Language modelling and language generation

Mariana Romanyshyn
Vsevolod Dyomkin

Contents

1. Language modelling
    - Statistical
    - Neural
    - Evaluation
2. Language generation
    - Free-form
    - Data-to-text
    - Text-to-text
    - Evaluation

1. Language modelling

- Statistical LM
- Neural LM
- Evaluation of LM

Language modelling

*What is the probability of sentence **S** in language **L**?*

Language modelling

*What is the probability of sentence **S** in language **L**?*

P(“Я люблю маму.”) = ?

P(“Я люблю Таню.”) = ?

P(“Я люблю пиво.”) = ?

P(“Я люблю пивом.”) = ?

Language modelling: why?

- Word choice, predictive typing / autocomplete
- Natural language generation
- Machine translation
- Spelling and grammatical error correction
- OCR, ASR, code breaking, paleolinguistics etc.

Language modelling: why?

I am pointing a gub at you.
I am pointing a gum at you.
I am pointing a gun at you.
I am pointing a gull at you.

Language modelling: why?

He is the center of attention.
He is the centaur of attention.
He is the cent or of attention.
He is the cento of attention.
He is the cental of attention.

Statistical LM

Chain rule + Markov assumption + given N

Statistical LM

Chain rule + Markov assumption + N = 3

P(*“<S> <S> I like kittens . </S>”*) =

*P(“I”|“<S> <S>”) **

*P(“like”|“<S> I”) **

*P(“kittens”|“I like”) **

P(“.”|“like kittens”)

Statistical LM

Chain rule + Markov assumption + N = 3

P(“<S> <S> I like kittens . </S>”) =

P(“I”|“<S> <S>”) *

P(“like”|“<S> I”) *

P(“kittens”|“I like”) *

P(“.”|“like kittens”)

Statistical LM

Chain rule + Markov assumption + N = 3

P(“<S> <S> I like kittens . </S>”) =

*P(“I”|“<S> <S>”) **

*P(“like”|“<S> I”) **

P(“kittens”|“I like”) = C(“I like kittens”) / C(“I like”)

P(“.”|“like kittens”)

What if we never saw “kittens”?

P(*“<S> <S> I like kittens . </S>”*) = *0 ?*

Smoothing techniques: add-k

- Add-1 (Laplace) smoothing
- Add-k smoothing
- Naive +1 smoothing

$$P_{Add-1}(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i) + 1}{c(w_{i-1}) + V}$$

** Add-one smoothing is generally a horrible choice.*

Smoothing techniques: backoff

- Stupid backoff

$$S(w_i|w_{i-2}\ w_{i-1}) = \begin{cases} \frac{C(w_{i-2}\ w_{i-1}\ w_i)}{C(w_{i-2}\ w_{i-1})} & \text{if } C(w_{i-2}\ w_{i-1}\ w_i) > 0 \\ 0.4 \cdot S(w_i|w_{i-1}) & \text{otherwise} \end{cases}$$

- Interpolation

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

Smoothing techniques: Kneser-Ney

Interpolation + discounting + word history

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

$$\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}v)}$$

** One of the best-performing smoothing techniques*

Smoothing techniques: Kneser-Ney

Discounting — subtracting a fixed number D from all n-gram counts.

$$A(w_1, \ldots, w_n) = C(w_1, \ldots, w_n) - D$$

Bigram count in training set	Bigram count in heldout set
0	0.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

Smoothing techniques: Kneser-Ney

Word history — count the number of contexts that the word appears in:

- *Kong* appears mostly as part of *Hong Kong*
- *cat* appears in multiple contexts

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

Smoothing techniques: Kneser-Ney

Discounting + interpolation + word history

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

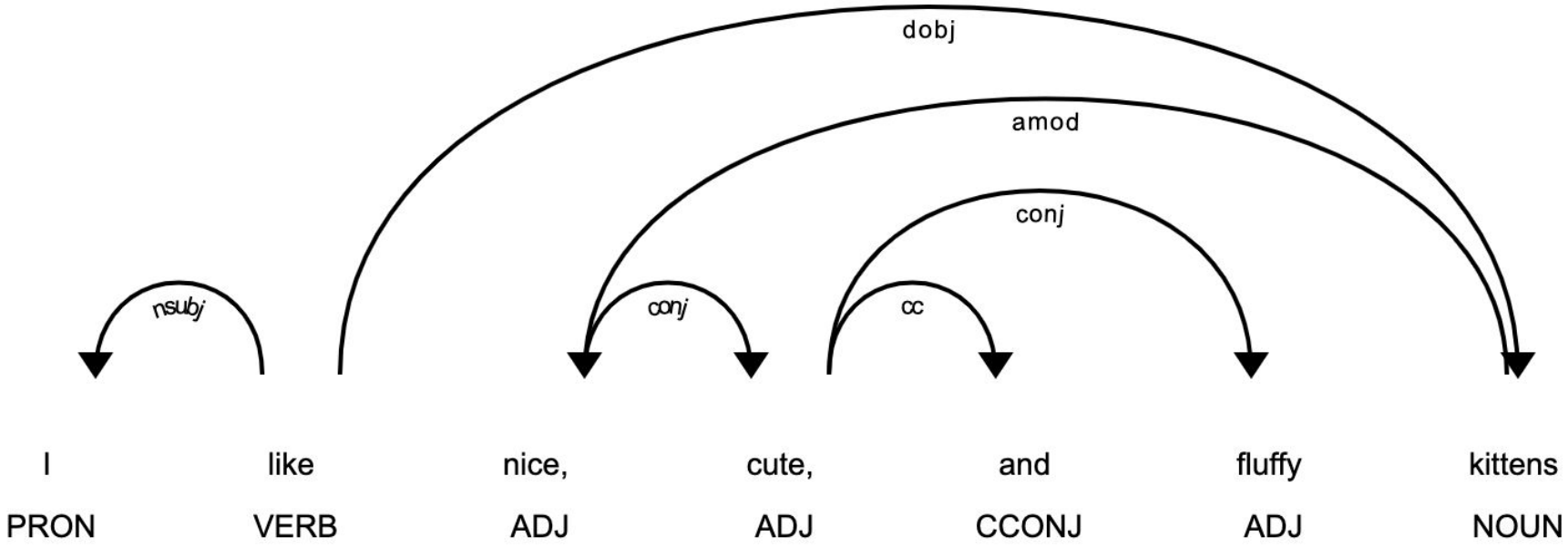
$$\lambda(w_{i-1}) = \frac{d}{\sum_v C(w_{i-1}v)}$$

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

Statistical LM challenges

- Does not generalize
  - *C(“red car”) = 2 390*
  - *C(“blue car”) = 1 113*
  - *C(“purple car”) = 0*
- Does not capture long-range dependencies
  - *I like nice, cute, and fluffy kittens.*
- Scaling to *N>5* is very expensive
- Needs intricate smoothing techniques

LM with syntactic ngrams



LM with syntactic ngrams

P(*“I like nice, cute, and fluffy kittens.”*) =

*P(“I”|“like_nsubj”) **

*P(“kittens”|“like_dobj”) **

*P(“nice”|“kittens_amod”) **

*P(“cute”|“nice_conj”) **

*P(“fluffy”|“nice_conj”) * ...*

SumLM

SumLM — simple score to compare different words in the same context.

- *Preposition Selection*
- *Spelling correction*
- *Word sense disambiguation*
- *Word choice*

SumLM

Sentence: “Every day I go *on* school.”

Candidates: *{from, to, with, in, ...}*

For each candidate:

- collect freqs of overlapping ngrams (N=1..5)
- weight each ngram by length
- sum weighted freqs

Ngram LMs

Implementation:

- cut-off
- efficient storage (binary trees, perfect hash-tables, …)
- quantization
- efficient estimation (MapReduce)

Ready-to-use:

- KenLM
- BerkeleyLM

KenLM

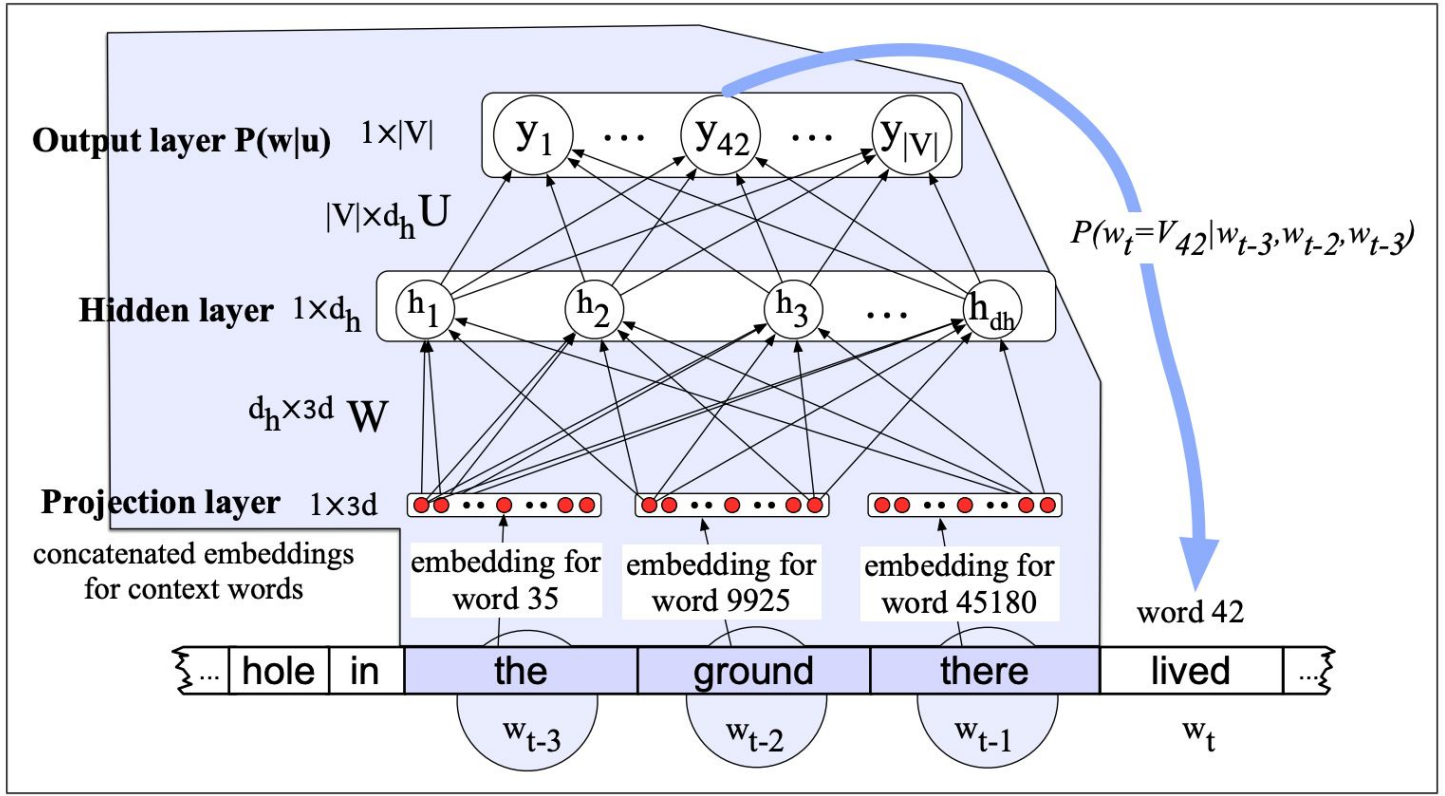
- Uses Kneser-Ney smoothing
- Fast and memory-efficient
- Returns a negative log probability

```
>>> (kenlm/scores ["<S> I like to swim . </S>"
                   "<S> Swim to like I . </S>"])
[-8.113167023286223 -17.18827013671398]
```

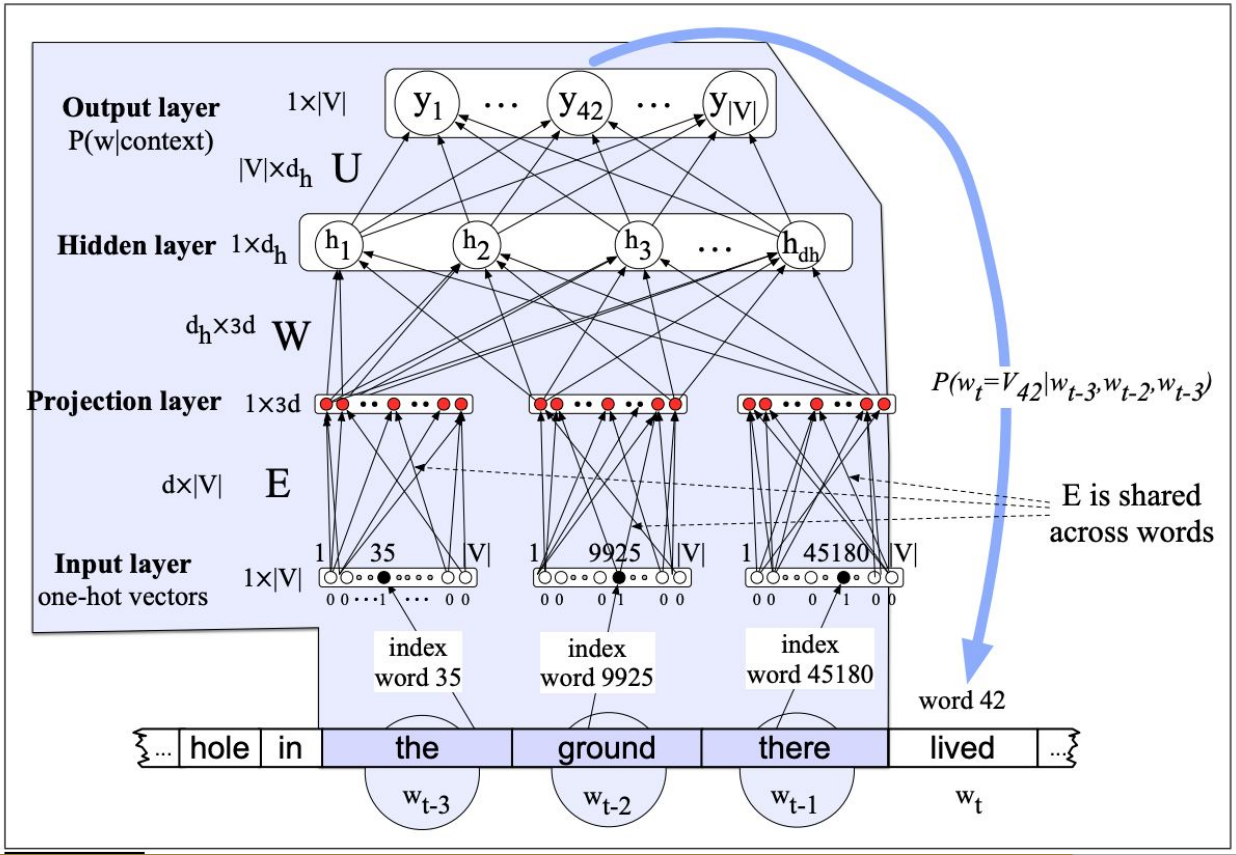
KenLM

```
>>> (kenlm/scores ["<S> That 's a nice idea ! </S>"
                   "<S> That 's a nice disease ! </S>"
                   "<S> That 's a nice tnetennba ! </S>"])
[-9.143143055960536 -13.877632185816765 -13.963078618049622]
```

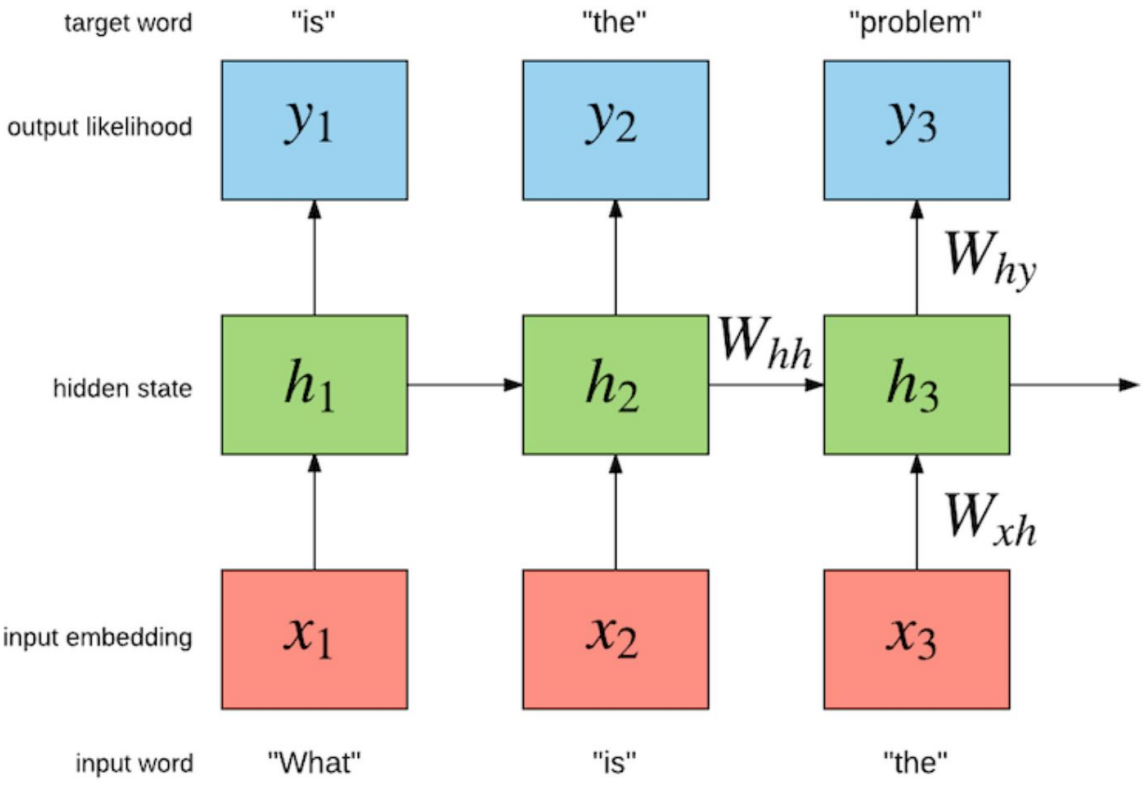
Neural LM: FFNN



Neural LM: FFNN



Neural LM: RNN



Taken from http://torch.ch/blog/2016/07/25/nce.html

Neural LM challenges

- Takes a long time to train
- Much more expensive
- May generalize too much
  - *brown horse, white horse, green horse 0_o*
- ***For many end tasks***, doesn't work that much better than statistical LMs.

Evaluation of LMs

- Intrinsic evaluation
  - perplexity

- Extrinsic evaluation
  - speech recognition
  - spelling correction
  - word choice

Perplexity

Perplexity — a measure of surprise per word.

- If the text belongs to the language, perplexity should ↓.
- If the text doesn't belong to the language, perplexity should ↑.

$$\mathrm{PP}(W) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_i|w_1 \ldots w_{i-1})}}$$

Perplexity

Text: *natural language processing*

w	p(w | <s>)
processing	0.4
language	0.3
the	0.17
natural	0.13

w	p(w | natural)
processing	0.4
language	0.35
natural	0.2
the	0.05

w	p(w | language)
processing	0.6
language	0.2
the	0.1
natural	0.1

$$PP(\text{natural language processing}) = \sqrt[3]{\frac{1}{0.13 \times 0.35 \times 0.6}} = 3.32$$

Perplexity SOTA (2016)

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

Taken from https://arxiv.org/pdf/1602.02410.pdf

Perplexity SOTA

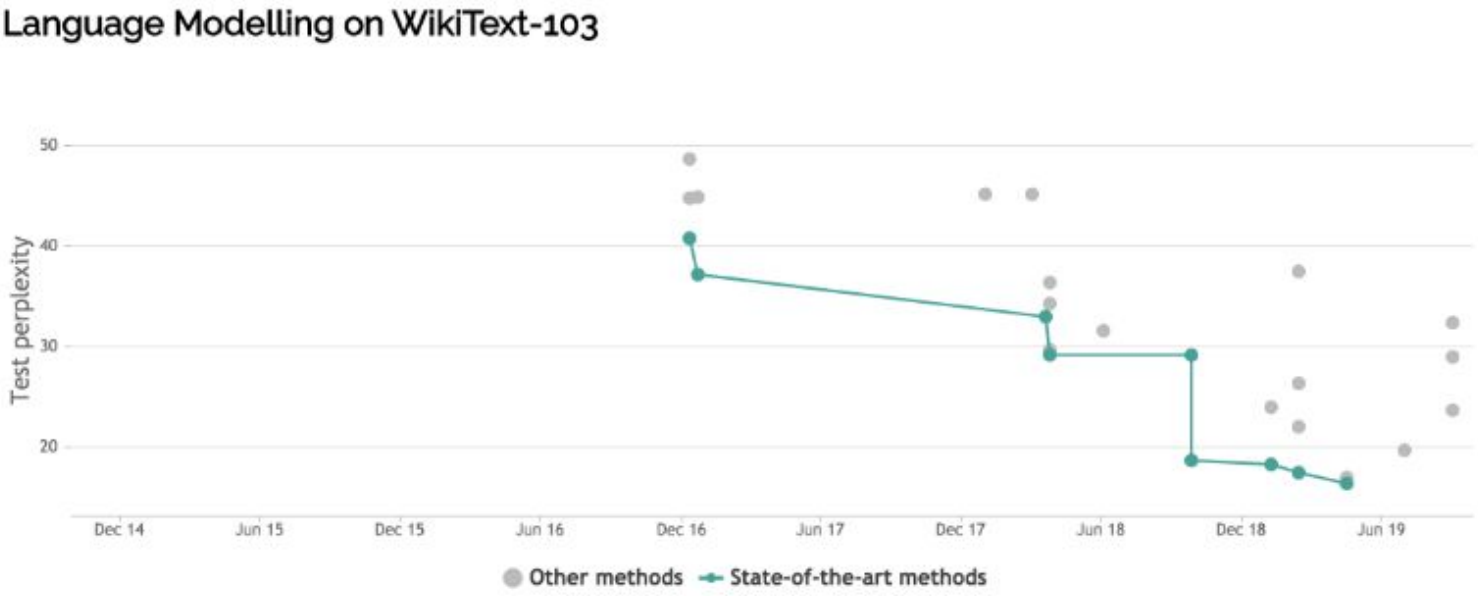


Figure 4: SOTA perplexity on the WikiText-103 dataset over time. paperswithcode.com

Taken from https://thegradient.pub/understanding-evaluation-metrics-for-language-models/

Perplexity SOTA

Check http://nlpprogress.com/

2. Language generation

- Free-form NLG
- Data-to-text NLG
- Text-to-text NLG
- Evaluation of NLG

Language generation

Types:

- General-purpose / domain-specific
- Free-form / data-to-text / text-to-text

Scale:

- Word-level
- Text-level
- Book-level

Language generation

Free-form NLG methods:

- Rule-based
- LM-based
- DL-based

Free-form NLG with rules

Nick Montfort's World Clock

```
for hour in range(24):
    print
    print '## ' + str(hour)
    print
    for minute in range(60):
        now = fresh(now)
        negative = fresh(negative)
        positive = fresh(positive)
        modifiers = [negative[0], positive[0], positive[0] + ' yet ' + negative[0], negative[0] +
        place_adj = choice(modifiers)

        pronoun = choice(['He', 'She'])
        if pronoun == 'He':
            male_names = fresh(male_names)
            name = male_names[0]
            a_man = fresh(a_man)
            person = a_man[0]
        else:
            female_names = fresh(female_names)
            name = female_names[0]
            a_woman = fresh(a_woman)
            person = a_woman[0]
```

Free-form NLG with LM

Having a language model:

- select a word based on the sequence so far
- add this word to the sequence
- repeat

Free-form NLG with LM

Having a language model:

- select a word based on the sequence so far
  - *argmax*
  - *sampling*
  - *beam search*
- add this word to the sequence
- repeat

2019 project on poetry generation

клямку , жовтаву скалочку знайду .
і зарості малини звільна линуть .
тяжке прокляття нам хвилини назавжди ,
що не в твої білі копита
б’ють , тебе в блакитні пюпітри

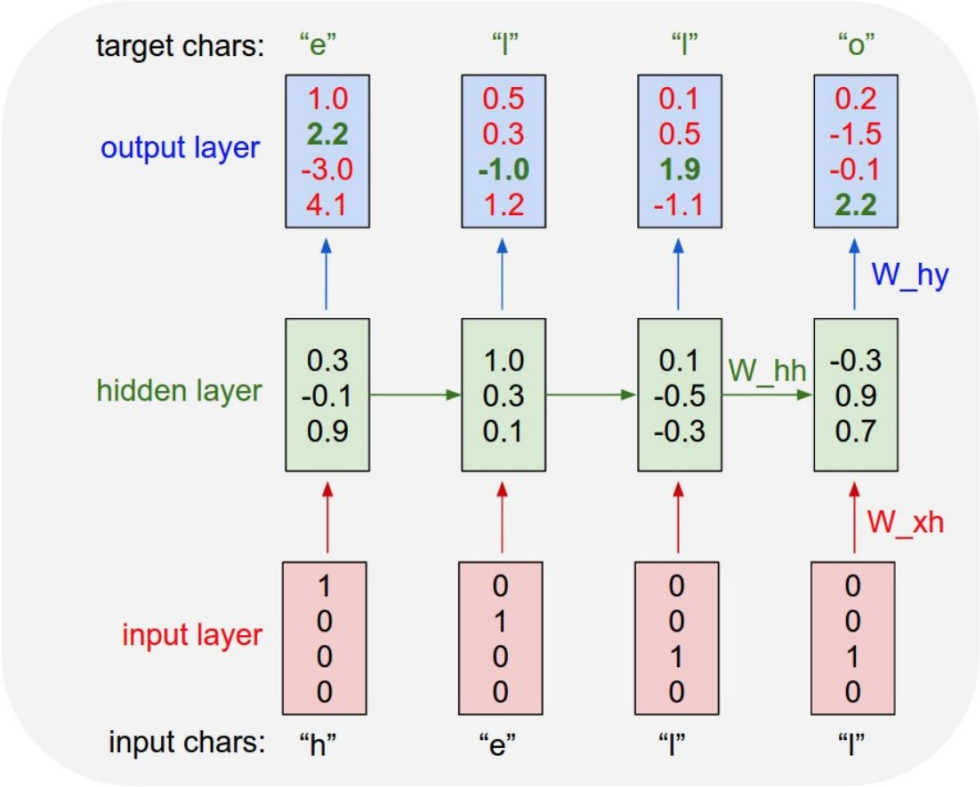
Borrowed from Anton Shpigunov’s presentation

Free-form NLG with LM

Input units:

- words *(V ≈ 500K)*
- characters *(V ≈ 100)*
- byte-pair encodings *(V ≈ 32K–64K)*

Character-level LM



Taken from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Byte-pair encoding

Algorithm:

1. Prepare a large corpus.
2. Count the frequency of each word.
3. Define a desired subword vocabulary size.
4. Split words into characters and count their paired frequencies.
5. Find pairs that occur together most frequently.
6. Merge the high-frequency pairs to form a new subword.
7. Repeat steps 5-6 until the needed vocabulary size is reached.

Byte-pair encoding

Example:

- Corpus: *“low: 5”, “lower: 2”, “newest: 6”* and *“widest: 3”*
- Pairs: *“lo: 7”, “ow: 7”, “we: 8”, “er: 2” … “es: 9”, “st: 9”*
- The most frequent subword pair is *“es: 9”*.
- Merge *“es”*.
- Next iteration:

  “lo: 7”, “ow: 7”, “we: 8”, “er: 2” … “wes: 6”, “est: 9”
- Repeat

OpenAI's GPT-2

- Data: WebText, scraped 8 mln documents (40 GB of text)
- Input units: BPEs *(V ≈ 50K)*
- Transformer-based (see next lecture)

SYSTEM PROMPT (HUMAN-WRITTEN)	*A train carriage containing controlled nuclear materials was stolen in Cincinnati today. Its whereabouts are unknown.*

MODEL COMPLETION (MACHINE-WRITTEN, FIRST TRY)	`The incident occurred on the downtown train line, which runs from Covington and Ashland stations.`
	
	`In an email to Ohio news outlets, the U.S. Department of Energy said it is working with the Federal Railroad Administration to find the thief.`

Taken from https://openai.com/blog/better-language-models/

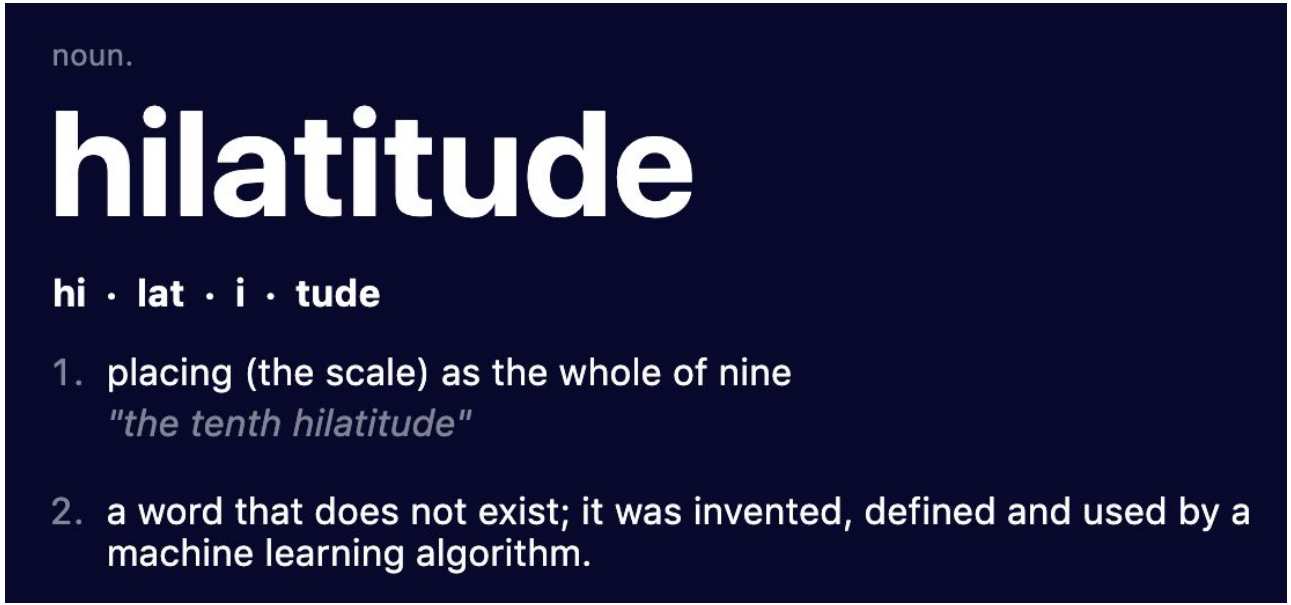
MIT-IBM-HarvardNLP's GLTR on GPT-2

Same model, different goal.

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexpl
ored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the
unicorns spoke perfect English.
The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned,
silver-white unicorns were previously unknown to science.
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally
solved.
Dr. Jorge PÃ©rez, an evolutionary biologist from the University of La Paz, and several companions,
were exploring the Andes Mountains when they found a small valley, with no other animals or
humans. PÃ©rez noticed that the valley had what appeared to be a natural fountain, surrounded
by two peaks of rock and silver snow.

Taken from http://gltr.io/

GPT-2-based word generation



Taken from https://www.thisworddoesnotexist.com//

Data-to-text generation

Data-to-text NLG tasks:

- generation of reports (news, weather, sports, finance, medical, etc.)
- image / video captioning
- dialogue response generation
- storytelling
- generation of exercises
- ASR, OCR, etc.

Data-to-text generation

Data-to-text NLG methods:

- Template-based / rule-based
- Hybrid
- Domain model + LM
- DL-based

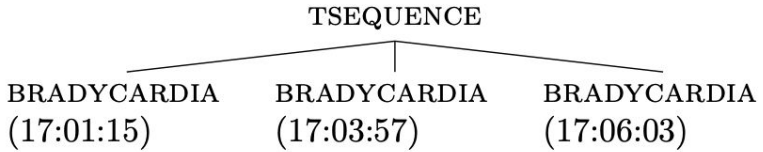
Template-based data-to-text NLG

Classical approach:

1. Content determination
2. Text structuring
3. Aggregation
4. Lexical choice
5. Realization

Template-based data-to-text NLG

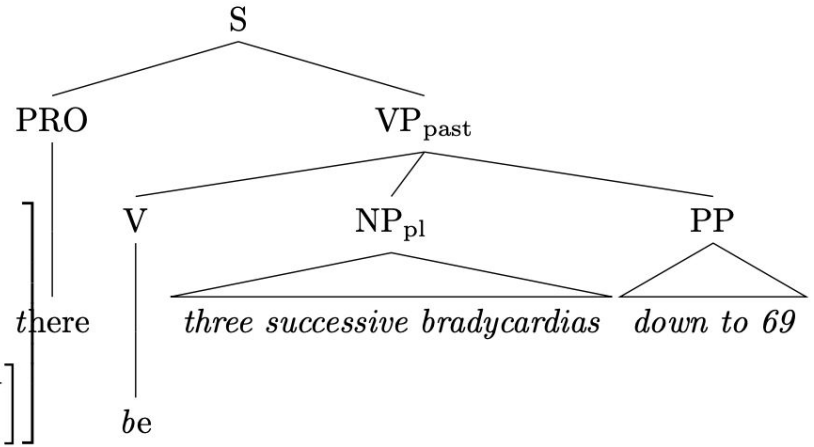
(a) Content Determination



(b) Text Structuring

$$\begin{bmatrix} \textit{Event} & \\ \text{TYPE} & \textit{existential} \\ \text{PRED} & \textit{be} \\ \text{TENSE} & \textit{past} \\ \text{ARGS} & \begin{bmatrix} \text{THEME} & \{b_1, b_2, b_3\} \\ \text{MIN-VAL} & 69 \end{bmatrix} \end{bmatrix}$$

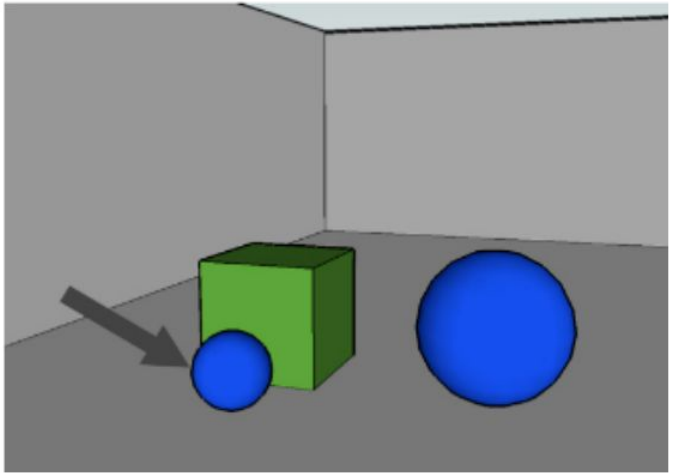
(c) Lexicalisation etc.



(d) Realisation

Taken from https://arxiv.org/pdf/1703.09902.pdf

Template-based data-to-text NLG



(a) Visual domain from the GRE3D corpus (Viethen & Dale, 2008)

	Domain objects		
Attr	d_1	d_2	d_3
Color	blue	green	blue
Shape	ball	cube	ball
Size	small	large	large
Rel	bef(d_2)	beh(d_1)	nt(d_2)

(b) Table of objects and attributes. *beh*: 'behind'; *bef*: 'before'; *nt*: 'next to'

Taken from https://arxiv.org/pdf/1703.09902.pdf

Template-based data-to-text NLG

Hotel review generation

- Template:
  “A [well-priced] <NAME> [is located] on <STREET> [close to <ATTR>].”
- Data:
  - NAME = Astoria Hotel
  - ADDRESS = {country: France, city: Paris, street: Rue Dunois}
  - ATTR = {Louvre: 100 m, city center: 200 m, airport: 5 km}
- Lexicon
  - *is located, can be found, is set, boasts convenient location*
  - *well-priced, famous, attractive, convenient*

Hybrid data-to-text NLG

Use ML to:

- select the most important data
- select the best template
- select the most suitable phrasing
- select the most grammatical phrasing

Hybrid data-to-text NLG

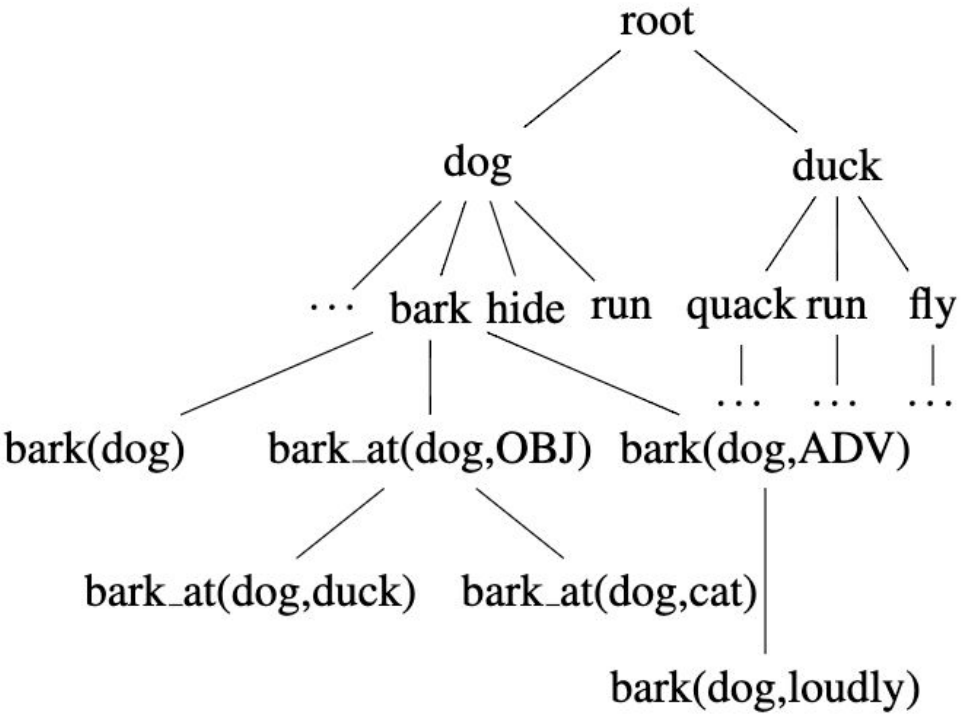
Use an LM to choose the best phrasing:

- *“wind speed 12-16 decreasing 8-12 then increasing 16-20"*
- *“wind speed 12-16 decreasing 8-12 then increasing to 16-20"*
- *“wind speed 12-16 decreasing 8-12 then rising 16-20"*
- *“wind speed 12-16 decreasing 8-12 then rising to 16-20"*
- *“wind speed 12-16 falling 8-12 then increasing 16-20"*
- *“wind speed 12-16 falling 8-12 then rising 16-20"*

Hybrid data-to-text NLG

Storytelling

- Input: *dog, duck*
- Build a graph of possible dependencies
- Generate sentences
- Rank sentences using ML



Taken from https://mafiadoc.com/learning-to-tell-tales-a-data-driven-approach-to-story-generation_59bd73d91723ddb63c355dee.html

Hybrid data-to-text NLG

Storytelling

	The family has the baby	*The giant guards the child*
Random	The family has the baby. The family is how to empty up to a fault. The baby vanishes into the cave. The family meets with a stranger. The baby says for the boy to fancy the creature.	The giant guards the child. The child calls for the window to order the giant. The child suffers from a pleasure. The child longer hides the forest. The child reaches presently.
Determ	The family has the baby. The family rounds up the waist. The family comes in. The family wonders. The family meets with the terrace.	The giant guards the child. The child rescues the clutch. The child beats down on a drum. The child feels out of a shock. The child hears from the giant.
Rank-based	The family has the baby. The baby is to seat the lady at the back. The baby sees the lady in the family. The family marries a lady for the triumph. The family quickly wishes the lady vanishes.	The giant guards the child. The child rescues the son from the power. The child begs the son for a pardon. The giant cries that the son laughs the happiness out of death. The child hears if the happiness tells a story.

Taken from https://mafiadoc.com/learning-to-tell-tales-a-data-driven-approach-to-story-generation_59bd73d91723ddb63c355dee.html

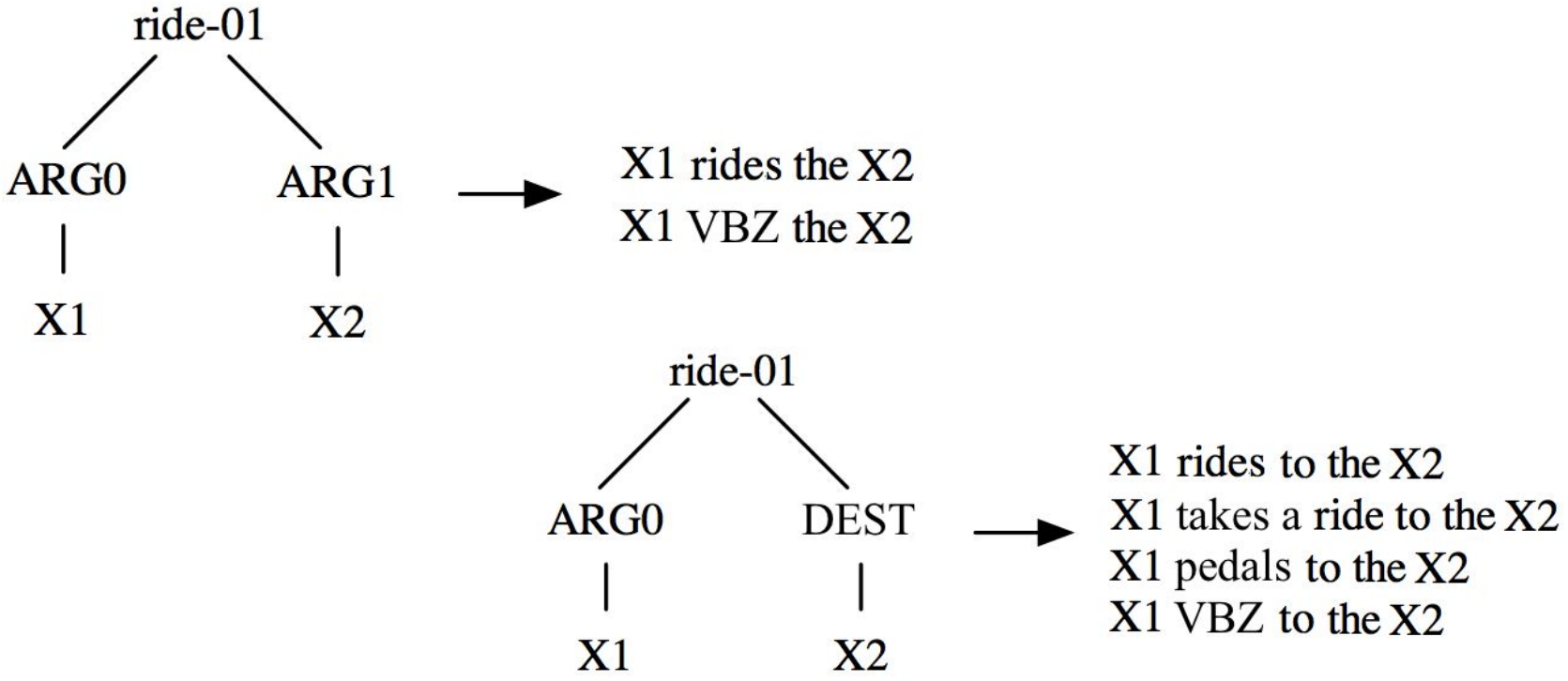
Hybrid data-to-text NLG

AMR to text:

- extract rules from AMR Banks
- use rules as features for a transducer
- find the best-scoring transduction (SMT-kind)
- pick the highest scoring sentence with a language model

** SemEval shared task on NLG with AMR (2017)*

AMR to text


ride-01
ARG0
ARG1
X1
X2
X1 rides the X2
X1 VBZ the X2
ride-01
ARG0
DEST
X1
X2
X1 rides to the X2
X1 takes a ride to the X2
X1 pedals to the X2
X1 VBZ to the X2

AMR to text: example

```
(a / and
    :op1 (r / remain-01
              :ARG1 (c / country :wiki "Bosnia_and_Herzegovina"
                                 :name (n / name :op1 "Bosnia"))
              :ARG3 (d / divide-02
                        :ARG1 c
                        :topic (e / ethnic)))
    :op2 (v / violence
              :time (m / match-03 :mod (f2 / football)
                        :ARG1-of (m2 / major-02))
              :location (h / here)
              :frequency (o / occasional))
    :time (f / follow-01
               :ARG2 (w / war-01
                         :time (d2 / date-interval
                                 :op1 (d3 / date-entity :year 1992)
                                 :op2 (d4 / date-entity :year 1995)))))
```

AMR to text: example

Source	Text
Reference	following the 1992-1995 war bosnia remains ethnically divided and violence during major football matches occasionally occurs here.
RIGOTRIO	following the 1992 1995 war, bosnia has remained an ethnic divide, and the major football matches occasionally violence in here.
CMU	following the 1992 1995 war , bosnia remains divided in ethnic and the occasional football match in major violence in here

Domain Model + Language Model

Works in ASR, OCR, (and also text-to-text - MT, GEC):

- the domain model generates many variants of resulting text based on source data (image, sound, incorrect text)
- the LM scores the variants and selects the best one

Noisy Channel Model

- A Noisy Channel Model has two components:

$P(e)$ **the language model**

$P(f \mid e)$ **the translation model**

- Giving:

$$P(e \mid f) = \frac{P(e, f)}{P(f)} = \frac{P(e)P(f \mid e)}{\sum_e P(e)P(f \mid e)}$$

and

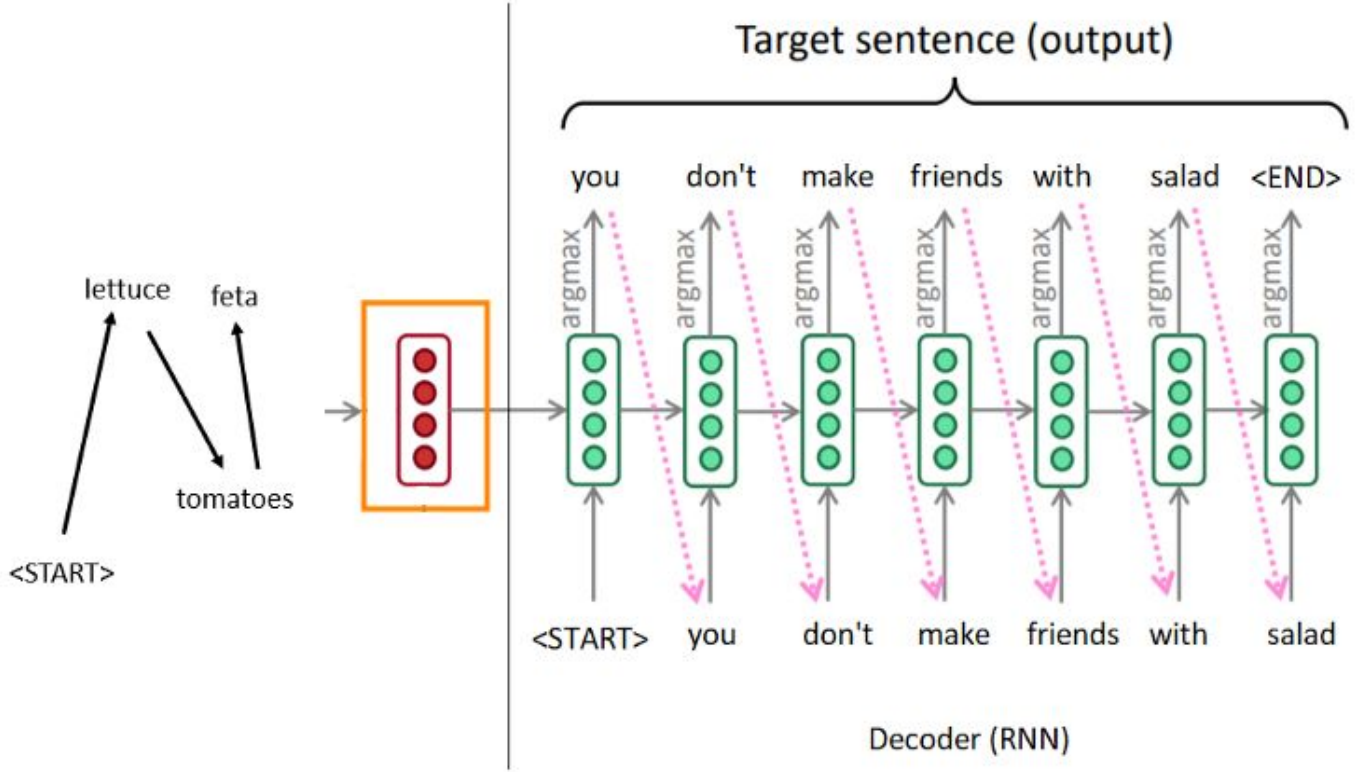
$$\text{argmax}_e P(e \mid f) = \text{argmax}_e P(e)P(f \mid e)$$

DL-based data-to-text NLG

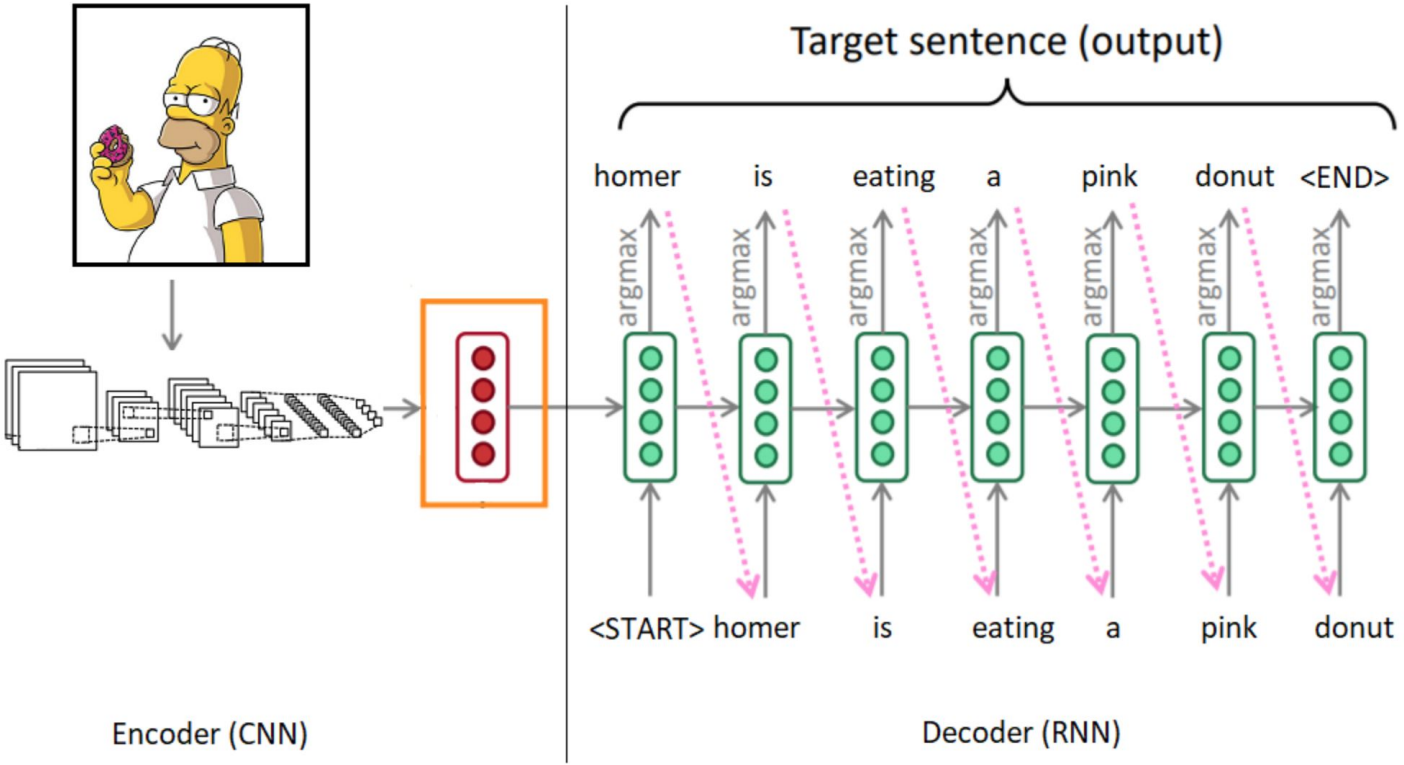
Encoder-decoder (seq2seq) models:

- **Encoder RNN:** extracts the information from the source sentence to produce an encoding (vector representation)
- **Decoder RNN:** a language model that generates the target sentence conditioned with the encoding created by the encoder

DL-based data-to-text: guided



DL-based data-to-text: multimodal



Text-to-text generation

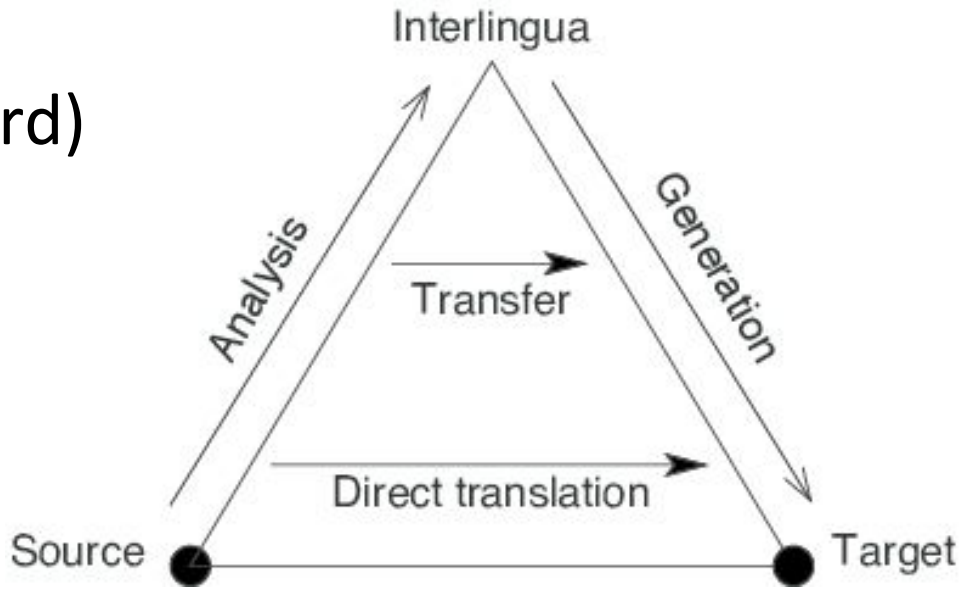
Text-to-text NLG tasks:

- machine translation
- text summarization
- paraphrasing
- text simplification
- dialogue response generation
- grammatical error correction
- style transfer
- email response generation

Machine Translation

Approaches:

- Direct (word-by-word)
- Phrase-based
- Via an abstract representation (“interlingua”)



Machine Translation Data

- Train on a parallel corpus (Europarl, UN, TEDTalks)
- Pivot Machine Translation (via an intermediary language)
- Zero-shot Machine Translation (joint training)

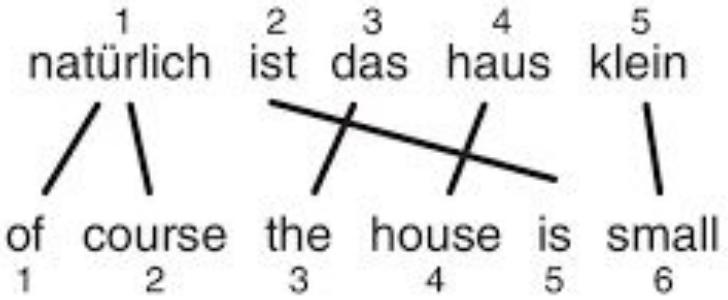
Statistical MT

Components:

- Alignment model
- Reordering model
- Phrase table
- Language model

SMT Alignment: IBM Models 1–5

- Model 1
  - One-to-one alignment
- Model 2
  - One-to-many alignment
- Model 3
  - …
- …
- Model 5



SMT Alignment: IBM Models 1–2

- Model 1

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = P(\mathbf{a} \mid e) P(\mathbf{f} \mid \mathbf{a}, e) = \frac{C}{(l+1)^m} \prod_{j=1}^{m} \mathbf{T}(f_j \mid e_{a_j})$$

- Model 2

$$P(\mathbf{f}, \mathbf{a} \mid \mathbf{e}) = P(\mathbf{a} \mid \mathbf{e}) P(\mathbf{f} \mid \mathbf{a}, \mathbf{e}) = C \prod_{j=1}^{m} \mathbf{D}(a_j \mid j, l, m) \mathbf{T}(f_j \mid e_{a_j})$$

IBM Models 1–2 Estimation

A Summary of the EM Procedure

- Start with parameters Θ^{t-1} as

$$\begin{aligned} &\mathbf{T}(f \mid e) \qquad \text{for all } f \in \mathcal{F}, e \in \mathcal{E} \\ &\mathbf{D}(i \mid j, l, m) \end{aligned}$$

- Calculate **alignment probabilities** under current parameters

$$a[i, j, k] = \frac{\mathbf{D}(a_j = i \mid j, l, m)\mathbf{T}(f_j \mid e_i)}{\sum_{i'=0}^{l} \mathbf{D}(a_j = i' \mid j, l, m)\mathbf{T}(f_j \mid e_{i'})}$$

- Calculate **expected counts** $tcount(e, f)$, $scount(e)$, $acount(i, j, l, m)$, and $acount(j, l, m)$ from the alignment probabilities

- Re-estimate $\mathbf{T}(f \mid e)$ and $\mathbf{D}(i \mid j, l, m)$ from the expected counts

SMT phrases & reordering

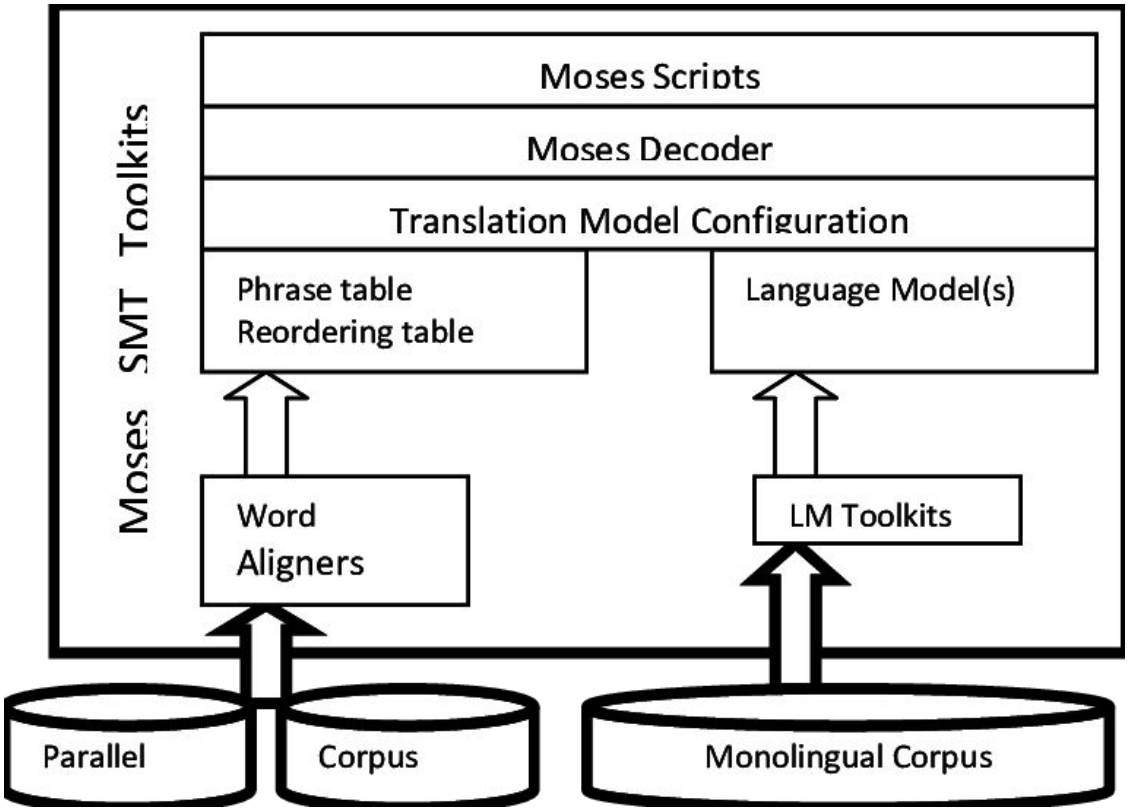
Generation part — Noisy Channel + LM.
During decoding, the source sentence is segmented into a sequence of phrases. Each source phrase is translated into a destination phrase. The destination phrases may be reordered. Phrase translation is modeled by a probability distribution. Reordering of the output phrases is modeled by a relative distortion probability distribution.

In order to calibrate the output length, we introduce a factor ω (called word cost) for each generated output word in addition to the trigram language model probability. This is a simple means to optimize performance. Usually, this factor is larger than 1, biasing toward longer output.

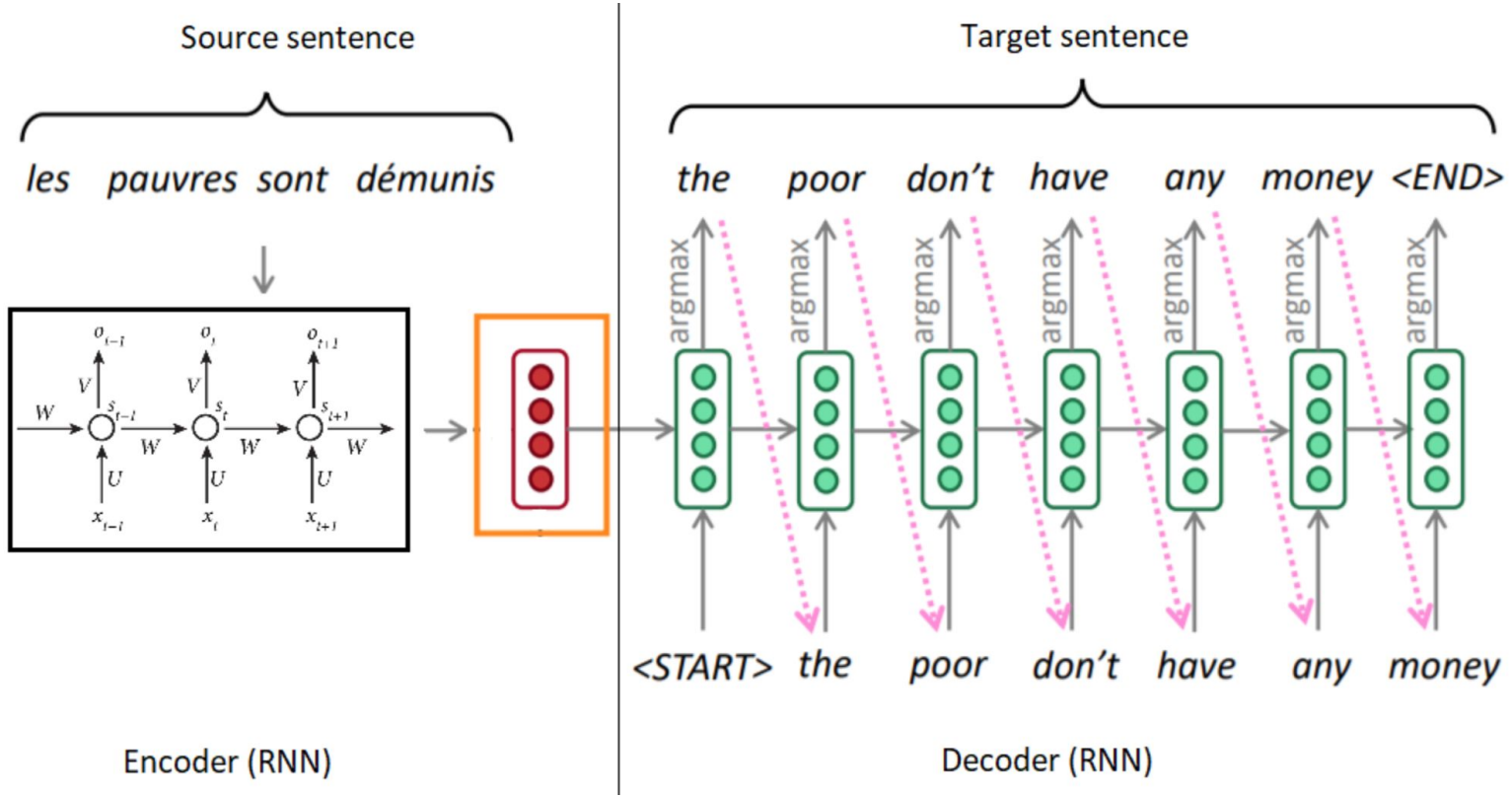
In summary, the best output sentence given an input sentence according to our model is:

$$\text{output} = \text{argmax}\ p(e|f) = \text{argmax}\ p(f|e)\ p\_LM(e)\ \omega\ \text{length}(e)$$

Moses SMT


Moses SMT Toolkits
Moses Scripts
Moses Decoder
Translation Model Configuration
Phrase table
Reordering table
Language Model(s)
Word
Aligners
LM Toolkits
Parallel
Corpus
Monolingual Corpus

Neural machine translation



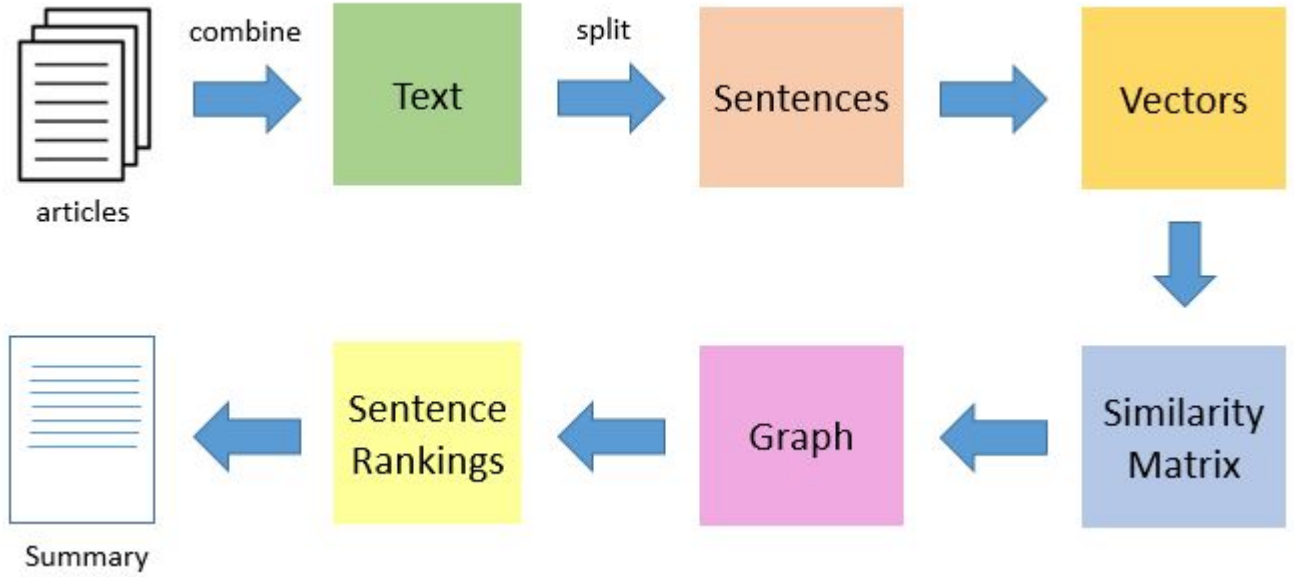
Summarization

Approaches:

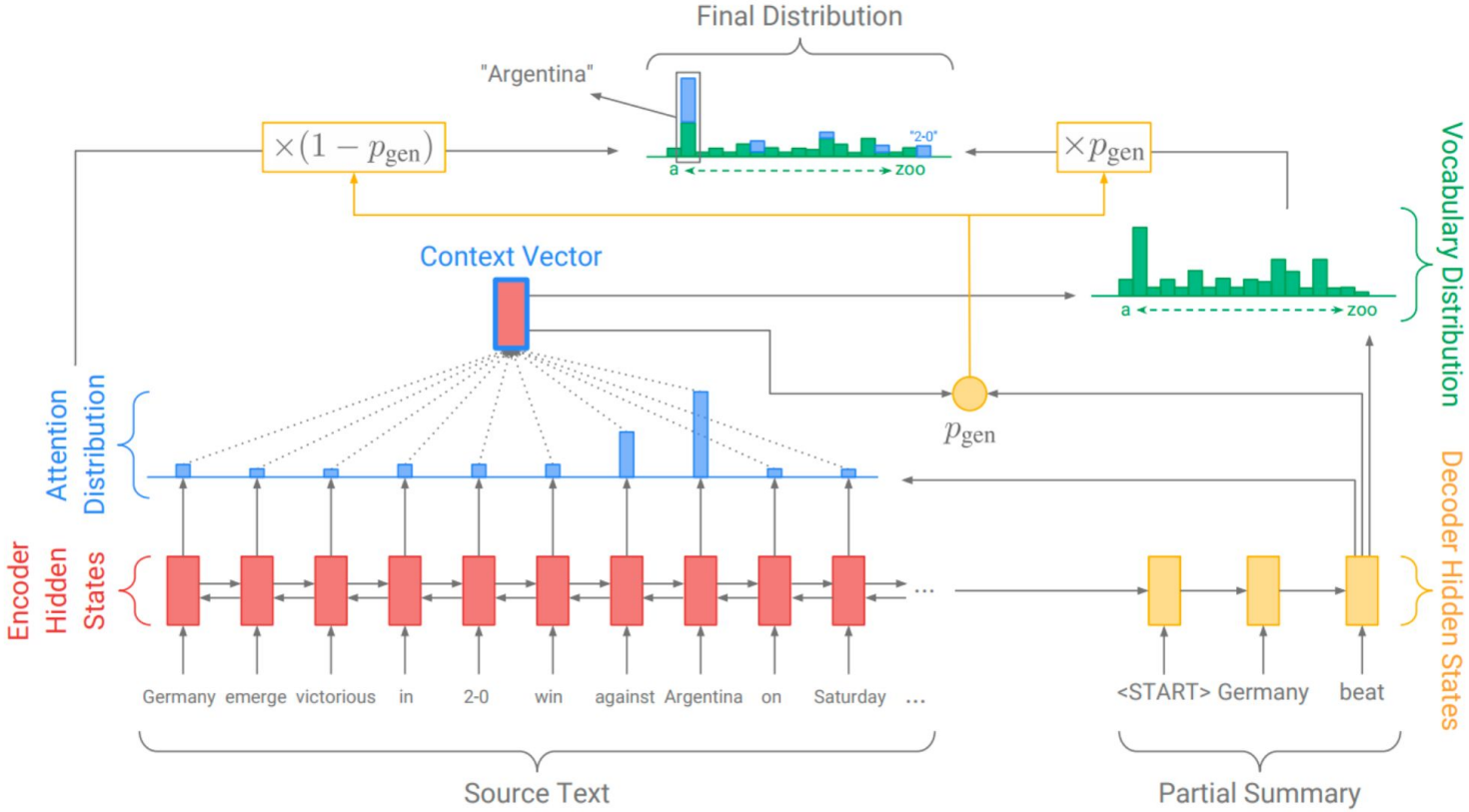
- Extractive
- Abstractive

Summarization Baseline: TextRank

Derived from PageRank

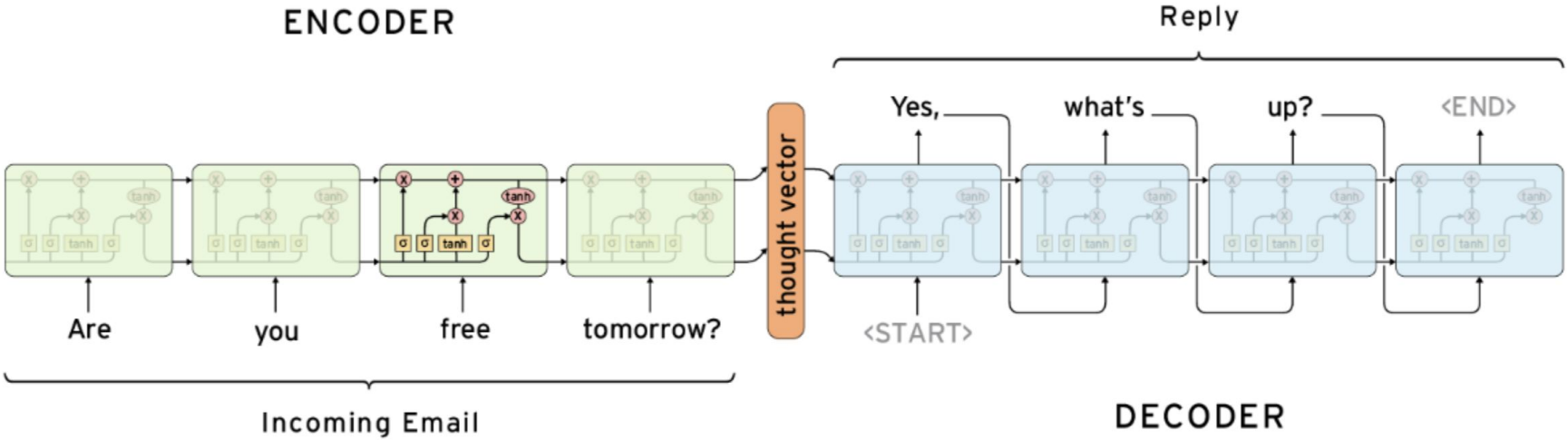


Neural summarization



Taken from https://tutorials.one/encoder-decoder-deep-learning-models-for-text-summarization/

Neural response generation



Taken from https://medium.com/@rndholakia/what-makes-alexa-siri-and-google-assistant-respond-intelligently-275043cd172e

Problem of seq2seq approach

Problem: all information (lexical, postional, syntactic etc.) needs to be combined into a single vector.

Solution: attention (see the next lecture).

Neural machine translation

NMT software:

- OpenNMT (Python; tf)
- MarianNMT (C++)
- Nematus (Python, tf)
- Sockeye (Python)

NLG evaluation

Need to capture quality & diversity

- (Best) Real-World Task-Based (Extrinsic)
- (Good) Laboratory Task-Based or Real-World Human Ratings
- (OK) Laboratory Human Ratings
- (Worst) Metrics

Challenges of automatic evaluation

- Hard to properly define
- Many possible correct variants
  - Need to capture “essence” and ignore surface differences
- Need to evaluate on several additional dimensions:
  - fluidity
  - grammaticality
  - etc.
- Different requirements for different NLG tasks

NLG metrics

WER (and Wacc=1-WER)

- Takes into account word order =>
- Not very well suited for MT, paraphrasing and other tasks that imply reordering

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

S - number of substitutions

D - number of deletions

I - number of insertions

C - number of correct words

N is the number of words in the reference (N=S+D+C)

NLG metrics

BLEU

- precision-oriented
- works on a text level

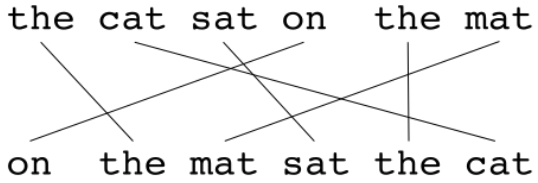
```
exp (1.0 * log p1 +
     0.5 * log p2 +
     0.25 * log p3 +
     0.125 * log p4 –
     max(words-in-reference / words-in-machine – 1, 0)

p1 = 1-gram precision
P2 = 2-gram precision
P3 = 3-gram precision
P4 = 4-gram precision
```

NLG metrics

- ROUGE
  - recall-oriented
  - counts how many n-grams in the reference translation show up in the output, rather than the reverse

- METEOR
  - aims to improve on BLEU
  - compares word stems and synonyms
  - uses a harmonic f0.11 metric adjusted by penalty



NLG metrics

- Perplexity
  - the surprise measure of the generated sentence
- BERTScore, BLUERT and other trained metrics
  - computes a similarity score for each token in the candidate sentence with each token in the reference sentence using contextual embeddings

NLG metrics

- HUSE

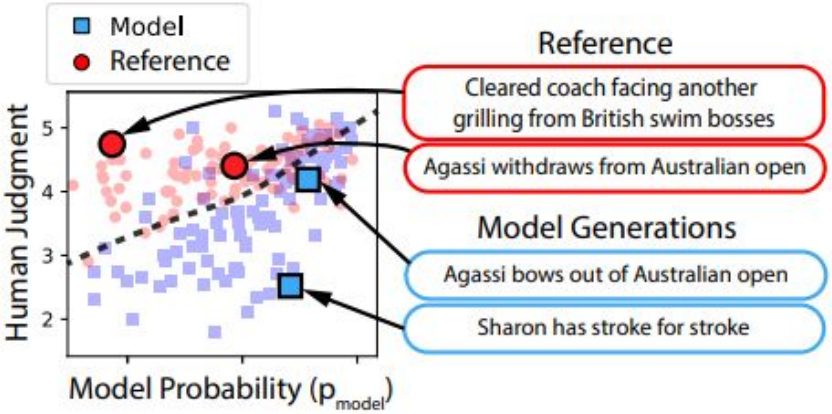
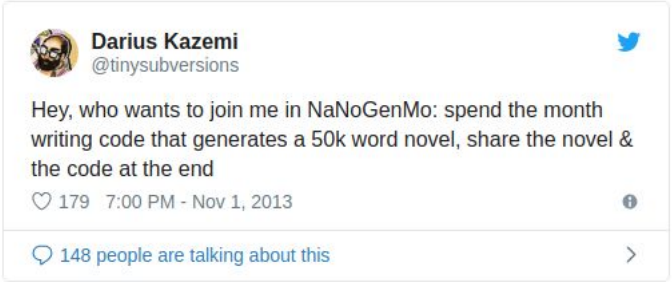


Figure 1. HUSE is twice the classification error of distinguishing reference and generated text based on human judgment scores and model probabilities. HUSE identifies samples with defects in quality (Sharon has stroke . . .) and diversity (Cleared coach facing . . .).

Taken from https://arxiv.org/pdf/1904.02792.pdf

P.S. Algorithmic book generation



Titles:

- Webster's Slovak – English Thesaurus Dictionary
- The 2007-2012 World Outlook for Wood Toilet Seats
- The World Market for Rubber Sheath Contraceptives (Condoms): A 2007 Global Trade Perspective
- Ellis-van Creveld Syndrome – A Bibliography and Dictionary for Physicians, Patients, and Genome Researchers
- Webster's English to Haitian Creole Crossword Puzzles: Level 1

https://singularityhub.com/2012/12/13/patented-book-writing-system-lets-one-professor-create-hundreds-of-thousands-of-amazon-books-and-counting/

References

- Chapters 3, 7 and 9 from Speech and Language Processing (Jurafsky and Martin)
- Web-Scale N-gram Models for Lexical Disambiguation (2009)
- KenLM
- The Unreasonable Effectiveness of Recurrent Neural Networks (Karpathy, 2015)
- Neural text generation: How to generate text using conditional language models (2018)
- Survey of the State of the Art in Natural Language Generation (2018)
- Types of NLG Evaluation: Which is Right for Me? (2017)
- MT for Low-Resource Languages