Natural Language Processing with Deep Learning
CS224N/Ling284

Michihiro Yasunaga
Lecture 13: Integrating Knowledge in Language Models

Slides coauthored with Megan Leszczynski
Lecture Plan (Integrating Knowledge in Language Models)

1. Recap of language models (LMs) and what do they know? [15 mins]
2. Techniques to add knowledge to LMs
   1. Add pretrained entity embeddings [20 mins]
   2. Use an external memory [15 mins]
   3. Modify the training data [15 mins]
3. Evaluating knowledge in LMs [20 mins]

Announcements:
• Project Milestone out today; due next Thursday
• Project Proposal feedback available on GradeScope by Thursday
• Hope your final projects are going well – stop by office hours for any questions or help!
Recap: LMs

- **Standard language models** predict the next word given a sequence of text and compute the probability of a sequence.

  \[ \text{The students opened their } \text{books}. \]

- **Masked language models** (e.g., BERT) predict a masked token in a sequence of text using bidirectional context.

  \[ \text{went } \quad \text{store} \]
  \[ I \ [MASK] \ to \ the \ [MASK]. \]

- In both cases, language models can be trained over large amounts of unlabeled text, without any human annotation.
Recap: LMs

• LMs are used for many tasks involving generating or evaluating the probability of text:
  • Summarization
  • Dialogue
  • Autocompletion
  • Machine translation
  • Fluency evaluation

• Recently, LMs are also commonly used to generate pretrained representations of text that encode some notion of language understanding for downstream NLP tasks
  • Text classification
  • Question answering

• Today: If an LM is trained over large amounts of text, can it even be used as a knowledge base?
What does a language model already know?

• iPod Touch is produced by _________.

• London Jazz Festival is located in _________.

• Dani Alves plays with _________.

• Carl III used to communicate in _________.

• Ravens can _________.

Examples taken from Petroni et al., EMNLP 2019
Check out what BERT-Large predicts
What does a language model already know?

- iPod Touch is produced by **Apple**.

- London Jazz Festival is located in **London**.

- Dani Alves plays with **Santos** -> **Barcelona**. Predictions generally look reasonable, but are not always factually correct!

- Carl III used to communicate in **German**. -> **Swedish**

- Ravens can **fly**.

Examples taken from **Petroni et al., EMNLP 2019**

Check out what BERT-Large predicts
What does a language model 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 do we want to build knowledge-aware language models?

- 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 bases and how to query them

• 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"
How to query language models as knowledge bases

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

Roberts et al., EMNLP 2020
Advantages of using language models over traditional KBs

• 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!
Section 2: Techniques to add knowledge to LMs
Techniques to add knowledge to LMs

Add pretrained entity embeddings
- ERNIE
- QAGNN/GreaseLM

Use an external memory
- KGLM

Modify the training data
- WKLM
- ERNIE (another!), salient span masking
Techniques to add knowledge to LMs

Add pretrained entity embeddings

- ERNIE
- QAGNN/GreaseLM

Use an external memory

- KGLM

Modify the training data

- WKLM
- ERNIE (another!), salient span masking
Method 1: Add pretrained 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*
Aside: What is entity linking?

• Link mentions in text to entities in a knowledge base

Washington was the first president of the United States.

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

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

More resources: Orr et al., CIDR 2021 & Li et al., EMNLP 2020
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

Bordes et al., NeurIPS 2013 & Yamada et al., 2020 & Wu et al., EMNLP 2020
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.

$$h_j = F(W_tw_j + W_e e_k + b)$$

- $w_j$ is the embedding of word $j$ in a sequence of words
- $e_k$ is the corresponding entity embedding

Intuition: there’s alignment between entities and words in the sentence such that projections $W_tw_j$ and $W_e e_k$ are in the same vector space
ERNI: Enhanced Language Representation with Informative Entities
[Zhang et al., ACL 2019]

- **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

\[
\begin{align*}
  h_j &= \sigma \left( \mathbf{W}_t (i) \mathbf{w}_j + \mathbf{W}_e (i) \mathbf{e}_k + \mathbf{b} (i) \right) \quad \text{fusion representation} \\
  w_j^{(i)} &= \sigma \left( \mathbf{W}_t (i) h_j + \mathbf{b}_t^{(i)} \right) \quad \text{token embedding output (fed to next block)} \\
  e_k^{(i)} &= \sigma \left( \mathbf{W}_e (i) h_j + \mathbf{b}_e^{(i)} \right) \quad \text{entity embedding output (fed to next block)}
\end{align*}
\]
Bob Dylan wrote 'Blowin’ in the Wind' in 1962.
How to train? Pretrain jointly with three tasks:

- **Masked language model and next sentence prediction** (i.e., BERT tasks)
- **Knowledge pretraining task (dEA\textsuperscript{1})**: randomly mask some token-entity alignments and predict which entity in the sequence should be linked to the given token

\[
p(e_j | w_i) = \frac{\exp(W\mathbf{w}_i \cdot e_j)}{\sum_{k=1}^{m} \exp(W\mathbf{w}_i \cdot e_k)}
\]

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

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

**ERNIE: Enhanced Language Representation with Informative Entities**  
[Zhang et al., ACL 2019]

- 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.

![Performance on downstream relation extraction task](chart.png)
ERNIE: Enhanced Language Representation with Informative Entities
[Zhang et al., ACL 2019]

• 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
**QAGNN/GreaseLM: Reasoning with Language Model and Knowledge Graph**

[Yasunaga et al. NAACL 2021; Zhang et al. ICLR 2022]

- **Key idea**: when adding entity embeddings to language model, dynamically update them together with neighbor or related entities in knowledge graph as well as text

  ![Diagram of entity relationships]

  **CTX**: If it is not used for hair, a round brush is an example of what? Art supplies.

- **Benefits**
  - **Robust** to non-perfect entity linking: can include all entity candidates and let the model figure out what to fuse
  - **Better contextualize knowledge**: helpful for joint reasoning about text and knowledge (e.g. question answering tasks)

- Get all entity candidates and their neighbors in KG to prepare a local KG
QAGNN/GreaseLM: Reasoning with Language Model and Knowledge Graph
[Yasunaga et al. NAACL 2021; Zhang et al. ICLR 2022]

- **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.

- **What is GNN?**
  - Neural network designed for encoding graph data.
  - GNN updates each node representation by aggregating message vectors from neighbor nodes.
  - Check out [CS224W](#) for more about GNN!

\[
\begin{align*}
  a_v^{(k)} &= \text{AGGREGATE}^{(k)} \left( \{ h_u^{(k-1)} : u \in \mathcal{N}(v) \} \right) \\
  h_v^{(k)} &= \text{COMBINE}^{(k)} \left( h_v^{(k-1)}, a_v^{(k)} \right)
\end{align*}
\]
QAGNN/GreaseLM: Reasoning with Language Model and Knowledge Graph
[Yasunaga et al. NAACL 2021; Zhang et al. ICLR 2022]

- **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.

[CTX] If it is not used for hair, a round brush is an example of what? **Art supplies.**
QAGNN/GreaseLM: Reasoning with Language Model and Knowledge Graph
[Yasunaga et al. NAACL 2021; Zhang et al. ICLR 2022]

- **Quantitative result:** QAGNN and GreaseLM outperform previous BERT-based models on knowledge-intensive question answering tasks

<table>
<thead>
<tr>
<th>Model</th>
<th>CommonsenseQA</th>
<th>OpenBookQA</th>
<th>MedQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-Large</td>
<td>55.4</td>
<td>60.4</td>
<td>-</td>
</tr>
<tr>
<td>RoBERTa-Large</td>
<td>68.7</td>
<td>64.8</td>
<td>35.0</td>
</tr>
<tr>
<td>SapBERT-Base</td>
<td>-</td>
<td>-</td>
<td>37.2</td>
</tr>
<tr>
<td>QAGNN</td>
<td>73.4</td>
<td>67.8</td>
<td>38.0</td>
</tr>
<tr>
<td>GreaseLM</td>
<td><strong>74.2</strong></td>
<td>66.9</td>
<td><strong>38.5</strong></td>
</tr>
</tbody>
</table>

Devlin et al., NAACL 2019 & Liu et al., 2019 & Liu et al., NAACL 2021 & Yasunaga et al., NAACL 2021 & Zhang et al., ICLR 2022
QAGNN/GreaseLM: Reasoning with Language Model and Knowledge Graph

[Yasunaga et al. NAACL 2021; Zhang et al. ICLR 2022]

- **Qualitative example:**
  By grounding language model to knowledge graph, models learn to perform *structured reasoning* (e.g. handling negation correctly)

- Vanilla LMs don’t handle negation well.. [Kassner et al., ACL 2020]

- New insight on how KG can help LM
  - Provide background knowledge
  - Provide *scaffold* for reasoning

---

If it is **not** used for *hair*, a *round brush* is an example of what?

A. *hair brush*  B. *art supplies*

---

QAGNN/GreaseLM:

After several layers of fusion, attention weight from text over *hair* decreases, but attention weight over *round brush* and *painting* increases, adjusting for the negation in text
Techniques to add knowledge to LMs

Add pretrained entity embeddings
- ERNIE
- QAGNN/GreaseLM

Use an external memory
- KGLM

Modify the training data
- WKLM
- ERNIE (another!), salient span masking
Method 2: Use an external memory

- Previous methods rely on the pretrained entity embeddings to encode the factual knowledge into the language model.
  - Pros: Convenient, as you can just plug in any available entity embeddings
  - Cons: if the KB is modified, you may need to re-train the entity embeddings and model

- Question: Are there more direct ways to provide factual knowledge for LM?
  - Answer: Yes! Give the model access to an external memory (a key-value store for KG triples or facts) in a way that is independent of learned model parameters

- Advantages:
  - Can directly update facts in the external memory without re-training the model
  - Interpretable
    - It’s more visible which fact in external memory the LM used to make prediction (↔ it’s harder to debug model predictions if we use entity embeddings)
Barack's Wife Hillary: Using Knowledge-Graphs for Fact-Aware Language Modeling (KGLM) [Logan et al., ACL 2019]

• Key idea: condition the language model on a knowledge graph (KG) when predicting next word

• Recall that (standard) language models predict the next word given previous words:

\[ P(x^{(t+1)}|x^{(t)}, ..., x^{(1)}) \text{, where } x^{(1)}, ..., x^{(t)} \text{ is a sequence of words} \]

• Goal: predict the next word and entity using both the previous word and entity info

\[ P(x^{(t+1)}, \mathcal{E}^{(t+1)}|x^{(t)}, ..., x^{(1)}, \mathcal{E}^{(t)}, ..., \mathcal{E}^{(1)}) \]

where \( \mathcal{E}^{(t)} \) is the set of KG entities mentioned at timestep \( t \)
**KGLM [Logan et al., ACL 2019]**

- Method: Build a **local knowledge graph** as you iterate over the sequence
  - Local KG is a subset of the full KG with only entities relevant to the sequence so far

*Super Mario Land* is a game developed by **Nintendo**.

Assumes entities are known during training!

- Local KG can provide a strong signal for predicting what comes as the next word
- How can the LM know when to use the local KG vs standard LM to predict the next word?
Super Mario Land is a game developed by Nintendo.

Classify: Is the next word...
1. Related entity (in the local KG)
2. New entity (not in the local KG)
3. Not an entity

- Instead of predicting next word directly, use the LM hidden state to first predict the type of the next word (3 classes)
- Once we predict the word type, how to predict the next entity and word in each of the 3 scenarios?
Super Mario Land is a game developed by Nintendo.

1. Related entity case (in the local KG)

Example
Top scoring parent entity: “Super Mario Land”
Top scoring relation: “publisher”
-> Next entity is “Nintendo”, due to KG triple (Super Mario Land, publisher, Nintendo).
Super Mario Land is a game developed by Nintendo.

1. Related entity case (in the local KG)

- Find the top-scoring parent and relation in the local KG using the LM hidden state and entity and relation embeddings

  $P(p_t) = \text{softmax}(v_p \cdot h_t)$, where $p_t$ is a potential parent entity, $v_p$ is the corresponding entity embedding, and $h_t$ is from the LM hidden state

- Similarly for predicting top relation

- **Next entity will be:** tail entity from KG triple (top parent entity, top relation, tail entity)

- **Next word will be:** most likely next token over the standard vocabulary expanded to include the tail entity and its aliases

[1] Phrases that could all refer to Nintendo (e.g. Nintendo, Nintendo Co., Koppai)
**KGLM** [Logan et al., ACL 2019]

Super Mario Land is a game developed by Nintendo.

2. New entity case (not in the local KG)
- Find the top-scoring entity in the full KG using the LM hidden state and entity embeddings
- Next entity will be: the predicted top-scoring entity
- Next word will be: most likely next token over standard vocabulary + entity aliases

3. Not an entity case
- Next entity will be: None
- Next word will be: most likely next token over standard vocabulary
Super Mario Land is a 1989 side-scrolling platform video game developed and published by Nintendo.
**KGLM [Logan et al., ACL 2019]**

- Outperforms GPT-2 and AWD-LSTM\(^1\) on a fact completion task (“fill-in-the-blank”)

- Qualitatively, KGLM tends to predict more specific tokens, whereas GPT-2 predicts more common, generic tokens

- Supports modifying/updating facts!
  - Modifying the KG has a direct change in the LM predictions

Barack Obama was born on __________.

**KG triples:**

- (Barack Obama, *birthDate*, 1961-08-04)   **Most likely next word:**
  - “August”, “4”, “1961”
- (Barack Obama, *birthDate*, 2013-03-21)   **Most likely next word:**
  - “March”, “21”, “2013”

- External memory can help LMs to do factually-grounded text generation!

---

\(^{1}\) Merity et al., ICLR 2018
Techniques to add knowledge to LMs

Add pretrained entity embeddings
- ERNIE
- QAGNN/GreaseLM

Use an external memory
- KGLM

Modify the training data
- WKLM
- ERNIE (another!), salient span masking
Method 3: 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)
Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model (WKLM) [Xiong et al., ICLR 2020]

- 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!
WKLM [Xiong et al., ICLR 2020]

Entity Boundary Representations

Stan Lee ➔ ✓
Steve Ditko ➔ ✓
Marvel Comics ➔ ✓

Transformer Encoders

Entity Boundary Representations

Bryan Johnson ➔ ❌
Steve Ditko ➔ ✓
DC Comics ➔ ❌

Transformer Encoders

Original Article:
Spider-Man is a fictional superhero created by writer-editor Stan Lee and writer-artist Steve Ditko. He first appeared in the anthology comic book American comic books published by Marvel Comics.

Replaced Article:
Spider-Man is a fictional superhero created by writer-editor Bryan Johnson and writer-artist Steve Ditko. He first appeared in the anthology comic book American comic books published by DC Comics.

Entity Replacement Procedure

Marvel Comics ➔ Q173496 ➔ Q1320047 ➔ type lookup ➔ book publishing company

Entities clustered by type Q1320047

DC Comics
Dark Horse Comics
Image Comics
DC Comics

random sample

WIKIDATA
WIKIPEDIA
The Free Encyclopedia
**WKLM [Xiong et al., ICLR 2020]**

- **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
**WKLM** [Xiong et al., ICLR 2020]

- 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

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

Much worse training for longer, compared to using the entity replacement loss.
Learn inductive biases through masking

• Besides corrupting data, another idea is: can we just do clever masking to help the LM learn factual knowledge?
  • ERNIE\(^1\): Enhanced Representation through Knowledge Integration, Sun et al., arXiv 2019
    • Uses phrase-level and entity-level masking, and shows improvements on downstream NLP tasks

• How Much Knowledge Can You Pack Into the Parameters of a Language Model?, Roberts et al., EMNLP 2020
  • Uses “salient span masking” (Guu et al., ICML 2020) to mask out salient spans (i.e. named entities and dates)
  • Shows that salient span masking improves T5’s performance on QA tasks

[1] Yes, another ERNIE paper!
ERNIE\textsuperscript{1}: Enhanced Representation through Knowledge Integration

\[\text{Sun et al., arXiv 2019}\]

Bert

Transformer

ERNEIE

phrase

entity

Transformer

[1] Yes, another ERNIE paper!
Salient span masking

Salient span masking has been shown to outperform other masking or corruption strategies on QA and document retrieval tasks.

QA/Retrieval performance on NaturalQuestions

<table>
<thead>
<tr>
<th>Masking technique</th>
<th>Exact Match</th>
<th>Retrieval Recall @5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random uniform masks (BERT)</td>
<td>32.3</td>
<td>24.2</td>
</tr>
<tr>
<td>Random span masks (SpanBERT)</td>
<td>35.3</td>
<td>26.1</td>
</tr>
<tr>
<td>Salient span masking</td>
<td>38.2</td>
<td>38.5</td>
</tr>
</tbody>
</table>

REALM, Guu et al., ICML 2020

Roberts et al., EMNLP 2020
Recap: Techniques to add knowledge to LMs

1. Use pretrained entity embeddings
   • Convenient to apply to existing architectures: just plug in entity features
   • Indirect handle of knowledge (e.g. embedding space instead of direct copy)

2. Add an external memory
   • Can support some updating of factual knowledge and easier to interpret
   • Tend to require more complex implementation and more memory

3. Modify the training data
   • Requires no model architecture changes, no additional computation/memory.
   • Still open question if this is always as effective as model changes

• It is also an active area of research to combine and get the best of those techniques!
Section 3:
Evaluating knowledge in LMs

• Probes
• Downstream tasks
**LLanguage Model Analysis (LAMA) Probe** [Petroni et al., EMNLP 2019]

• **Idea:** How much relational (commonsense and factual) knowledge is already in off-the-shelf language models?
  - Without any additional training or fine-tuning

• Manually constructed a set of “cloze” statements (fill-in-the-blank) to assess a model’s ability to predict a missing token. *Examples:*

  - The theory of relativity was developed by [MASK].
  - The native language of Mammootty is [MASK].
  - Ravens can [MASK].
  - You are likely to find a overflow in a [MASK].
LAmguage Model Analysis (LAMA) Probe [Petroni et al., EMNLP 2019]

- Generate cloze statements from KG triples and question-answer pairs in QA datasets
- **Goal:** evaluate knowledge in off-the-shelf pretrained LMs (Note: this means they may have used different pretraining corpora)
- Compare the unsupervised LMs to supervised relation extraction (RE) and QA systems

**Mean precision at one (P@1)**

<table>
<thead>
<tr>
<th>Corpus</th>
<th>DrQA</th>
<th>RE baseline</th>
<th>ELMo (5.5B)</th>
<th>ELMo (300M)</th>
<th>GPT2 (1.5B)</th>
<th>BERT-large</th>
<th>BERT-base</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google-RE</td>
<td>-</td>
<td>7.6</td>
<td>2.0</td>
<td>3.0</td>
<td>4.9</td>
<td>6.5</td>
<td>9.8</td>
</tr>
<tr>
<td>T-REx</td>
<td>-</td>
<td><strong>33.8</strong></td>
<td>4.7</td>
<td>7.1</td>
<td>20.3</td>
<td>25.1</td>
<td>31.1</td>
</tr>
<tr>
<td>ConceptNet</td>
<td>-</td>
<td>-</td>
<td>6.1</td>
<td>6.2</td>
<td>9.7</td>
<td>12.8</td>
<td>15.6</td>
</tr>
<tr>
<td>SQuAD</td>
<td><strong>37.5</strong></td>
<td>-</td>
<td>1.6</td>
<td>4.3</td>
<td>5.9</td>
<td>11.5</td>
<td>14.1</td>
</tr>
</tbody>
</table>

BERT struggles on N-to-M relations

LMs are NOT finetuned!
Using LAMA library, you can try out examples to assess knowledge in popular/your favorite LMs!
[https://github.com/facebookresearch/LAMA](https://github.com/facebookresearch/LAMA)

The cat is on the [MASK].

[1] Example courtesy of the authors at link above.
• **Limitations** of the LAMA probe:
  • Hard to understand *why* models perform well when they do
    • LM could just be memorizing word co-occurrence patterns rather than “understanding” the cloze statement and “recalling” knowledge
    • LM could just be identifying similarities between the surface forms of the subject and object (e.g., *Pope Clement VII* has the position of *pope*)
  • LMs are sensitive to the phrasing of the statement
    • e.g. sometimes rephrasing the template makes LMs suddenly perform better
    • But LAMA has only one manually defined template for each relation
    • This means probe results are a lower bound on knowledge encoded in the LM
  • We will talk about two works that address these limitations
A More Challenging Probe: LAMA-UnHelpful Names (LAMA-UHN)  
[Poerner et al., EMNLP 2020]

- Key idea: Remove the examples from LAMA that can be answered without relational knowledge

- Motivation: BERT may rely on surface forms of entities to make predictions
  - String match between subject and object
  - “Revealing” person name: Name can be a (possibly incorrect) prior for native language, nationality, etc.

- Removing these examples helps to evaluate whether BERT is really knowing the fact

- With LAMA-UHN, BERT’s score drops ~8%
  - Knowledge-enhanced model E-BERT drops only <1%
Developing better prompts to query knowledge in LMs

[Jiang et al., TACL 2020]

• Problem: LMs may know the fact, but fail on completion tasks (LAMA) due to the query phrasing
  • Pretraining text may have had different sentence structures/contexts than the query
    Example: “The birth place of Barack Obama is Honolulu, Hawaii” (pretraining corpus) versus “Barack Obama was born in _____” (query)

• Solution
  • Generate more LAMA prompts by mining templates from Wikipedia¹ and generating paraphrased prompts by using back-translation
    • Increases the chance of getting a prompt similar to what was seen in pretraining
  • Ensemble prompts: LM’s output probability is averaged over different prompts

[1] One mining approach uses dependency parsing to build the template!
Developing better prompts to query knowledge in LMs
[Jiang et al., TACL 2020]

• **Results:** Performance on LAMA for BERT-large increases 7% when using top-performing query for each relation. Ensembling leads to another 4% gain.
  - Original LAMA really was a lower bound on knowledge encoded in LM!

• Small changes in the query phrasing lead to large gains.
  - LMs are very sensitive to the query phrasing => research opportunity for robust LM!

<table>
<thead>
<tr>
<th>ID</th>
<th>Modifications</th>
<th>Acc. Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>P413</td>
<td>( x ) plays in \rightarrow at \ y ) position</td>
<td>+23.2</td>
</tr>
<tr>
<td>P495</td>
<td>( x ) was \textit{created} \rightarrow \textit{made} in \ y</td>
<td>+10.8</td>
</tr>
<tr>
<td>P495</td>
<td>( x ) was \rightarrow \textit{is} created in \ y</td>
<td>+10.0</td>
</tr>
<tr>
<td>P361</td>
<td>( x ) is a part of \ y</td>
<td>+2.7</td>
</tr>
<tr>
<td>P413</td>
<td>( x ) plays \textit{in} \ y ) position</td>
<td>+2.2</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>LM</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-GCN</td>
<td>-</td>
<td>69.9</td>
<td>63.3</td>
<td>66.4</td>
</tr>
<tr>
<td>BERT-LSTM-base</td>
<td>BERT-Base</td>
<td>73.3</td>
<td>63.1</td>
<td>67.8</td>
</tr>
<tr>
<td>ERNIE (Zhang et al.)</td>
<td>BERT-Base</td>
<td>70.0</td>
<td>66.1</td>
<td>68.0</td>
</tr>
<tr>
<td>KnowBert-W+W</td>
<td>BERT-Base</td>
<td><strong>71.6</strong></td>
<td><strong>71.4</strong></td>
<td><strong>71.5</strong></td>
</tr>
</tbody>
</table>

Peters et al., EMNLP 2019
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

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NFGEC</strong> (LSTM)</td>
<td>68.8</td>
<td>53.3</td>
<td>60.1</td>
</tr>
<tr>
<td><strong>UFET</strong> (LSTM)</td>
<td>77.4</td>
<td>60.6</td>
<td>68.0</td>
</tr>
<tr>
<td>BERT-Base</td>
<td>76.4</td>
<td>71.0</td>
<td>73.6</td>
</tr>
<tr>
<td><strong>ERNIE</strong> (Zhang et al.)</td>
<td>78.4</td>
<td>72.9</td>
<td>75.6</td>
</tr>
<tr>
<td>KnowBert-W+W</td>
<td><strong>78.6</strong></td>
<td><strong>73.7</strong></td>
<td><strong>76.1</strong></td>
</tr>
</tbody>
</table>

Zhang et al., ACL 2019 & Peters et al., EMNLP 2019
Knowledge-intensive Question Answering

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

<table>
<thead>
<tr>
<th>Model</th>
<th>CommonsenseQA</th>
<th>OpenBookQA</th>
<th>MedQA</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-Large</td>
<td>55.4</td>
<td>60.4</td>
<td>-</td>
</tr>
<tr>
<td>RoBERTa-Large</td>
<td>68.7</td>
<td>64.8</td>
<td>35.0</td>
</tr>
<tr>
<td>SapBERT-Base</td>
<td>-</td>
<td>-</td>
<td>37.2</td>
</tr>
<tr>
<td>QAGNN</td>
<td>73.4</td>
<td>67.8</td>
<td>38.0</td>
</tr>
<tr>
<td>GreaseLM</td>
<td><strong>74.2</strong></td>
<td>66.9</td>
<td><strong>38.5</strong></td>
</tr>
</tbody>
</table>

Devlin et al., NAACL 2019 & Liu et al., 2019 & Liu et al., NAACL 2021 & Yasunaga et al., NAACL 2021 & Zhang et al., ICLR 2022
Recap: Evaluating knowledge in LMs

- **Probes**
  - Evaluate the knowledge already present in models without more training
  - Challenging to construct benchmarks that really test factual knowledge
  - Challenging to construct the query prompts used in the probe

- **Downstream tasks**
  - Evaluate the usefulness of the knowledge-enhanced representation in applications
  - Typically requires finetuning the LM further on the downstream task
  - Less direct way to evaluate the knowledge in the LM, but perhaps more practically useful in terms of applications
Other exciting progress & what’s next?

• Retrieval-augmented language models [More details in Kelvin Guu’s lecture!]
  • REALM, Guu et al., ICML 2020
  • RAG, Lewis et al., NeurIPS 2020
  • Retro, Borgeaud et al., 2022
• Modifying knowledge in language models [More details in Eric Mitchell’s lecture!]
  • Fast Model Editing at Scale, Mitchell et al., 2021
• More knowledge-aware pretraining for language models
  • KEPLER, Wang et al., TACL 2020
• More efficient knowledge systems
  • NeurIPS Efficient QA challenge
• Better knowledge benchmarks
  • KILT, Petroni et al., NAACL 2021
Good luck with your projects!