Combining pre-trained language models and structured knowledge

Pedro Colon-Hernandez*
MIT Media Lab

Catherine Havasi
Dalang Health

Jason Alonso
Dalang Health

Matthew Huggins
MIT Media Lab

Cynthia Breazeal
MIT Media Lab

In recent years, transformer-based language models have achieved state of the art performance in various NLP benchmarks. These models are able to extract mostly distributional information with some semantics from unstructured text, however it has proven challenging to integrate structured information, such as knowledge graphs into these models. We examine a variety of approaches to integrate structured knowledge into current language models and determine challenges, and possible opportunities to leverage both structured and unstructured information sources. From our survey, we find that there are still opportunities at exploiting adapter-based injections and that it may be possible to further combine various of the explored approaches into one system.

1. Introduction

Recent developments in Language Modeling (LM) techniques have greatly improved the performance of systems in a wide range of Natural Language Processing (NLP) tasks. Many of the current state of the art systems are based on variations to the transformer (Vaswani et al. 2017) architecture. The transformer architecture, along with modifications such as the Transformer XL (Dai et al. 2019) and various training regimes such as the Masked Language Modeling (MLM) used in BERT (Devlin et al. 2018) or the Permutation Language Modeling (PLM) used in XLNet(Yang et al. 2019), uses an attention based mechanism to model long range dependencies between text. This modeling encodes syntactic knowledge, and to a certain extent some semantic knowledge contained in unstructured texts.

There has been interest in being able to understand what kinds of knowledge are encoded in these models' weights. Hewitt et al. (Hewitt and Manning 2019) devise a system that generates a distance metric between embeddings for words in language models such as BERT. They show that there is some evidence that there are syntax trees embedded in the transformer language models and this could explain the performance of these models in some tasks that utilize syntactic elements of text.

Petroni et al. (Petroni et al. 2019) build a system (LAMA) to gauge what kinds of knowledge are encoded in these weights. They discover that language models em-

---

* 75 Amherst St, Cambridge, MA, 02139. E-mail: pe25171@mit.edu.

bed some facts and relationships in their weights during pre-training. This in turn can help explain the performance of these models in semantic tasks. However, these transformer-based language models have some tendency to hallucinate knowledge (whether through bias or incorrect knowledge in the training data). This also means that some of the semantic knowledge they incorporate is not rigidly enforced or utilized effectively.

Avenues of research have begun to open up on how to prevent this hallucination and how to inject additional knowledge from external sources into the transformer-based language models. One promising avenue is through the integration of knowledge graphs such as Freebase(Bollacker et al. 2008), WordNet(Miller 1998), ConceptNet(Speer, Chin, and Havasi 2017), and ATOMIC(Sap et al. 2019).

A knowledge graph (used somewhat interchangeably with knowledge base although they are different concepts) is defined as "a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent relations between these entities" (Hogan et al. 2020). Formally, a knowledge graph is a set of triples that represents nodes and edges between these nodes. Let us define a set of vertices (which we will refer to as concepts) as V, a set of edges as E (which we will refer to as assertions as per Speer and Havasi(Speer and Havasi 2012)), and a set of labels L (which we will refer to as relations). A knowledge graph is a tuple $G := (V, E, L)$[1]. The set of edges (E) or assertions is composed of triples $E \subseteq V \times L \times V$ which are seen as a subject (a concept), a relation (a label), and object (another concept) respectively (e.g. $(subject, relation, object)$). These edges in some cases can have weights to represent the strength of the assertion. Broadly speaking, knowledge graphs (KGs) are a collection of tuples that represent things that should be true within the knowledge of the world that we are representing. An example assertion is "a dog is an animal" and its representation as a tuple would be: (dog,isA,animal).

Ideally, we would want to "*inject*" this structured collection of confident information (i.e. knowledge graph) into that of the high-coverage, contextual information found in language models. This injection would permit the model to incorporate some of the information found in the KG to improve its performance in inference tasks.

There are currently various approaches that try to achieve this injection. The approaches in general take either one or combinations of three forms: *input* focused injections, *architecture* focused injections, and *output* focused injections. We define an *input* focused injection as any technique that modifies the data pre-processing or the pre-transformer layer inputs that the base model uses(i.e. injecting knowledge graph triples into the training data to pre-train/fine tune on them or combining entity embeddings into the static word embeddings that the models have). We define *architecture* focused injections as techniques that alter a base model's transformer layers (i.e. adding additional layers that inject in some representation). Lastly, we define an *output* focused injection as any techniques that either modify the output of the base models or that modify/add custom loss functions. In addition to these three basic types, there are approaches that utilize combinations of these (i.e. a system that uses both input and output injections), which we call *combination* injections. Figure 1 gives an abstract visualization of the types of injections that we describe.

To be consistent throughout the type of injections, we will now give some definitions and nomenclature. Let us define a sequence of words (unstructured text) as S. Typically in a transformer-based model, this sequence of words is converted to a

---

1 We use the formal definitions found in Appendix B of (Hogan et al. 2020)

sequence of tokens that is then converted into some initial context-independent embeddings. To a word sequence we can apply a tokenization technique $\mathcal{T}$ to convert the word sequence into a token sequence T. This can be seen as $\mathcal{T}(S) = T$. This sequence T is used as a lookup in an embedding layer $\mathcal{E}$ to produce context independent token vector embeddings: $\mathcal{E}(T) = E$. These are then passed sequentially through various contextualization layers (i.e. transformers) which we define as the set $\mathcal{H}$, $\mathcal{H} = (H_1, ..., H_n)$. The successive application of these ultimately produces a sequence of contextual embeddings C: $C = H_n(H_{n-1}(...H_1(E)))$. We additionally define $\mathcal{G}$ as graph embeddings of a knowledge graph G that are the result of some embedding function $\mathcal{E}_g$: $\mathcal{G} = \mathcal{E}_g(G)$. This final sequence is run through a final layer L_{LM} that is used to calculate the language modeling loss function $\mathcal{L}$ that is optimized through back-propagation. The notation that we utilize is intentionally vague on the definition of the functions, in order for us to fit the different works that we survey.

In the following sections we will look at attempts of injecting knowledge graph information that fall into the aforementioned categories. Additionally, we will highlight relevant benefits in these approaches. We conclude with possible opportunities and future directions for injecting structured knowledge into language models.
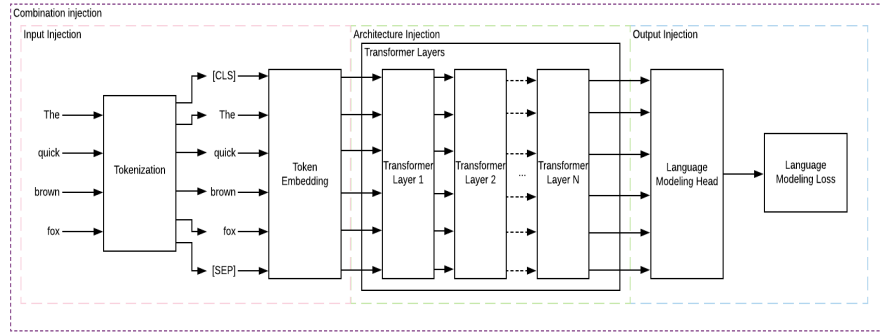
Figure 1
Visualization of boundaries of the different categories of knowledge injections. *Combination* injections involve combinations of the three categories.

2. Input Focused Injections

In this section we will describe knowledge injections whose techniques center around modifying either the structure of the input or the data that is selected to be fed into the base transformer models. A common approach to inject information from a knowledge graph is by converting its assertions into a set of words (possibly including separator tokens) and pre-training or fine-tuning a language model with these inputs. We discuss two particular papers that focus on structuring the input in different ways as to capture the semantic information from triples found in a KG. These approaches start from a pre-trained model, and fine-tune on their knowledge infusing datasets. A summary of these approaches can be found in table 1.

Input focused injections can be seen any technique whose output is a modified E, hereby known as E'. This modification can be achieved either by modifying $S, \mathcal{T}, T, \mathcal{E}$, or directly E. (i.e. the word sequence, the token sequence, the tokenization function, the context-less embedding function, or the actual context-less embeddings). The hope of

input focused injections is that the knowledge in E' will be distributed and contextualized through $\mathcal{H}$ as the language models are trained.

2.1 Align, Mask, Select (AMS) (Ye et al. 2019)

AMS is an approach in which a question answering dataset is created, whose questions and possible answers are generated by aligning a knowledge graph (in this particular case ConceptNet) with plain text. A BERT model is trained on this dataset to inject it with the knowledge.

Taking an example from their work, the ConceptNet triple *(population, AtLocation, city)* is aligned with a sentence from the English Wikipedia (i.e. "The largest **city** by **population** is Birmingham, which has long been the most industrialized city.") that contains both concepts in the triple. They then proceed to mask out one of the concepts with a special token ($[QS]$) and produce 4 plausible concepts as answers to the masking task by looking at the neighbors in ConceptNet that have the same masked token and relationship. Lastly, they concatenate the generated question with the plausible answers and run it through a BERT model tailored for question answering (QA) (following the same approach as the architecture and loss for the SWAG task in the original BERT). At the output, they run the classification token ($[CLS]$) through a softmax classifier to determine if the selected concept is the correct one or not.

The authors note that the work is sensitive to what it has seen in the pre-training because when asked a question that needs to disambiguate a pronoun it tries to match what it has seen the most in the training data. This may mean that the generalization of the structured knowledge (here commonsense information) or the understanding of the it is overshadowed by the distributional information that it is learning, however more testing would need to be done to verify this. Overall some highlights of the work are:

- Automated pre-training approach which constructs a QA dataset aligned to a KG

- Utilization of graph-based confounders in generated dataset entries

2.2 COMmonsEnse Transformers (COMET) (Bosselut et al. 2019)

COMET is a GPT(Radford et al. 2018) based system which is trained on triples from KGs (ConceptNet and Atomic) to learn to predict the object of the triple (the triples being defined as (subject, relation, object)). The triples are fed as a concatenated sequence of words into the model (i.e. the words for the subject, the relationship, and the object) along with some separators.

The authors initialize the GPT model to the final weights in the training from Radford et al.(Radford et al. 2018) and proceed to train it to predict the words that belong to the object in the triple. A very interesting part of this work is that it is capable directly of performing knowledge graph completion for nodes and relations that may not have been seen during training, in the form of sentences.

Some plausible shortcomings of this work is that you are still having the model extract the semantic information from the distributional one and possibly suffering from the same bias as AMS. In addition to this, by training on the text version of these triples, it may be the case that we lose some of the syntax that the model learns due to awkwardly formatted inputs (i.e. "cat located at housing" rather than "a cat is located at a house"), however further testing of these two needs to be performed.

There is some relevant derivative work for COMET by Bosselut et al.(Bosselut and Choi 2019) which looks into how effective COMET is at building KGs on the fly given a certain context, a question, and a proposed answer. They combine the context with a relation from ATOMIC and feed it into COMET to represent reasoning hops. They do this for multiple relations and keep redoing this with the generated outputs to represent a reasoning chain for which they can derive a probability. They use this in a zero-shot evaluation of a question-answering system and find that it is effective. Overall some highlights of COMET are:

- Generative language model that can provide natural language representations of triples

- Useful model for zero-shot KG completion

- Simple pre-processing of triples for training

Model	Summary of Injection	Example of Injection
Align, Mask, Select	Aligns a knowledge base with textual sentences, masks entities in the sentences, and selects alternatives with confounders to create a QA dataset	**KG Assertion**: (population, AtLocation, city) **Model Input**: The largest [QW] by population is Birming- ham, which has long been the most industrialized city? city, Michigan, Petrie dish, area with people inhabiting, country
COMET	Ingests a formatted sentence version of a triple from ConceptNet and Atomic	**KG Assertion**: (PersonX goes to the mall, xIntent, to buy clothes) **Model input**: PersonX goes to the mall [MASK] $\langle$ xIntent $\rangle$ to buy clothes

Table 1
Input Injection System Comparisons

3. Architecture Injections

In this section we describe approaches that focus on architectural changes to language models. This involves either adding additional layers that integrate knowledge in some way with the contextual representations or modifying existing layers to manipulate things such as attention mechanisms. We discuss two approaches within this category that fall under layer modifications. These approaches utilize adapter-like mechanisms to be able to inject information into the models. A summary of these approaches can be found in table 2.

3.1 KnowBERT(Peters et al. 2019)

KnowBERT modifies BERT's architecture by integrating some layers that they call the Knowledge Attention and Recontextualization (KAR). These layers take graph entity embeddings, that are based on Tucker Tensor Decompositions for KG completion (Balažević, Allen, and Hospedales 2019), and run them through an attention mechanism to generate entity span embeddings. These span embeddings are then added to the regular BERT contextual representations. The summed representations are then uncompressed

and passed on to the next layer in a regular BERT. Once the KAR entity linker has been trained, the rest of the BERT model is unfrozen and is trained in the pre-training. These KAR layers are trained for every KG that is to be injected, in this work they use data from Wikipedia and Wordnet.

An interesting observation is that the injection happens in the later layers, which means that the contextual representation up to that point may be unaltered by the injected knowledge. This is done to stabilize the training, but could present an opportunity to inject knowledge in earlier levels. Additionally, the way the system is trained, the entity linking is first trained, and then the whole system is unfrozen to incorporate the additional knowledge into BERT. This strategy could lead to the catastrophic forgetting(Kirkpatrick et al. 2017) problem where the knowledge from the underlying BERT model or the additional structured injection may be forgotten or ignored.

This technique falls into a broader category of what is called Adapters(Houlsby et al. 2019). Adapters are layers that are added into a language model and are subsequently fine tuned to a specific task. The interesting aspect of adapters is that they add a minimal amount of additional parameters, and freeze the original model weights. The added parameters are also initialized to produce a close to identity output. It is worth noting that the KnowBERT is not explicitly an Adapter technique as the model is unfrozen during training. Some highlights of KnowBERT are the following:

- Fusion of contextual and graph representation of entities

- Attention enhanced entity spanned knowledge infusion

- Permits the injection of multiple KGs in varying levels of the model

3.2 Common sense or world knowledge? investigating adapter-based knowledge injection into pre-trained transformers(Lauscher et al. 2020)

This work explores what kinds of knowledge are infused by fine tuning an adapter equipped version of BERT on ConceptNet. They generate and test models trained on sentences from the Open Mind Common Sense (OMCS)(Singh et al. 2002) corpus and from walks in the ConceptNet graph. They note that with simple adapters and as little as 25k/100k update steps on their training sentences, they are able greatly improve the encoded "World Knowledge" (another name for the knowledge found in ConceptNet).

However, it is worth noting that the information is presented as sentences to which the adapters are fine tuned. This may mean that the model may have similar possible shortcomings such as with the approaches that are input-focused (model may rely more on the distributional rather than the semantic information), however testing needs to be performed to confirm this. Overall some highlights of this work are the following:

- Adapter based approach which fine-tunes a minimal amount of parameters

- Shows that a relatively small amount of additional iterations can inject the knowledge in the adapters

- Show that adapters, trained on KGs, do indeed boost the semantic performance of transformer-based models

Model	Injected in Pre-Training	Injected in Fine-Tuning	Summary of Injection
KnowBERT	Yes	Yes	Sandwich Adapter-like layers which sum contextual representation of layer with graph representation of entities and distributes it in an entity span
Common sense or world knowledge?[...]	No	Yes	Use sandwich adapters to fine tune on a KG

Table 2
Architecture Injection System Comparisons

4. Output Injections

In this section we describe approaches that focus on changing either the output structure or the losses that were used in the base model in some way to incorporate knowledge. Only one model falls strictly under this category, the model injects entity embeddings into the output of a BERT model.

4.1 SemBERT(Zhang et al. 2019)

SemBERT uses a subsystem that generates embedding representations of the output of a semantic role labeling(Màrquez et al. 2008) system. They then concatenate this representation with the output of the contextualized representation from BERT to help incorporate relational knowledge. The approach, although clever, may fall short in that although it gives a representation for the roles, it leaves the model to figure out the exact relationship that the roles are performing, however testing would need to be performed to check this. Some highlights of SemBERT are:

- Encodes semantic role in an entity embedding that is combined at the output

5. Combination and Hybrid Injections

Here we describe approaches that use combinations of injection types such as input/output injections or architecture/output injections, etc. We start by looking at models that perform input injections and reinforce these with output injections (LIBERT, KALM), We then look at models that manipulate the attention mechanisms to mimic graph connections (BERT-MK,K-BERT). We follow this by looking into KG-BERT, a model that operates on KG triples, and K-Adapter, a modification of RoBERTa that encodes KGs into Adapter Layers and fuses them. After this, we look into the approach presented as Cracking the Contextual Commonsense Code[...] which determines that there are areas lacking in BERT that could be addressed by supplying appropriate data, and we look at ERNIE 2.0, a framework for multi-task training for semantically aware models. Lastly, we look at two hybrid approaches which extract LM knowledge and leverage it for different tasks. A summary of these injections can be found in table 3.

5.1 Knowledge-Aware Language Model (KALM) Pre-Training (Rosset et al. 2020)

KALM is a system that does not modify the internal architecture of the model that it is looking to inject the knowledge into, rather it modifies the input of the model by fusing entity embeddings with the normal word embeddings that the language model (in KALM's case, GPT-2) uses. They then enforce the model in the output to uphold the entity information by adding an additional loss component in the pre-training that uses a max margin between the cosine distance of the output contextual representation and the input entity embedding and the cosine distance of the contextual representation and a confounder entity. Altogether what this does is that it forces the model to notice when there is an entity and tries to make the contextual representation have the semantics of the correct input entity. Some highlights of KALM are:

- Sends an entity signal in the beginning and and enforces it in the output of a generative model to notice its semantics

5.2 Exploiting structured knowledge in text via graph-guided representation learning(Shen et al. 2020)

This work masks informative entities that are drawn from a knowledge graph, in BERT's MLM objective. In addition to this, they have an auxiliary objective which uses a max-margin loss for a ranking task which is composed of using a bilinear model that calculates a similarity score between a contextual representation of an entity mention and the representation of the $[CLS]$ token for the text. The use for this is to determine if it is a relevant entity or a distractor. Both KALM and this work are very similar, but a key difference is that KALM uses a generative model without any kind of MLM objective, and KALM does not do any kind of filtering for the entities. Some highlights of this work are:

- Filters relevant entities to incorporate their information into the model

- Enforces entity signal at beginning and end of the model through masking and max-margin losses

5.3 Lexically Informed BERT (LIBERT)(Lauscher et al. 2019)

LIBERT converts batches of lexical constraints and negative examples, into a BERT-compatible format. The lexical constraints are synonyms and direct hyponyms-hypernyms (specific,broad) and take the form of a set of tuples of words: ($C = \{(w_1, w_2)_i\}_{i=1}^N$). In addition to this set, the authors generate some negative examples by finding the words that are semantically close to (w_1) and (w_2) in a given batch. They then format the examples into something BERT can use which is simply the wordpieces that pertain to words in the batch separated by the separator token. They pass this input through BERT and use the $[CLS]$ token as an input to a softmax classifier to determine if the example is a valid lexical relation or not.

During pre-training they alternate between a batch of sentences and a batch of constraints. LIBERT outperforms BERT with lesser (1M) iterations of pre-training. It is worth noting that as the amount of iterations of training increase, the gap between the two systems, although present, becomes smaller. This may indicate that, although the additional training objective is effective, it may be getting overshadowed by the

regular MLM coupled with large amounts of data, however more testing needs to be performed. It is also worth noting that the authors do not align the sentences with the constraint batches, combine the training tuples which may hinder training as BERT has to alternate between different training input structures, and lastly, they do not incorporate antonymy constraints in their confounder selection, so further experimentation would be required to verify the effects of these. Some highlights of LIBERT are the following:

- Incorporate lexical constraints from entity embeddings

- Good performance with constrained amounts of data

5.4 BERT-MK(He et al. 2019)

BERT-MK utilizes a combination of architecture injection and an output injection (additional training loss). In BERT-MK, they utilize the KG-transformer modules which are transformer layers that are combined with learned entity representations. These entity representations are generated from another set of transformer layers that are trained on a KG converted to natural language sentences. The interesting aspect is that these additional layers incorporate an attention mask that mimics the connections in the KG, so to a certain extent, incorporating the structure of the graph and propagating it back into the embeddings. These additional layers are trained to reconstruct the input set of triples. The authors evaluate the system for medical knowledge (MK) however it may be interesting to evaluate this on the GLUE benchmark along with utilizing other KGs such as ATOMIC or ConceptNet. Some highlights of BERT-MK are:

- Utilization of a modified attention mechanism to mimic KG structure between terms

- Incorporation of triple reconstruction loss to train the KG-transformer modules

- Merges KG-transformer with regular transformer for contextual+knowledge-informed representation

5.5 K-BERT (Liu et al. 2020)

K-BERT uses a combination of input injection and architecture injections. For a given sentence, they inject relevant triples for the entities that are present in the sentence and in a KG. They inject these triples in between the actual text and utilize a soft-position embedding to determine the order in which the triples are evaluated. These soft position embeddings simply add positional embeddings to the injected triple tokens. This in turn creates a problem that the tokens are injected as entities appear in a sentence, and hence the ordering of the tokens is altered.

To remedy this the authors utilize a masked self attention similar to BERT-MK. What this means is that the attention mechanism should only be able to see everything up to the entity that matched in the injected triple. This attention mechanism helps the model focus on what relevant knowledge it should incorporate. It would have been good to see a comparison of just adding these as sentences in the input rather than having to

fix the attention mechanism to compensate for the erratic placement. Some highlights of K-BERT are:

- Utilization of attention mechanism to mimic connected subgraphs of injected triples

- Injection of relevant triples as text inputs

5.6 KG-BERT (Yao, Mao, and Luo 2019)

The authors present a combination approach which fine-tunes a BERT model with the text of triples from a KG similar to COMET. The authors also feed confounders in the form of random samples of entities into the training of the system. It utilizes a binary classification task to determine if the triple is valid and a relationship type prediction task to determine which relations are present between pairs of entities. Although this system is useful for KG completion, there is no evidence of its performance on other tasks. Additionally they train one triple at a time which may limit the model's ability to learn the extended relationships for a given set of entities. Some highlights of KG-BERT are the following:

- Fine tunes BERT into completing triples from a KG

- Uses a binary classification to predict if a triple is valid

- Uses multi-class classification to predict relation type

5.7 K-Adapter (Wang et al. 2020)

A work based on adapters, K-Adapter works by adding projection layers before and after a subset of transformer layers. They do this only for some specific layers in a pre-trained RoBERTa model (the first layer, the middle layer, and the last layer). They then freeze RoBERTa as per the Adapter work in (Houlsby et al. 2019) and train 2 adapters to learn factual knowledge from Wikipedia triples (Elsahar et al. 2019) and linguistic knowledge from outputs of the Stanford parser(Chen and Manning 2014). They then train the adapters with a triple classification (whether the triple is true or not) task similar to KG-BERT.

It is worth noting that the authors compare RoBERTa and their K-Adapter approach against BERT, and BERT has considerably better performance on the LAMA probes. The authors attribute RoBERTa's byte pair encodings(Shibata et al. 1999) (BPE) as the major performance delta between their approach and BERT. Another possible reason may be that they only perform injection in a few layers rather than throughout the entire model, although testing needs to be done to confirm this. Some highlights of K-Adapter are:

- Approach provides a framework for continual learning

- Use a fusion of trained adapter outputs for evaluation tasks

5.8 Cracking the Contextual Commonsense Code: Understanding Commonsense Reasoning Aptitude of Deep Contextual Representations(Da and Kusai 2019)

The authors analyze BERT to determine that it is deficient in certain attribute representations of entities. The authors use the RACE (Lai et al. 2017) dataset, and based

on five attribute categories (Visual, Encyclopedic, Functional Perceptual, Taxonomic), select samples from the dataset that may be help a BERT model compensate deficiencies in the areas. They then fine tune on this data. In addition to this, the authors concatenate the fine tuned BERT embeddings with some knowledge graph embeddings. These graph embeddings are generated based on assertions that involve the entities that are present in the questions and passages they train their final joint model on (MCScript 2.0(Ostermann, Roth, and Pinkal 2019)). Their selection of additional fine-tuning data for BERT improves their performance in MCScript 2.0, highlighting that their selection addressed missing knowledge.

It is worth noting that the graph embeddings that they concatenate boost the performance of their system which shows that there is still some information in KGs that is not in BERT. We classify this approach as a combination approach because they concatenate the result of the BERT embeddings and KG embeddings and fine tune both at the same time. The authors however gave no insight as to how the KG embeddings could have been incorporated in the fine-tuning/pre-training of BERT with the RACE dataset. Some highlights of this work are:

- BERT has some commonsense information in some areas, but is lacking in others

- Fine-tuning on the deficient areas increases performance accordingly

- The combination of graph embeddings plus contextual representations are useful

5.9 ERNIE 2.0 (Sun et al. 2020)

The authors develop a framework that constructs pre-training tasks that center around Word-aware Pre-training, Structure-aware Pre-training, Semantic-aware Pre-training Tasks, and proceeds to train a transformer based model on these tasks. An interesting aspect is that as they finish training on tasks, they keep training on older tasks in order for the model to not forget what it has learned. In ERNIE 2.0 the authors do not incorporate KG information explicitly. They do have a sub-task within the Word-aware pre-training that masks entities and phrases with the hope that it learns the dependencies of the masked elements which may help to incorporate assertion information.

A possible shortcoming of this model is that some tasks that are intended to infuse semantic information into the model (i.e. the Semantic aware tasks which are a Discourse Relation task and an information retrieval (IR) relevance) rely on the model to pick it up from the distributional examples. This could have the same possible issue as with the Input Injections and would need to be investigated further. Additionally, they do not explicitly use KGs in the work. Some highlights of ERNIE 2.0 are:

- Continual learning platform keeps training on older tasks to maintain their information

- Framework permits flexibility on the underlying model

- Wide variety of semantic pre-training tasks

5.10 Graph-based reasoning over heterogeneous external knowledge for common-sense question answering (Lv et al. 2020)

A hybrid approach in which the authors do not inject knowledge into a language model (namely XLNet(Yang et al. 2019), rather they utilize a language model as a way to unify graph knowledge and contextual information. They combine XLNet embeddings as nodes in a Graph Convolutional Network (GCN) to answer questions.

They generate relevant subgraphs of ConceptNet and Wikipedia (from ConceptNet the relations that include entities in a question/answer exercise and the top 10 most relevant sentences from ElasticSearch on Wikipedia). They then perform a topological sorting on the combined graphs and pass them as input to XLNet. XLNet then generates contextual representations that are then used as representations for nodes in a Graph Convolutional Network (GCN). They then utilize graph attention to generate a graph level representation and combine it with XLNet's input ([CLS] token) representation to determine if an answer is valid for a question. In this model they do not fine-tune XLNet, which could have been done on the dataset to give better contextual representations, and additionally they do not leverage the different levels of representation present in XLNet. Some highlights of this work are the following

- Combination of GCN, Generative Language Model, and Search systems to answer questions
- Use XLNet as contextual embedding for GCN nodes
- Perform QA reasoning with the GCN output

5.11 Commonsense knowledge base completion with structural and semantic context(Malaviya et al. 2020)

Another hybrid approach, the authors fine tune a BERT model on a list of the unique phrases that are used to represent nodes in a KG. They then take the embeddings from BERT and from a sub-graph in the form of a GCN and run it through an encoder/decoder structure to determine the validity of an assertion.[2]

They then take this input and concatenate it with node representations for for a subgraph (in this case a combination of ConceptNet and Atomic). They treat this concatenation as an encoded representation, and run combinations of these through a convolutional decoder that additionally takes an embedding of a relation type.

The result of the convolutional decoder is run through a bilinear model and a sigmoid function to determine the validity of the assertion. It seems interesting that the authors only run the convolution through one side: convolution of (e_i, e_{rel}) rather than both the convolution of (e_i, e_{rel}) and (e_{rel}, e_j) (where (e_i, e_j) are the entity embeddings for entity i and j respectively and (e_{rel}) is the embedding for a specific relationship) and then a concatenation. They rely on the bilinear model joining the two representations.

Some highlights of this work are the following:

- Use a GCN and a LM to generate contextualized assertions representations

---

2 It is worth noting that the two hybrid projects possibly benefited from the ability for these language models to encode assertions as shown by Feldman et al. (Davison, Feldman, and Rush 2019) and Petroni et al.(Petroni et al. 2019).

- Use BERT to generate contextual embeddings for nodes
- Use an encoder-decoder structure to learn triples

6. Future Directions

6.1 Input Injections

Most input injections are to format KG information into whatever format a transformer model can ingest. Although KALM has explored incorporating a signal to the input representations, it would be interesting to add additional information such as the lexical constraints mentioned in LIBERT, to the word embeddings that are trained with the transformer based models like BERT. A possible approach could be to build a post-specialization system that could generate retrofitted(Faruqui et al. 2014) representations that can then be fed into language models.

6.2 Architecture Injections

Adapters seem to be a promising field of research in language models overall. The idea that one can fine tune a small amount of parameters may simplify the injection of knowledge and KnowBERT has explored some of these benefits. It would be interesting to apply a similar approach to generative models and see the results.

Another possible avenue of research would be to incorporate neural memory models/modules such as the ones by Munkhdalai(Munkhdalai et al. 2019) into adapter-based injections. The reasoning would be that the model can simply look up relevant information encoded into a memory architecture and fuse it into a contextual representation.

6.3 Combined Approaches

There are a variety of combined approaches, but none of them tackle all three areas (input, architecture, and output) at the same time. It seems promising to test out a signaling method such as KALM and see how this would work with an adapter based method similar to KnowBERT. The idea being that the input signal could help the entity embeddings contextualize better within the injected layers. Additionally, it would be interesting to see how the aforementioned combination would look with a system similar to LIBERT; such that you could fuse entity embeddings with some semantic information.

7. Conclusion

Infusing structured information from Knowledge Graphs into pre-trained language models has had some success. Overall, the works reviewed here give evidence that the models benefit from the incorporation of the structured information. By analyzing the existing works, we give some research avenues that may help to develop more tightly coupled language/KG models.

References

Balažević, Ivana, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Bodenreider, Olivier. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Bollacker, Kurt, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Bosselut, Antoine and Yejin Choi. 2019. Dynamic knowledge graph construction for zero-shot commonsense question answering. *arXiv preprint arXiv:1911.03876*.

Bosselut, Antoine, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. *arXiv preprint arXiv:1906.05317*.

Chen, Danqi and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Da, Jeff and Jungo Kusai. 2019. Cracking the contextual commonsense code: Understanding commonsense reasoning aptitude of deep contextual representations. *arXiv preprint arXiv:1910.01157*.

Dai, Zihang, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Davison, Joe, Joshua Feldman, and Alexander M Rush. 2019. Commonsense knowledge mining from pretrained models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1173–1178.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Elsahar, Hady, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Elena Simperl, and Frederique Laforest. 2019. T-rex: A large scale alignment of natural language with knowledge base triples.

Faruqui, Manaal, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Gabrilovich, Evgeniy, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). *Note: http://lemurproject. org/clueweb09/FACC1/Cited by*, 5:140.

He, Bin, Di Zhou, Jinghui Xiao, Qun Liu, Nicholas Jing Yuan, Tong Xu, et al. 2019. Integrating graph contextualized knowledge into pre-trained language models. *arXiv preprint arXiv:1912.00147*.

Hewitt, John and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Hogan, Aidan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. 2020. Knowledge graphs. *arXiv preprint arXiv:2003.02320*.

Houlsby, Neil, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. *arXiv preprint arXiv:1902.00751*.

Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Lai, Guokun, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Lauscher, Anne, Olga Majewska, Leonardo FR Ribeiro, Iryna Gurevych, Nikolai Rozanov, and Goran Glavaš. 2020. Common sense or world knowledge? investigating adapter-based knowledge injection into pretrained transformers. *arXiv preprint arXiv:2005.11787*.

Lauscher, Anne, Ivan Vulić, Edoardo Maria Ponti, Anna Korhonen, and Goran Glavaš. 2019. Informing unsupervised pretraining with external linguistic knowledge. *arXiv preprint arXiv:1909.02339*.

Liu, Weijie, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *AAAI*, pages 2901–2908.

Lv, Shangwen, Daya Guo, Jingjing Xu, Duyu Tang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, and Songlin Hu. 2020. Graph-based reasoning over heterogeneous external knowledge for commonsense question answering. In *AAAI*, pages 8449–8456.

Malaviya, Chaitanya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *AAAI*, pages 2925–2933.

Màrquez, Lluís, Xavier Carreras, Kenneth C Litkowski, and Suzanne Stevenson. 2008. Semantic role labeling: an introduction to the special issue.

Miller, George A. 1998. *WordNet: An electronic lexical database*. MIT press.

Munkhdalai, Tsendsuren, Alessandro Sordoni, Tong Wang, and Adam Trischler. 2019. Metalearned neural memory. In *Advances in Neural Information Processing Systems*, pages 13331–13342.

Ostermann, Simon, Michael Roth, and Manfred Pinkal. 2019. Mcscript2. 0: A machine comprehension corpus focused on script events and participants. *arXiv preprint arXiv:1905.09531*.

Peters, Matthew E, Mark Neumann, Robert L Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A Smith. 2019. Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164*.

Petroni, Fabio, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Rosset, Corby, Chenyan Xiong, Minh Phan, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Knowledge-aware language model pretraining. *arXiv preprint arXiv:2007.00655*.

Sap, Maarten, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A Smith, and Yejin Choi. 2019. Atomic: An atlas of machine commonsense for if-then reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3027–3035.

Shen, Tao, Yi Mao, Pengcheng He, Guodong Long, Adam Trischler, and Weizhu Chen. 2020. Exploiting structured knowledge in text via graph-guided representation learning. *arXiv preprint arXiv:2004.14224*.

Shibata, Yusuxke, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. 1999. Byte pair encoding: A text compression scheme that accelerates pattern matching. Technical report, Technical Report DOI-TR-161, Department of Informatics, Kyushu University.

Singh, Push, Thomas Lin, Erik T Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1223–1237, Springer.

Speer, Robert and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686.

Speer, Robyn, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Sun, Yu, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *AAAI*, pages 8968–8975.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Wang, Ruize, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, Ming Zhou, et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*.

Yang, Zhilin, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Yao, Liang, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Ye, Zhi-Xiu, Qian Chen, Wen Wang, and Zhen-Hua Ling. 2019. Align, mask and select: A simple method for incorporating commonsense knowledge into language representation models. *arXiv preprint arXiv:1908.06725*.

Zhang, Zhuosheng, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2019. Semantics-aware bert for language understanding. *arXiv preprint arXiv:1909.02209*.

Model	Injected in Pre-Training	Injected in Fine-Tuning	Summary of Input Injection	Summary of Architecture Injection	Summary of Output Injection
KALM	Yes	No	Combines an entity embedding with the model's word embeddings	N/a	Incorporates a max-margin loss with cosine distances to enforce semantic information
Exploiting structured knowledge in text [...]	Yes	No	Uses a KG informed masking scheme to exploit MLM learning	N/A	Incorporates a max-margin loss with distractors from a KG and a bilinear scoring model for the MM loss.
LIBERT	Yes	No	Alternates between batches of sentences, and batches of tuples that are lexically related	N/a	Adds a binary classifier as a third training task to determine if the tuples form a valid lexical relation
BERT-MK	Yes	No	N/A	Combines a base transformer with KG transformer modules which are trained to learn contextual entity representations and have an attention mechanism that mimics the connections between a graph	Use a triple reconstruction loss, similar to MLM, but for triples
K-BERT	Yes	No	Incorporate as part of their training batches, assertions from entities present in a sample	Modify the attention mechanism and position embeddings to reorder injected information, and mimic KG connections	N/A
KG-BERT	No	Yes	Feeds triples from a knowledge graph as input examples	N/A	Uses a binary classification objective to determine if a triple is correct and a multi-class classification objective to determine what kind of relation the triple has.
K-Adapter	No	Yes	N/A	Fine tune adapter layers to store information from a KG	Combine the model and different adapter layers to give a contextual representation with information from different sources, additionally use a relation classification loss for each trained adapter.
Cracking the contextual commonsense code[...]	Yes	No	Pre-process the data to address commonsense relation properties that are deficient in BERT	N/A	Concatenates a graph embedding to the output of the BERT model
ERNIE 2.0	Yes	No	Construct data for pre-training tasks	N/A	Provides a battery of tasks that are trained in parallel to enforce different semantic areas in a model

Table 3
Combination Injection Systems Comparisons

Knowledge Injection Approach	Underlying Language Model	Type of Injection	Knowledge Sources	Training Objective
Align, Mask, Select	BERT	Input	ConceptNet	Binary Cross Entropy
COMET	GPT	Input	Atomic, ConceptNet	Language Modeling
KnowBERT	BERT	Architecture	Wikipedia, WordNet	Masked Language Modeling
Common sense or world knowledge?[...]	BERT	Architecture	ConceptNet	Masked Language Modeling
SemBERT	BERT	Output	Semantic Role Labeling of Pre-Training data	Masked Language Modeling
KALM	GPT-2	Combination (Input,Output)	FACC1 and FAKBA entity annotation(Gabrilovich, Ringgaard, and Subramanya 2013)	Language Modeling + Max Margin
Exploiting Structured[...]	BERT	Combination (Input+Output)	ConceptNet	Masked Language Modeling, Max Margin
LiBERT	BERT	Combination (Input, Output)	Wordnet, Roget's Thesaurus	Masked Language Modeling + Max Margin
Graph-based reasoning over heterogeneous external knowledge	XLNet + Graph Convolutional Network	Hybrid (Language Model + Graph Reasoning)	Wikipedia, Concept-Net	Cross Entropy
Commonsense knowledge base completion with structural and semantic context	BERT+Graph Convolutional Net	Hybrid (Language Model + GCN Embeddings)	Atomic,ConceptNet	Binary Cross Entropy
ERNIE 2.0	Transformer-Based Model	Combination (Input+Output)	Wikipedia, Book-Corpus, Reddit, Discovery Data (Various types of relationships extracted from these datasets)	Various tasks, among them Knowledge Masking, Token-Document Relation Prediction, Sentence Distance Task, IR Relevance Task
BERT-MK	BERT	Combination (Architecture+Output)	Unified Medical Language System(Bodenreider 2004)(UMLS)	Masked Language Modeling, Max Margin
K-Bert	BERT	Combination (Input + Architecture)	TBD	Same As BERT
KG-Bert	BERT	Combination (Input + Output)	Freebase, Wordnet, UMLS	Binary + Categorical Cross Entropy
K-Adapter	RoBERTa	Combination (Architecture+Output)	Wikipedia, Dependency Parsing from Book Corpus	Binary Cross Entropy
Cracking the Commonsense Code	BERT	Combination (Input+Output)	N/A:Fine Tuning on RACE dataset subset	Binary Cross Entropy

Table 4
Knowledge Injection Models Overview

Knowledge Injection Approach	Benchmark Name	Base Model	Base Model Benchmark Performance	Knowledge Injected Model Performance	Percent Difference
BERT-CS$_{base}$ (Align, Mask, Select)	GLUE (Average)	Bert-Base	78.975	79.612	0.81%
BERT-CS$_{large}$ (Align, Mask, Select)	GLUE (Average)	Bert-Base-large	81.5	81.45	-0.06%
LIBERT (2M)	GLUE (Average)	BERT Baseline Trained with 2m examples	72.775	74.275	2.06%
SemBERT$_{base}$	GLUE(Average)	BERT-Base	78.975	80.35	1.74%
SemBERT$_{large}$	GLUE(Average)	BERT-Base	81.5	84.262	3.39%
K-Adapter F+L	CosmosQA, TACRED	RoBERTa + Multitask training	81.19,71.62	81.83,71.93	1.54%, 0.95%
Ernie 2.0 (large)	GLUE (Average)	BERT-Base-Large	81.5	84.65	3.87%
BERT-MK	Entity Typing, Rel. Classification (using UMLS)	BERT-base	96.55 ,77.75	97.26,83.02	0.74%,6.78%
K-BERT	XNLI	BERT-base	75.4	76.1	0.93%
Cracking the contextual commonsense code (Bert large + KB+RACE)	MCScript 2.0	BERT-Large	82.3	85.5	3.89
KnowBERT	TACRED	BERT-Base	66	71.5	8%
Common Sense or World Knowledge? (OM-ADAPT 100K)	GLUE (Average)	BERT-Base	78.975	79.225	0.40%
Common Sense or World Knowledge? (CN-ADAPT 50K)	GLUE (Average)	BERT-Base	78.975	79.225	0.32%

Table 5
Knowledge Injection Models Performance Comparison