# Semantics

## Meaning representation

represent questions, and knowledge drawn from text. Anything that answer questions, determine truths and draw inferences.

Examples:

* Categories/entities
* events
* time
* aspect
* beliefs, desires, intentions

## Semantic roles

Types:

* agent: initiator or doer in the event
* patient: affected entity in the event; undergoes the action
* theme: object in the event undergoing a change of state or location, or of which location is predicated
* experiencer: feels or perceive the event
* stimulus: the thing that is felt or perceived
* goal
* recipient (may or may not be distinguished from goal)
* benefactive (may be grouped with recipient)
* source
* instrument
* location

Abstract Meaning Representation (AMR): Roles are given theory-neutral names. example:



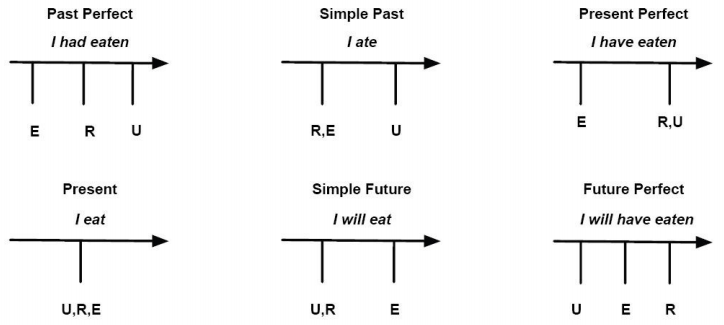
## Time/temporal representations

from verb tense, temporal expressions and sequence of presentation

utterance time: when the utterance occurs

reference time: the temporal point-of-view of the utterance

event time: when events described in the utterance occur



## aspect: verbs and event types

Statives: states or properties of objects at a particular point in time

Activities: events with no clear endpoint

Accomplishments: events with durations and endpoints that result in some change of state

Achievements: events that change state but have no particular duration (occur in an instant, e.g., received something)

## Beliefs, desires and intentions (BDI)

Very hard to represent internal speaker states like believing, knowing, wanting, assuming, imagining. Not well modeled by a simple DB lookup approach

Truth in the world vs. truth in some possible world

Augment representations with special modal operators that take logical formulae as arguments, e.g., believe, know

Mutual belief: Practical importance: modeling belief in dialogue

## Lexical relations

synonyms (same meaning), antonyms (opposite meaning), homonyms (the same spelling or pronunciation but different meanings and origins)

# Machine Translation MT (Intro + Statistical)

## Challenges

Orthographic Variations:

* Ambiguous spelling (Arabic)
* Ambiguous word boundaries (Chinese)

Lexical Ambiguity:

* a homonym in a language might correspond to different words in another language
* Phrase/word combination differences

Morphological Variations

* Affixation vs. Root+Pattern
* Tokenization

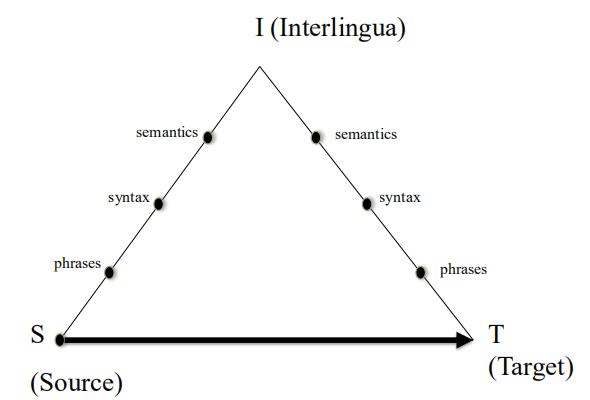
Translation Divergences: conflation

Language Differences – vocabulary

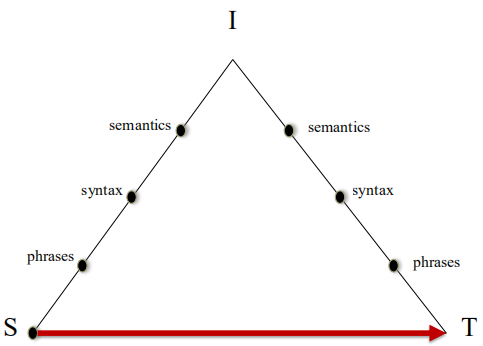
Language Differences – Syntax:

* word order: SVO, VSO, SOV
* word order in phrases
* word order in sentences
* Prepositions

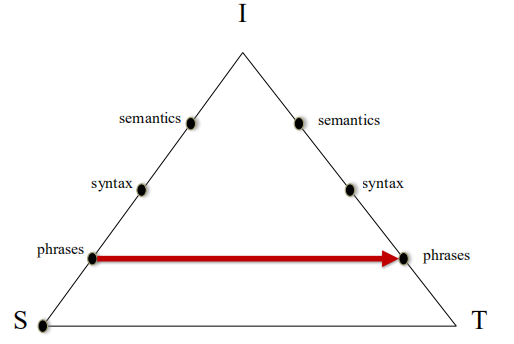
## MT Approaches and Pyramid



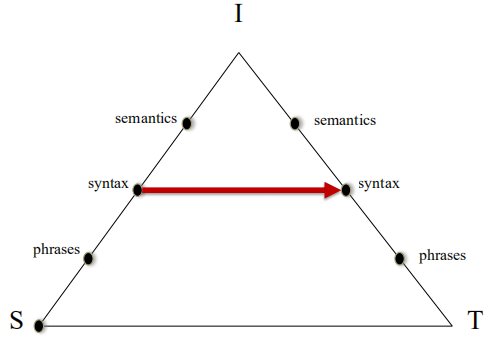
String-to-string translation

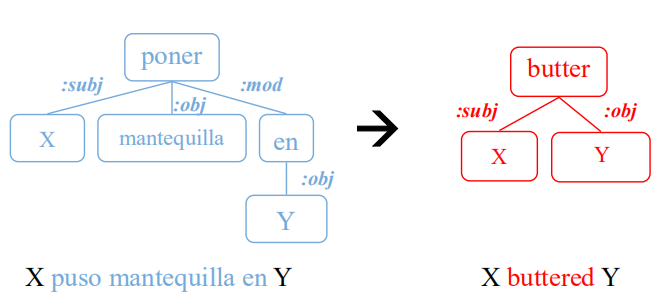


Phrase-based translation

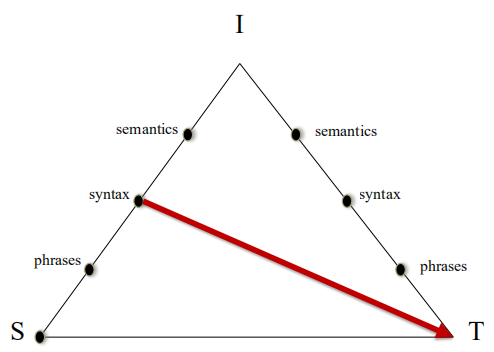


Tree-to-tree translation





Tree-to-string translation



## AMR characteristics

Rooted, labeled graphs

Abstract away from syntactic differences. Example:

He described her as a genius -> His description of her: genius

Use Propbank framesets. e.g., “bond investor”: “invest-01”

Heavily biased towards English

Graph details:

* Variables (or nodes) for entities, events, properties, states
* Leaf nodes are labeled with concepts. e.g., “(b/boy)” is an instance of the concept “boy”
* Relations link entities. e.g. “(d/die-01 :location(p/park))” means “there was a death in the park”
* Concepts:
  + English words (e.g., boy)
  + Propbank framesets (e.g., want-01)
  + special keywords (entity-types, quantities or conjunctions)

AMR relations:

* frame arguments: arg0, arg1, … (propbank)
* general semantic relations:

:Accompanier, :age, :beneficiary, :cause, :compared-to, :employed-by,

:concession, :condition, :consistof, :degree, :destination, :direction, :domain,

:duration, :example, :extent, :frequency, :instrument, :li, :location, :manner,

:medium, :mod, :mode, :name, :part, :path, :polarity, :poss, :purpose, :source,

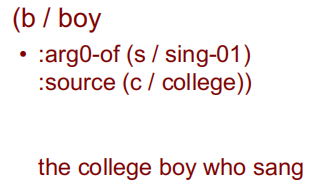
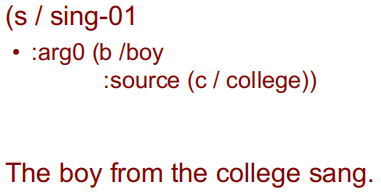
:subevent, :subset, :time, :topic, :value

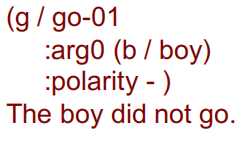
* relations for quantity: :quant, :unit, :scale
* relations for date entity: :day, :month, :year, :weekday, :time, :timezone, :quarter,

:season, :decade, :century, :era, :calendar, :dayperiod

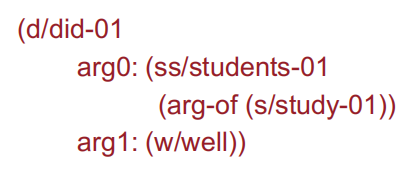
* relation for lists: :op1, :op2, …
* plus inverses: :arg0-of, :location-of
* negation: :polarity –

Examples:



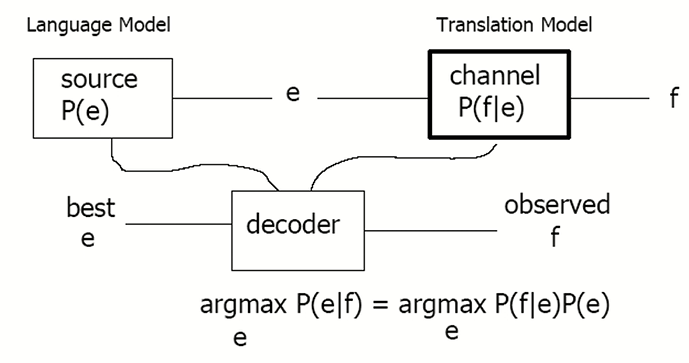


Students who studied did well.



## Statistical MT word alignment

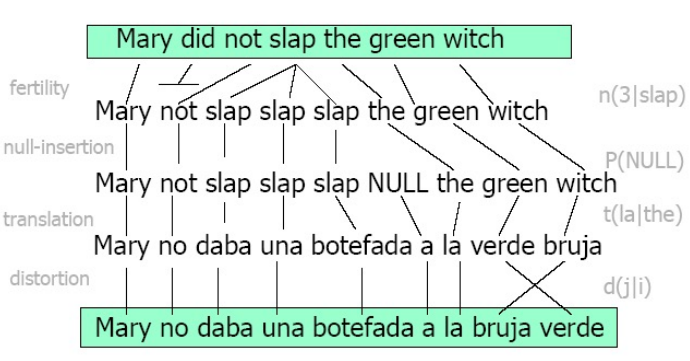
What words must be placed together in the target/translation?





automatic word alignment: GIZA++ is a toolkit used to train word alignments via expectation-maximization (EM) algorithm with various constraints to bootstrap alignments. Constraints include names/proper nouns (untranslated) and some words that remain unchanged in both languages.

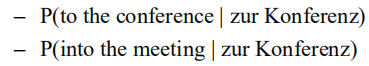
IBM model (word-based or string-to-string?):



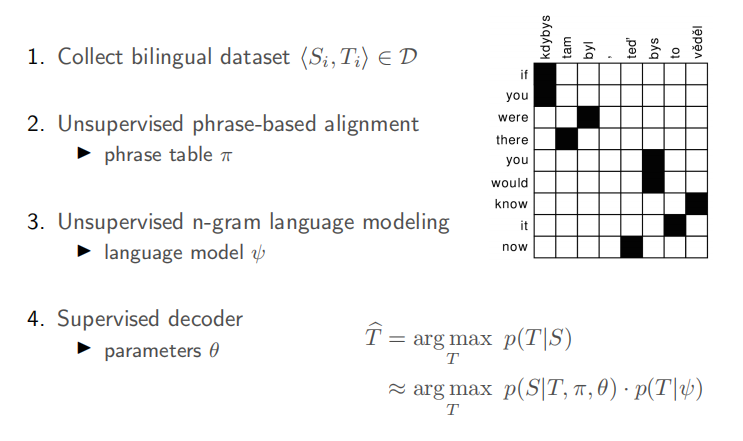
5 separate models trained via EM: word translation, local alignment, fertilities, class-based alignment, re-ordering

## Phrase-based Statistical MT

(Assuming target language is English) Foreign input segmented into phrases (phrase is any sequence of words). Each phrase is probabilistically translated into English:



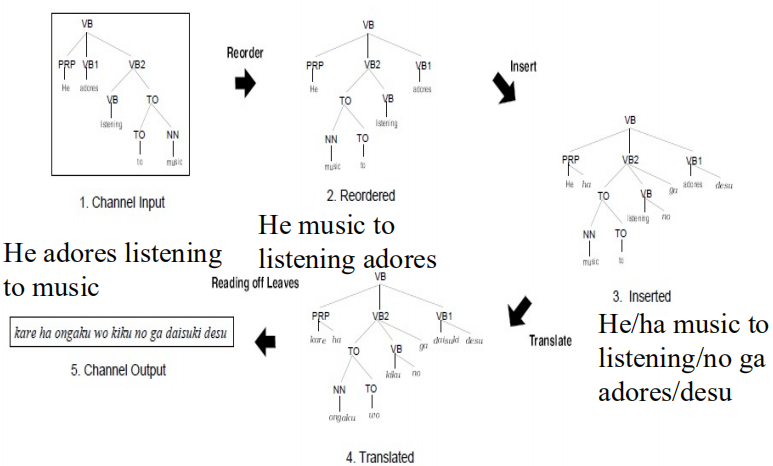
Phrases are probabilistically re-ordered. SOTA before neural MT.



Advantages of phrase-based SMT:

* Many-to-many mappings can handle non-compositional phrases (a non-compositional phrase is a phrase where the meaning cannot be inferred from individual words of the phrase. e.g., real estate).
* local context is very useful for disambiguating
* the more data, the longer the learned phrases
* might be less computationally expensive than neural models?

## String-to-tree translation



synchronous grammars: generate parse trees in parallel in two languages using different rules.

## MT practical considerations

resource availability:

* parsers and generators, input/output compatibility
* translation lexicons: word-based vs transfer/interlingua
* parallel corpora: domain of interest; bigger is better

time availability:

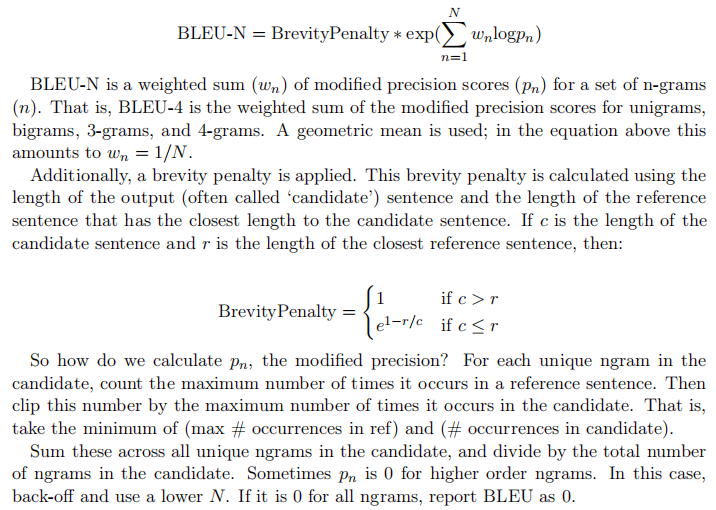
* statistical training, resource building

## MT Evaluation

human-based (golden but slow) vs machine-based (dumb/not well defined)

Human-based: adequacy criteria (check how much meaning has been conveyed, any missing words/phrases?), fluency criteria (grammar, clear, terminology, sentence structure)

BLEU metric: (might remove exp and log, and replace sum with product below, if the sequence is not long. exp and log are for numerical stability)



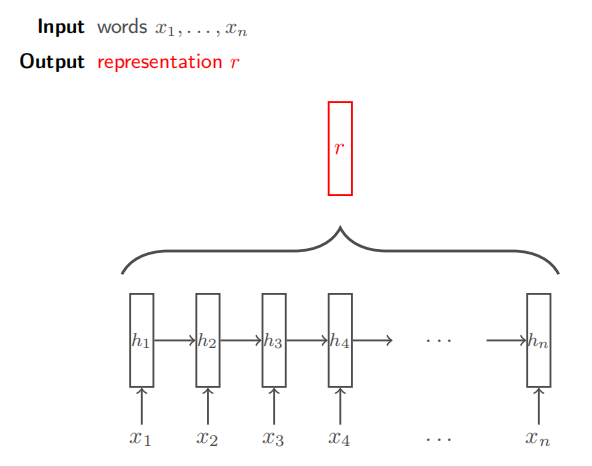
* quick, inexpensive and language independent
* Correlates highly with human evaluation
* many SOTA papers use it, very popular, easy to compare performance with them
* modified n-gram precision (instead of taking directly the count of an n-gram in candidate as the numerator, take min(the count of that n-gram in candidate, the maximum count of that n-gram in any reference))
* why modified precision instead of original precision: penalize repetition/duplicate words in candidate
* Why include brevity penalty: penalize shorter sentences. shorter sentences are more likely to produce high precision (small denominator)
* Why multiple references important: any sentence can have multiple good translations.
* Problem: penalizes synonyms, paraphrasing, or inflectional variations/morphology. no explicit model of semantics. This means it cannot give high scores to out-puts that are semantically correct but don’t word-for-word match something in a reference sentence. It tends to favor fluent outputs over correct ones, as a mixture of reference sentences will score highly even if this mixture introduces unexpected meanings. tends to over-estimate performance by prioritizing fluency. As you collect more and more reference sentences, BLEU may begin overestimating the

quality of your candidates. Allows changes in critical words that can completely change the meaning of a sentence. BLEU treats all words equally and does not prioritize content or topic words. The brevity penalty may not appropriately select a good length; it just restricts the model to choosing outputs of the same length as the references.

# Neural MT

supervised encoder-decoder architecture

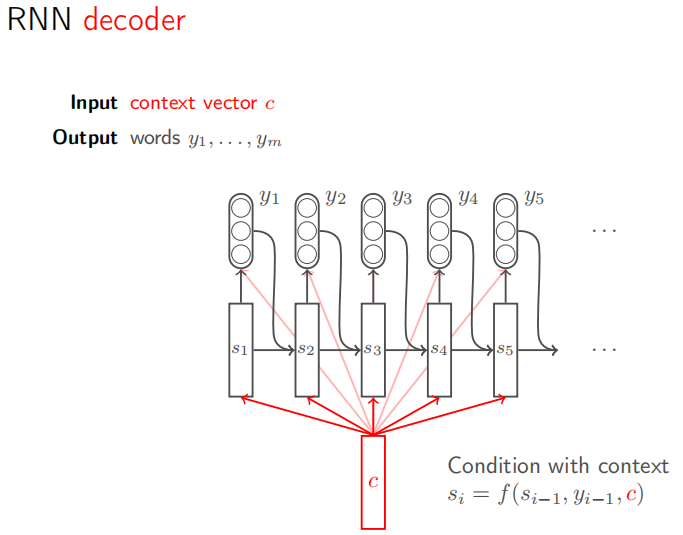
rnn encoder



basic rnn decoder (similar to rnn language model, but input is a context vector c instead of a sequence of input words)



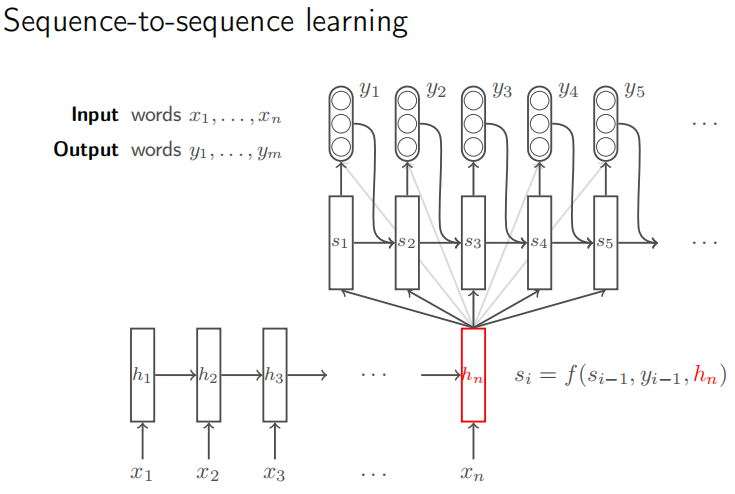
rnn decoder with context condition

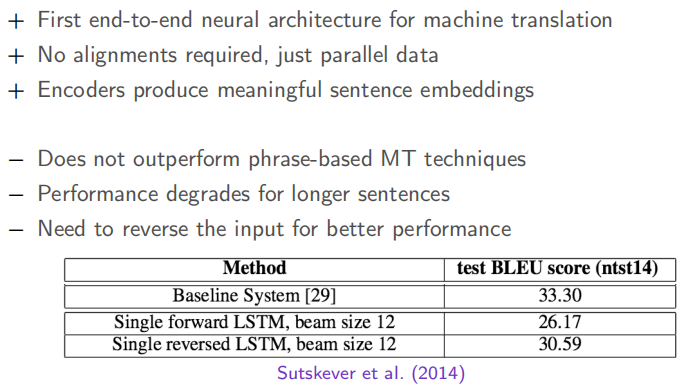


## seq2seq

combine a sequence encoder for the source lang with a seq. decoder for the target lang.

* Encode source language tokens until <EOS> obtained
* Use final encoder hidden state as context vector
* Decode target language tokens until <EOS> obtained



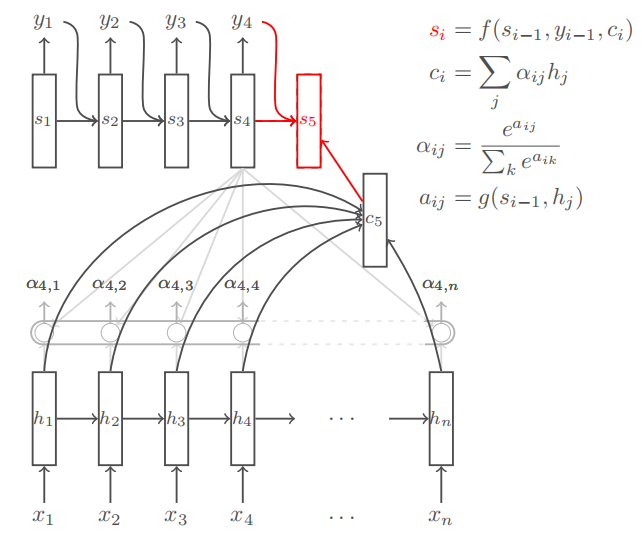


## seq2seq with attention

Fixed context vector is a bottleneck for performance in encoder-decoder architectures

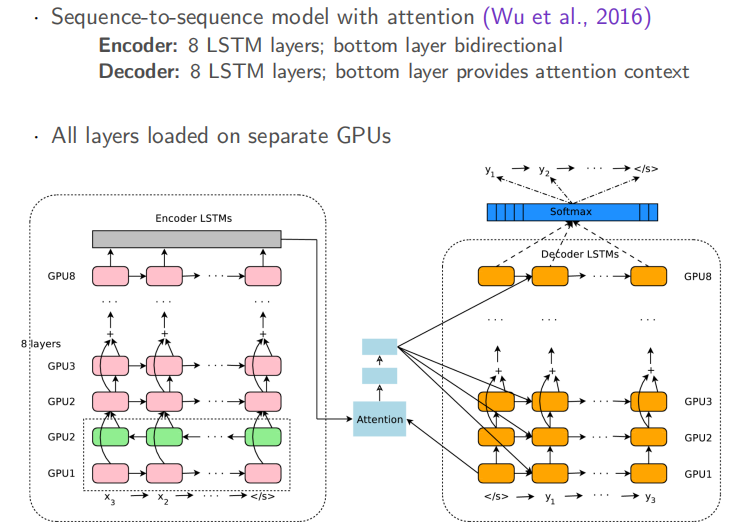
Attention:

* a dynamic context vector that changes with each decoder timestep
* Weighted average over all encoder hidden states
* Weights (“attention”) conditioned on current decoder hidden state
* Allows gradients to flow directly from decoding errors to relevant encoder hidden states, thus robust to vanishing gradients. Consistent performance as sentence length increases
* Performance competitive with phrase-based MT
* However, runtime for inference is O(mn) instead of O(m + n) without attention



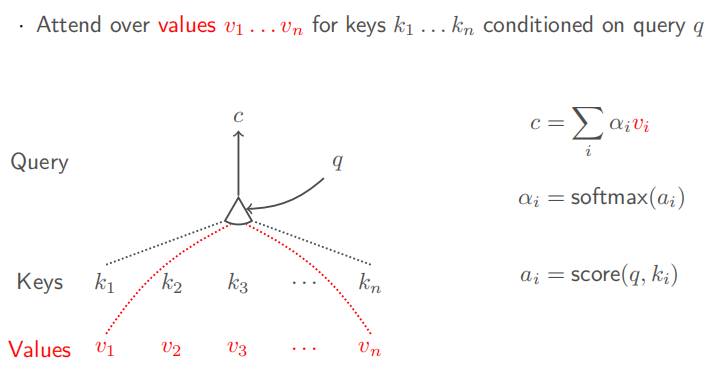


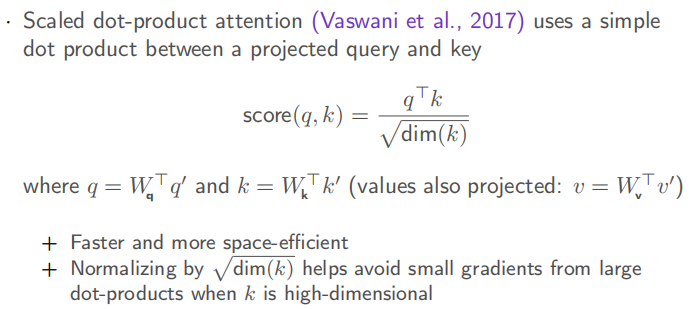
## Scaling up seq2seq

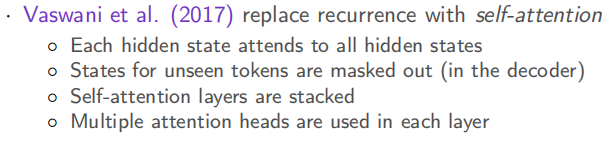
* GNMT:
* stacked LSTM with residual connections, more robust to vanishing gradients in deep networks
* Sub-word units: Infrequent words replaced with sub-words to reduce vocabulary
* sequence-level training: originally models are trained at the word level by maximizing word likelihood estimation, but evaluation metrics like BLEU and METEOR are sequence-level and typically non-differentiable. model parameters can also be optimized for any non-differentiable reward using reinforcement learning (computing gradients with REINFORCE). this trick might improve BLEU slightly but not human judgments
* multilingual-MT: 

## Transformer

transformer attention





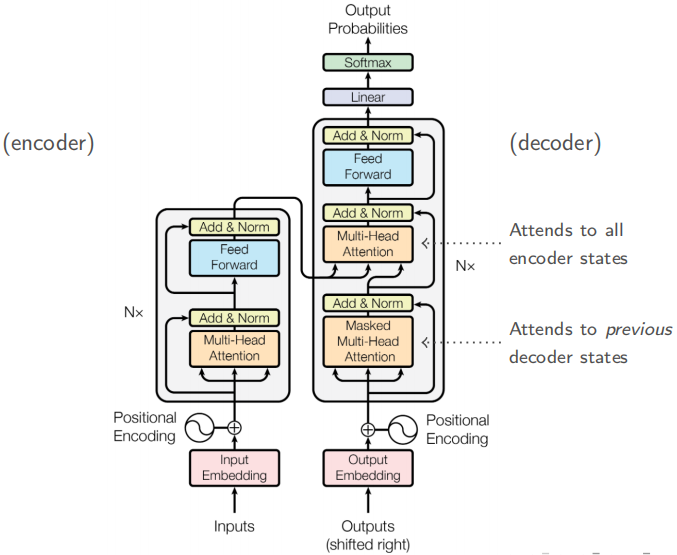


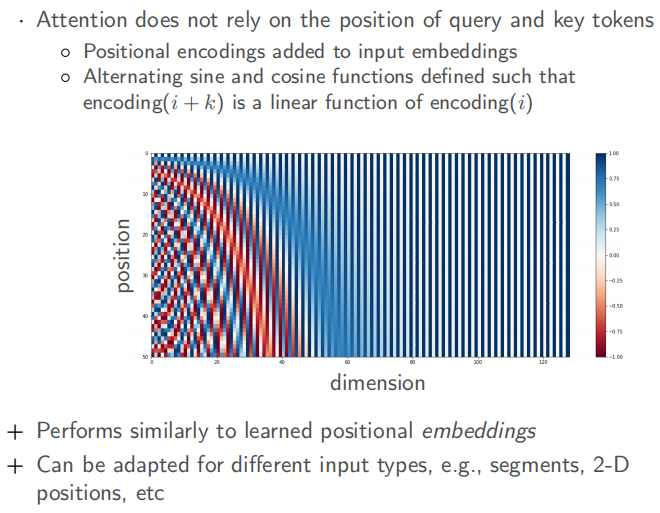
self-attention: q, k and v are projections of the same input

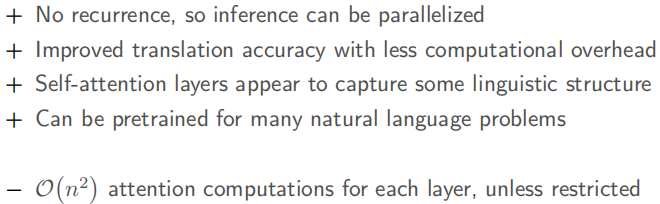


cross-attention: q and k based on encoder input, v based on decoder input

full transformer:







## Transformer pretraining

Autoregressive LM (Encoder only)

* pretrain by predicting every subsequent token
* fine-tune from state of the final token
* examples: GPT, GPT-2

Masked LM (Encoder only)

* pretrain by predicting masked or corrupted tokens
* fine-tune from state of a special classification token
* examples: BERT, RoBERa, AlBERT

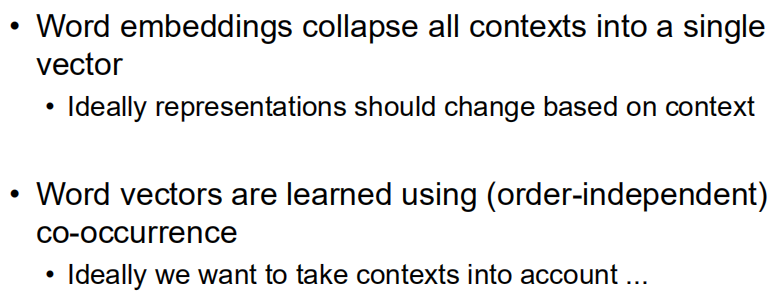
Encoder-decoder pretraining

* pretrain from noisy inputs (reorder permuted tokens, fill in masked spans, etc.)
* fine-tune through the decoder
* examples: BART, E5

# Advanced word embedding and pretrained LM

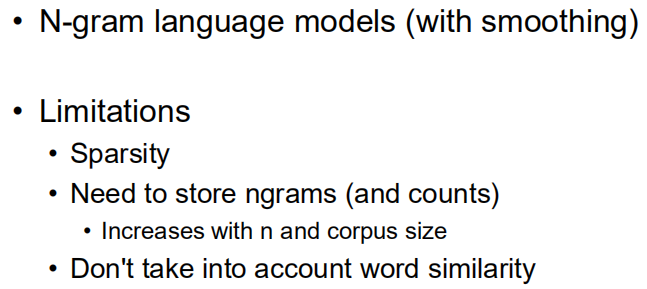
transfer learning: expensive label, new domain/task

problem with basic word embedding:

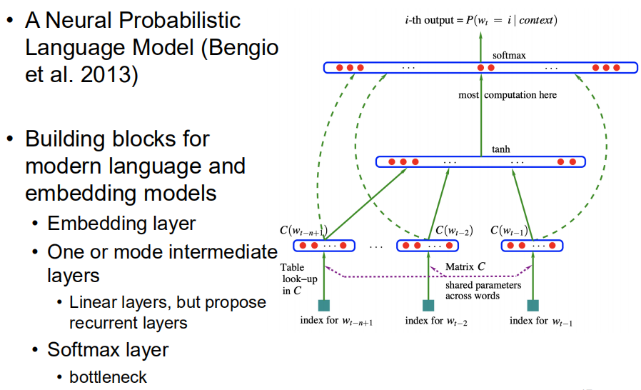


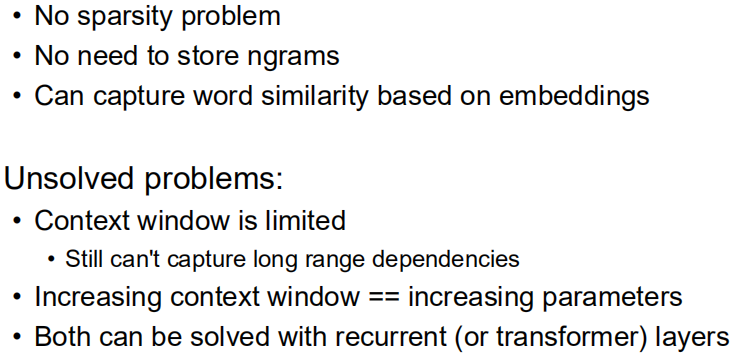
Improvement: instead of pretrain the first layer (basic lang embedding), pretrain the entire network/multiple layers which take into account things like context (basic lang embedding can’t represent/encode). A model would need to learn about syntax, semantics, and even some world knowledge in order to do well at this task

N-gram LM problems:

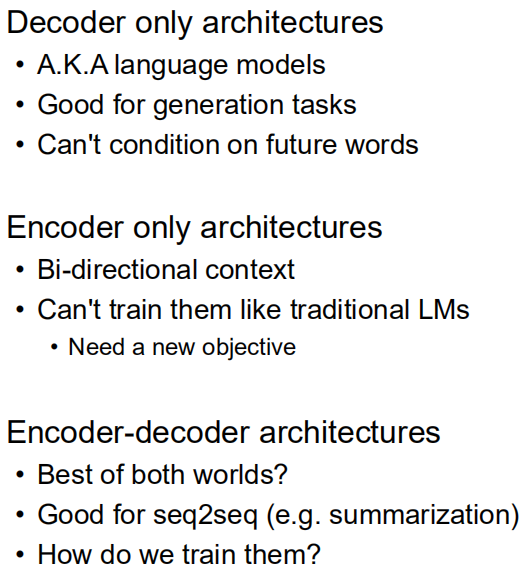


Early neural LM:



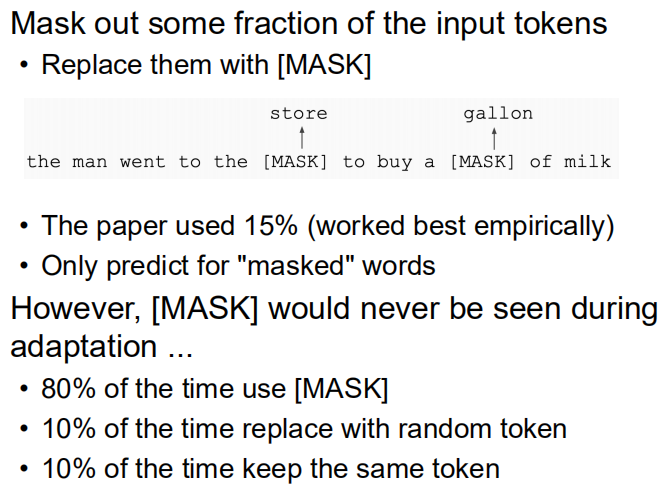


Architecture choices and considerations

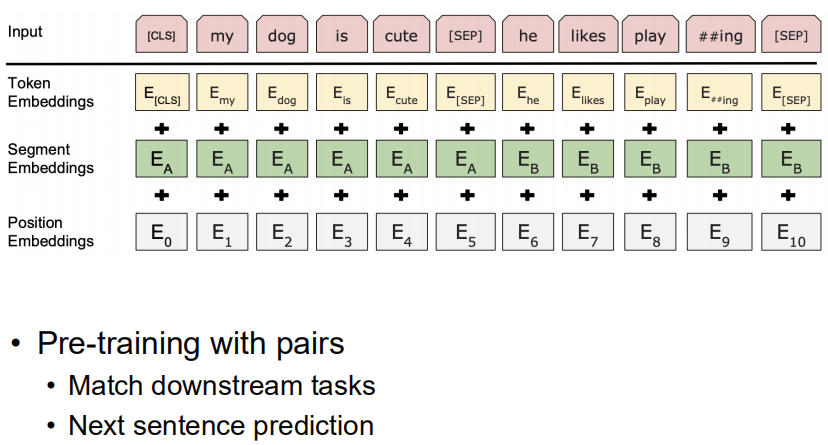


Decoder-only: GPT, GPT-2, at a minimum, only need to train an extra FC layer in transfer learning

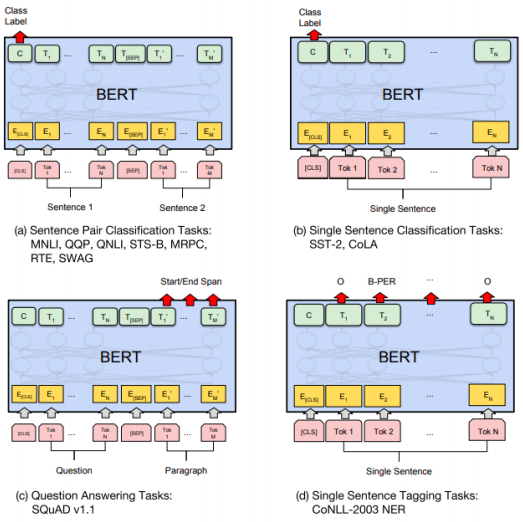
Encoder-only: BERT. pretrain tasks: masked language model & next sentence prediction



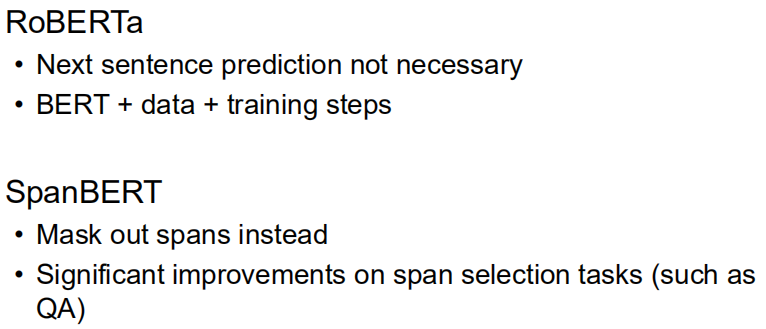
BERT input representation:



Fine-tuning/transfer learning with BERT: a class token is added to the front for many-to-one classification downstream tasks.



some BERT variants:



Research trend: larger datasets and bigger model. Still computationally expensive.

# Lexical Semantics and Word Sense Disambiguation

lexeme: an entry in a lexicon consisting of a pairing of a form with a single meaning representation

lexicon: a collection of lexemes

lemma or citation form: the grammatical form that is used to represent a lexeme. examples:



sense: discrete representation of one aspect of the meaning of a word

Relationships between word meanings:

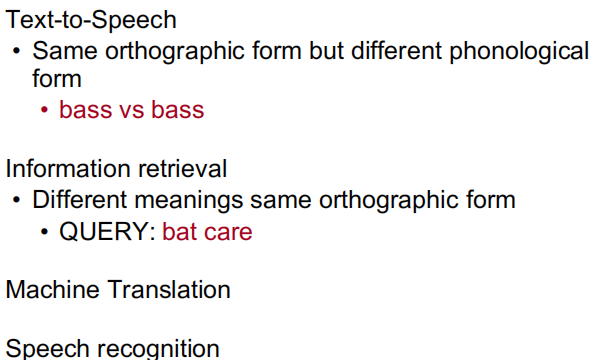
* homonymy
* polysemy
* synonymy
* antonymy
* hypernymy
* hyponymy
* meronomy

## Homonymy

Lexemes that share a form: phonological, orthographic or both, but have **unrelated**, distinct meanings

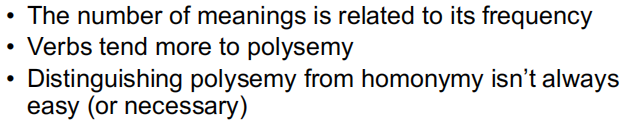
同形/同音多义，各含义之间无明显联系

cause problems in NLP:



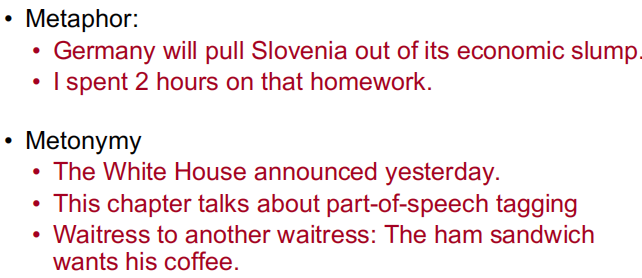
## Polysemy

a single lexeme with multiple **related** meanings



## Metaphor and Metonymy

specific types of polysemy. examples:



## Synonyms

words that have the same meaning in some or all contexts同义词

Two lexemes are synonyms if they can be successfully substituted for each other in all situations. If so, they have the same propositional meaning. Synonymy is a relation between senses rather than words.

## Antonyms

senses that are opposites with respect to one feature of their meaning反义词

## Hypernyms and Hyponyms

hyponym – subclass

hypernym – superclass

e.g., car is a hyponym of vehicle and vehicle is a hypernym of car

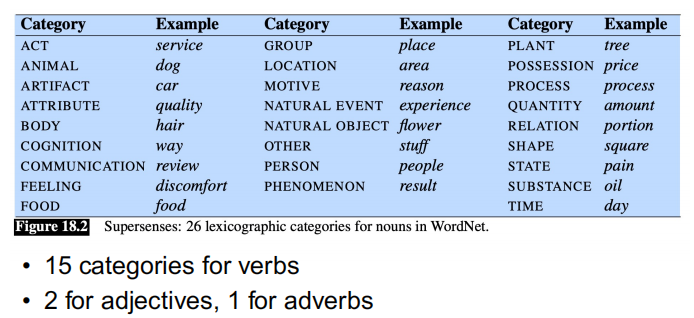
hyponymy is transitive: A hypo B and B hypo C entails A hypo C

## WordNet

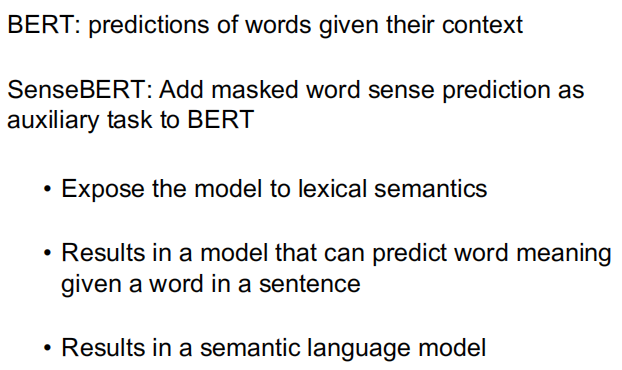
A hierarchically organized lexical database, On-line thesaurus + aspects of a dictionary

The set of near-synonyms for a WordNet sense is called a synset (synonym set); it’s their version of a sense or a concept

Supersenses:



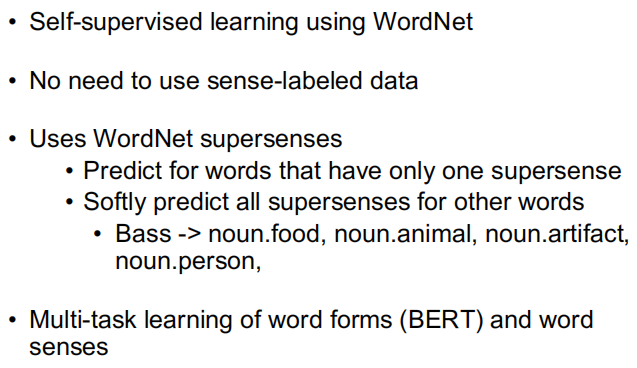
## Word Sense Disambiguation with SenseBERT



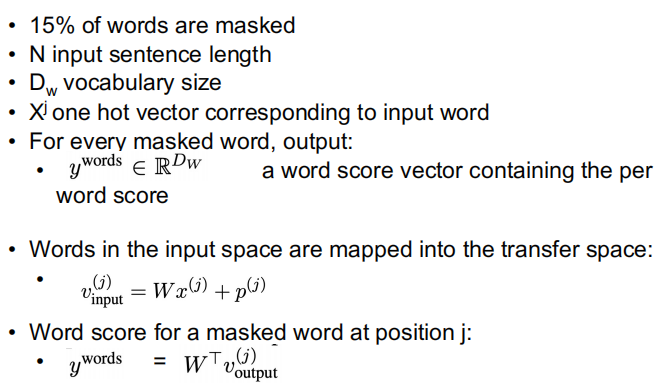
Why semantic LM useful:

* Selectional Restrictions: constraints on the types of arguments verbs take
* Predict word meaning in given contexts
* help in predicting semantics of words that can fill an argument.

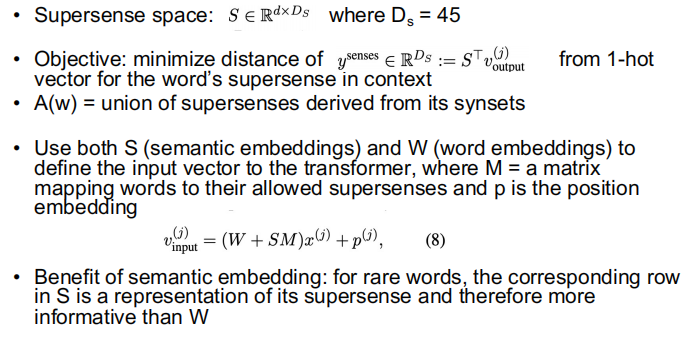
Overview of Pretrain of SenseBERT:

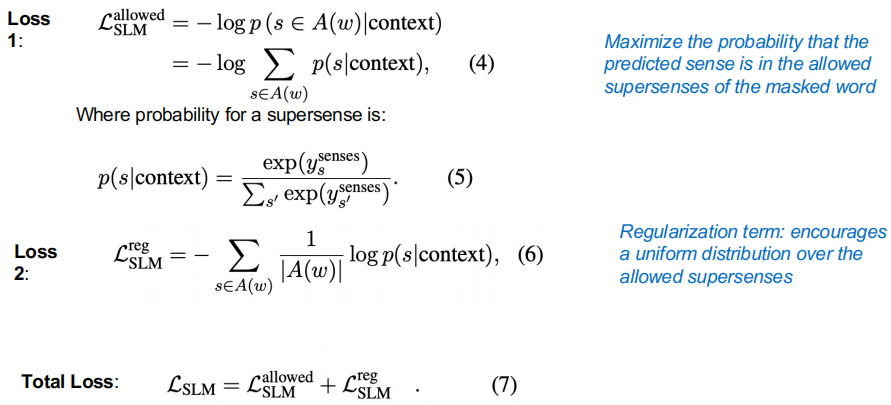


BERT pretrain part:



SenseBERT extras:

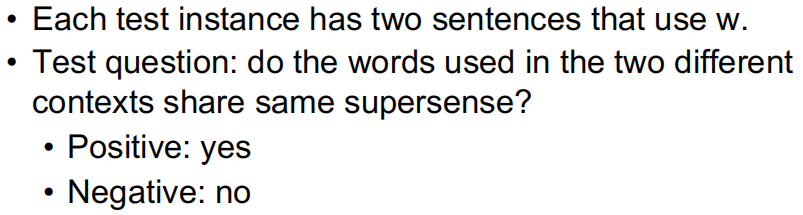






Evaluation:

* SemEval Supersense (SemEval-SS) disambiguation: given a word in a sentence, predict its supersense
* Word in Context (WIC)

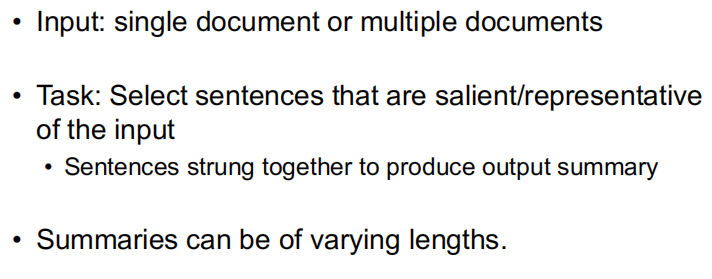


# Text Summarization (Extractive)

Extractive vs. Generative (abstractive)

* Choosing bits of the source vs. generating something new

Extractive summarization



Need to determine salient sentences/words. Methods:

* Unsupervised
  + term frequency \* inverse document frequency (TF\*IDF)
  + log likelihood ratio
* Supervised with large corpus of article/summary pairs

## TF\*IDF

Important terms are those that are frequent in this document but not frequent across all documents. Extract sentences that have high TF\*IDF words – these are words that indicate what is unique/striking about this article. Term Frequency (TF) is the ratio of number of times the word appears in a document compared to the total number of words in that document. Inverse document frequency:

N is the number of docs; n\_i is the number of documents with term/token i

## log likelihood based / Topic Signature Words

Uses the log ratio test to find words that are highly descriptive of the input.

Threshold to divide all words in the input into either descriptive or not

* H1: the probability of a word in the input is the same as in the background
* H2: the word has a different, higher probability, in the input than in the background

Binomial distribution used to compute the ratio of the two likelihoods

Probability p of w occurring k times in N Bernoulli trials

Log likelihood ratio

Where the counts with subscript I occur in the input corpus and those with subscript B occur in the background corpus.

## Graph-based methods

Lexrank, Textrank

Sentences vote for other sentences

* Frequently occurring words link many sentences

Input represented as highly connected graph

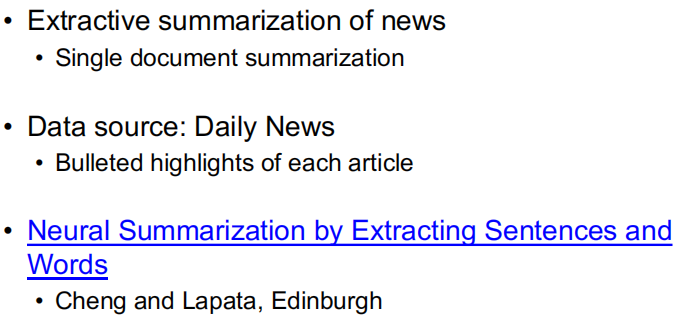
* Vertices represent sentences
* Edges between sentences weighted by similarity between two sentences
* Cosine similarity with TF\*IDF weights for words

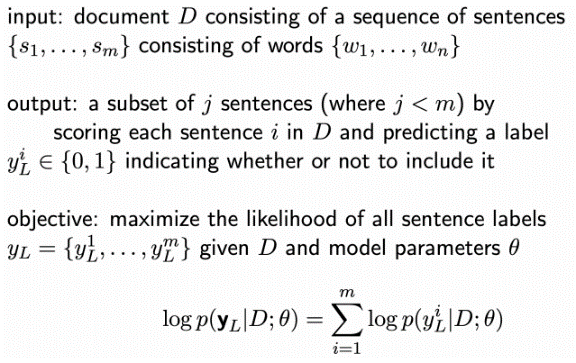
Vertex importance (centrality) computed using graph algorithms

* Edge weights normalized to form probability distribution -> Markov chain
* Compute probability of being in each vertex of graph at time t while making consecutive transitions from one vertex to next
* As more transitions made, probability of each vertex converges -> stationary distribution

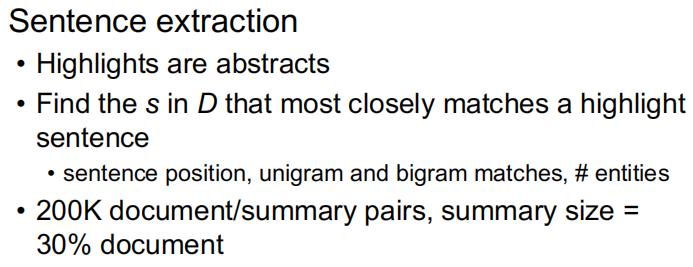
Vertices with higher probability = more important sentences

## Neural approach 1 – CNN + LSTM + Attention





training data:

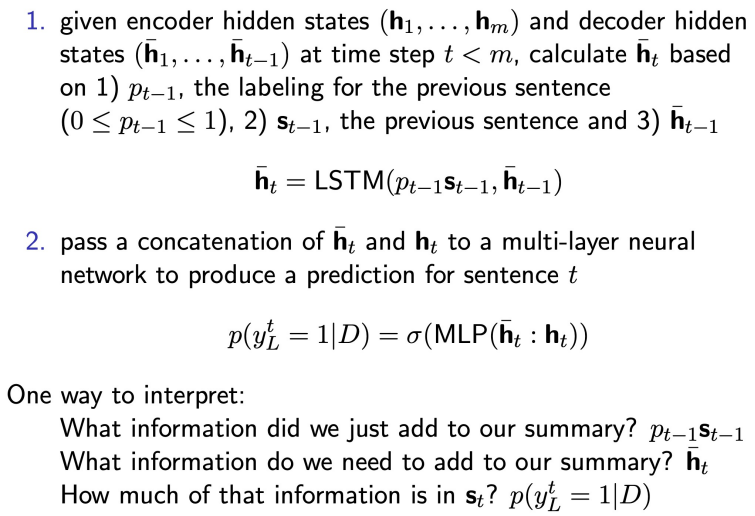


Model arch:

* Hierarchical document reader
  + Derive meaning representation of document from its constituent sentences
* Attention based hierarchical content extractor
* Encoder-decoder architecture
* Document reader (encoder, CNN + RNN):
  + CNN sentence encoder
    - Useful for sentence classification
    - Easy to train
    - Operates at word level to produce a sentence representation
    - Embed a doc sentence s of n words into a dense matrix (d is the size of word embedding, concat word vectors as input image), do convolutions, and then max pool over time
    - concat feature maps after pooling to produce a sentence vector
  + LSTM document encoder
    - avoids vanishing gradients
    - uses CNN-based sentence representation to produce a document representation
    - LSTM to compose a sequence of sentence vectors into a document vector
    - The hidden states of the LSTM = a list of partial representations, each focuses on the corresponding input sentence given previous content
    - Altogether constitute document representation



Sentence extractor



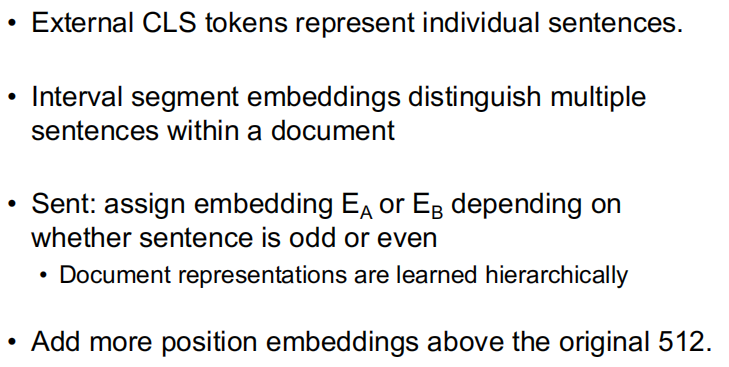
## Neural Approach 2

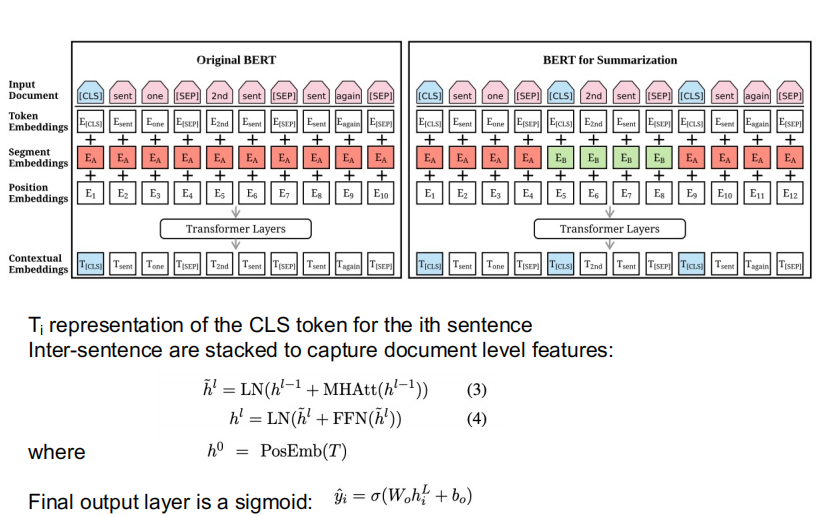
BertSum

Challenges in using BERT for summarization:

* Output vectors grounded to tokens in a masked language model but in summarization, we want representations of sentences
* Segmentation embeddings represent sentence pairs. In summarization, we want to encode and manipulate multiple sentence sequences.

Approach:





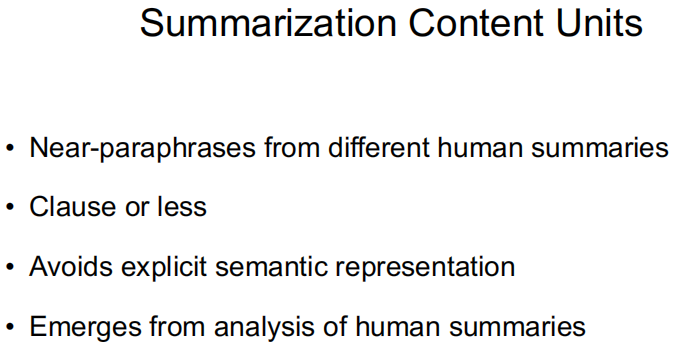
## Evaluation

ROUGE:

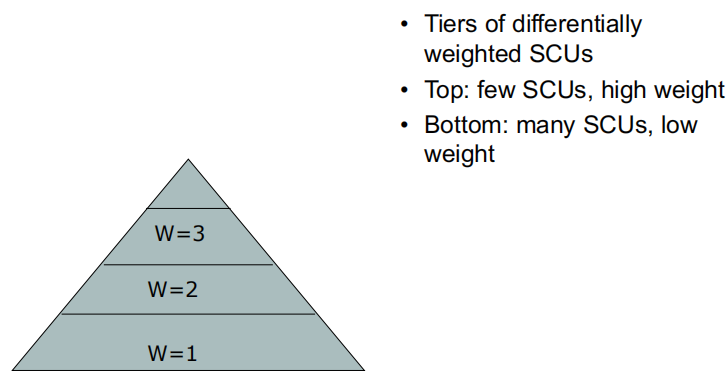
* Measures recall
* Rouge-N: How many N-grams in the human summary did the system summary find
* Pros
  + Automatic metric: Can be used for tuning
  + With enough examples or enough human models, differences are significant
* Cons
  + In practice, there often aren’t enough examples
  + Measures word overlap so re-wording a problem
  + Semantic similarity is not captured

Pyramid

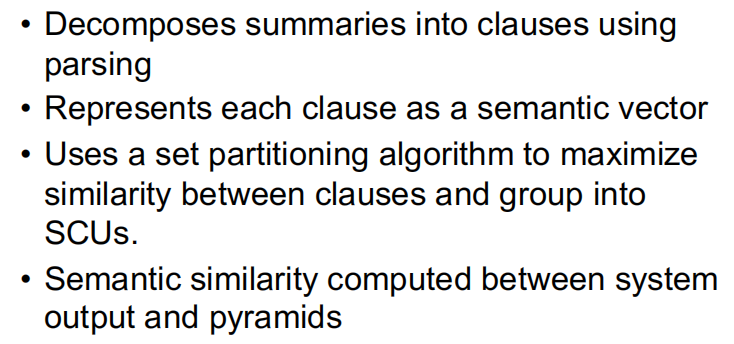
* Uses multiple (>=5) human summaries
* Information is ranked by its importance
* Allows for multiple good summaries
* created from the human summaries
  + Elements of the pyramid are content units
  + System summaries are scored by comparison with the pyramid
  + summarization content units (SCUs)



* + idealized representation: the higher weight, the more important

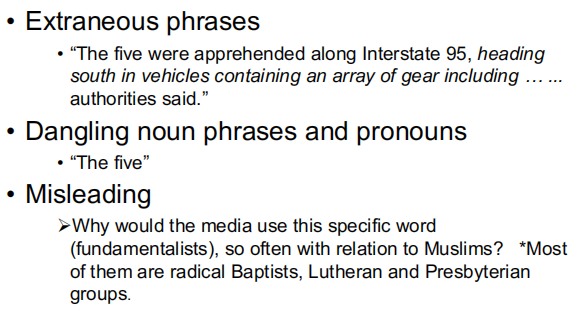


* + Pyramid score = D/MAX, D is the sum of the weights of the SCUs in a summary extracted/generated, MAX is the sum of the weights of the SCUs in a ideally informative summary (golden)
  + Originally needs much labor work (crowd sourcing), can be automated

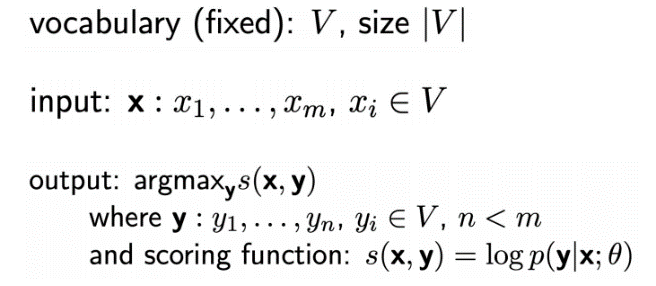


# Text Summarization (Abstractive)

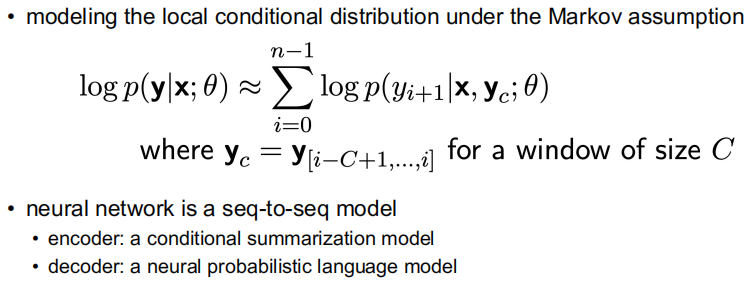
Problems with sentence extraction:



Problem framework

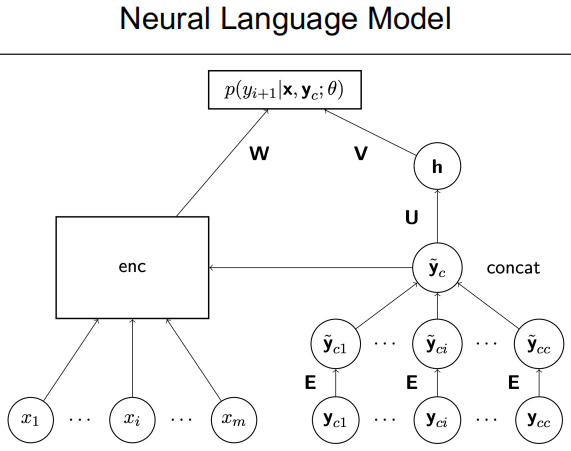


main focus

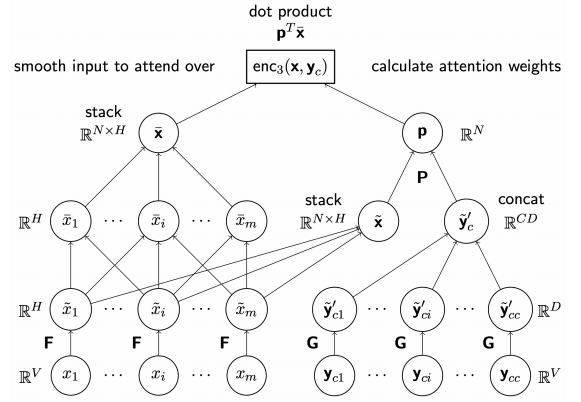


## RNN/CNN-based seq2seq

Neural LM:



Attention based encoder



Extensions:

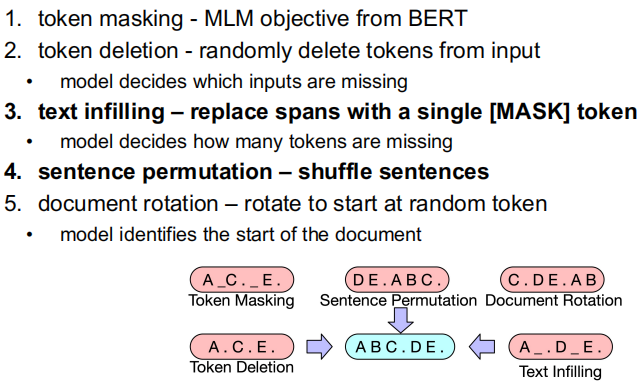
* use a beam search decoder
  + maintains full vocabulary
  + limits to k potential hypotheses at each position of the summary
  + compare to greedy decoding:
    - avoids making a bad choice early on
    - allows for consideration of a wider possibility of paraphrases
    - more computationally expensive
* Add features to promote using words of input (extractive features)
  + combine the local conditional probability with indicator features for n-grams from input

## BART

pre-trained “denoising” auto-encoder for seq2seq models

* modifies the original transformer, and uses a bidirectional encoder and left-to-right autoregressive decoder
* pre-trained on noisy documents; optimize cross-entropy loss between the decoder’s output and the original (reconstruction)
* flexible – allows any noise transformation

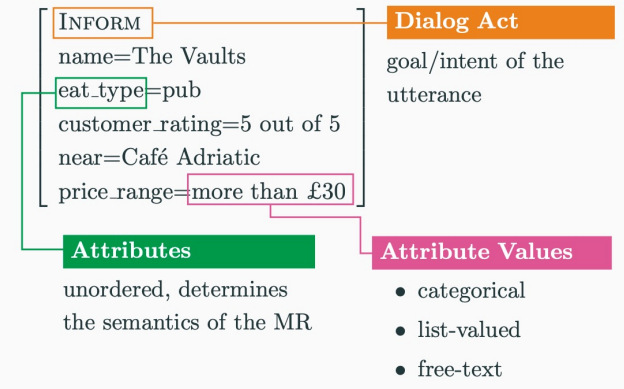
BART pretrain tricks:



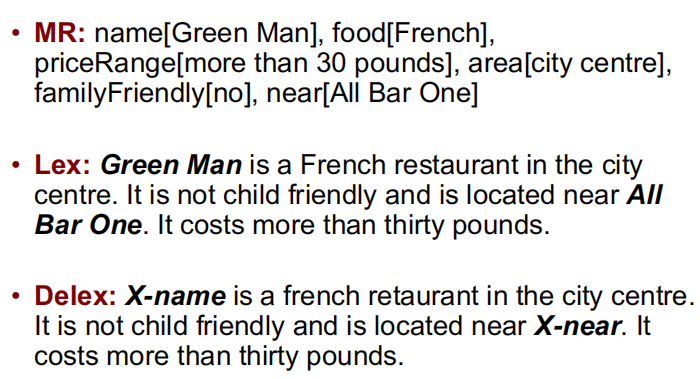
# Language Generation

end-to-end: generate description from meaning representation (MR)

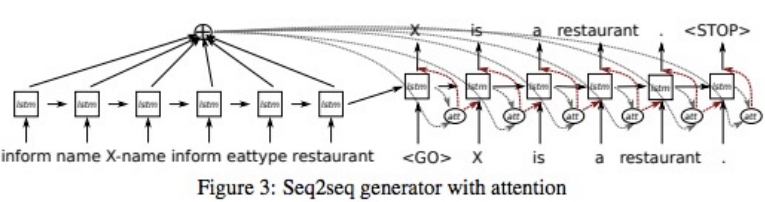
example MR:



Delexicalization example



Can use encoder-decoder RNN, need to convert input dialog act (DA) and output tree into sequences.

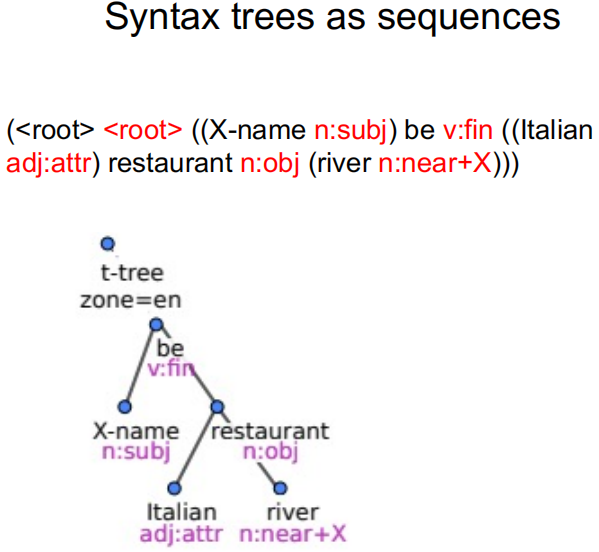


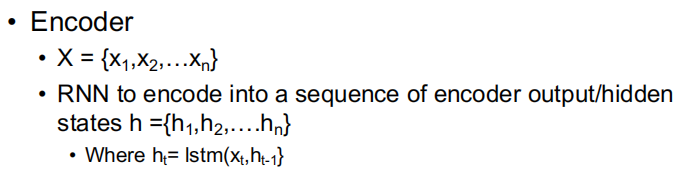
## Dialog Act (DA)

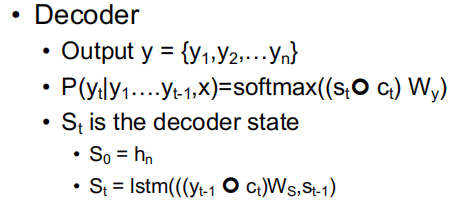
triple: DA type, slot, valie

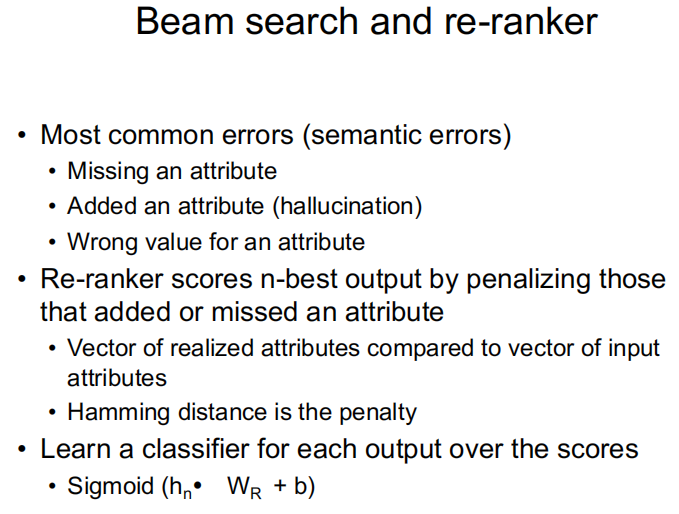
concatenate triples all slots

each token is an embedding

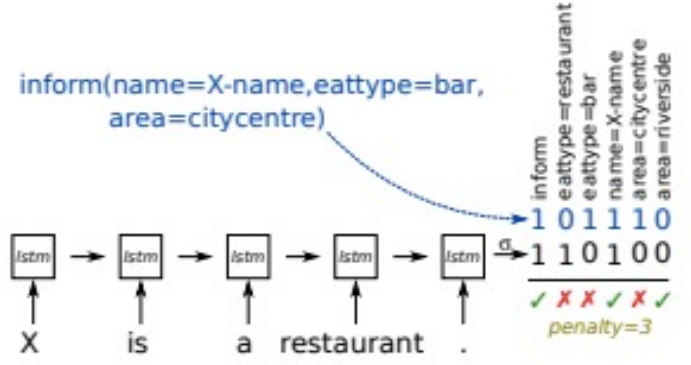








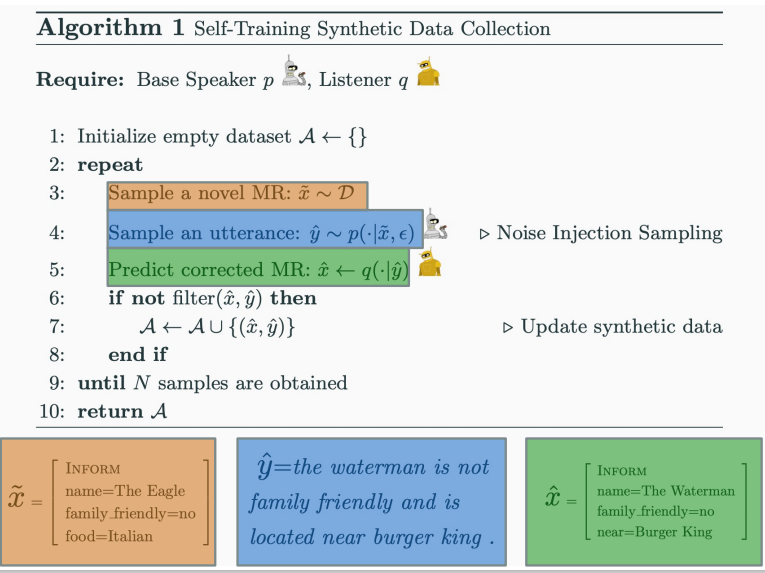
Hamming distance: find the number of unmatched attributes/attribute values



seq2seq often fail to correctly express a meaning representation and can be outperformed by hand-engineered methods. Why failed? Idiosyncratic MRs not well modeled and some attribute values never in the n-best list! RNNs have limited systematicity. To get a faithful generation model, need data augmentation to create diverse MR/utterance pairs not seen in the training distribution.



Get a novel MR from badly generated description and use these as a new training pair. This trick is task specific and probably won’t work in other tasks/scenarios. Example:



Noise injection sampling:

* Randomness is moved from next word selection to hidden states
* Greedy decoding will work well, selecting the most likely next word.
* Reduces the risk of drawing a disfluent word
* Easier to obtain a topically diverse but syntactically well-formed output
* Train new seq2seq model on original + synthetic data and the trained model is the faithful speaker.

Evaluation:

* BLEU
* ROUGE-L
* Semantic Error Rate (SER) = (n\_miss + n\_wrong + n\_added) / n\_attr

# Info Extraction

# Dialog

# Bias