Model Design

Example model: LaserTagger (2019)

Malmi, Krause, Rothe, Mirylenka, Severyn. *Encode, Tag, Realize: High-Precision Text Editing*. EMNLP 2019 (pdf)

High-Overlap Example: Sentence Fusion

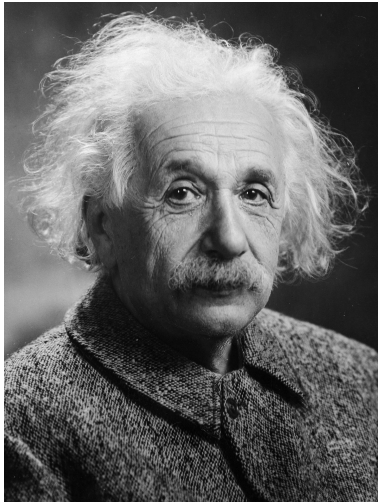
Given two or more answers, fuse them into a single coherent answer.

-- Example --

Query: [einstein birth and death]

Answers:

- Albert Einstein was born in 1879.
- Albert Einstein died in New Jersey.
- Albert Einstein died at the age of 76.



Fusion: Albert Einstein was born in 1879 and he died in New Jersey at the age of 76.

Sentence Fusion via Text Editing

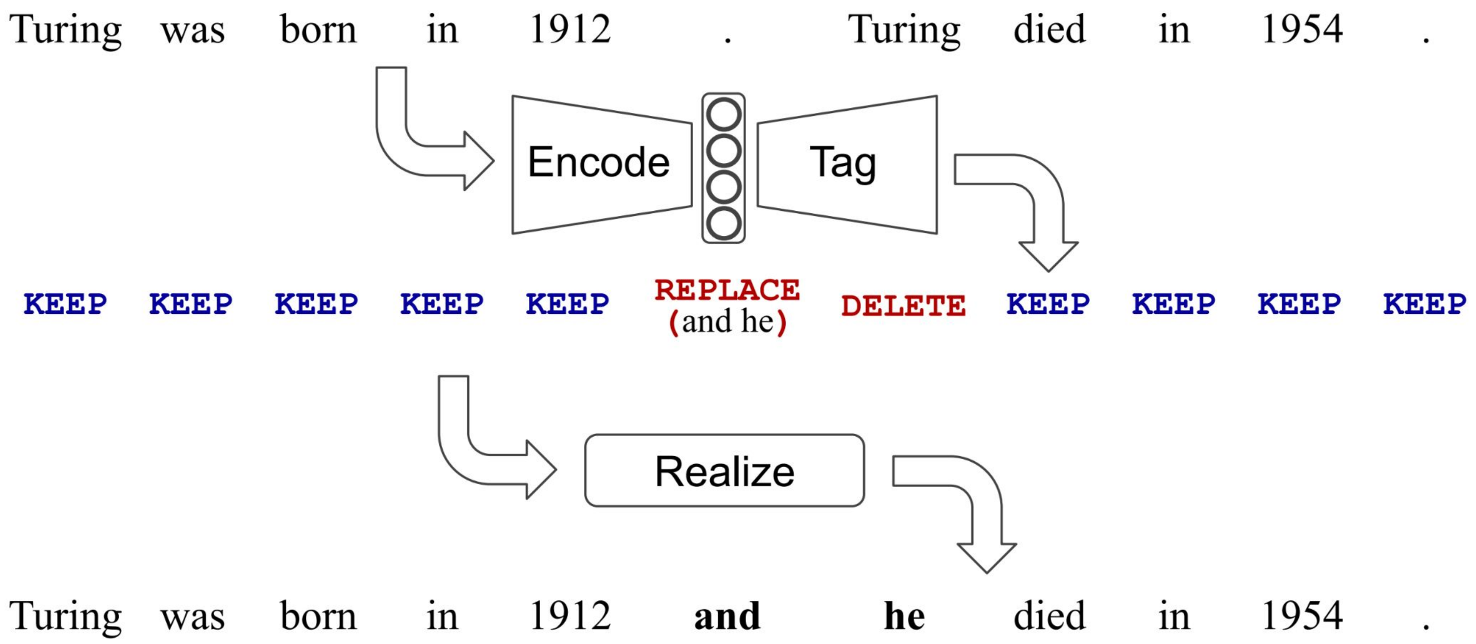
Observation: High overlap between the answers and the fusions.

Fusion requires mainly:

- Deleting repeated phrases
- Adding short glue phrases

Solution: Predict edit operations instead of generating from scratch.

LaserTagger



Malmi, Krause, Rothe, Mirylenka, Severyn. *Encode, Tag, Realize: High-Precision Text Editing*. EMNLP 2019 (pdf)

LaserTagger: Key Ingredients

- **Convert** training target texts into **target tag** sequences.
  - Tag = Base tag {KEEP, DELETE} + added phrase
  - Additionally: SWAP tag to reverse sentence order
- **Phrase vocabulary:** Set of phrases the model can add.
  - Counters hallucination
  - Optimized to cover as many training examples as possible
- **Tagging Model:** BERT (+ 1-layer Transformer decoder)

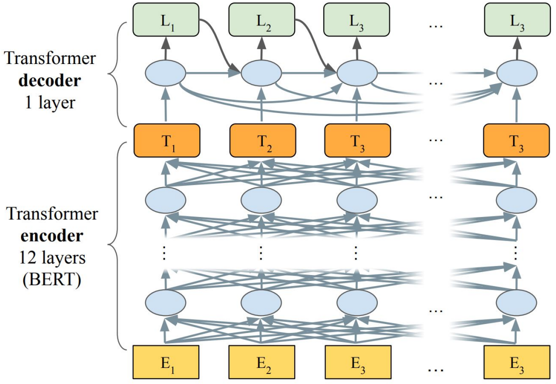


Figure 3: The architecture of LASERTAGGER$_{AR}$.

Source: Malmi et al. 2019 ([pdf](pdf)).

LaserTagger's Limitations

1. Realized text is sometimes unnatural since we only pretrain the encoder.
2. Limited phrase vocabulary can be too restrictive.
3. Reordering words is difficult.

Model landscape

Anatomy of a text-editing model

Encoder

- What edit operations to use?
- Tagging architecture?
- Auto-regressive vs. feed-forward?

Pointer

- How to reorder words?

Decoder

- How to insert words / phrases?

Anatomy of a text-editing model

Encoder

- What edit operations to use?
- Auto-regressive vs. feed-forward?
- Tagging architecture?

Pointer

- How to reorder words?

Decoder

- How to insert words / phrases?

Edit-operation types

Basic Edit-Operation Types

1. KEEP: Keeps the current token
2. DELETE: Deletes the current token
3. REPLACE: Replaces the current token
    a. REPLACE_X: Replace with a **specific token/phrase X** (e.g. LaserTagger, GECToR)
    b. REPLACE: Replace with a placeholder and use a **separate insertion component** to fill the blank (e.g. EditNTS, Felix, LEWIS)
4. APPEND / PREPEND: Inserts new token(s) next to the current token

REPLACE_X, APPEND_X, PREPEND_X

- Separate edit operation for each insertion $x \in X$ where X is a predefined set of possible insertions
  - REPLACE_the, REPLACE_a, etc.
- Pros
  - Counters hallucinations (more on this later)
- Cons
  - X can become very large when having to do multi-word insertions
  - Hard to leverage pre-trained LMs to determine a good insertion

DfWiki	WikiSplit	AS	GEC
,	.␣⟨::::⟩	,	,
and	,	.	.
however␣,	.␣⟨::::⟩␣he	the	the
,␣but	.␣⟨::::⟩␣it	a	a
he	the	&	to
because	and	and	in
,␣although	was	is	of
but	is	in	on
,␣and	"	"	at
although	.␣⟨::::⟩␣she	’s	for
his	.␣⟨::::⟩␣it␣is	with	have
,␣while	a	for	is
it	.␣⟨::::⟩␣they	of	was
,␣which	.␣⟨::::⟩␣however	n’t	and
she	he	an	that

Table 1: The 15 most frequently added phrases in the datasets studied in this work, in order of decreasing frequency. ⟨::::⟩ marks a sentence boundary. “AS”/“GEC” is short for Abstractive Summarization/Grammatical Error Correction.

Source: LaserTagger paper (Malmi et al. 2019).

Other Edit-Operation Types

- SWAP: Swap the order of this and the previous sentence

Source:	Dylan	won	Nobel	prize	.	Dylan	is	an	American	musician	.
Tags:	DELETE	KEEP	KEEP	KEEP	SWAP	KEEP	^{comma}DELETE	KEEP	KEEP	KEEP	^{comma}DELETE
Realization:	Dylan , an American musician , won Nobel prize .										

- PRONOMINALIZE: Replace this entity with a pronoun (look up gender from a knowledge base)
- NOUN_NUMBER_SINGULAR: Convert noun to singular form
    a. Other grammar-related edit operations discussed in the Applications section

Tagging architecture + auto-regressiveness

Types of Models

Two major types of models used for tagging

Autoregressive (AR)

- Condition on previous predictions
- Seq2Seq
- Slow*

Non-Autoregressive (NAR)

- Predict simultaneously
- Feedforward NN
- More prone to errors
- Fast*

Models

Method	Non-autoregressive
EdiT5 (Mallinson et al., 2022)	(✓)
EditNTS (Dong et al., 2019)	
Felix (Mallinson et al., 2020)	✓
GECToR (Omelianchuk et al., 2020)	✓
LaserTagger (Malmi et al., 2019)	✓
LevT (Gu et al., 2019)	(✓)
LEWIS (Reid and Zhong, 2021)	
Masker (Malmi et al., 2020)	✓
PIE (Awasthi et al., 2019)	✓
Seq2Edits (Stahlberg and Kumar, 2020)	
SL (Alva-Manchego et al., 2017)	✓

Case study: LaserTagger

LaserTagger supports **AR** and **NAR** version allowing for a direct comparisons

- Across **4** tasks AR outperforms NAR
  - Up to **7%** difference but as little as **1%**
- At a **40x** increase in latency (on GPU)
  - **13ms to 535ms**
- Trade off between **speed** and **performance**

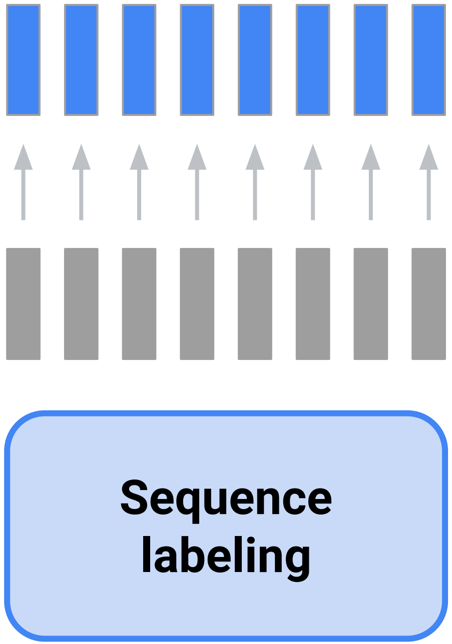
Non-AutoRegressive

Sequence labeling task

1. Encode source
2. For each token **predict a label**
   a. Maximize gold tag probability in training

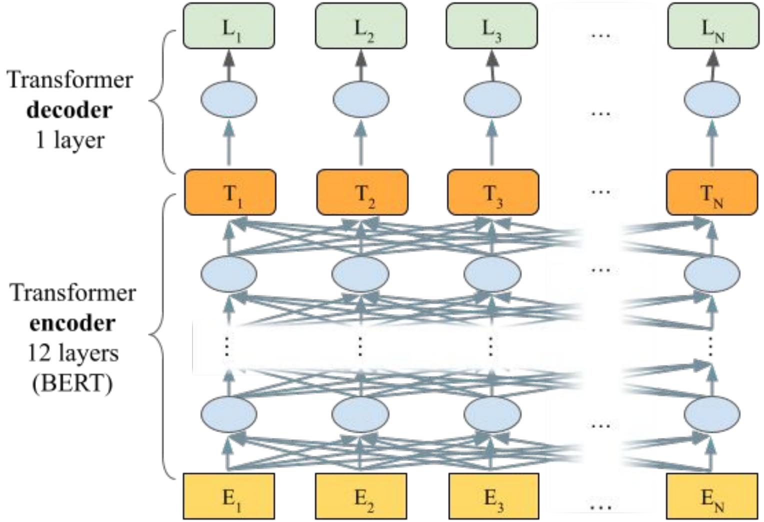
$$P(y|x) = \prod_{i}^{|y|} P(y_i|x)$$

   b. Argmax in prediction



Non-AutoRegressive

1. **Encode source sentence**
    a. **Pre-trained** NAR models
    b. **BERT**: Felix, LaserTagger, GECToR
    c. **XLNet** : GECToR
2. **Predict the tags**
    a. Single layer Feedforward
    b. Output size: **2 - 1000** tags
    c. Each hidden state gets a single tag



Difficult to generate arbitrary outputs

Source: Malmi et al. 2019 (pdf).

Non-AutoRegressive Agreement

NAR runs the risk of the edits not agreeing with each other

```
Source:          "We have an apples"
```

```
NAR Prediction:  "We have some apple"
```

```
AR Prediction:   "We have an apple"
                 "We have some apples"
```

NAR don't condition on past edits

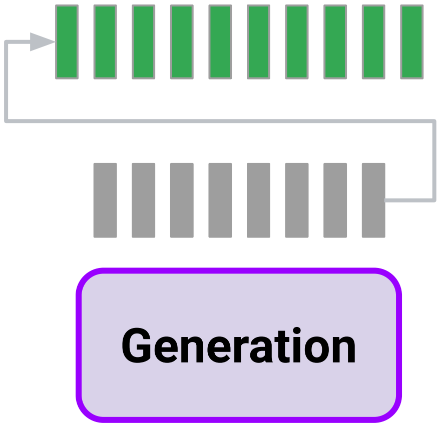
Agreement issues:

- Direction
- Grammar
- Subwords

NAR don't apply layers multiple times

Auto-Regressive

- Encode source
- Decode edit-by-edit
  - Condition on **previously decoded edit**
- RNN/Transformer
  - **Pre-trained AR**
  - **T5**: EdiT5
  - **BART**: LEWIS



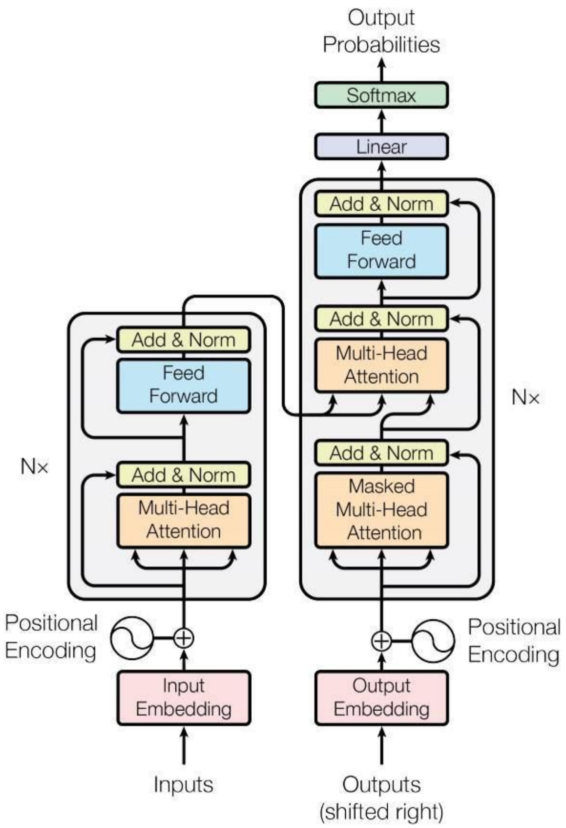
$$P(y|x) = \prod_{i}^{|y|} P(y_i|y_{<i}, x)$$

Auto-Regressive

Decoding is non-parallelizable

Each step:

1. Input previously predict token
2. Self-attention * Number of layers
3. Cross-attention * Number of layers
4. Argmax
    a. **For tagging this can be small**



Source: Vaswani, et al. 2017

Iterative Refinement

- Apply the model to its **own output**
  - Each iteration increases performance but adds latency
- Commonly used for GEC
- Can be used with any type of model
  - GECToR, PIE, Seq2Edits

Iteration #	**P**	**R**	**$F_{0.5}$**	**# corr.**
Iteration 1	72.3	38.6	61.5	787
Iteration 2	73.7	41.1	63.6	934
Iteration 3	74.0	41.5	64.0	956
Iteration 4	73.9	41.5	64.0	958

Table 4: Cumulative number of corrections and corresponding scores on CoNLL-2014 (test) w.r.t. number of iterations for our best single model.

[GECToR – Grammatical Error Correction: Tag, Not Rewrite (Omelianchuk et al., BEA 2020)]

Anatomy of a text-editing model

Encoder

- What edit operations to use?
- Auto-regressive vs. feed-forward?
- Tagging architecture?

Pointer

- How to reorder words?

Decoder

- How to insert words / phrases?

Reordering

Most text-editing apply their models left-to-right

Reordering allows us to model

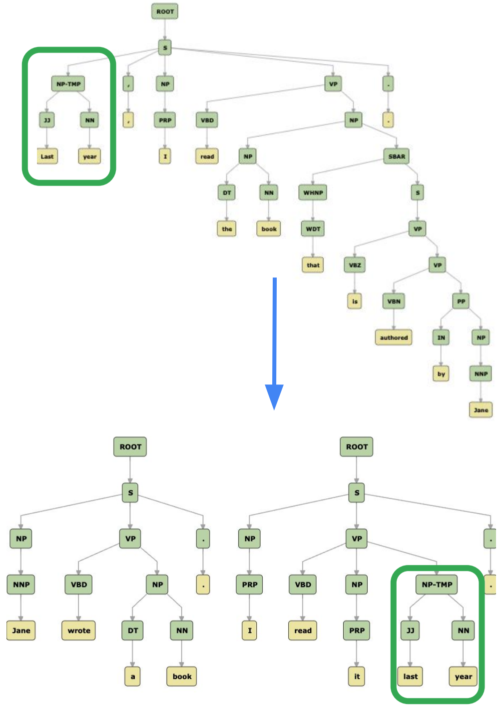
- Large syntactic changes
- Local changes

Without the need to **delete then insert**

Last year, I read the book that is authored by Jane.

[Original sentence]

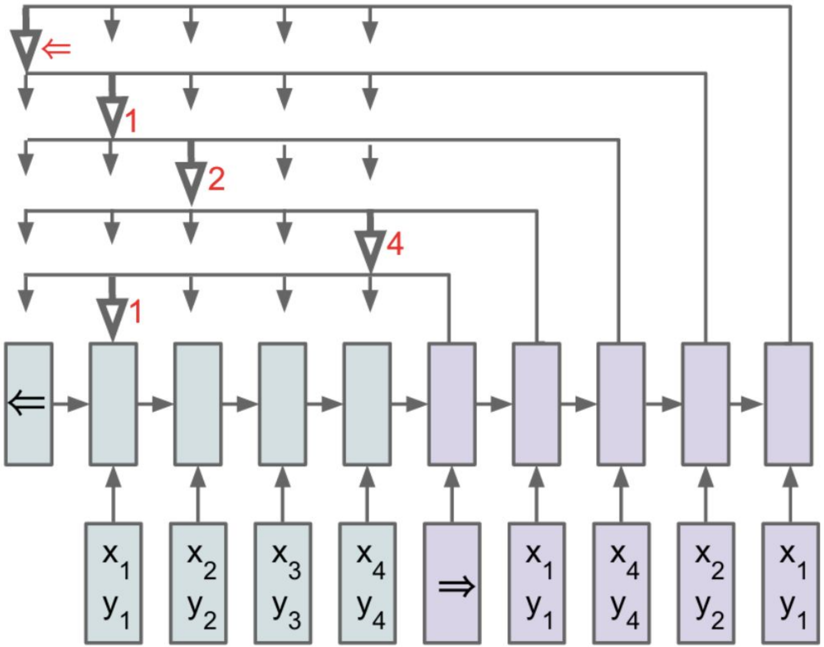
Jane wrote a book.
I read it **last year**.



AutoRegressive Reordering

Implemented using pointer network

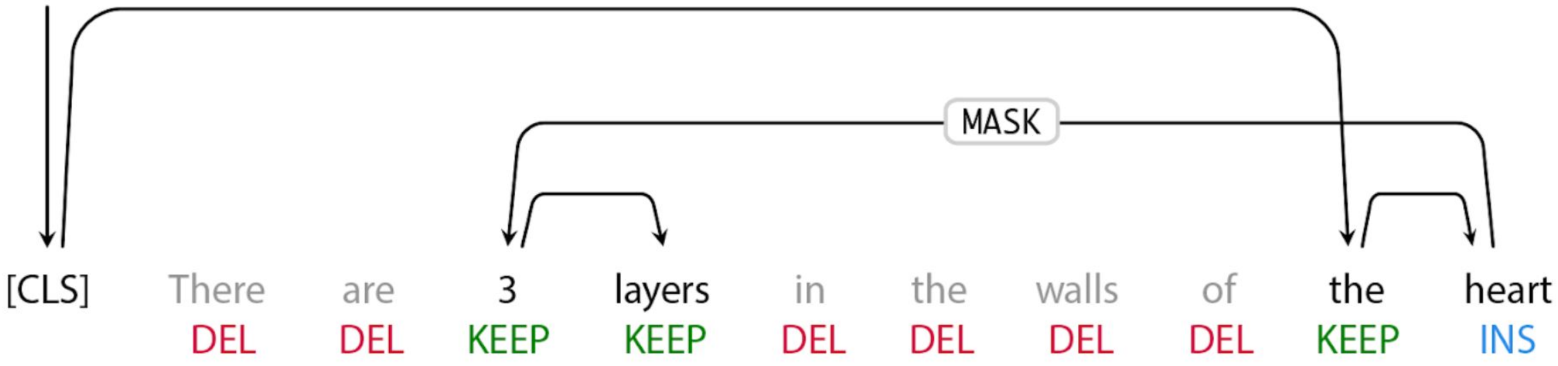
- When decoding use a **cross-attention**
- The source token with the highest attention is copied to the target
- Can copy the same source token multiple times



Source: Pointer networks paper (Vinyals et al., 2015).

Non-AutoRegressive Reordering

- **Self-attention** pointer network which are **daisy chained**
  - Attention between encoder hidden states
  - Felix & EdiT5
- Can only copy each source token once



Source: Felix paper (Mallinson et al. 2020).

Anatomy of a text-editing model

Encoder

- What edit operations to use?
- Auto-regressive vs. feed-forward?
- Tagging architecture?

Pointer

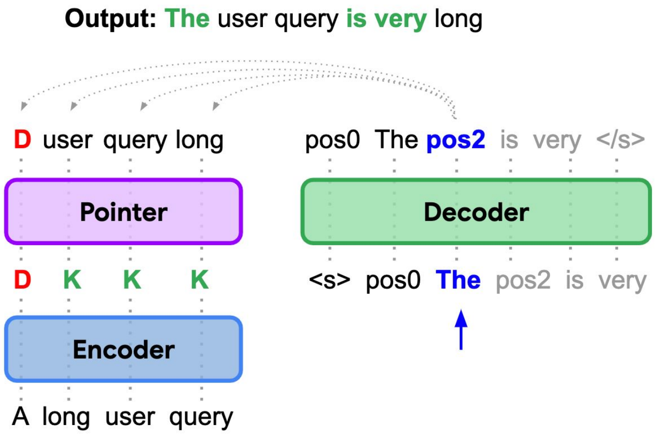
- How to reorder words?

Decoder

- How to insert words / phrases?

Separate Insertion Component

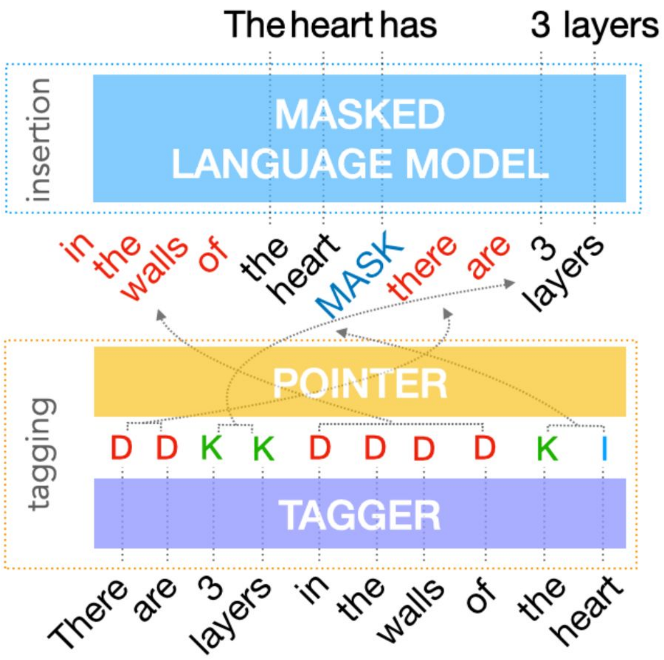
- Tagger predicts **where** to insert;
  a separate component **what** to insert
- Different insertion architectures
  - RNN (EditNTS)
  - BERT MLM (Felix, Masker)
  - Transformer decoder (Seq2Edits, EdiT5, LEWIS)



Source: EdiT5 paper (Mallinson et al. 2022).

Felix

- Idea 1: **Separate** insertion from tagging
  - Leverage pretrained BERT
- Idea 2: Predict word order using a **pointer network**



Source: Felix paper (Mallinson et al. 2020).

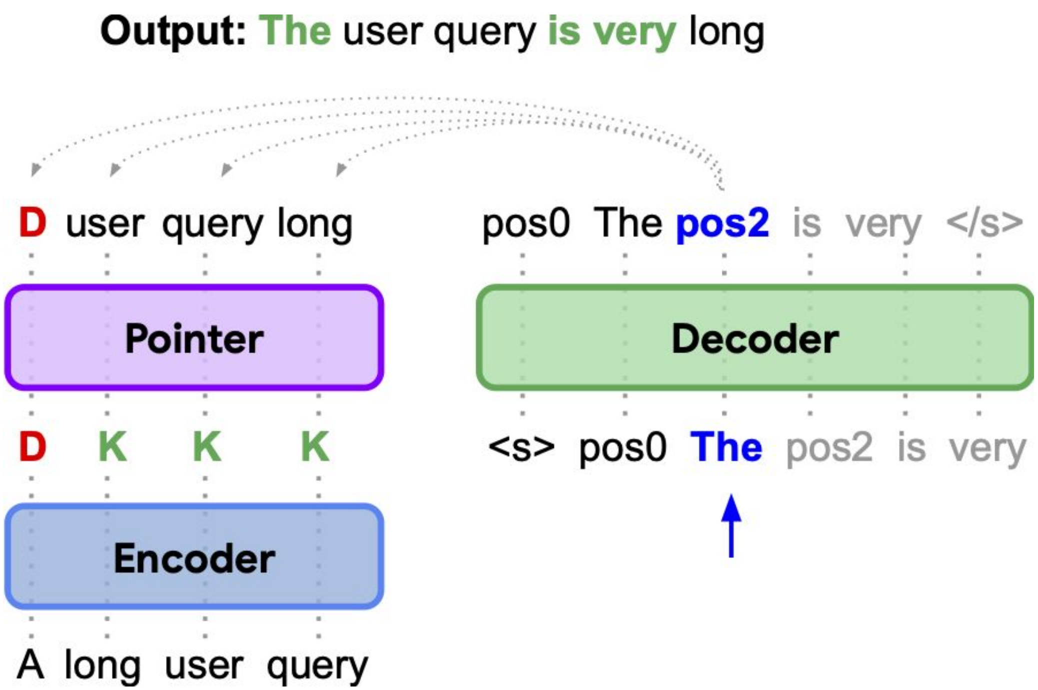
Insertion

- The output of the tagging model is the reordered input text with deleted words and MASK tokens
- The insertion model predict the content of MASK tokens
- Very similar to the **pretraining objective of BERT**

Source:	The	hearts	consist	of			layers
Tags:	DEL	KEEP	KEEP	$\text{KEEP}^{\text{INS\_2}}$			KEEP
Insertion input:	[R] The [/R]	hearts	consist	of	MASK	MASK	layers
Prediction:		hearts	consist	of	many	different	layers

Source: Felix paper (Mallinson et al. 2020).

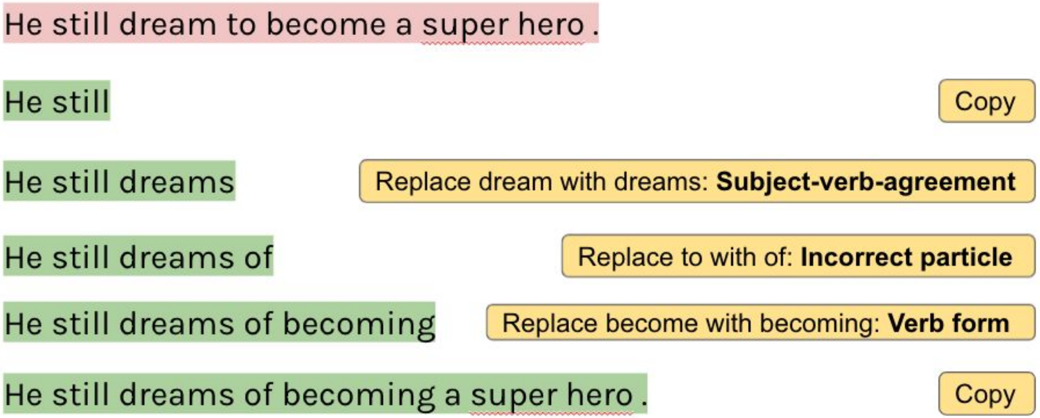
EdiT5



- **Idea 1:** Join insertion and tagging
  - Leverage **pretrained T5 models**
- **Idea 2:** Use autoregressive decoding on the small number of inserted tokens
- Decoder first predicts the location of the new text then decodes new tokens

Source: EdiT5 paper (Mallinson et al. 2022).

Seq2Edits: A model that can rewrite and explain



Source: Seq2Edits paper (Stahlberg and Kumar, 2020).

- Contains 3 sub-models for predicting tags, span-end positions and replacement tokens
- The model is able to provide explanations for each edit operation
- By avoiding unnecessary copying of input spans, it is up to 5 times faster than a regular seq2seq model

Method overview

Text-Editing Models

Method	Non-autoregressive	Pre-trained decoder	Reordering	Unsupervised	Language-agnostic	Application(s)
EdiT5 (Mallinson et al., 2022)	(✓)	✓	✓		✓	*multiple*
EditNTS (Dong et al., 2019)					✓	Simplification
Felix (Mallinson et al., 2020)	✓	✓	✓		✓	*multiple*
GECToR (Omelianchuk et al., 2020)	✓	(✓)				GEC
LaserTagger (Malmi et al., 2019)	✓				✓	*multiple*
LevT (Gu et al., 2019)	(✓)	✓			✓	*multiple*
LEWIS (Reid and Zhong, 2021)		✓		✓	✓	Style Transfer
Masker (Malmi et al., 2020)	✓	✓		✓	✓	*multiple*
PIE (Awasthi et al., 2019)	✓	✓				GEC
Seq2Edits (Stahlberg and Kumar, 2020)					(✓)	*multiple*
SL (Alva-Manchego et al., 2017)	✓		✓		✓	Simplification

Text-Editing Models (discussed so far)

Method	Non-autoregressive	Pre-trained decoder	Reordering	Unsupervised	Language-agnostic	Application(s)
EdiT5 (Mallinson et al., 2022)	(✓)	✓	✓		✓	*multiple*
EditNTS (Dong et al., 2019)					✓	Simplification
Felix (Mallinson et al., 2020)	✓	✓	✓		✓	*multiple*
GECToR (Omelianchuk et al., 2020)	✓	(✓)				GEC
LaserTagger (Malmi et al., 2019)	✓				✓	*multiple*
LevT (Gu et al., 2019)	(✓)	✓			✓	*multiple*
LEWIS (Reid and Zhong, 2021)		✓		✓	✓	Style Transfer
Masker (Malmi et al., 2020)	✓	✓		✓	✓	*multiple*
PIE (Awasthi et al., 2019)	✓	✓				GEC
Seq2Edits (Stahlberg and Kumar, 2020)					(✓)	*multiple*
SL (Alva-Manchego et al., 2017)	✓		✓		✓	Simplification

Text-Editing Models (discussed later)

Method	Non-autore-gressive	Pre-trained decoder	Reorde-ring	Unsuper-vised	Language-agnostic	Application(s)
EdiT5 (Mallinson et al., 2022)	(✓)	✓	✓		✓	*multiple*
EditNTS (Dong et al., 2019)					✓	Simplification
Felix (Mallinson et al., 2020)	✓	✓	✓		✓	*multiple*
GECToR (Omelianchuk et al., 2020)	✓	(✓)				GEC
LaserTagger (Malmi et al., 2019)	✓				✓	*multiple*
LevT (Gu et al., 2019)	(✓)	✓			✓	*multiple*
LEWIS (Reid and Zhong, 2021)		✓		✓	✓	Style Transfer
Masker (Malmi et al., 2020)	✓	✓		✓	✓	*multiple*
PIE (Awasthi et al., 2019)	✓	✓				GEC
Seq2Edits (Stahlberg and Kumar, 2020)					(✓)	*multiple*
SL (Alva-Manchego et al., 2017)	✓		✓		✓	Simplification

Converting target texts to target edits

Edit types

There are multiple different ways that one sentence can be edited into another:

- Edit operation types (insert, delete, replace, append, prepend, reorder…)
- Token-level vs. span-level edits
- Tagged vs. untagged edits
- Alignment algorithm

Example

Source: i like films when i was younger i watched on **TV**

Target: i like films i watched on **television** when i was younger

- We could delete everything and then use *insert* everything
- is TV a delete and insert after on? or singe a replace
- Do we want to reorder or delete everything after i like films?
- What should the i align to?

Questions?