Machine Translation

Lukas Bach - lbach@outlook.de - lukasbach.com

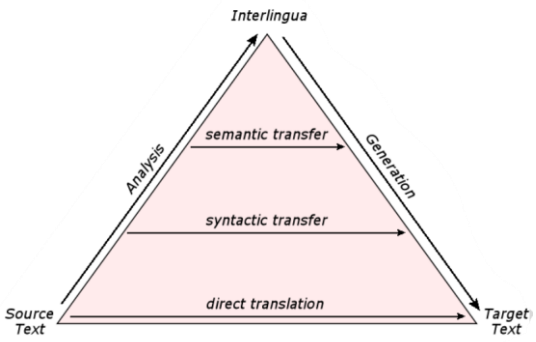
Preliminaries

1 Introduction

Machine Translation approaches:

- Rule-based MT: Translate using handcrafted rules
- Corpus-based MT: Learn from corpus how to translate between languages

Vauquois triangle:



- Direct translation: Dictionary-based or translation memory, e.g. just look up words or store many translated sentences in memory.
- Transfer based: Analyze into syntactic structure
- Interlingua based MT: Language independent structure, find pure meaning of sentence. Semantic grammar and logic calculus. No n-to-n translations.
  - Syntactic Transfer: transfer syntactic tree, regenerate sentence in target language
  - Semantic Transfer: transfer semantic representation, e.g. logic formula.
- Pivot MT: Use natural language, e.g. english, as intermediate representation (more data available)

1.1 Translation Memory

Store many translations (source-target sentence pairs). Only useful when you expect the same/similar sentence to come up again, but very useful then. Closest match of sentence is presented.

1.2 EBMT

Example-based Machine Translation. Direct Translation. Uses parallel-sentence data. Similar to translation memory, but not closest match, rather translate phrases within matched sentences. Higher coverage than translation memory (because you can not just match identical sentences).

2 Linguistic Approaches

2.1 Linguistic Background

- **Word Level**
  - *Tokenization*: Segment sentence of characters into words. Western languages: Split on spaces.
  - *Morphology*: Internal structure of words.
    Morpheme: Smallest meaning-carrying units of a language.
    E.g. un-predict-able, house-s, …
    Morpheme types:
    - Stem morphemes: Morpheme as complete word
    - Functional/bound morpheme: prefix, suffix, infix, circumfix

    Compositionabilities:
    - Inflection: tenses, count, person, case.
      Kauf-st, kauf-te, Freund-e
    - Derivation: Form a new word
      Happi-ness, un-predict-able, un-brauch-bar
    - Composition: Create new words from stem morphemes
      Rain-bow, water-proof

    Morpho-phonologicial processes: e.g. Fugen-S
  - *Word classes*
    Role of word in sentence determined by its POS (part of speech)
    - Noun, verb, adjective, adverb, …

    POS tagger: assign POS to words based on relative frequency counts and context in corpus
  - *Grammatical Categories*
    - (Pro)nouns, adjectives
      Person, number, gender, case
    - Verbs
      Tense, aspect, mood, voice
  - *Lexical Semantics*
    Whjch semantic meaning does a word have?
    - Polysemy: same words with different but related meanings, e.g. “bank”
    - Homonymy: same words have completely different meanings, e.g. “can”

    Word sense disambiguation necessary
- **Sentence Level**
  - $SUBJECT\ VERB\ (OBJECT)^*$
  - Syntactic theory: Form rules on how sentences are formed. Context-free grammar.
  - Syntactic parsing: Bottom-up or top-down.
    See F04-27_22-29.
  - Phenomena: cannot be covered by simple CFGs

…F04-28_29ff

2.2 Translation Challenges
2.3 Linguistic approaches to MT

3 Evaluation

Problems:

- Ambiguity (Multiple correct translations)
- Subjective
- Small changes can be very important
- Application dependent

Criteria: Adequacy and Fluency.

Granularity: Sentence-based, Document-based or Task-based (e.g. understanding questions after lecture).

3.1 Manual Evaluation

Gold standard, but subjective and expensive.

- Inter annotator agreement
- Intra annotator agreement

Assessment types:

- Direct (manually score adequacy/fluency)
- human ranking of hypothesis
- measure post-editing effort
- task based (sufficient for task?).

3.2 Automatic Evaluation

Cheap and fast, but often only using human reference generated only once.

- BLEU: Compare machine translation with human translation
  - N-Gram overlap. E.g. 4-BLEU uses 1- to 4-grams
  - $BLEU = \sqrt[4]{P_1 \cdot P_2 \cdot P_3 \cdot P_4}$
  - Brevity Penalty (to punish shorter outputs):
    $$BP = \begin{cases} 1 & c > r \\ e^{1-\frac{r}{c}} & c \leq r \end{cases}$$
    r =reference length, c = output length
  - Adequacy modeled by word-precision, fluency modelled by n-gram precision (larger n-grams).
  - Weaknesses: Brevity penalty compensating lack of recall, requires exact word matches, all matched words weigh equally, geometric average is volatile to zero-scores.
- TER (Translation Error Rate): See WER
  - Compare using edit operations (insert, del, subst. shift)
- METEOR (Metric for Evaluation of Translation with Explicit Ordering)
  - Recall and Precision as scores
  - Only unigrams, align output with each reference individually and take scores of best pairings
  - Fluency scored by measuring fragmentation
- Tunable Metrics
  - TODO?

Ideal goals for metrics:

- High correlation with human performance
- Interpretable
- Sensitive
- Consistent
- Reliable
- Easy to run
- General

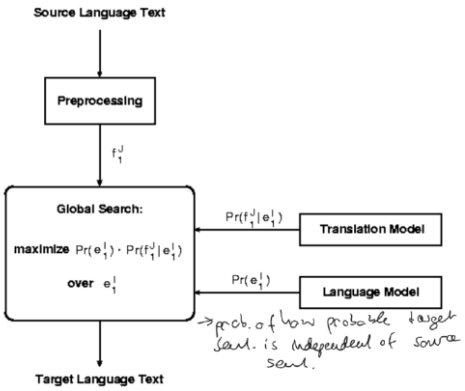
SMT

4 Word Alignments

Target: Find most probable translation

$$\hat{e} = argmax_e\, p(e|f) = argmax_e \frac{(p(f|e)p(e))}{p(f)}$$
$$= argmax_e\, p(f|e)p(e)$$

- $p(f|e)$ is modelled by translation model
- $p(e)$ is modelled by language model



4.1 Preliminaries

4.1.1 Alignment

$$a\colon [e] \mapsto [f], j \to i$$

For a specific alignment a, maps the position of a translated English word to its original foreign word.

The IBM2 alignment probability distribution $a(i|j, l_e, l_f)$ maps into the same direction, predicting foreign input word positions conditioned on English output word positions.

4.1.2 Distortion

The IBM3 distortion probability distribution $d(j|i, l_e, l_f)$ maps into the translation direction, predicting output word positions based on input word positions.

4.2 IBM Model 1

Break into words, translate word-to-word. t denotes the translation probability, l_i the sentence lengths, $(l_f + 1)^{l_e}$ the number of possible alignments, +1 due to NULL token.

ε is a normalization constant s.t. all probabilities sum up to 1.

$$p(e,a|f) = \frac{\varepsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$

4.2.1 Training

Problem: No word-aligned data to train from.

Solution: EM

1. Initialize model from uniform distributions
2. Expectation: Apply model to data
   - Compute alignment probabilities, i.e. how likely is it that $House \mapsto Haus$?
   - Dynamic programming: $O(e^n) \Rightarrow O(n^2)$
3. Maximization: Learn model from data
   - Learn translation probabilities from alignment, use best alignment according to probabilities
   - Learn probabilities by counting sentence pairs
4. Repeat steps 2&3 until convergence

Dynamic programming reformulation:

$$p(e|f) = \sum_a p(e,a|f)$$
$$= \sum_{a(1)=0}^{l_f} \dots \sum_{a(l_e)=0}^{l_f} \frac{\varepsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})$$
$$= \frac{\varepsilon}{(l_f+1)^{l_e}} \prod_{j_1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)$$

Uses perplexity ("loss function") to evaluate/minimize for. Converges to global minimum.

TODO Johanna: Formula 4.14, page 91, lower sum, which e?

4.2.2 Language Model

Ensure fluency. TODO

4.2.3 Noisy Channel Model

Combine language model and translation model. TODO

4.3 IBM Model 2

Model likelihood of absolute alignment based on positions of input and output words (rather than uniform likelihoods as in model 1). Uses alignment probability distribution $a(i|j,l_e,l_f)$. Two steps:

1. Lexical translation step
2. Alignment step

4.4 IBM Model 3

Introduce fertility, i.e. likelihood that more/less than one English word is generated by one foreign word.

$$n(\phi|\mathrm{f}) = p$$

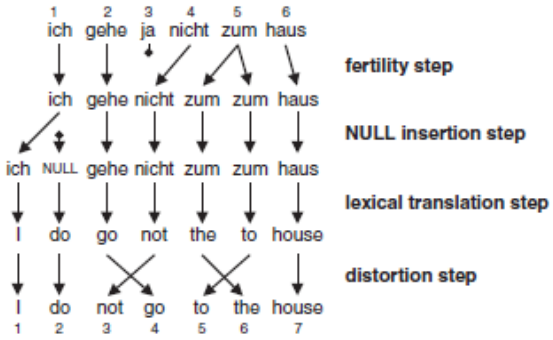
i.e. $n(2|natürlich) \cong 1, n(1|natürl.) \cong 0, n(0|ja) \cong 1$

Word insertions modelled "add NULL insertion step", adds NULL token after every word with probability p_1 or not with probability $p_0 = 1 - p_1$.

Four steps:

1. Fertility step, remove/duplicate words
2. NULL insertion step, insert NULL tokens after words
3. Lexical translation step, as in IBM1
4. Distortion step, reorder words

Why distortion instead of alignment? If we duplicate "zum" in step 1, translate both "zum" tokens in step 3 to "to" and "the", two alignments are possible (either "to" or "the" at first). But for both alignments, the alignment function would be identically defined (always maps to "zum").



If fertility is greater than one, we have different tableaus (possible productions) for the same distortion.

TODO Formula? F05-07-01_48.

Dynamic programming no longer feasible (exp. # of distortions)

4.5 IBM Model 4

Model relative alignments. Model 3 suffers for long sentences, when movement estimates become sparse and unrealistic.

Each input word f_i aligned to at least one output word $[e_{j_1}, \dots, e_{j_k}]$ (subsequent words with potential gaps), forms a cept π_i with center $\odot_i = \lceil \frac{1}{k} \sum j \rceil$. The relative distortion for

- Target words generated by NULL token is uniformly distributed.
- The first output word e_j in cept π_i is $d_1(j - \odot_{i-1})$, i.e. target position relative to center of preceding cept.
- Subsequent output word $e_{j+k}, k > 0$ in cept π_i is $d_{>1}(j - \pi_{i,k-1})$, i.e. target position relative to position of previous word within the same cept.

TODO Johanna: Exists a definition for the d function?

4.5.1 Word classes

Problem: Some words are much more often reordered than others. We may be tempted to condition distortions

for translations ($d_1(j - \odot_{i-1} | f_{[i-1]}, e_j)$), but we don't have sufficient statistics for that.

Instead group words into classes using POS tags, and condition for word class instead.

4.6 IBM Model 5

Problem: Multiple output words can be placed in the same position, thus we have positive probability for impossible alignments.

Prevent those in Model 5. Not that often used.

4.7 HMM Viterbi Alignments

Viterbi-path describes a word-alignment in the $f - e$-Matrix and can be outputted in the last iteration of IBM training.

Problem: IBMs can align multiple English words to one foreign (input) word, but not one English word to multiple foreign words.

5 Language Modelling

Goal: Produce fluent english. Outputs soft probabilities. Trainable on large databases. Decomposed into per-word-predictions:

$$p(w_1, w_2, \dots, w_N) = p(w_1) \cdot p(w_2|w_1) \cdot \dots \cdot p(w_N|w_1, \dots, w_{N-1})$$

Arbitrarily long "histories" (probability condition) not feasible, thus limit to specific number of previous words taken into account (Markov assumption: Word probability depends only on n preceding words)

5.1 N-Grams

$$p(w_3|w_1, w_2) = \frac{count(w_1, w_2, w_3)}{\sum_w count(w_1, w_2, w)}$$

(for 3-grams)

5.2 Measuring LM Quality

For a word sequence, compute its cross entropy $H(p_{LM})$ on a language model LM, and from that its perplexity $PP = 2^{H(p_{LM})}$ ("branching factor").

TODO?

5.3 Count Smoothing

Problem: N-Gram has not seen many combinations. Maximum-Likelihood estimation ➔ probability of zero. Idea: Assign positive probabilities to unseen n-grams.

- Add-One Smoothing: Add fixed number, e.g. 1, to every count
  - No zero counts, no zero probabilities
  - $p = \frac{c}{n} \Rightarrow p = \frac{c+1}{n+v}$, v = #possible n-grams
  - Too much weight to unseen examples
- Add-α Smoothing: Add $\alpha < 1$
  - $p = \frac{c}{n} \Rightarrow p = \frac{c+\alpha}{n+\alpha v}$
  - α optimized on perplexity and held-out data
- Deleted Estimation
  - TODO?
- Good-Turing
  - TODO?

5.4 Interpolation and Backoff

Idea: Say we use 3-grams, we have never seen "Scottish beer drinkers", but have seen "beer drinkers".

- Interpolation
  - $p_I(w_3|w_1w_2) = \lambda_1 p_1(w_3) + \lambda_2 p_2(w_3|w_2) + \lambda_3 p_3(w_3|w_1w_2), \sum\lambda_i = 1, 0 \leq \lambda_i \leq 1.$
  - Weights optimized on held-out data
- Backoff
  - Trust highest n-gram with counts > 0.

5.5 Other LM Approaches

5.5.1 POS-based LM

Replace words by Part-of-Speech Tags (PP, VP, PREP, …). Much smaller vocabulary, can model long-range dependencies.

5.5.2 Cache Language Model

- Static LM component: n-gram on trained LM
- Dynamic LM component: n-gram trained on the fly, updated with newly translated text
- Final model: $P_T = \lambda P_D + (1 - \lambda) P_S$

5.5.3 Trigger Language Model

Problem: Some words depends on word far back in history.

For every word w keep Trigger-list $L(w)$ of words that are likely to occur some tome later. E.g.

$$L(Money) = \{Bank, Savings, Account, Cost\}$$

5.5.4 Morpheme-based LM

N-gram of word components, not complete words, e.g.

$$P(Moving\ forward) = P(mov\ \ ing\ \ for\ \ ward)$$

5.5.5 Grammar-based LM

Write grammar of possible sentences. Use grammar to verify fluency. Models long context, but requires effort to write.

5.5.6 Special problems

- Spontaneous speech (Uhm, yeah, you know…)
  - Narrow the context of n-grams
  - Solutions: Ignore, Skip spontaneous effects in history, or increase context width (4-grams instead of 3-grams)
- Different Languages
  - Languages differ in potential to create new words, or differ in notion of words
  - Solutions: Use morpheme-based language models, decompose compound nouns, use syllable based LMs

6 Phrase-based MT

Probability, that phrase $\overline{f}_1$ is translated to phrase $\overline{e}_1$:

$$p\left(\overline{f}_1^I \middle| \overline{e}_1^I\right) = \prod_{i=1}^{I} \phi\left(\overline{f}_i \middle| \overline{e}_i\right) d(start_i - end_{i-1} - 1)$$

All segmentations are modelled with equal probability. $\phi\left(\overline{f}_i \middle| \overline{e}_i\right)$ describes the phrase translation probability from english to foreign (direction was inverted in noisy channel).

6.1.1 Advantages

- Captures $n \times m$ alignments
- Encapsulates context
- Allows local reordering without extra modelling
- Compensates segmentation errors (e.g. chinese text which is hard to segment into words)
- Much simpler (no fertility, no word deletion/additions)

6.2 Phrase extraction

- Generate word alignments with IBMs, HMMs, …
- Calculate Viterbi path in both directions
  - Viterbi has only ≥ 1 to 1 alignments, no 1 to ≥ 1 alignments
  - Thus, train in both directions, combine alignments
- Combine paths from both directions
  - Use intersection: High precision, but low recall
  - Use union: Lower precision, but higher recall
  - Heuristics, e.g. GROW
    - GROW:
      INIT-Step: Start with intersection as current selection
      GROW-Step: Add neighbors of current selection which belong to union and which are in yet uncovered rows/columns.
      FINAL-Step: add points in union which row (OR/AND) column is empty

Extract only phrase pairs, which are consistent with the generated alignment:

$$\left(\overline{e}, \overline{f}\right) \; consistent \; with \; A \Leftrightarrow$$

$$\forall e_i \in \overline{e}: \left(e_i, f_j\right) \in A \Rightarrow f_j \in \overline{f}$$

$$AND \; \forall f_j \in \overline{f} : \left(e_i, f_j\right) \in A \Rightarrow e_i \in \overline{e}$$

$$AND \; \exists e_i \in \overline{e}, f_j \in \overline{f} : (e_i, f_i) \in A$$

i.e. all aligned English words of foreign words are included, all aligned foreign words for English are included, and at least one foreign word is aligned to one English word. Note that gaps are allowed.

Unaligned words can be attached to all surrounding phrases.

6.3 Scoring

- Based on corpus statistics:
  - Compute $p(e|f)$ and $p(f|e)$ for phrases e, f
    $$p(e|f) = \frac{count(e,f)}{\sum_{f_i} count(e, f_i)}$$
- Lexical Weights:
  $$p(e|f,a) = \prod_{i=1}^{length(e)} \frac{1}{|\{j|(i,j) \in a\}|} \sum_{(i,j)\in a} w(e_i|f_i)$$
  - Fallback when a phrase does not occur in corpus (then p would always be 1). For every english word in the phrase, take the average word probability in the translations of that word and multiply them together.
  - Book: 5.3.3

6.4 Weighting models

Combine Language Model and Translation Model. Find

$$\hat{e} = argmax_e \; p\left(e_1^J \middle| f_1^I\right)$$

$$= argmax_e \; \frac{p\left(f_1^I \middle| e_1^J\right) \cdot p\left(e_1^J\right)}{p\left(f_1^I\right)}$$

$$= argmax_e \; p\left(f_1^I \middle| e_1^J\right) \cdot p\left(e_1^J\right)$$

for decoding. Log-Linear Model. General formulation for weighting:

$$\hat{e} = argmax_e \; \frac{\exp\left(\sum_{i=1}^{n} \lambda_i h_i(x)\right)}{Z}$$

Variate λ_i to weight individual models. Possible models:

- Translation Model
- Language Model
  - $h_2 = (-\log p(w_0| < s >) - \log p(w_1|w_0) - \log p(w_2|w_1) - \cdots)$
- Distortion Model
  - $h_3 = c(start_i - end_{i-1} - 1)^2$
  - Relative distortions
- Word-count model
  - $h_4 = \#words$
  - Prefer short translations
- Phrase-count model
  - $h_5 = \#phrases$
  - Prefer longer phrases
  - Prefer less phrases

6.5 Decoding

Decoding: Find best translation.

- Model errors: Translation with best score is not the best translation
- Search errors: Find suboptimal translation according to model

6.5.1 Decoder: Two Level Approach

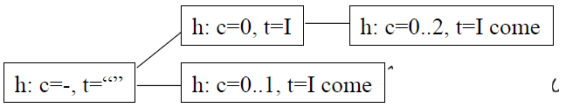
- Build translation lattice

  - Run left-right over test sentence, search matching phrases, insert edges for translations
- First best search on lattice
  - N-gram language model on lattice, left-right. Trace back best hypothesis.
  - Builds tree structure that expands previous translation leafs.

Search space grows exponentially with input sentence length.

6.5.2 Complexity reduction

- Limit possible reorderings (only small jumps)
- Hypothesis recombination
  - Matching hypothesis reached with varying paths
  - Expand only the one with better score
  - Only when no future information can switch their current ranking, i.e. $Cost(H1) < Cost(H2) \Rightarrow Cost(H1 + E) < Cost(H2 + E)$ for all expansions E.



- Hypothesis pruning (expand only the best partials)
  - Keep partials in stacks, limit stack sizes (drop if too large)
  - Histogram Pruning: Direct relation to decoding speed
  - Threshold Pruning: Keep if

$$cost(h) < cost(h_{best}) + const$$

6.6 Parameter estimation

Goal: Estimate weighting parameters λ_i that yield best performance on validation data ("Tuning Set". use BLEU as metric), but too many parameters/values.

6.6.1 N-best list

TODO F05-14_96f, book chapter 9.1.3

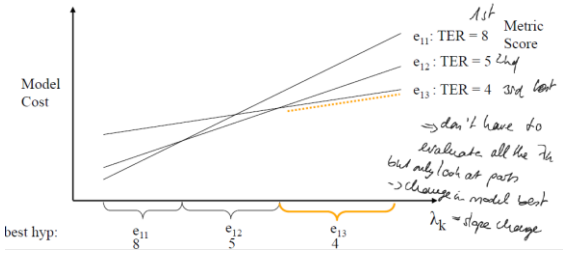
6.6.2 Powell Search

Reduce number of points considered. Consider one parameter at once, find maximum, fix and find maximum for next parameter. At the end, repeat from first parameter again until convergence.

- No guarantee for global optimum found
- Cannot go diagonally (one at a time)
- Start each iteration at random starting point to have varying end points

Optimize number of points evaluated per dimension:

- Total cost for one hypothesis considering one changing weight is a linear function
- Set λ_k s.t. metric-best hypothesis is also model-best
- Which hypothesis is model-best can only change in intersection points
- Throw away intersection points which are not on bottom of graph



7 Advanced Systems SMT

7.1 N-Gram based MT

TODO?

7.2 Hierarchical Phrase Based

Phrase-based can capture local reorderings, but not over long range.

Idea: Use phrases of phrases that learn reorderings of phrases. Learns structure in sentences.

X gives Y to Z ➔ X gibt Z Y.

Hierarchical rules learnt by synchronous context free grammar.

Grammar rule extraction:

1. Start with word alignment
2. Find conventional phrases
3. Find phrases that contain phrases and replace subphrases with nonterminal symbols
   e.g. $X\ gives\ Y\ to\ Z$

Problems: Too many extracted rules, different rules produce same translations.

Solution: Heuristics to filter extracted rules, i.e. don't grow phrases to include unaligned words on phrase boundaries, max length of 10 words, max 2 nonterminals, …

Decoding: Parse input sentence by build hierarchical tree, e.g. CYK algorithm (TODO?).

Add Language Model either by rescoring translations at the end or intersecting CFG with finite state machine of language model, or as compromise cube pruning.

7.3 Syntax based MT

TODO?

Neural MT

8 Word Representation

- Character Model with Word representation
  - Each word fed into word model, each word model fed into language model

- Character Model with Char sequence representation
  - Each sequence of k chars fed into char model, each char model fed into language model
- Direct character model
  - Each character fed into word model
- Character-based encoding – Bag of letter representation
  - Problem: flow and wolf have the same represent
  - Represent word by set of character n-grams
  - House = ho ou se
- Byte Pair Encoding (BPE) ?

9 Neural Machine Translat.

9.1 Encoder-Decoder-Model

Sequence-to-Seqence task, i.e. input and output are sequences of potentially different lengths.

Solve via Encoder-Decoder. Encoder models source sentence, decoder generates target sentence word by word.

- Encoder: Typical RNN. Uses input words as input for every step.
- Decoder: Conditional RNN, conditioned on memory vector (source sentence representation). *Uses output of previous step as input for next one.*

Typically using LSTMs for both, initialize hidden state of decoder with last encoder hidden state.

9.2 Finding most probable word

Given output probabilities by the decoder.

- Greedy Search: Select best target word. But might not find most probable sentence (best overall solution).
- Beam Search: Keep best n hypothesis. Only small beam needed.
  - Calculate output probabilities in decoder step
  - Select best n translations
  - Extend all hypothesis in beam
  - Prune hypothesis not in beam

  In NMT, large beam can decrease quality. In SMT, it just slows the process.

9.3 Other approaches

- Vector Space Encoder
- Recurrent-baesd Encoder
- Convolutional Encoder

10 Attentional NMT

Problem: Gradient flows indirectly, entire source sentence cannot be crammed into vector.

Attention module gives decoder context of current state.

10.1 Calculating attention

Measure the relevance between source and target states.

Decoder at timestep j has query q_j (decoder state) and key k_i (encoder state). For every query-key-pair (q_j, k_i), compute the weight $\alpha_{ij} = softmax\left(att\_func(q_j, k_i)\right)$, and get context vector $c_j = \sum_i \alpha_{ij} v_i$ for values v_i (encoder hidden-states).

10.2 Integrating context vector into decoder

For every step in the decoder, query the attention module with current query. Attention module computes computes context, context is used to compute output.

11 Domain adaption

?

12 Transformers

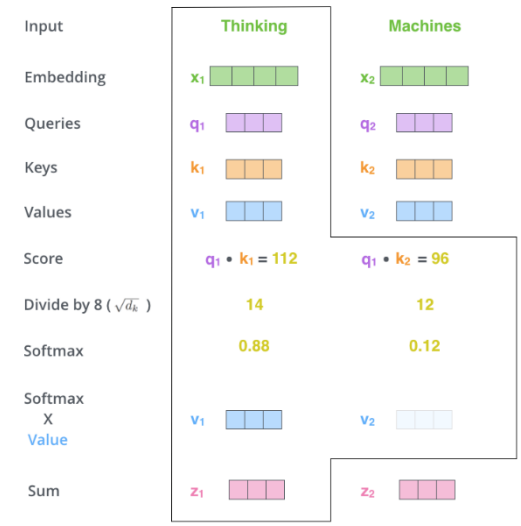
See http://jalammar.github.io/illustrated-transformer

12.1 Dropout, Normalization

- Dropout
  - Problem: We give NNs many neurons to increase capacity, but that causes overfitting
  - Zero-out random layer-inputs during training
- Batch Normalization
  - Problem: Updating a lower layer during training changes input distribution for next layer
  - Normalize for mean/variance between layers
  - Gradients less dependent on scale of parameters
  - $x^1 = \frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}}, x^N = \gamma \cdot x + \beta$
- Layer Normalization
  - Not as effective as BN, but more efficient.
  - Compute μ and σ over feature dimension

12.2 Transformers

12.2.1 Computing Self-Attention in the Encoder



Multiple input words can also be stacked on top of one another to transform those calculations into matrix computations, which is more convenient for CUDA.

12.2.2 Multi-Headed Self-Attention

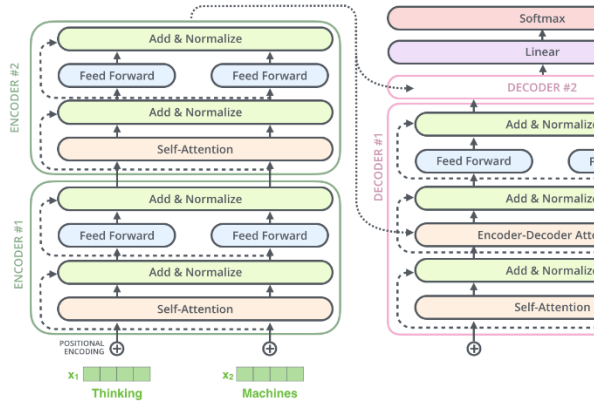
Multiple heads i with distinct W_i^Q, W_i^K, W_i^V weights, each computing distinct z_i's which are concatenated to a big vector, multiplied with a big weight W^O trained jointly with the model to get the output z with the typical size.

12.2.3 Positional encodings

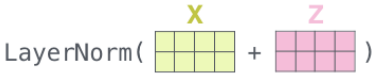
Prior to inputting embeddings to first encoder, add positional encoding vectors that give a sense of order and distance between words. Left half of positional vector is modulated by sine, right half by cosine. Halfs are commonly interweaved instead of concatenated.

12.2.4 Residual Connections

Residual Connection between every step in every encoder and decoder.



Add&Normalize Step:



12.2.5 Decoding process

- First decoding time step: Have every input word run through encoders, compute K and V. Run first input word through decoder, using K and V.
- Subsequent decoding time steps: Have every other word run through the decoder one at a time, using K and V.
    - Self-attention: Decoder may only self-attend past words, not future ones (masked by $-inf$)
    - Encoder-Decoder attention: Multi-headed, use queries from decoder layer below and keys/values from encoder.

13 Advanced Topics of NMT

13.1 Open Vocab and rare Words

Vocabulary changes over time (names, compounds, new word creations), but NMT uses fixed vocabulary, where modelling of rare words is hard.

Solution: Use fixed vocab for frequent n words, replace all other by UNK. Then

- Copy word over. Use attention to align UNK between source and target.
- Subword Segmentation. Segment into smaller word-pieces.
    - Linguistic-driven Subword Segmentation
      Use Stemmer/Lemmatizer. But Language dependent.
    - Byte pair Encoding
      Use n symbols to encode all words, find most frequent bi-gram, replace with new symbol. But not all splits are reasonable.

13.2 Monolingual Data

Parallel Data used to train translation model, monolingual data can be used to train language model. NMT only uses parallel data. Monolingual easy to gather in large quantities.

Integration of language model into RNN-based translation model:

- Parallel combination (combine outputs)
- Serial combination (Use LM out as input to decoder)

Training:

- Train independently, combine afterwards
- Pretrain language model, then train combined NMT
- Jointly train both models

Monolingual data can also be used to synthesize parallel data (translate target to source, then train on that).

13.3 Advanced Architectures

- Tying embeddings and weights
    - Tie source and target embedding weights
- Residual Connections
- Batch/Layer Normalization
    - Improves gradient flow, allows higher learn rates, reduce dependence on initialization
- Copy-Generator
    - Improves Copy-mechanism (UNK)
- Neural Coverage
    - Problem: Some words are translated multiple times, some not at all.
    - Soft Coverage: Store how much model as already attended to one word.
        - But some words don't need any attention or need several attendings
        - Use in combination with source representation to compute attention weights
        - Update weight after every target word production

13.4 Training Tricks

- Regularization against Overfitting
    - Early Stopping

- Data Augmentation
- L1/L2 Regularization (Punish large weights)
- Ensemble Models (Reduces variance of individual models)
- Dropout

14 Multilingual, Multitask, Multimodal

14.1 Multilinguality

- Human-based Interlingua
  Difficult to achieve
- Rule-based systems
  Expensive, n^2 systems
- SMT systems
  Many bilingual corpora, expensive, not scalable.

14.1.1 Pivot Approaches

- Direct Pivot: Output of MT1 is input of MT2
- Model Pivot: Combine models MT1 and MT2
- Data Pivot: Use MT1 to generate synthetic data to train MT2

14.2 Multi-task Learning

14.3 Multi-modal Learning