**Machine Translation: Key Concepts for NLP Course – summary of pdf 13** *Speech and Language Processing. Daniel Jurafsky & James H. Martin. Copyright © 2024. All*

*rights reserved. Draft of January 12, 2025.*

**What is Machine Translation?**

Machine Translation (MT) is the use of computers to translate from one language to another. Modern MT focuses on practical tasks rather than the full complexity of literary translation.

**Main Applications**

1. **Information Access**: Translating web content, recipes, instructions (Google Translate processes hundreds of billions of words daily)
2. **Computer-Aided Translation (CAT)**: Human translators post-edit MT output for localization
3. **Real-time Communication**: Simultaneous interpretation, image translation (OCR + MT)

**Language Challenges**

**Word Order Differences**

* **SVO languages**: English, German, French, Mandarin (Subject-Verb-Object)
* **SOV languages**: Japanese, Hindi (Subject-Object-Verb)
* **VSO languages**: Irish, Arabic (Verb-Subject-Object)

Example:

* English: "He wrote a letter to a friend"
* Japanese: "tomodachi ni tegami-o kaita" (friend to letter wrote)

**Lexical Divergences**

* **One-to-many mappings**: English "bass" → Spanish "lubina" (fish) or "bajo" (instrument)
* **Many-to-many mappings**: Complex conceptual overlaps between languages
* **Lexical gaps**: Words that don't translate directly (e.g., Japanese "oyakokō")

**Morphological Complexity**

Languages vary from:

* **Isolating**: Vietnamese, Cantonese (one morpheme per word)
* **Polysynthetic**: Siberian Yupik (many morphemes per word)

**Modern MT Architecture: Encoder-Decoder**

**Core Components**

1. **Encoder**: Processes source language input into contextual representations
2. **Decoder**: Generates target language output using cross-attention to encoder

**Key Innovation: Cross-Attention**

* Decoder attends to all encoder hidden states
* Allows flexible alignment between source and target words
* Handles complex reordering between languages

**Training Process**

* Uses parallel corpora (bitexts)
* Self-supervised learning with teacher forcing
* Maximizes probability: P(target|source)

**Tokenization**

Modern systems use **subword tokenization**:

* **BPE (Byte Pair Encoding)**: Merges frequent character pairs
* **WordPiece**: Chooses merges that increase language model probability
* **Unigram/SentencePiece**: Starts large, iteratively removes tokens
* Vocabulary size: typically 8K-32K tokens

**Decoding Algorithms**

**Beam Search**

* Maintains k best hypotheses at each step
* Explores k×V possibilities, keeps best k
* Uses length normalization to avoid bias toward shorter sequences
* Typical beam width: 5-10

**Minimum Bayes Risk (MBR)**

* Chooses translation most similar to other candidates
* Uses similarity metrics like chrF or BERTScore
* Often outperforms beam search

**Low-Resource Solutions**

**Backtranslation**

1. Train target→source MT with limited bitext
2. Translate monolingual target data to source
3. Add synthetic bitext to training data
4. Retrain source→target MT

**Multilingual Models**

* Single model handles multiple language pairs
* Uses language tokens to specify source/target
* Lower-resource languages benefit from similar high-resource languages

**Evaluation Methods**

**Human Evaluation (Gold Standard)**

* **Adequacy**: How well meaning is preserved
* **Fluency**: How natural the target text sounds
* Can use direct scoring or post-editing approaches

**Automatic Metrics**

**chrF (Character F-score)**

* Measures character n-gram overlap with human reference
* Combines precision and recall with F-score
* Robust across languages, correlates well with human judgment

**Embedding-based Methods**

* **BERTScore**: Uses BERT embeddings to match tokens
* **COMET**: Trained on human-labeled datasets
* Handle synonyms and paraphrases better than overlap metrics

**Ethical Considerations**

**Gender Bias**

* MT systems often default to male pronouns
* Reinforce occupational stereotypes
* Problem amplified beyond actual employment statistics

**Confidence and Safety**

* Need for confidence measures in critical applications (medical, legal)
* "Do no harm" principle for high-stakes translations

**Historical Context**

* 1940s: First proposed alongside early computers
* 1966: ALPAC report dramatically cut funding
* 1990s: Statistical MT revolution (IBM Models, phrase-based)
* 2015+: Neural encoder-decoder dominance
* Current: Transformer-based systems

**BEAM SEARCH and MBR**

**Where Beam Search and MBR Are Used**

**During Training: Teacher Forcing**

During training, the encoder-decoder model uses **teacher forcing** - it doesn't use beam search or MBR at all. The decoder is given the correct target tokens as input at each step and learns to predict the next token using cross-entropy loss.

**During Inference/Decoding: Post-Model Processing**

Beam search and MBR are **inference-time decoding algorithms** that operate **after** the encoder-decoder model has done its work:

1. **Encoder**: Processes source sentence → produces contextual representations
2. **Decoder**: At each timestep, outputs probability distributions over the vocabulary
3. **Decoding Algorithm** (Beam/MBR): Uses these probabilities to select the final output sequence

**Beam Search Process:**

Encoder → Decoder outputs probabilities → Beam Search selects best k sequences

* The decoder transformer blocks (with cross-attention) generate probability distributions
* Beam search then uses these probabilities to maintain k hypotheses and select the best path

**MBR Process:**

Encoder → Decoder → Sampling → MBR scoring → Final selection

* First, generate multiple candidate translations (using beam search, sampling, etc.)
* Then MBR evaluates all candidates using a utility function (like chrF, COMET)
* Finally, selects the candidate with highest expected utility

**The Core Problem They Solve**

The transformer decoder gives you **probability distributions**, but you need to convert those into **one actual sentence**. This is harder than it sounds!

**Why Not Just Pick the Highest Probability Words?**

**Greedy decoding problem:**

At step 1: Pick "good" (prob=0.6)

At step 2: Given "good", best next word is "morning" (prob=0.8)

Final: "good morning" (total prob = 0.6 × 0.8 = 0.48)

But what if:

At step 1: Pick "hello" (prob=0.5)

At step 2: Given "hello", best next word is "world" (prob=0.9)

Final: "hello world" (total prob = 0.5 × 0.9 = 0.45)

Greedy picked "good morning" but "hello world" might actually be better overall!

**What Beam Search Does (Smart Choosing)**

**Keeps multiple options alive:**

* At each step, maintain k best partial sequences
* Don't commit to one choice too early
* Find globally better sequences that greedy would miss

Step 1: Keep top 3 words: ["good", "hello", "hey"]

Step 2: Expand all 3, keep top 3 complete sequences

Result: Better global optimization

**What MBR Does (Even Smarter Choosing)**

**Says "probability ≠ quality":**

* Generate many candidate translations
* Score each candidate using **quality metrics** (not just probability)
* Pick the one with highest expected quality

Candidate 1: High probability, but awkward phrasing

Candidate 2: Lower probability, but more natural/accurate

MBR picks: Candidate 2 (better quality despite lower probability)

**The Key Insight**

As noted in the document: "model estimates and translation quality only vaguely correlate" [Beam search - Wikipedia](https://en.wikipedia.org/wiki/Beam_search)

The neural network might assign highest probability to translations that humans would rate as worse quality!

**Your "Choosing" Metaphor is Perfect**

Think of it as:

* **Neural network**: Generates many possible answers with confidence scores
* **Beam search**: Smart way to explore the space of possibilities
* **MBR**: Even smarter way to pick based on actual quality, not just confidence

They're the "decision-making" layer on top of the "thinking" layer.

Retry

AC

so how do they work? and without them the quality is bad?

Edit

Great question! Let me break down exactly how they work and what happens without them.

**How They Work**

**Beam Search: Step-by-Step**

Let's say we're translating "Hello" to Spanish with beam width = 2:

Step 1: Generate first word

Decoder outputs: P("Hola")=0.6, P("Buenos")=0.3, P("Saludos")=0.1

Keep top 2: ["Hola", "Buenos"]

Step 2: For each kept word, generate next word

From "Hola": P("amigo")=0.8, P("mundo")=0.7, P(".")=0.9

From "Buenos": P("días")=0.9, P("tardes")=0.4

Calculate total probabilities:

- "Hola ." = 0.6 × 0.9 = 0.54

- "Hola amigo" = 0.6 × 0.8 = 0.48

- "Buenos días" = 0.3 × 0.9 = 0.27

- "Hola mundo" = 0.6 × 0.7 = 0.42

Keep top 2: ["Hola .", "Hola amigo"]

Continue until <END> token...

**MBR: Step-by-Step**

Step 1: Generate candidate set (using beam search, sampling, etc.)

Candidates: ["Hola amigo", "Hola mundo", "Buenos días", "Saludos"]

Step 2: Score each candidate against all others using utility function

For "Hola amigo":

- vs "Hola mundo": chrF = 0.75

- vs "Buenos días": chrF = 0.30

- vs "Saludos": chrF = 0.20

Average utility = (0.75 + 0.30 + 0.20) / 3 = 0.42

Repeat for all candidates...

Step 3: Pick candidate with highest average utility

**Without Them, Quality IS Bad**

**Without Beam Search (Just Greedy):**

**Real example problems:**

* **Repetition:** "The the the cat sat sat"
* **Local optima:** Picks words that sound good locally but create bad global sentences
* **No backtracking:** Once you pick "good" you can't go back to try "hello" even if "hello world" would be better overall

From the document: "A problem with greedy decoding is that what looks high probability at word t might turn out to have been the wrong choice once we get to word t+1" [Beam search - Wikipedia](https://en.wikipedia.org/wiki/Beam_search)

**Without MBR (Just Beam Search):**

**Problems with probability-based selection:**

* "model estimates and translation quality only vaguely correlate" [Beam search - Wikipedia](https://en.wikipedia.org/wiki/Beam_search)
* High-probability translations might be:
  + Grammatically correct but unnatural
  + Missing important meaning
  + Too literal/not fluent

**Real example:**

High probability: "I have hunger" (literal)

Lower probability: "I am hungry" (natural)

**The Quality Impact**

**Greedy vs Beam Search:** Typically 2-5 BLEU points improvement **Beam Search vs MBR:** Can be 5-10+ points improvement with good utility functions

**Concrete Example:**

Source: "Il fait beau aujourd'hui"

Greedy: "It makes beautiful today" (grammatical but wrong)

Beam: "It is beautiful today" (better, but still awkward)

MBR: "It's a beautiful day today" (natural and accurate)