Y')\right)} \qquad F_k(X, Y) = \sum_{i=1}^{n} f_k(y_{i-1}, y_i, X, i)$$

restriction to current and previous tags: "linear chain" → variants of HMM algorithms (e.g. Viterbi) can be used

They use the Viterbi algorithm for inference

C9: DL Architectures for Sequence Processing

Why we need it ? Language is a temporal phenomenon. Sequence of spoken/written words

RNNs

$$\mathbf{h}_t = g(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t) \qquad W \in \mathbb{R}^{d_h \times d_{in}}$$
$$\mathbf{y}_t = f(\mathbf{V}\mathbf{h}_t) \qquad U \in \mathbb{R}^{d_h \times d_h}, \quad V \in \mathbb{R}^{d_{out} \times d_h}$$

Advantages	Disadvantages
-Can process any length input	-Recurrent computation isn't parallel, slow
-Model size same for longer input	-Hard to access info from many steps back
-Step t depends step t-many steps	-Vanishing gradient: Repeated multiplications drive gradient to zero
-Weights are shared across timestamps	

Teacher Forcing is giving correct word embedding for each state because if we made a mistake in first word prediction, all remaining states meaningless.

For Sequence Labelling -> assigning a label to each element of a sequence, softmax each output

For Sequence Classification-> Last Hidden layer + softmax

- Alternative mean hidden layers
- no intermediate outputs = no loss terms for those elements
- loss is computed only on final task (end-to-end training)

LSTM

○ Removing unneeded information: **forget gate** controls how much of the previous memory to keep (kt)
○ Adding new information: **input gate** controls how much of the proposed update to keep(jt)
○ Keeping separate states memory C and hidden state H

$$\mathbf{k}_t = \mathbf{c}_{t-1} \odot \mathbf{f}_t \quad \mathbf{c}_t = \mathbf{j}_t + \mathbf{k}_t$$

$$\mathbf{j}_t = \mathbf{g}_t \odot \mathbf{i}_t \quad \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

Advantages	Disadvantages
-Gates allow more distant information flow	-Still information loss - Still slow

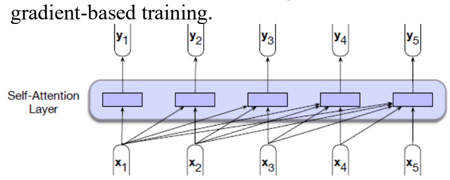
Self-Attention

- Extract information from arbitrary length context
- No passing through intermediary recurrent connections
- Access to all information from previous inputs =
- availability of long term context
- parallelizable
- Query: current focus of attention
- Key: preceding input being compared to the query
- Value: used to compute the output
- Attention is quadratic w.r.t. length of the input

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^{\mathbf{Q}};\ \mathbf{K} = \mathbf{X}\mathbf{W}^{\mathbf{K}};\ \mathbf{V} = \mathbf{X}\mathbf{W}^{\mathbf{V}}$$

$$SelfAttention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d_k}}\right)\mathbf{V}$$

- **Residual Connections** are passing information from a lower layer to a higher layer without going through the intermediate layer. Allowing information from the activation going forward and the gradient going backwards to skip a layer improves learning and gives higher level layers direct access to information from lower layers
- **Layer normalization** can be used to improve training performance in deep neural networks by keeping the values of a hidden layer in a range that facilitates gradient-based training.



Multihead Attention

- input is copied across all heads
- computed in parallel at the same depth of the model
- at the end of a block reduced back to the original dimensionality by
- concatenation and linear projection via weight matrix !"

Transformers

Normalizedan sonra Value ile carpim

Advantages	Disadvantages
- Calculations are no longer serial	-Still information loss - Still slow
- Parallelization possible!	
- Resulting model can be evaluated via perplexity	
- New text can be autoregressively generated	

C10: Machine Translation and Encoder-Decoder Models

Problems: Word Order Typology, Lexical Divergences, Morphological Typology, Referential density

Approach

- Give source text to model

- Starting with separator token: train autoregressively to predict next target word
- Calculate loss for each token
- Average token losses for sentence level loss

Teacher forcing

- Inference: Previous prediction as input for next word
- Training: Gold target token instead of prediction as input
- not propagate wrong predictions
- speed up training

Problem: Encoder final state only info about source sentence folder

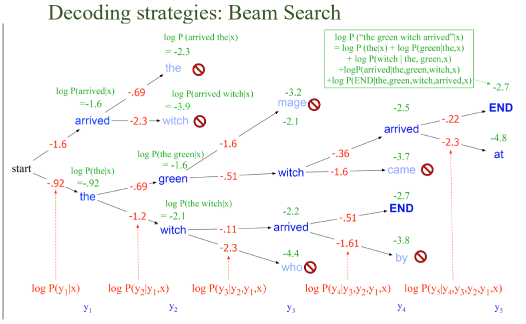
Dot-Product Attention: Include all encoder hidden states in context vector

- Weighted sum of encoder hidden states -> focus on relevant part in source text
- Context vector different for each decoding step 21

$$score(\mathbf{h}^d_{i-1}, \mathbf{h}^e_j) = \mathbf{h}^d_{t-1}\mathbf{W}_s\mathbf{h}^e_j$$

- Weights optimized during training
- Allow different dimensions in encoder and decoder
- Learn aspects relevant in current application

Decoding strategies: Beam Search



- Select most probable k in each stage.
- Generate hypothesis until </s> produced
- Problem: Generally lower probability for longer strings Solution: length normalization

Cross-attention: keys and values from encoder output, query is from decoder

What is achieved by employing masked attention?

This self-attention is masked to prevent the positions to attend to subsequent positions, so as tohavea"forwarding" attention effect, instead of orienting backwards.

Wordpiece algorithm:

○ Input: Training corpus and desired vocabulary size V
○ 1. Init lexicon with all characters
○ 2. Repeat until V wordpieces in lexicon
– Train n-gram LM on training corpus using current lexicon
– New wordpieces by concatenating two from current lexicon
- add most probable piece according to language model to lexicon

Evaluation Metrics: **Character F-score (chrF)**

$$\text{chrF}\beta = (1+\beta^2)\frac{\text{chrP}\cdot\text{chrR}}{\beta^2\cdot\text{chrP}+\text{chrR}}$$

Commonly B is 2
ChrP -> Mean of 1-gram, 2gram,… precision (x/len(hypo))
ChrR -> Mean of 1-gram, 2gram,… precision (x/len(ref))
Precision -> hypothesis that occur in the reference
Recall -> reference that occur in the hypothesis
Reference: Target sentence

Chapter11: Pretrained Models and Contextual Embed

- Casual (left-to-right) transformers are powerful for autoregressive generation
- For e.g. sequence classification usage of information only from the left context is not enough. Bidirectional Attention

How to train bidirectional transformers?

Masked Language Modeling (MLM):

- Change with [MASK]
- It is changed by random token to learn better. I ate sandwich lunch. If you change lunch to random, model learn "ate" better

Masking Spans

A span (one or more words) selected for masking

Span Boundary Objective (SBO):

$$L(x) = L_{MLM}(x) + L_{SBO}(x)$$
$$L_{SBO}(x) = -logP(x|x_s, x_e, p_x)$$

s denotes the word before the span and e denotes the word after the span, p_x: positional embedding (which word in the spam is currently predicted).

Next Sentence Prediction: given a pair of sentences, to predict whether a pair consists of a pair of adjacent sentences or a pair of unrelated sentences.

Chapter12: Constituency Grammars

Terminal : Words

Non-terminal : The symbols that express abstractions

- Nominal -> Noun | Nominal Noun
- Left part of error cannot be terminal\
- Two grammars are **weakly equivalent** if they generate the same language
- **Strongly equivalent** if they generate the same language and the same phrase structure
- X/Y : function that seeks its argument to the right

$$X\ CONJ\ X \Rightarrow X \ <\Phi>$$

function composition: $X/Y\ Y/Z \Rightarrow X/Z$ $\quad \xrightarrow{\quad} B$ S/NP
$Y\backslash Z\ X\backslash Y \Rightarrow X\backslash Z$

type raising: $X \Rightarrow T/(T\backslash X)$ $\quad \xrightarrow{\quad} T$ S/(S\NP)
$X \Rightarrow T\backslash(T/X)$

Why what questionlarinda long term dependency var CCG -> Az rule var. Her bir lexion elementi cok anlam tasir. Little emphasis on lexicon, many non terminal. CFG-> Cok rule var. Her bir bir rule az anlam tasiyor..

Chapter13: Constituency Parsing

Syntactic Parsing: sentence -> parse tree

- Applications: information extraction + question answering
- **Ambiguity**: assign more than one parse tree to a sentence
  - **Attachment ambiguity**: a particular constituent can be attached to the parse tree at more than one place
  - **coordintion ambiguity**: join different sets of phrases by and. Example: [old [men and women]] [old men] and [women]

CKY Algorithm: two parts: recognizer + actual parser.

Typically: convert to Chomsky Normal Form (CNF) first

- Convert into Chomsky Normal Form (CNF) :
  1. **eliminate terminals** which are not alone in the right side. Each terminal should be alone.
  2. **eliminate unit productions:** Eliminate non-terminal to non-terminal.

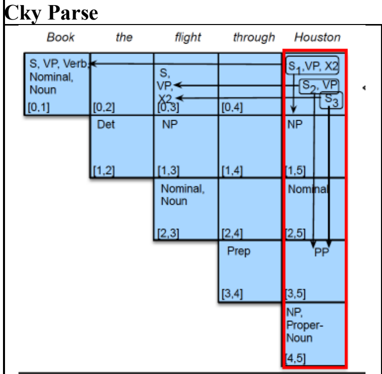
if $A \overset{*}{\Rightarrow} B$ by a chain of one or more unit productions and $B \to \gamma$ is a non-unit production in our grammar, then we add $A \to \gamma$ for each such rule in the grammar and discard all the intervening unit productions

3. **iteratively shorten long productions**

$$A \to B\, C\, \gamma \implies \begin{cases} A \to X1\ \gamma \\ X1 \to B\ C \end{cases}$$

motivation for subcatagorization of verb-phrases in ml cons parses? it enables the parsers to learn the syntactic frames in which verbs can appear and accurately identify the constituents of a sentence.

Cky Parse



Left-to-right Bottom-to-upper

Cyk parse can lead to exponential number of parse trees. To fit that we use SpanBERT. SpanBERT Gives creates 2 embeddings for each token.

They represent Left and right span.We give span indexes and map resulting embedding to one of constituency (NP,VP)

Chunking: Sadece belirli consituencyleri labellamak. VP,NP Her bir consituent bir chunk BIO tagleri ile etiketlenir.

2 ways to fix ambiguity

Supertagging: Select most likely constituency for word.

Better Search Algorithm:

$$f(w_{i,j},t_{i,j}) = g(w_{i,j}) + h(w_{i,j})$$
$$= \sum_{k=i}^{j} -\log P(t_k|w_k) +$$
$$\sum_{k=1}^{i-1} \min_{t \in tags}(-\log P(t|w_k)) + \sum_{k=j+1}^{N} \min_{t \in tags}(-\log P(t|w_k))$$

G(n)'de kurala uyani al, h(n) diger hepsinin minimunu

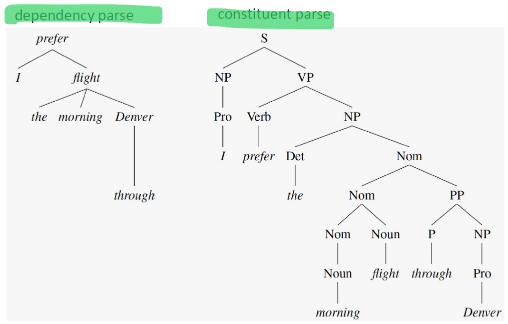
Chapter14: Dependency Parsing:

syntactic structure of a sentence is described solely in terms of the words + associated set of directed binary relations among the words projective if ∃ path from h to every word w that in the sentence lies between h and d.

Projectivity : An arc from a head to a dependent is said to be projective projective if there is a path from the head to every word that lies between the head and the dependent in the sentence. If a language has flexible word order, it is lead to non-projective.

LAS - > Look head and head relation h

UAS -> Just look if head is True



Eger baban Head degil ise head olan kuzenine git. Eger baban head ise babanin soyundan non head bulana kadar git. Non head buldugun zamanki, kuzenine bagla. Eger hic bulamazsan direct root a bagla.

Transition-Based Dependency Parsing

Graph-Based Dependency Parsing advantages compared to transition-based approaches:

- non-projective trees possible
- better with long-range dependencies (especially in other languages than English) (scoring entire trees than just making greedy local decisions)

Maximum spanning tree

1. Subtract best incoming edge for each node.
2. Select highest edges If there is a cycle exists, Merge cycle edges nodes as a 1 node.
   Find a solution.
   When extracting you have already incoming edge for that node, you can delete other one.

C15:Logical Representations of Sentence Meaning

Reichenbach's Reference Point:
E -> Eventin oldugu zaman
u-> soylemin gerceklestigi zaman
r-> olayin bittigi zaman

C17:Information Extraction

- Since NER provides previous constitute label affects current token label, Sequence labelling is ideal for Information Extraction.
- **Bootstrapping** here means utilizing known relations to automatically learn new rules from unlabeled text. Vodofone(ORG) has a hub in Istanbul(LOC)
  Seed: hub(vodofone,Istanbul)
- This is done by extracting seed tuples with a specific relation and then using pattern matching to find new entities that adhere to the same pattern as present in the seed tuples. (Look 3 words contains sentences and extract patterns)
- You can assign confidence scores based on how close the matching fits to the seed tuples. Hits ->set of tuples in T that p matches while looking in D Finds-> total set of tuples that p finds in D (|hit| / |find|)* log(|finds|)
- Since open relation extraction is a self reinforcing systems, errors in the extracted relations will propagate in future steps and can multiply the error. We can label some data, calculate estimated precision and recall.
- **Distant Supervision** for Relation Extraction-> use relation databases to produce large number of seeds
- **Temporal Normalization->** tomorrow = anchor of today + 1d

C18:Word Senses and WordNet

1. We could use contextual embedding and use cosine similarity for finding closest sense in Semcore or wordnet.
2. LESK: look the sentence and find overlap words in definition or example in dictionary
3. For each occurrence token, compute a context vector, cluster them. Find closest sense in test time by cosine.

C20: Lexicons for Sentiment, Affect and Connotation

SentProp defines a similarity graph on word embeddings. A word w is connected to its k nearest neighbors by cosine angle. Select 2 seed words positive and negative are known. Then propagate polarities by a random walk on the graph, determining the polarity of a new word by the number of visits from other positive or negative seeds. At the end you need to normalize the estimated scores.

C21: Coreference Resolution

$$J = -\sum_{i=2}^{N}\sum_{j=1}^{i} y_{ij}\log p(m_j, m_i)$$

Iterate through mentions

Iterate through candidate antecedents (previously occurring mentions)

Coreferent mentions pairs should get high probability, others should get low probability

Even though the model may not have predicted this coreference link, I and my are coreferent due to transitivity

C22: Discourse Coherence