CIS 6930 Special Topics in Large Language Models

Knowledge Augmented Language Model

Outline

- Introduction to Knowledge-augmented LM
- Methods to add knowledge into LM
  1. Add pretrained entity embeddings
  2. Modify the training data
- Evaluating knowledge in LM

Introduction to Knowledge-augmented LM

What does LM know?

- iPod Touch is produced by _____Apple_____.
- London Jazz Festival is located in _____London_____.
- Dani Alves plays with _____Santos_____ -> Barcelona.
- Carl III used to communicate in _____German_____ -> Swedish.
- Ravens can _____fly_____.

Predictions generally look reasonable, but are not always factually correct!

What does LM know?

- Observation: predictions generally make sense (e.g. the correct types), but are not all factually correct.

- Why might this happen?
  - Unseen facts: some facts may not have occurred in the training corpora at all
  - Rare facts: LM hasn't seen enough examples during training to memorize the fact
  - Model sensitivity: LM may have seen the fact during training, but it was phrased in a different way than how we are testing, so the LM is confused
    - Fails to answer "x was *created* in y" but correctly answers "x was *made* in y"

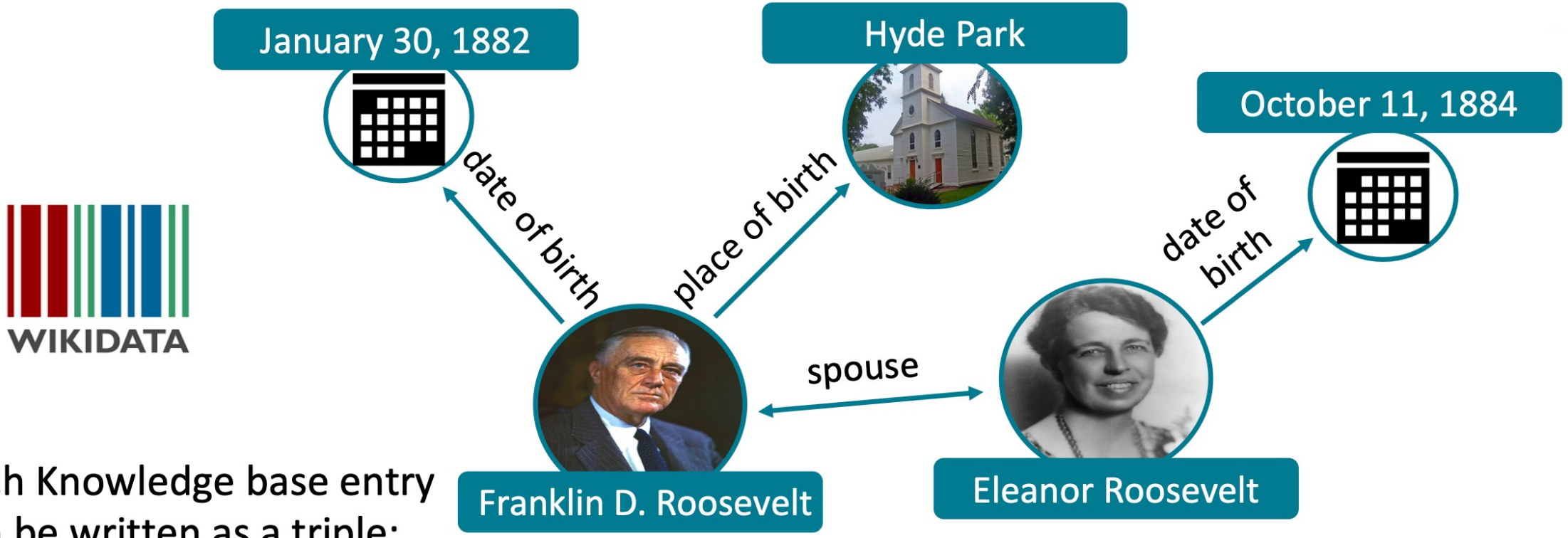
- Takeaway: LMs have *some* knowledge, but fail to *reliably* recall knowledge
  - We will talk about how to address this key challenge facing LMs!

Why build Knowledge-aware LM

- LM's pretrained representations can benefit downstream tasks that leverage knowledge
  - e.g. Question answering and relation exaction (extracting the relations between two entities in a sentence) are much easier with knowledge about the entities
  - We'll come back to this when we talk about evaluation!

- Stretch goal: can LMs ultimately replace traditional knowledge bases?
  - Instead of querying a knowledge base with formal query (e.g. SQL), query the LM with a natural language prompt!
    - Of course, this requires LM to have high quality on recalling facts, and this is an active area of research

Traditional Knowledge Base



WIKIDATA

- Each Knowledge base entry can be written as a triple:

  (parent entity, relation, tail entity), e.g. (“FDR”, “date of birth”, “Jan 30, 1882”)

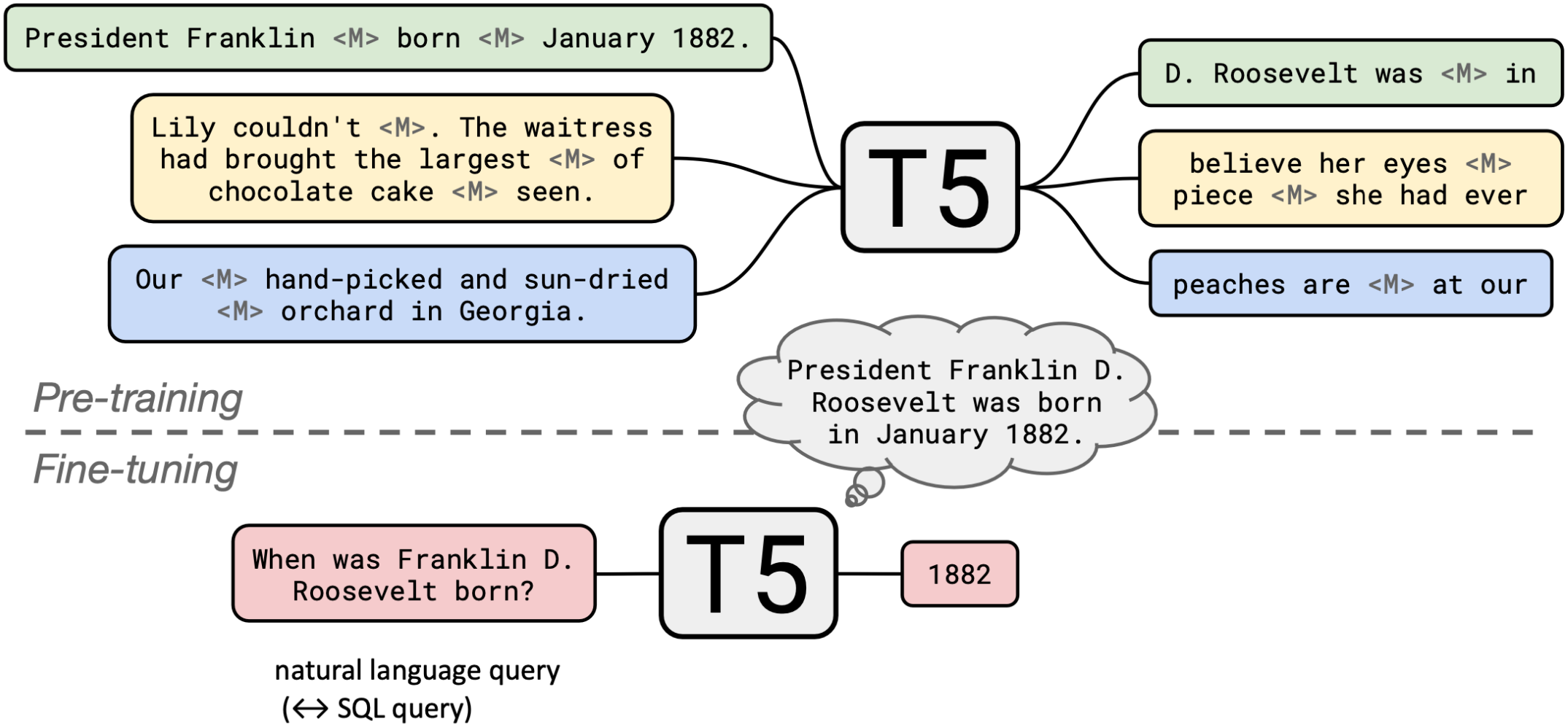
- You can query knowledge base with a formal query such as SQL statement:

  “What is the date of birth of FDR?”

```
SELECT date of birth
WHERE person = “Franklin D. Roosevelt”
```

Query Language Models as Knowledge Bases

- Pretrain LM over unstructured text and then query with natural language.



Advantage of using LM over traditional KB

- LMs can be pretrained over large amounts of unstructured and unlabeled text
  - ↔ KBs typically require manual annotation
- LMs support more flexible natural language queries
  - Example: *What does the final F in the song U.F.O.F. stand for?*
    - Traditional KB may not have a specific relation "final F"; LM *may* learn it implicitly
- However, there are also many open challenges to using LMs as KBs:
  - Hard to interpret (it's unclear why LM produces this answer ↔ KB has provenance)
  - Hard to trust (LM may produce a realistic but incorrect answer ↔ KB either returns the correct answer or returns no answer)
  - Hard to modify (hard to update knowledge in LM ↔ KB is directly editable)

=> Open up exciting opportunities for further research!

Methods to add knowledge into LM

Methods to add knowledge into LM

- Method 1: Add pretrained entity embeddings

ERNIE: Enhanced Language Representation with Informative Entities [Zhang et al., ACL 19]

GreaseLM: Reasoning with Language Model and Knowledge Graph [Zhang et al. ICLR 2022]

- Method 2: Modify the training data

WKLM [Xiong et al., ICLR 2020]

ERNIE1: Enhanced Representation through Knowledge Integration [Sun et al., 2019]

Method 1: Add pretrained entity embeddings

Method 1: Add pretrained entity embeddings

- Observation: Facts about the world are usually in terms of entities
  - Example: Washington was the first president of the United States.
- However, the typical word embeddings we use do **not** have a notion of entities
  - We use different word embeddings for "U.S.A.", "United States of America" and "America" even though they all refer to the same entity
- What if we assign a single embedding per entity?
  - Single entity embedding for "U.S.A.", "United States of America" and "America"
- **Goal**: Get pretrained entity embeddings that encode factual knowledge, and add to language model
- Note: To use entity embeddings for text, we need to do a task called entity linking

What is Entity Linking

- Link mentions in text to entities in a knowledge base

mention *mention*

Washington was the first president of the United States.

candidate *candidate* *candidate*

Q23 (Wikidata) Q1223 (Wikidata) Q30 (Wikidata)

- Entity linking involves resolving ambiguous mentions (e.g. using context)
- Takeaway: Entity linking tells us which entity embeddings are relevant to the text

Method 1: Add pretrained entity embeddings

Summary: Entity embeddings are like word embeddings, but for entities in a knowledge base!

George Washington = $\begin{pmatrix} 0.111 \\ -0.345 \\ 0.876 \\ -0.201 \end{pmatrix}$

Many techniques for training entity embeddings:

- Knowledge graph embedding methods (e.g., TransE)
- Word-entity co-occurrence methods (e.g., Wikipedia2Vec)
- Transformer encodings of entity descriptions (e.g., BLINK)

Any of those entity embeddings can be used for the knowledge integration methods we will talk about today

Method 1: Add pretrained entity embeddings

Question: How do we incorporate pretrained entity embeddings when they're from a *different embedding space* than the language model?

Answer: Learn a fusion layer *h* that combines word info (from LM) and entity info.

$$\boldsymbol{h}_j = F\left(\boldsymbol{W}_t \boldsymbol{w}_j + \boldsymbol{W}_e \boldsymbol{e}_k + \ b\right)$$

- $\boldsymbol{w}_j$ is the embedding of word j in a sequence of words
- $\boldsymbol{e}_k$ is the corresponding entity embedding

Intuition: there's alignment between entities and words in the sentence such that projections $\boldsymbol{W}_t \boldsymbol{w}_j$ and $\boldsymbol{W}_e \boldsymbol{e}_k$ are in the same vector space

Method 1: Add pretrained entity embeddings (ERNIE)

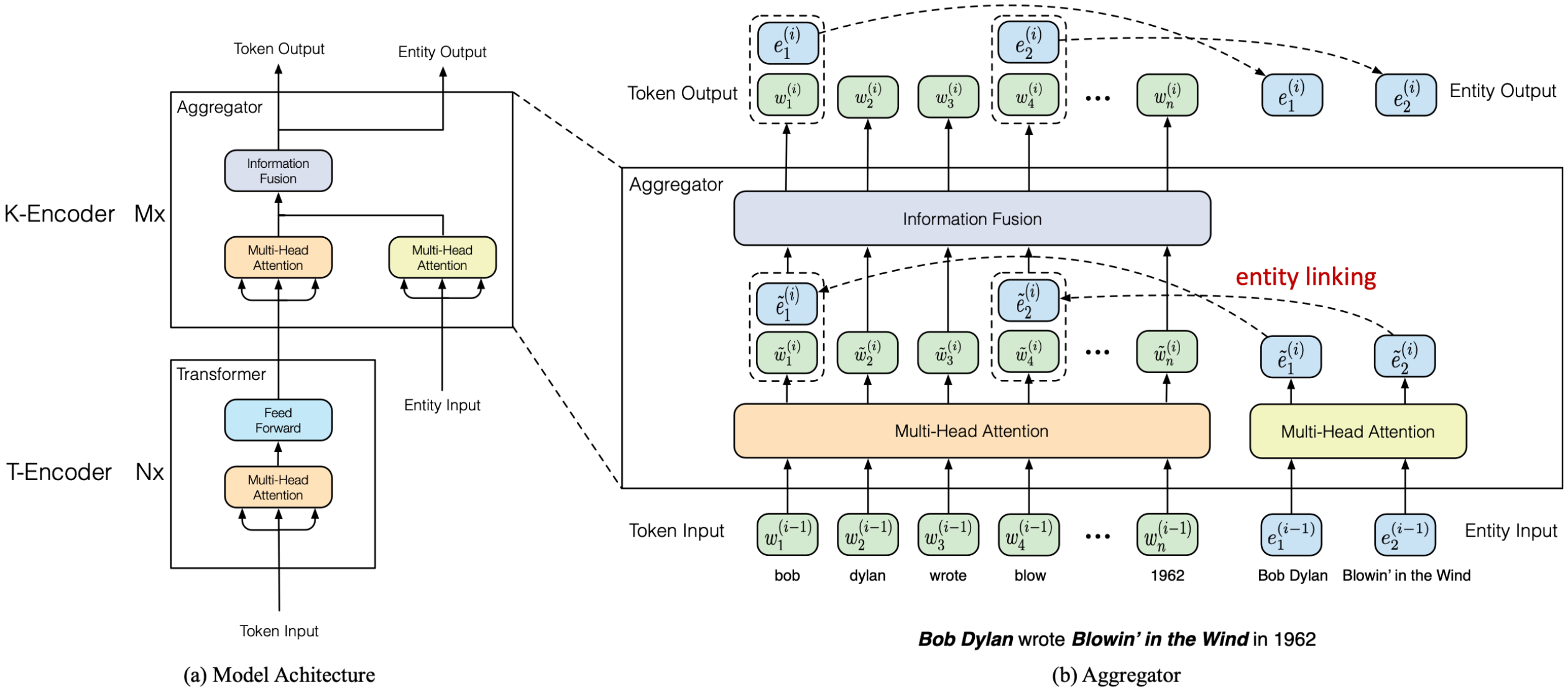
- Text encoder: multi-layer bidirectional Transformer encoder over the token in the sentence
- Knowledge encoder: each block is composed of:
  - Two self-attention layers – one for entity embeddings and one for token embeddings
  - A fusion layer to combine the output of the self-attention layers

$$\boldsymbol{h}_j = \sigma\left(\widetilde{\boldsymbol{W}}_t^{(i)}\widetilde{\boldsymbol{w}}_j^{(i)} + \widetilde{\boldsymbol{W}}_e^{(i)}\widetilde{\boldsymbol{e}}_k^{(i)} + \widetilde{\boldsymbol{b}}^{(i)}\right) \qquad \text{fusion representation}$$

$$\boldsymbol{w}_j^{(i)} = \sigma\left(\boldsymbol{W}_t^{(i)}\boldsymbol{h}_j + \boldsymbol{b}_t^{(i)}\right) \qquad \text{token embedding output (fed to next block)}$$

$$\boldsymbol{e}_k^{(i)} = \sigma\left(\boldsymbol{W}_e^{(i)}\boldsymbol{h}_j + \boldsymbol{b}_e^{(i)}\right) \qquad \text{entity embedding output (fed to next block)}$$

Method 1: Add pretrained entity embeddings (ERNIE)



Method 1: Add pretrained entity embeddings (ERNIE)

- How to train? Pretrain jointly with three tasks:
  - Masked language model and next sentence prediction (i.e., BERT tasks)
  - Knowledge pretraining task (dEA[1]): randomly mask some token-entity alignments and predict which entity in the sequence should be linked to the given token

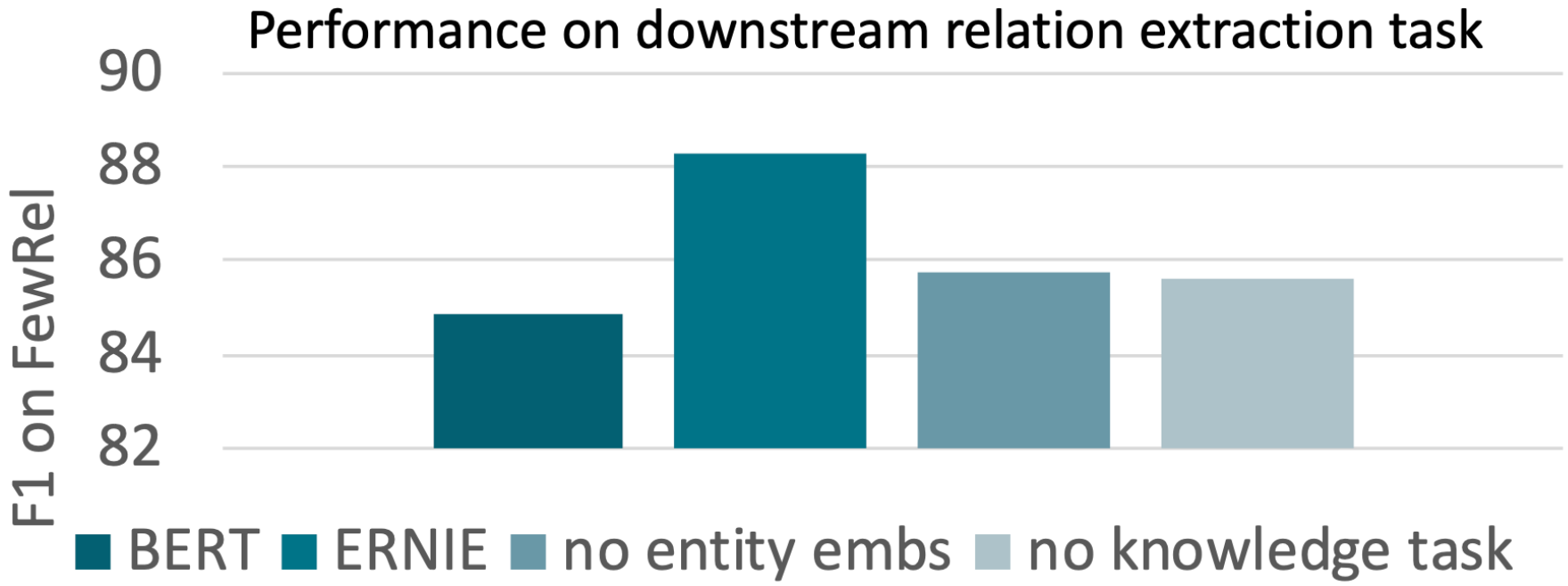
$$p(e_j \mid w_i) = \frac{\exp(\boldsymbol{W}\boldsymbol{w}_i \cdot \boldsymbol{e}_j)}{\sum_{k=1}^{m} \exp(\boldsymbol{W}\boldsymbol{w}_i \cdot \boldsymbol{e}_k)}$$

    - Motivations: better learn word-entity alignments; and prevent overfitting to pre-given (ground-truth) entity linking inputs
- Final objective:

$$\mathcal{L}_{ERNIE} = \mathcal{L}_{MLM} + \mathcal{L}_{NSP} + \mathcal{L}_{dEA}$$

Method 1: Add pretrained entity embeddings (ERNIE)

- Analysis to see the effect of model components (entity embs and knowledge task)
  - Knowledge pretraining task is necessary to make the most use of the pretrained entity embeddings.



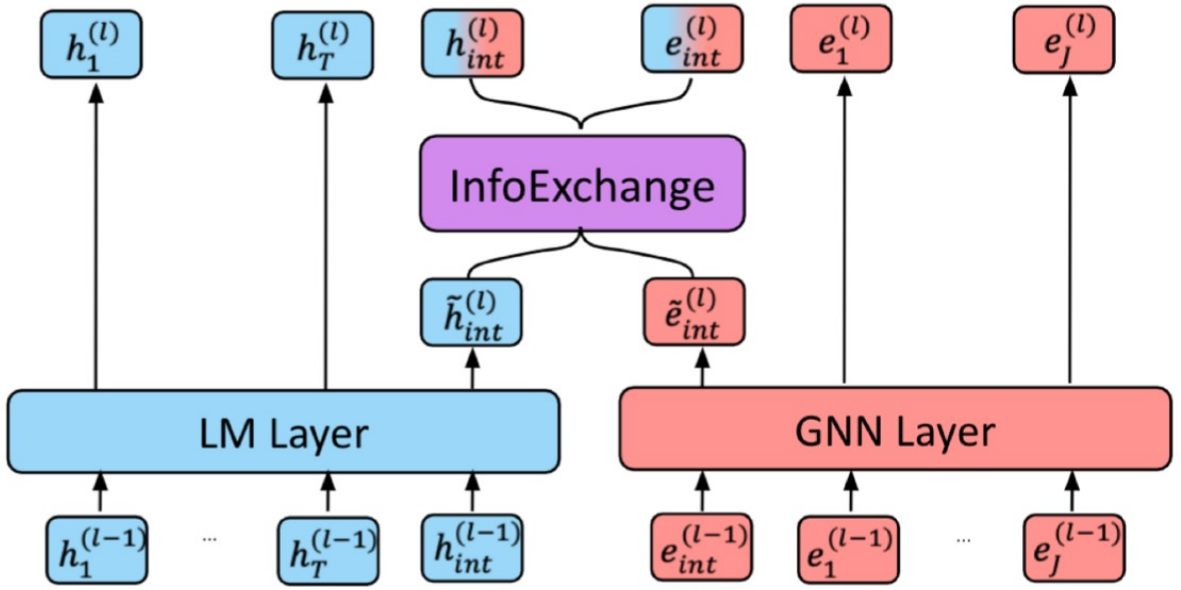
Method 1: Add pretrained entity embeddings (ERNIE)

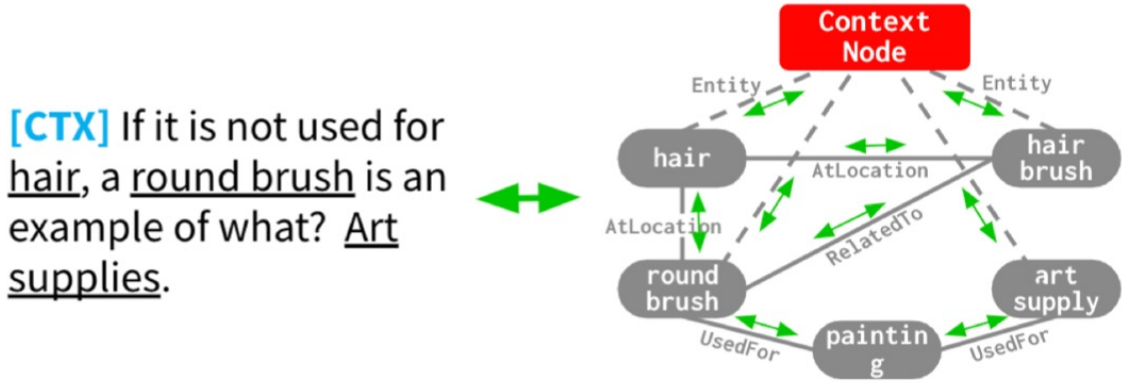
- Strengths:
  - Combines entity + text info through fusion layers and a knowledge pretraining task
  - Improves performance on knowledge-intensive downstream tasks

- Limitation:
  - Needs to link each entity mention in input text to knowledge base in advance
    - For instance, "Bob Dylan wrote Blowin' in the Wind" needs entities linked to Wikidata knowledge base
    - It's challenging to get a good entity linker for any domain of text or tasks
    - We will next talk about a more recent method that mitigates this issue

Method 1: Add pretrained entity embeddings (GreaseLM)

- **Model architecture**:
  Text is encoded by a language model, knowledge graph (KG) is encoded by a graph neural network (GNN), and they are fused together for multiple rounds.





Text

Local KG

Method 2: Modify the training data

Method 2: Modify the training data

- Previous methods incorporated knowledge explicitly through pretrained embeddings and/or an external memory.

- Question: Can knowledge also be incorporated implicitly through the unstructured text?
- Answer: Yes! Mask or corrupt the data to introduce additional training tasks that require factual knowledge.

- Advantages:
  - No need for additional memory/computation (e.g. no need to carry a local KG)
  - No need for modifying the architecture (e.g. no need for a fusion layer)

Method 2: Modify the training data (WKLM)

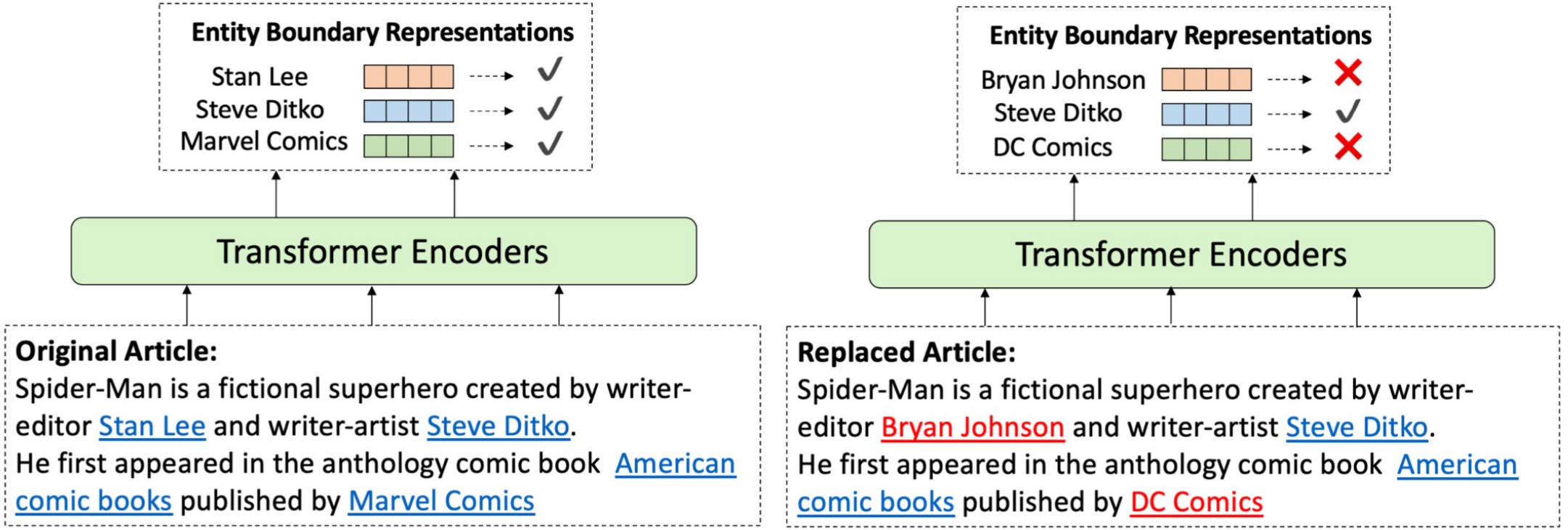
- Key idea: train the model to distinguish between true and false knowledge
- Method: Replace mentions in the text with mentions that refer to different entities of the same type to create negative knowledge statements
  - Make the model predicts whether entity has been replaced or not
  - Need type-constraint to enforce linguistically correct replacement. Otherwise the model may trivially predict "replaced" using linguistic signal instead of knowledge

True knowledge statement: J.K. Rowling is the author of Harry Potter.

Negative knowledge statement: J.R.R. Tolkien is the author of Harry Potter.

=> Requires the model to have background knowledge to be able to distinguish!

Method 2: Modify the training data (WKLM)



Entity Replacement Procedure

Marvel Comics —*entity linking*→ Q173496 —*type lookup*→ Q1320047

book publishing company

WIKIPEDIA
The Free Encyclopedia

WIKIDATA

Entities clustered by type Q1320047

DC Comics
Dark Horse Comics
Image Comics
....

random sample → *DC Comics*

Method 2: Modify the training data (WKLM)

- **Training**: Uses an entity replacement loss (binary classification) to train the model to distinguish between true and false mentions

$$\mathcal{L}_{entRep} = \mathbb{I}_{e \in \mathcal{E}^+} \log P(e \mid C) + (1 - \mathbb{I}_{e \in \mathcal{E}^+}) \log(1 - P(e \mid C))$$

where e is an entity, C is the context, and $\mathcal{E}^+$ represents a true entity mention

- Total loss is the combination of standard masked language model loss (MLM) and the entity replacement loss.

$$\mathcal{L}_{WKLM} = \mathcal{L}_{MLM} + \mathcal{L}_{entRep}$$

- MLM is defined at the token-level; entRep is defined at the entity-level
  - Treating a whole entity (could be multi-word) instead of a token as one unit can make LMs more knowledge-aware

Method 2: Modify the training data (WKLM)

- Improves over BERT and GPT-2 in fact completion tasks
- Improves over ERNIE on downstream tasks
- Ablation experiments (see the effect of model components, MLM and EntRep)
  - EntRep loss is essential because it makes WKLM outperform BERT
  - MLM loss is also essential for downstream task performance
  - On knowledge-intensive tasks, WKLM even outperforms training BERT longer with just MLM loss

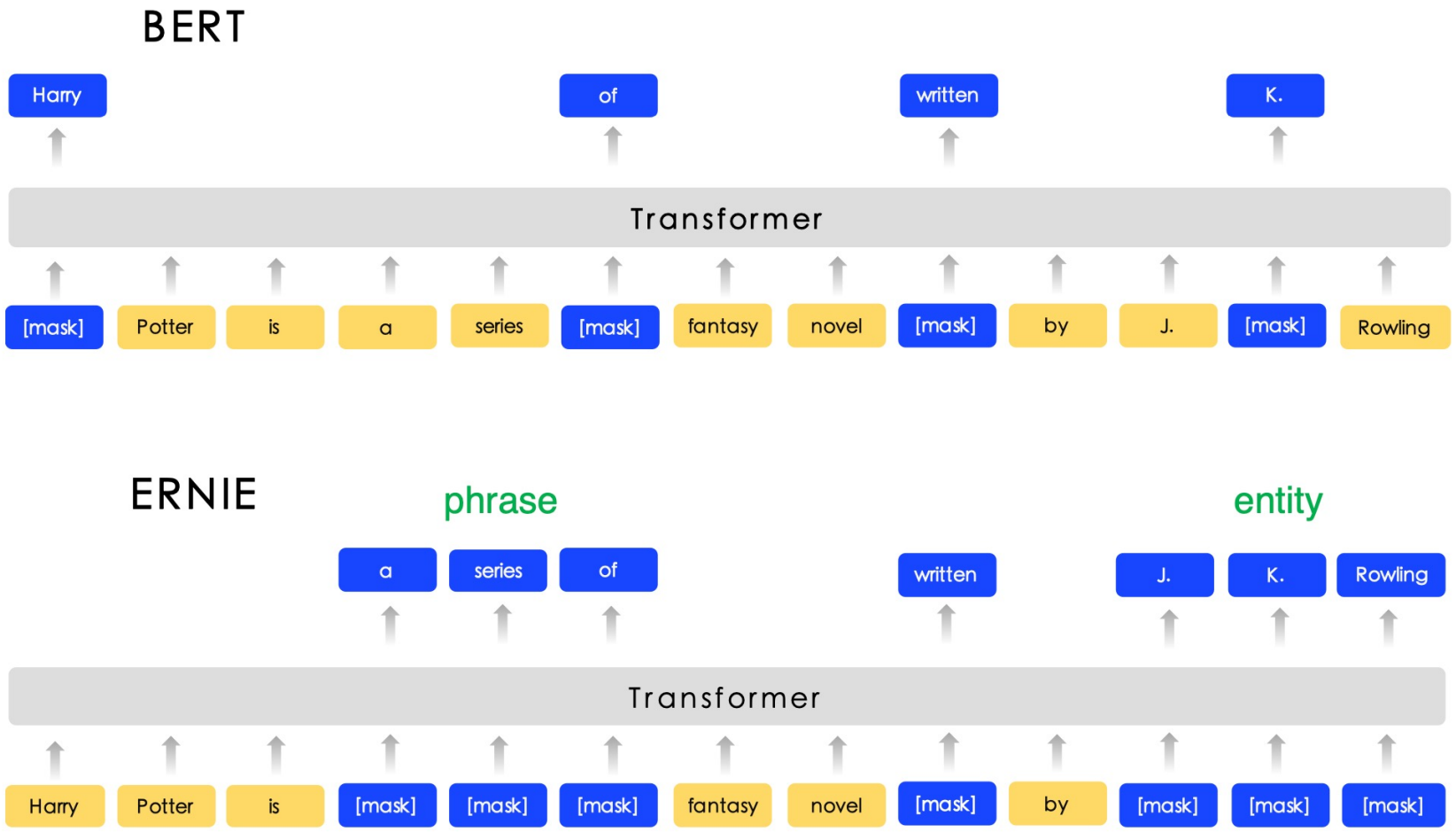
Model	SQuAD (F1)	TriviaQA (F1)	Quasar-T (F1)	FIGER (acc)
WKLM	91.3	56.7	49.9	60.21
WKLM w/o MLM	87.6	52.5	48.1	58.44
BERT + 1M Updates	91.1	56.3	48.2	54.17

Much worse without MLM

Much worse training for longer, compared to using the entity replacement loss

Method 2: Modify the training data (ERNIE1)

- Uses phrase-level and entity-level masking, and shows improvements on downstream NLP tasks



Evaluation

Knowledge-intensive downstream tasks

- Measures how well the knowledge-enhanced LM transfers its knowledge to downstream tasks
- Unlike probes, this evaluation usually involves finetuning the LM on downstream tasks, like evaluating BERT on GLUE tasks

- Common knowledge-intensive tasks:
  - Relation extraction
    - Example: *[Bill Gates] was born in [Seattle]; label: "city of birth"*
  - Entity typing
    - Example: *[Alice]* has donated billions to eradicate malaria*; label: "philanthropist"*
  - Question answering
    - Example: *"What kind of forest is the Amazon?"; label: "moist broadleaf forest"*

Relation Extraction Performance on TACRED

- Knowledge-enhanced systems (ERNIE, KnowBERT) improve over previously state-of-the-art models for relation extraction

Model	LM	Precision	Recall	F1
C-GCN	-	69.9	63.3	66.4
BERT-LSTM-base	BERT-Base	73.3	63.1	67.8
ERNIE (Zhang et al.)	BERT-Base	70.0	66.1	68.0
KnowBert-W+W	BERT-Base	**71.6**	**71.4**	**71.5**

Entity Typing Performance on OpenEntity

- Knowledge-enhanced LMs (ERNIE, KnowBERT) improve over prior LSTM and BERT-Base models on entity typing
- Impressively, previous models (NFGEC, UFET) were designed for entity typing

Model	Precision	Recall	F1
NFGEC (LSTM)	68.8	53.3	60.1
UFET (LSTM)	77.4	60.6	68.0
BERT-Base	76.4	71.0	73.6
ERNIE (Zhang et al.)	78.4	72.9	75.6
KnowBert-W+W	**78.6**	**73.7**	**76.1**

Knowledge-intensive Question Answering

- Knowledge-enhanced LMs (QAGNN, GreaseLM) improve over previous BERT-based models on question answering

Model	CommonsenseQA	OpenBookQA	MedQA
BERT-Large	55.4	60.4	-
RoBERTa-Large	68.7	64.8	35.0
SapBERT-Base	-	-	37.2
QAGNN	73.4	**67.8**	38.0
GreaseLM	**74.2**	66.9	**38.5**


Thank you!
UF
Herbert Wertheim
College of Engineering
UNIVERSITY of FLORIDA
LEADING THE CHARGE, CHARGING AHEAD