Artefact Retrieval: Overview of NLP Models with Knowledge Base Access

Anonymous authors

Abstract

Many NLP models gain performance by having access to a knowledge base. A lot of research has been devoted to devising and improving the way the knowledge base is accessed and incorporated into the model, resulting in a number of mechanisms and pipelines. Despite the diversity of proposed mechanisms, there are patterns in the designs of such systems. In this paper, we systematically describe the typology of *artefacts* (items retrieved from a knowledge base), retrieval mechanisms and the way these artefacts are *fused* into the model. This further allows us to uncover combinations of design decisions that had not yet been tried. Most of the focus is given to language models, though we also show how question answering, fact-checking and knowledgable dialogue models fit into this system as well. Having an abstract model which can describe the architecture of specific models also helps with transferring these architectures between multiple NLP tasks.

1. Introduction

For multiple NLP tasks and primarily language modelling, the BERT [Devlin et al., 2019], GPT-2 [Radford et al., 2019], GPT-3 [Brown et al., 2020] and T5 [Raffel et al., 2020] models have seen great success. Their performance is, however, limited on rare or unseen entities [Logan et al., 2019, Schick and Schütze, 2020] and knowledge-intensive NLP tasks [Chen, 2020]. As a consequence, they perform poorly on the task of fact-aware language modelling, where the words to be predicted are named entities [Logan et al., 2019].

Traditionally question answering systems relied heavily on knowledge base access. This has been expanded by other NLP tasks for which models have been proposed (Appendix A) that make use of knowledge bases. Not only do they perform better on e.g. fact-aware language modelling, but they can also provide a degree of explainability (which fact was retrieved) and allow for the model and knowledge base components to be trained separately. The latter is a strong prerequisite for efficient knowledge base manipulation and control, such as removing misinformation or biases [Bhardwaj et al., 2020, Metz, 2019] or adding new information. This is problematic for models like BERT or T5 because their knowledge is tightly coupled with their generative ability, which is stored implicitly in the parameters. Such models make use of parametric knowledge representations as opposed to non-parametric ones, which are typically used in retrieval-based approaches (separation

of the memory and generation components). Separating memory from the generative component allows for more flexible architectures, though at the cost of increased complexity.

While using knowledge bases for language modelling is not immediately intuitive, it is very prevalent in the area of open-domain question answering. Commonly in this case the retrieved knowledge (passed to a generative model) is useful for producing an answer to a question in natural language as the model is trained to condition its generation on the retrieved artefact. Having access to a knowledge base is mandatory for the task of extractive question answering where the output is a span from the available data. Pre-trained language models have also been shown to be able to perform question-answering in the form of being primed by the question [Liu et al., 2019]. This demonstrates the amount of knowledge that is stored in their parameters, though these models are usually outperformed by models with explicit knowledge base access [Fan et al., 2020]. Models in the task of slot-filling and fact-checking also benefit from explicit memory access largely because of the same reasons.

Recently there has been an effort in the community to phrase any NLP task in a unified fashion as a sequence to sequence task (seq-2-seq). Examples of this are phrasing multiple tasks as question answering [McCann et al., 2018], the T5 model [Raffel et al., 2020], the focus on prompt priming [Le Scao and Rush, 2021, Lin et al., 2021] and GPT3 and its few-shot learning approach for multiple tasks [Brown et al., 2020]. There is also a separate line of research that aims to inject commonsense or factual knowledge into language models by training them on data derived from knowledge graphs. An example of this is injecting numerical reasoning skills [Geva et al., 2020] or commonsense knowledge [Bosselut et al., 2019]. This is orthogonal to retrieval-based approaches (such as dialogue generation with commonsense knowledge base access by Young et al. [2018]) on which we focus in this paper.

Contribution. The goal of this paper is to provide a formalism for an abstract model which underlies many specific models for knowledge-intensive tasks with knowledge base access. Similarly to KILT [Fan et al., 2020], we hope that this provides a foundation for future research into task-agnostic memory and model architectures. Lastly, this systematic approach to model description allows us to uncover (1) combinations of mechanisms tried only on one task, which could, however, also work on other NLP tasks and (2) combinations of mechanisms that had not yet been explored for any task.

Outline. In Section 3 we introduce the abstract model underlying other systems together with the different components and the way they can be instantiated. The limitations of this schema and also future work with methods not yet explored are discussed in Section 4. We conclude in Section 5.

An important part is also Appendix A in which different models and approaches for language modelling, question answering, fact-checking and knowledgable dialogue are examined to show how disparate models fit into this schema.

2. Related Work.

A comprehensive overview of earlier work on neural model-based information retrieval systems together with a general introduction has been done by Mitra and Craswell [2017]. More recently models such as BERT have been utilized successfully for the task of retrieval itself [Nogueira et al., 2019b, Soleimani et al., 2020].

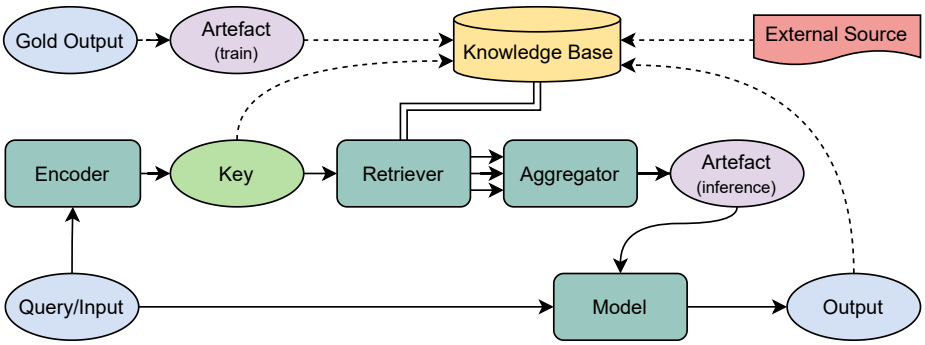

Figure 1: General scheme of NLP models utilizing artefacts by retrieving them from a knowledge base and fusing them into the model in order to produce a better output. Dashed links are utilized only in knowledge base creation and usually not all at once.

The aim of KILT [Fan et al., 2020] is to provide a common knowledge base for a number of different NLP tasks (ranging from question answering to fact verification) and to stimulate research in task-agnostic memory architectures. Reformulating various NLP tasks to all use the same knowledge base format provides a stepping stone for the formalisms of artefact retrieval.

Defining a task-agnostic abstract model is closely related to multi-task learning. The goal of this approach is to improve the performance by training the model on multiple tasks rather than on individual ones [Maillard et al., 2021]. The hope is that representations and generalizations learned for one task will help on another one and vice versa. A strong requirement for this is that the instantiations for different tasks (in the multi-task setup) share significant portions of the model.

An edge-case of this is using a pre-trained BERT model and then fine-tuning it for the new task and/or possibly adding extra layers to match the input and output shapes. Even for BERT, however, it was shown several times [Fan et al., 2020, Liu et al., 2019, Sun et al., 2019, Kim et al., 2019] that training on multiple tasks improves the performance [Aghajanyan et al., 2021]. This would not be possible without a common model shared among the tasks. Further related work is discussed in the respective sections when presenting individual NLP models and how they fit into this schema.

3. Artefact Retrieval

Systems utilizing knowledge bases usually contain up to four main components (excluding the knowledge base itself): **encoder**, **retriever**, **aggregator** and **model**. In other works, some of these parts are joined together, most commonly the encoder, the retriever and the aggregator. We separate them for clarity.

A *query* or an *input* is specific to the given task. For language modelling, it is the previous context, for question answering the question, for slot-filling usually the entity and the relation, and for fact-checking the fact to be verified. In the context of this work, a *knowledge base* is a collection of items, usually (but not necessarily) with a pre-built index that maps keys to values. Prototypically, it is a collection of *documents*, though it can also be a collection of gold training data input-output pairs or a knowledge graph. *Candidates* are values retrieved from the knowledge base, which may be later post-processed (e.g. reranking or averaging) by the aggregator to form an artefact. *key* is an object through which the retriever finds suitable artefacts. Commonly the key is a dense vector representation of the input, though it is not necessarily a vector and may be dependent also on

an intermediate model computation. An *artefact* is an object which is (1) dependent on elements retrieved from the knowledge base (e.g. the concatenation of k retrieved documents) and (2) can be used to improve the performance during training and inference. In the simplest example, it is the retrieved value itself, though it can also be multiple retrieved values or their combination.

On a simplified level, models with artefact retrieval usually work the following way given a query/input q and knowledge base $\mathcal{B}$:

$$\text{[Key] } k = \texttt{Encoder}(q) \tag{1}$$

$$\text{[Candidates] } C = \texttt{Retriever}(\mathcal{B}, k) \tag{2}$$

$$\text{[Artefact] } a = \texttt{Aggregator}(C) \tag{3}$$

$$\text{[Output] } \hat{y} = \texttt{Model}(q, a) \tag{4}$$

A diagram of this pipeline can be seen in Figure 1 (ignore dashed connectors). Pipelines without knowledge base access would only make use of the bottom line (Query/Input → Model → Output). The aggregator is shown to only depend on the retriever output but in some scenarios may also take the key or the query itself as an input.

3.1 Artefact Typology

Most systems used in the literature differ in the encoder, retriever, and model design. We bring attention to four properties that characterize the differences between such systems.

- **Fusion** (early, late, other)
- **Specificity** (sample, task, class)
- **KB source** (train, external, dynamic)
- **Key & value type** (dense, sparse, other)

3.2 Fusion

Formally the model estimates $p(y|x, a)$ where y is the ground-truth output, x is the query/input and a is a retrieved artefact. Fusion Sun et al. [2018] concerns with which point is the artefact made available to the model. It can be presented to the model at the same time as the query/input, e.g. by concatenating x and y (early fusion), just before the output is created by the model (late fusion), or somewhere in between. Formally, the model computation is a composition of functions $f_1, \ldots, f_n$. In the simplest example of feed-forward networks, these correspond to single layers and activation functions and on a higher level, they correspond to whole encoder/decoder blocks. The distinction as to what counts as early and late is not clear and for presentation purposes, we consider early fusion at the level of f_1 and late fusion at the last stage, f_n. These functions themselves may, however, still be composed of multiple others. In **early fusion**, the artefact is the input together with the query to the first function f_1, while in **late fusion** the query is the single input to f_1 and artefact is considered only for f_n.

$$\begin{aligned}
&\text{No fusion:} && f_n \circ \ldots \circ f_2 \circ f_1(q) \\
&\text{Early fusion:} && f_n \circ \ldots \circ f_2 \circ f_1(q, a) \\
&\text{Late fusion:} && f_n(f_{n-1} \circ \ldots \circ f_1(q), a) \\
&\text{Intermediate fusion:} && f_n \circ \ldots \circ f_k(f_{k-1} \circ \ldots \circ f_1(q), a)
\end{aligned}$$

Intuitively it makes more sense to prefer early fusion, to maximize the model's access to extra information [Izacard and Grave, 2020, Karpukhin et al., 2020]. However, this can also be a disadvantage, as the signal from the artefact can get lost during the long computation. In the case of an artefact which is the gold output of a similar query from the training data, later fusion makes more sense. This also allows for a degree of explainability. By examining the forward pass of the last function we could determine what the contribution of the artefact was to the produced output.

The decision of how late the fusion should be depends heavily on the artefact type. The application of every function in the chain of computation projects the input to some latent space. The final function f_n is special because it projects the output of previous functions to the space of possible outputs for the whole model. In this space, there is the prediction $\hat{y}$ and also the true output y. The loss function used during training is defined in this space. Adding an artefact should ideally move the prediction in the output space closer to the correct output. Assuming c is the intermediate computation and there are two (overloaded) functions that produce a prediction: $\hat{y}_x = f_n(c)$ and $\hat{y}_a = f_n(c, a)$. In circumstances in which adding the artefact helps, $L(\hat{y}_a, y) < L(\hat{y}_x, y)$, where L is a loss function such as cross-entropy. This is illustrated in the first row of Figure 2 for $n = 2$.

Assume that we can create an inverse of the last projection and see where the correct output lies in the intermediate representation. There may be multiple such $c_t : f_n(c_t) = y$ or none, if too much information was lost by the first projection f_1. Further, assume that there is always at least one such c_t. We may then define an intermediate loss L_i for each model computation by measuring the distance of the partial computations to the back-projection. Similarly to late fusion, we consider two overloaded functions that produce the intermediate representation $c_x = f_1(q)$ and $c_a = f_1(q, a)$. Adding the artefact then ideally moves the intermediate representation closer to the back-projection and reduces the intermediate loss: $L^i(c_a, c_t) < L^i(c_x, c_t)$.[1]

This is illustrated in the second row of Figure 2, which depicts a model with only two computational steps: $f_2 \circ f_1$. Early fusion (second row) adds the artefact to f_1, while late fusion adds it in the next step. For simplicity in the figure, we consider the standard L^2 distance loss between the points. In both cases, adding the artefact reduced the target loss. For early function, the intermediate loss was also reduced and the target loss was lower. This does not always happen and complex computations may still at some point project the intermediate computation to the same point regardless of whether an artefact was added earlier or not.

Even though the invertibility of projections in this paper is only used as an illustration for the artefact fusion and an intermediate loss does not have to be defined in practice, invertible neural networks are an ongoing topic of research [Ardizzone et al., 2018, Behrmann et al., 2021]. The combination of artefact retrieval and invertible neural networks has not yet been explored to our knowledge.

Fusion Mechanisms. There are common patterns with respect to artefact fusion. *Priming*, as used by many question-answering systems [Guu et al., 2020, Lewis et al., 2020, Karpukhin et al., 2020], is an example of an early fusion technique. In the simplest scenario, the retrieved paragraphs are prepended to the actual question and put together to the model as one input.

An extension to this is *key-aware priming*, in which the keys of the retrieved values are also put to the model in an early fusion. The motivation for this in the context of question answering

1. In case of multiple elements that map to y, we can define a loss that considers the minimum distance to any of them: $L^{i'} = \min L^i(c, c_t)$. If there is no such element in the projection space, then we may consider the elements that project close to the target $C_t = \arg\min L(f_2(c), y)$.

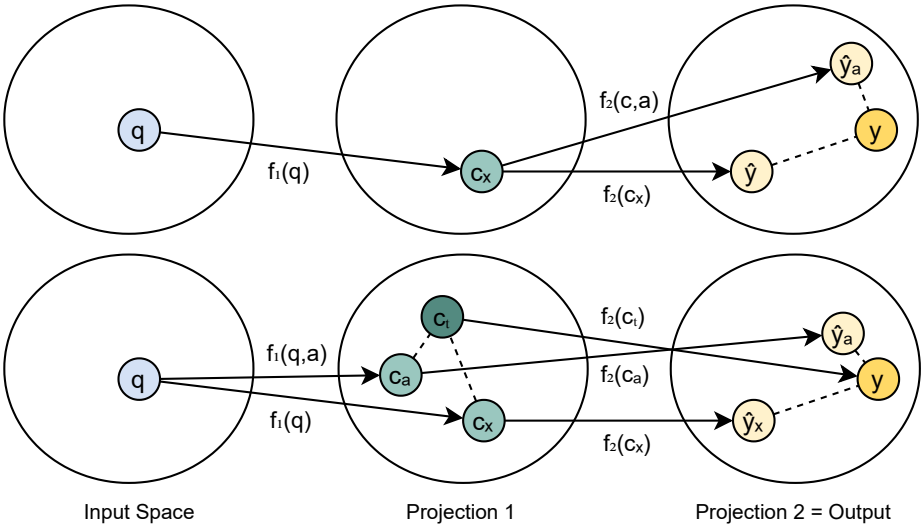

Figure 2: Example of how late (top) and early (bottom) fusions affect the projections into intermediate spaces. The lengths of dashed lines correspond to the loss (longer is greater loss). Adding an artefact a to the computation decreases the loss in both early and late fusions.

is that the keys may be encoded questions and the retrieved value is simply the answer. Then at inference time, the system can receive a specific question of which the *negation* is stored in the knowledge base. Knowing the answer to the negation of a question may be beneficial in answering the original question, though knowing the question to which the answer corresponds is vital. Key-aware priming can be also conceptualized as simply storing the key-value pairs as: (key, key+value) and then retrieving the second item of the tuple by the retriever which would also retrieve the key.

For multi-class predictors[2] like language models it is possible to consider the training data as a knowledge base and when retrieving, suggest k nearest neighbours. This can be followed up by averaging the probabilities of the output classes based on the neighbour distance (performed by the aggregator). For this late fusion approach, *output gating*, the last function f_n can either be a simple convex combination $\hat{y} = \lambda \cdot c + (1 - \lambda) \cdot a$ or a slightly more complex gating mechanism, which computes λ dynamically based on the current input [Khandelwal et al., 2019, Yogatama et al., 2021]. Both are discussed in detail in Appendix A.1.

Filtration/masking can be seen as super late fusion. For slot-filling or fact-aware language modelling, the retriever may provide relevant documents out of which a term incidence vector[3] is constructed. The output of the model is then masked and normalized to prevent outputs unrelated to the query ($\hat{y} = \frac{c \odot a}{|c \odot a|}$). An example of this is to constrain beam-search decoding to valid names only as done in the recently proposed GENRE model for entity retrieval tasks [De Cao et al., 2020].

3.3 Specificity

The property of specificity is not directly bound to the pipeline of models using artefact retrieval. In a wider sense of an artefact being something that improves inference-time performance, the

2. Single-token predictors can be treated as multi-class predictors with the classes being the vocabulary.
3. A vector v of the length of the vocabulary filled with ones and zeroes. Then $v_i = 1$ if the i-th term occurred in the retrieved documents, otherwise zero.

retriever can produce artefacts that help the model, though which are not necessarily dependent on a knowledge base. In this specific case, the retriever can simply ignore the conditioning on the knowledge base and can be directly fused together with the encoder. In the usual scenario, the encoder makes decisions based on the current query. It may, however, ignore significant parts of the query or be conditioned on a specific task and not take the query into consideration at all.

For the task of open-domain question-answering, retrieving documents based on their embedding's inner product similarity to the query embedding [Karpukhin et al., 2020] is an example of the sample-specific property. Presumably significant portions of the query are considered when producing the key for the retriever and different queries produce different artefacts (sets of documents).

Pre-trained language models, such as BERT or GPT-3, are either fine-tuned on a new task or are primed on prompts either crafted by humans [Haley, 2020, Misra et al., 2020, Petroni et al., 2019] or found automatically [Shin et al., 2020, Jiang et al., 2020]. These prompts (also called stimuli, constraints, or demonstrations) are then dependent only on the specific task at hand and nothing else. As an example, Radford et al. [2019] affix the prompt `TL;DR` after a passage to induce summarization behaviour. Crafting task-specific artefact, specifically prompts, is known as *prompt programming* or *prompt engineering* [Reynolds and McDonell, 2021] and allows the models to perform a wide variety of NLP tasks.

The prompts may be in a structured form with only some parts of it being considered by the encoder. In the case of fact retrieval [Shin et al., 2020], the input is in the form of (subject, relation). It is then possible to have different prompts based only on the relation (class). Different conditioning which results in distinct encoder specificities are summarized in the following equations:

Query/input $q = \{q_1, q_2, \dots q_n\}$	
Sample-specific:	$\texttt{Encoder}(q, \text{task})$
Task-specific:	$\texttt{Encoder}(\text{task})$
Class-specific:	$\texttt{Encoder}(q' \subset q, \text{task})$

3.4 Knowledge base source

The way the knowledge base is created is tightly coupled with its fusion purpose. There are, however, three common underlying patterns that describe the possible creation schemes. In the general architecture overview of pipelines with artefact retrieval in Figure 1, they are depicted with dashed lines. The most common example is in the task of open-domain question-answering, where the knowledge base, a collection of documents, is a vital component to produce the answer. A collection of documents or knowledge graph structures are examples of knowledge bases populated externally.

A specific contrast to this is having access to the training data during inference. Here, the knowledge base is created as:

$$\mathcal{B} = \{(\ \texttt{Encoder}_{\text{query}}(q), y) | (q, y) \in D_{\text{train}}\}$$

At inference time, the model can then refer back to examples it has already seen with the retriever providing e.g. a weighted combination of nearest neighbours [Khandelwal et al., 2019] or any other aggregation of found samples. This memorization based approach is usually used with late fusion because, at the end of the computation, the model has access to its proposed output and also to the gold outputs of similar queries. These can be then combined together using e.g. gating.

Finally, the knowledge base can also be created dynamically during inference, such as by keeping a track of already predicted words in language modelling. This knowledge base creation is depicted by the dashed line from *Output* to *Knowledge Base* in Figure 1. This usage of knowledge base lies, however, slightly out of the traditional meaning and understanding of the term.

3.5 Key & value type

Commonly the knowledge base is either (1) a collection of candidates (e.g. documents, paragraphs or gold outputs from the training dataset), in which case they are retrieved by similarity to the query or (2) a more structured source of information, such as a knowledge graph.

Vector Space Model. Traditionally, question-answering and other NLP systems would use a vector space model for the values (documents) and queries. Such representations usually describe the algorithm for computing the vector representations for both the document and the query. These include standard statistical methods such as TF-IDF [Salton and McGill, 1986] or BM25 [Robertson et al., 1995]. Their disadvantage is that the resulting vectors have the dimension the same as the number of words in the vocabulary. A contrast to this is dense vector representation. LSA [Dumais, 2004] is one of the oldest of such methods, but it also includes learned embedding methods based on word2vec [Mikolov et al., 2013], doc2vec [Le and Mikolov, 2014], docT5query [Nogueira et al., 2019a], CLSM [Shen et al., 2014] or BERT [Devlin et al., 2019]. The latter allows for the use of the gradient signal of the latent variable (documents) to fine-tune the index, as done asynchronously by Guu et al. [2020].

Splitting the retrieval mechanism into an `Encoder` and `Retriever` is not strictly necessary and only follows a common pattern found in many systems. This is usually done for training and speed purposes because e.g. recomputing embeddings for all documents bears too high of a cost for just single retrieval. In the vector space model, an index (keys for the documents) would be built usually in the pre-processing phase and the knowledge base would constitute a mapping from this index to the documents. Then a vector similarity, usually the cosine similarity, is used which results in the following pipeline:

$$\begin{aligned}
\mathcal{B} &= \{(\texttt{Encoder}_{\text{doc}}(d), d) | d \in C\} \\
k &= \texttt{Encoder}_{\text{query}}(q) \\
a &= \texttt{Retriever}(B, k) = \arg\max_{(v,d)\in B} \text{sim}(k, v)
\end{aligned}$$

Usually, instead of retrieving just the $\arg\max$, the top-k scoring documents are returned. When using cosine similarity with normalized vectors, we may substitute the similarity by the inner product $k \odot v$, which can be approximated efficiently by Maximum Inner Product Search (Approximate Nearest Neighbour Search) algorithms in sublinear query time [Johnson et al., 2019, Guo et al., 2020, Yang et al., 2021]. Instead of documents, which are replaced by paragraphs or spans of texts, depending on the specific work. It is also possible to store query representation from the training data, as described in Section 3.4. The setup is then similar as for document retrieval, with the exception that instead of documents, the retrieved values are gold outputs from the training dataset. The inner product search remains the same.

Knowledge Graphs. A vastly different approach has to be chosen when the knowledge base is more structured, such as a knowledge graph. In this case, the knowledge base is usually a directed labelled graph: a set of triples (parent, relation, entity). The parent and the entity are elements of a

fixed set of entities and the relation is an element of a fixed set of possible relations (usually orders of magnitudes smaller than the set of entities).

Question-answering over knowledge graphs is a specific, vastly explored [Bao et al., 2016, Lukovnikov et al., 2017] subfield of non-extractive open-domain question answering. Popular knowledge graphs, based on Wikipedia, include DBpedia [Auer et al., 2007], Wikidata [Vrandečić and Krötzsch, 2014] and YAGO [Rebele et al., 2016]. A comparison of them has been composed by Ringler and Paulheim [2017]. Specific datasets for testing question answering over knowledge graphs are WebQuestions [Berant et al., 2013] and SimpleQuestions [Bordes et al., 2015]. Knowledge graphs are, however, also used for fact-aware language modelling [Logan et al., 2019] or fact-checking [Ciampaglia et al., 2015, Tchechmedjiev et al., 2019]. Simple slot-filling without any reasoning (either multi-source facts or resolving aliases) would be a trivial task. It is then used in the opposite direction, for automatically creating knowledge graphs [Yu et al., 2014]. These knowledge bases found their use even in more distant tasks, such as Word Sense Disambiguation [Bevilacqua and Navigli, 2020]. Knowledge graph retrieval is then reduced to finding variables given constraints (a subgraph with free variables that needs to be matched over the knowledge base).

The query construction (which entity and relation should be selected) is handled by the `Encoder` and is commonly limited to a single restriction (single triplet). The `Retriever` is built on top of a database, which stores the knowledge graph. This is vastly faster than MIPS and computing the index in vector space models, but at the cost of a more complicated encoder and constraints to the type of knowledge stored.

4. Discussion

Descriptions of specific systems and approaches for language modelling, knowledge graph language modelling, question answering, fact-checking and knowledgable dialogue in the paradigm of artefact retrieval can be found in Appendix A. Summary charasteristics of the different models is shown in Table 1.

4.1 Limitations

Even though the abstract model presented in this paper is versatile in the description of various models, it may be lacking in subtle ways to describe more complex systems. This complexity may be included in one of the components in order to make it fit, though this obscures clear understanding and comparison across models. An example of this is the implicit aggregation in the form of re-ranking as the final step of the DPR question answering system [Karpukhin et al., 2020]. We include it in the core model component, though a better understanding of this model would be achieved by adding an additional aggregator element to the abstract model. This would, however, conflict with most other systems which would not make use of this and for full description, this part of the pipeline would have to be set to simply identity or NOP. This has happened with other models, as documented in Table 1.

4.2 Future work

An immediate observation from Table 1 is that models for knowledge-intensive tasks, e.g. question answering, do not tend to utilize late fusion nor training-time knowledge base sources yet. In the opposite directions, experiments could be made with early fusion for language models. They could also make greater use of knowledge bases with external sources.

Model	Fusion	KB Source	Keys	Values	Aggregation
k-NN LM [Khandelwal et al., 2019]	Very late Static convex combination	Train-time	Prefix embedding, L^2	Target word	Softmax
Continuous Cache LM [Grave et al., 2016]	Very late Static convex combination	Dynamic	Prefix embedding, inner product	Target word	Softmax
Dynamic Gating LM [Yogatama et al., 2021]	Late Dyn. convex combination	Train-time	Prefix encoding, inner product	Target word	Softmax sum
Knowledge Graph LM [Logan et al., 2019]	Intermediate Constraints	External	Entity + relation Discrete structure	Matching entity	None
Dense Passage Retrieval [Karpukhin et al., 2020]	Early Input	External	Passage embd., inner product	Passages	None
FAKTA [Nadeem et al., 2019]	Early Input	External Online	Condensed query	Documents	Re-ranking, Filtering
Wizards of Wikipedia [Dinan et al., 2018]	Intermediate Addition	External	Context & topic, inverted index	Passages	Attention (topic)

Table 1: Categorization of described NLP systems in terms of the artefact retrieval typology. *Fusion* describe both where it occurs and what mechanism it employs, *Keys* describes not only the key type but also the retrieval mechanism (e.g. metric).

Invertible neural networks should be studied in the context of artefact retrieval to determine the exact properties of different fusion mechanisms (e.g. quantifying the discussion in Section 3.2).

Finally, the area of using multiple separate artefact retrieval pipelines is currently unexplored. This would mean utilizing either (1) multiple knowledge bases with retrieval systems based on the same principle, e.g. dense vector representations or (2) a vastly different knowledge bases with separate retrieval systems, e.g. one with dense vector representation and the other with a knowledge graph.

5. Summary

In this paper, we presented an abstract model which can describe various systems for NLP tasks utilizing knowledge bases. The pipeline consists of multiple components: encoder, retriever, aggregator and the core model, typically a generative model, itself. The abstract model description leads to several key characteristics: fusion, specificity, knowledge base source and key & value type which share important properties across approaches to model design.

In the Appendix we showed how several increasingly complex language modelling models proposed in the recent years can be described in this paradigm. This is followed by model descriptions in the context of question answering, fact-checking and knowledgable dialogue.

Acknowledgments

Omitted for blind review.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*, 2021.

Lynton Ardizzone, Jakob Kruse, Sebastian Wirkert, Daniel Rahner, Eric W Pellegrini, Ralf S Klessen, Lena Maier-Hein, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. *arXiv preprint arXiv:1808.04730*, 2018.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

Alexei Baevski and Michael Auli. Adaptive input representations for neural language modeling. *arXiv preprint arXiv:1809.10853*, 2018.

Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, (2): 179–190, 1983.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514, 2016.

Jens Behrmann, Paul Vicol, Kuan-Chieh Wang, Roger Grosse, and Jörn-Henrik Jacobsen. Understanding and mitigating exploding inverses in invertible neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR, 2021.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

Michele Bevilacqua and Roberto Navigli. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2864, 2020.

Rishabh Bhardwaj, Navonil Majumder, and Soujanya Poria. Investigating gender bias in bert, 2020.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, 2019.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

Charles L Chen. *Neural Network Models for Tasks in Open-Domain and Closed-Domain Question Answering*. Ohio University, 2020.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, 2017.

Giovanni Luca Ciampaglia, Prashant Shiralkar, Luis M Rocha, Johan Bollen, Filippo Menczer, and Alessandro Flammini. Computational fact checking from knowledge networks. *PloS one*, 10(6): e0128193, 2015.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*, 2020.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*, 2018.

Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*, 2015.

Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

Fabio Petroni1 Aleksandra Piktus1 Angela Fan, Patrick Lewis, Majid Yazdani1 Nicola De Cao James, Thorne4 Yacine Jernite5 Vladimir Karpukhin, Jean Maillard1 Vassilis Plachouras1 Tim Rocktäschel, and Sebastian Riedel. Kilt: a benchmark for knowledge intensive language tasks. 2020.

Mor Geva, Ankit Gupta, and Jonathan Berant. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, 2020.

Edouard Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

Edouard Grave, Moustapha Cissé, and Armand Joulin. Unbounded cache model for online language modeling with open vocabulary. *arXiv preprint arXiv:1711.02604*, 2017.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pages 3887–3896. PMLR, 2020.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020.

Coleman Haley. This is a bert. now there are several of them. can they generalize to novel words? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 333–341, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*, 2020.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2020.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*, 2019.

Hyun Kim, Joon-Ho Lim, Hyun-Ki Kim, and Seung-Hoon Na. Qe bert: bilingual bert using multi-task learning for neural quality estimation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, pages 85–89, 2019.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. Hurdles to progress in long-form question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4940–4957, 2021.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

Teven Le Scao and Alexander M Rush. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, 2021.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, 2019.

Nayeon Lee, Belinda Z Li, Sinong Wang, Wen-tau Yih, Hao Ma, and Madian Khabsa. Language models as fact checkers? *arXiv preprint arXiv:2006.04102*, 2020.

Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

Junyang Lin, Rui Men, An Yang, Chang Zhou, Yichang Zhang, Peng Wang, Jingren Zhou, Jie Tang, and Hongxia Yang. M6: Multi-modality-to-multi-modality multitask mega-transformer for unified pretraining. 2021.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, 2019.

Robert Logan, Nelson F Liu, Matthew E Peters, Matt Gardner, and Sameer Singh. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971, 2019.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220, 2017.

Jean Maillard, Vladimir Karpukhin, Fabio Petroni, Wen-tau Yih, Barlas Oguz, Veselin Stoyanov, and Gargi Ghosh. Multi-task retrieval for knowledge-intensive tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1098–1111, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.89. URL https://aclanthology.org/2021.acl-long.89.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Cade Metz. We teach ai systems everything, including our biases. *The New York Times*, 2019.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Kanishka Misra, Allyson Ettinger, and Julia Rayz. Exploring bert's sensitivity to lexical cues using tests from semantic priming. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4625–4635, 2020.

Bhaskar Mitra and Nick Craswell. Neural models for information retrieval. *arXiv preprint arXiv:1705.01509*, 2017.

Moin Nadeem, Wei Fang, Brian Xu, Mitra Mohtarami, and James Glass. Fakta: An automatic end-to-end fact checking system. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 78–83, 2019.

Yixin Nie, Haonan Chen, and Mohit Bansal. Combining fact extraction and verification with neural semantic matching networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6859–6866, 2019.

Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. From doc2query to doctttttquery. *Online preprint*, 2019a.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*, 2019b.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. Language models as knowledge bases? 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 2463–2473, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *International semantic web conference*, pages 177–185. Springer, 2016.

Nils Reimers and Iryna Gurevych. The curse of dense low-dimensional information retrieval for large index sizes. *arXiv preprint arXiv:2012.14210*, 2020.

Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7, 2021.

Daniel Ringler and Heiko Paulheim. One knowledge graph to rule them all? analyzing the differences between dbpedia, yago, wikidata & co. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 366–372. Springer, 2017.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

Timo Schick and Hinrich Schütze. Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8766–8774, 2020.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 101–110, 2014.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. 2020.

Amir Soleimani, Christof Monz, and Marcel Worring. Bert for evidence retrieval and claim verification. *Advances in Information Retrieval*, 12036:359, 2020.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. Open domain question answering using early fusion of knowledge bases and text. 2018.

Andon Tchechmedjiev, Pavlos Fafalios, Katarina Boland, Malo Gasquet, Matthäus Zloch, Benjamin Zapilko, Stefan Dietze, and Konstantin Todorov. Claimskg: A knowledge graph of fact-checked claims. In *International Semantic Web Conference*, pages 309–324. Springer, 2019.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, 2018.

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

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Sixing Wu, Ying Li, Dawei Zhang, Yang Zhou, and Zhonghai Wu. Topicka: Generating commonsense knowledge-aware dialogue responses towards the recommended topic fact. In *IJCAI*, volume 2020, pages 3766–3772, 2020.

Brian Xu, Mitra Mohtarami, and James Glass. Adversarial domain adaptation for stance detection. *arXiv preprint arXiv:1902.02401*, 2019.

Shuo Yang, Tongzheng Ren, Sanjay Shakkottai, Eric Price, Inderjit S Dhillon, and Sujay Sanghavi. Linear bandit algorithms with sublinear time complexity. *arXiv preprint arXiv:2103.02729*, 2021.

Dani Yogatama, Cyprien de Masson d'Autume, and Lingpeng Kong. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373, 2021.

Tom Young, Erik Cambria, Iti Chaturvedi, Hao Zhou, Subham Biswas, and Minlie Huang. Augmenting end-to-end dialogue systems with commonsense knowledge. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Dian Yu, Hongzhao Huang, Taylor Cassidy, Heng Ji, Chi Wang, Shi Zhi, Jiawei Han, Clare Voss, and Malik Magdon-Ismail. The wisdom of minority: Unsupervised slot filling validation based on multi-dimensional truth-finding. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1567–1578, 2014.

Appendix A. System Description

This section provides a brief survey of different NLP systems and how their retrieval and fusion mechanisms work in the paradigm of artefact retrieval. Summaries with respect to this typology are shown in Table 1.

A.1 Language Modelling

The common goal of language modelling is to predict the next word (distribution) given the context of the previous tokens [Bahl et al., 1983]. The performance is measured by perplexity: the inverse of the geometric average of the whole sequence probability (lower is better). The input to the model is the text prefix and the output of the following word. The prefix is usually the object that is passed to the encoder.

k-Nearest Neighbours. The language model proposed by Khandelwal et al. [2019] utilizes memorization of the training data to decrease the perplexity. In this approach, the authors first compute representations of all sentence prefixes (keys) and store them together with the following word (values). They then use this for the next word prediction by softmaxing negative L^2 distances[4] of 1024 neighbours. The representations (1024 dimensions) are the output of the last self-attention layer of the trained Transformer-based language model [Vaswani et al., 2017, Baevski and Auli, 2018]. The importance of the retrieved artefact in the output is determined by the manually set hyperparameter $\lambda \in [0, 1]$, resulting in linear (convex) interpolation.

The symbolic working of the model is shown in the following set of equations which is adapted from the original paper (the aggregator is the softmax function; $\text{LM}^{rep.}(X_{<i})$ is the vector representation of the prefix $X_{<i}$ by the trained language model).

$$
\begin{aligned}
&\texttt{Encoder:} && k = \text{LM}^{rep.}(X_{<i}) \\
&\texttt{Knowledge base:} && \mathcal{B} = \{(\texttt{Encoder}(X_{<i}), X_i) | X \in D_{\text{train}}, i < |X|\} \\
&\texttt{Retriever:} && S = \{(r, v) | (r, v) \in \mathcal{N}^{L^2}_{1024}(k)\} \\
&\texttt{Aggregator:} && p_a(\hat{X}_i) \propto \sum_{(r,v)\in S} \exp(-||r - k||_2 \cdot v) \\
&\texttt{Model:} && p_m = \lambda \cdot p_a + (1 - \lambda) \cdot \text{LM}(X_{<i})
\end{aligned}
$$

The authors also showed that using the training data as a knowledge base outperforms using them for training. This approach was built on the work of Grave et al. [2017] which, however, does not use the state-of-the-art Transformer-based language model but an RNN-based one. Furthermore, they use the inner product (IP) for similarity instead of the L^2 distance.

Continuous Cache. The previous approach is a continuation of using local vocabulary cache for language modelling, as proposed by Grave et al. [2016]. The effect of specific history size is examined as well. The usage of cache can be interpreted as using a small local dynamic knowledge base that is being updated after every prediction. This is motivated by the fact that especially rare words tend to occur more probably than by overall uniform distribution, given that they appeared

4. The negative L^2 distance is used instead of the common inner product as the similarity measure. This was shown empirically by the authors to work better.

in recent history. The LSTM [Hochreiter and Schmidhuber, 1997] hidden state size is again 1024-dimensional, although this time, it is not used for retrieval but only by the aggregator. For the fusion, the authors propose two methods: (1) linear interpolation, as seen in the previous language model and (2) joint softmax over the artefact and the language model output distribution.

The simplest of the two proposed models (linear combination) is symbolically described in the following equations (the aggregator is the softmax function; θ is a hyperparameter of the cache distribution).

$$
\begin{aligned}
&\texttt{Encoder:} && k = \text{LSTM}^{hid.}(X_{<i}) \\
&\texttt{Retriever:} && S = \bigcup_{n<N} B_{i-n} \\
&\texttt{Aggregator:} && p_a(\hat{X}_i) \propto \sum_{(r,v)\in S} \mathbb{1}_{v=\hat{X}_i} \exp(\theta \cdot k^T r) \\
&\texttt{Model:} && p_m = \lambda \cdot p_a + (1-\lambda) \cdot \text{LSTM}(X_{<i}) \\
&\texttt{Knowledge base update:} && \mathcal{B}_{i+1} = \{(k, \arg\max \hat{v})\}
\end{aligned}
$$

Dynamic Gating. The previously described language models use very late fusion, which is controlled by hyperparameter λ. Yogatama et al. [2021] propose an approach in which the model itself determines this parameter (now a vector) dynamically based on the current sample. The knowledge base (called long-term memory) is constructed in the same way from the training data as in k-Nearest Neighbours LM. They also introduce short-term memory in the model, which is able to attend to extended local context. Another difference is using two different models for the encoder (vanilla transformer) and the language model itself. The keys are 512-dimensional vectors and the retriever uses inner product for nearest neighbour lookup.

The following set of equations, adapted from the original paper, describes the behaviour of the model with respect to the long-term (episodic) memory:

$$
\begin{aligned}
&\texttt{Encoder:} && k = \text{Transformer}^{rep.}(X_{<i}) \\
&\texttt{Knowledge base:} && \mathcal{B} = \{(\texttt{Encoder}(X_{<i}), X_i) | X \in D_{\text{train}}, i < |X|\} \\
&\texttt{Retriever:} && S = \mathcal{N}_4^{IP}(k) \\
&\texttt{Aggregator:} && p_a(v') = \sum_{(r,v)\in S} \text{softmax}(v,k) \cdot v' \\
&\texttt{Model:} && g = \sigma(w_g^T k) \\
& && z = (1-g) \odot p_a + g \odot \text{LM}(X_{<i}) \\
& && p_m = \text{softmax}(z, W)
\end{aligned}
$$

Since the artefact is processed slightly more by the model compared to the other systems which we describe, e.g. Grave et al. [2016], Khandelwal et al. [2019], we classify it as *late* fusion as opposed to *very late* fusion. This dynamic gating was also applied to the cache knowledge base structure by Merity et al. [2016].

Knowledge Graph LM. All of the previous models utilized train-time knowledge base creation with keys representing the contexts and values of the next word predictions. Logan et al. [2019] utilize knowledge graphs to specifically increase language modelling performance on named entities.

We only describe the simplified retrieval component and omit the details, such as entity rendering and aliases.

At every position (for every query), the model makes a decision as to what type the following token will be: (1) non-entity, (2) unrelated entity or (3) related entity. In the first case, the token is predicted by the standard language model. The other two, however, utilize the knowledge graph access.

Formally there are two knowledge bases used: a static knowledge graph $\mathcal{KG}$ and a local graph $\mathcal{KG}_{<i}$ containing already encountered entities and their relations in the prefix. The model uses an LSTM unit, the hidden state of which is split into three components $[h_x; h_p; h_r]$ used for: (1) the token type decision, (2) parent entity prediction and (3) relation prediction. When the model makes a decision to predict an unrelated entity, it is sampled by a simple projection to the entity embedding space:

$$\text{softmax}(v_e \cdot (h_p + h_r))$$

The predicted entity e is then added together with its immediate neighbours to the local graph:

$$\mathcal{KG}_{<i+1} = \mathcal{KG}_{<i} \cup \{(e, r, x) | (e, r, x) \in \mathcal{KG}\}$$

In case of a related entity, the model first predicts the parent, then the relation and finally the entity itself. When there are multiple entities in the local graph matching the restriction of the parent and the relation, it is sampled at random. The following set of equations describes the model behaviour in case of a new entity.

$$\begin{aligned}
&\texttt{Encoder:} && p_p(e_p) = \text{softmax}(v_p \cdot h_p) \\
& && p_r(r) = \text{softmax}(v_r \cdot h_p) \\
& && \text{constrained by } \exists e : (e_p, r, e) \in \mathcal{KG}_{<i} \\
&\texttt{Retriever:} && p_e \underset{\text{sample}}{\sim} \{(e_p, r, e) | (e_p, r, e) \in \mathcal{KG}_{<i}\} \\
&\texttt{Model:} && p_m = \text{renderer}(p_e) \\
&\texttt{Update knowledge base:} && \mathcal{KG}_{<i+1} = \mathcal{KG}_{<i} \cup \{(e, r, x) | (e, r, x) \in \mathcal{KG}\}
\end{aligned}$$

The main motivation for this approach is factual correctness in language modelling. Furthermore, it grants a higher degree of explainability and also allows for tighter manipulation and control of the data the model is working with (changing an entity in a relation has a direct impact on the produced output).

A.2 Question Answering

There has been a lot of progress in question answering systems. However, the task of long-form question answering was recently shown to be problematic even with state-of-the-art Transformer-based systems [Krishna et al., 2021]. Furthermore, it has been suggested that dense representations are inadequate when scaled up to large index sizes [Reimers and Gurevych, 2020]. Despite that, the current trend is based on this retrieval mechanism [Lee et al., 2019, Guu et al., 2020, Lewis et al., 2020].

Dense Passage Retrieval. We focus on DPR [Karpukhin et al., 2020] which describes a prototypical question-answering system built with knowledge base access and dense document embeddings.

DPR uses two BERT models to compute the document and question embeddings at the `[CLS]` token (768-dimensional). They are then fine-tuned so that the inner product (or L^2 distance or cosine similarity) of these two vectors is a good measure for document relevancy. The retrieved passages are reranked by a combination of the original similarity and the BM25 model [Robertson et al., 1995]. For every retrieved passage, the probability of a span (up to fixed maximum length) being selected is computed as the product of two tokens (representation from another BERT model) being the start and end ones, respectively. Reranking of the answers is done implicitly by choosing the span with the highest probability across all spans. The maximum similarity search is approximated to make it computationally feasible using FAISS [Johnson et al., 2019].

`Encoder:`	$k = \text{BERT}_{\text{query}}(q)$
`Knowledge base:`	$\mathcal{B} = \{(\text{BERT}_{\text{doc}}(d) \mid d \in KB\}$
`Retriever:`	$C = \arg\max^{n}_{d \in B} \text{sim}(q, d)$
	$C' = \text{Reranker}(C)$
	score given by: $\text{BM25}(q, d) + \lambda \cdot \text{sim}(q, d)$
`Model:`	$P_{\text{start},i}(s) = \text{softmax}(C'_i w_{\text{start}})_s$
	$P_{\text{end},i}(t) = \text{softmax}(C'_i w_{\text{end}})_t$
	$P_{\text{selection}}(i) = \text{softmax}(C' w_{\text{selection}})_i$

For every passage, $P_{\text{selection}}(i)$ is the score that this passage contains the answer and for every span $P_{\text{start},i}(s) \cdot P_{\text{end},i}(t)$ is the score of a single span (s to t) in a passage. In this specific model, the aggregator simply passes on all the retrieved passages, though some models simply concatenate all passages and pass it to the model as single string input [Izacard and Grave, 2020].

A.3 Fact Checking

Despite the research in using the masked language modelling capabilities of pre-trained models for fact checking [Lee et al., 2020], traditional methods rely on external knowledge base access to verify claims. This has the advantage of higher explainability, which is essential in the context of fact-checking. The pipelines then usually consist of evidence retrieval and verification which maps well to our proposed abstract artefact retrieval model.

The release of FEVER: a large-scale dataset for fact extraction and verification [Thorne et al., 2018] introduced a benchmark which was utilized by multiple works, such as by Nie et al. [2019] or Nadeem et al. [2019]. We describe the components of the latter.

FAKTA The query encoder of FAKTA filters out words that are not verbs, nouns or adjectives. And appends to it named entities from the claim. The retriever then fetches relevant documents to this query and if none are found, relaxes the query by incrementally omitting the last tokens. The model uses several sources of knowledge bases, some of them external such as Google, Bing and Yahoo). This step is followed up by a re-ranker based on important words (determined by their

POS tags) from the document title and the claim. Relevant documents are further filtered by a CNN based on the work of Xu et al. [2019]. Another CNN network determines the document stance (agree, disagree or discuss). Given a claim q the pipeline is as follows:

$$
\begin{aligned}
&\texttt{Encoder:} && k = \text{Filter}_{\text{N,V,ADJ}}(q) \frown \text{Filter}_{\text{named entity}}(q) \\
&\texttt{Retriever:} && C = \text{SearchEngine}(k_{<i}) \\
& && \text{(such that } i \text{ maximal and results non-empty)} \\
& && C' = \text{Reranker}(C) \\
& && C'' = \{d | \text{CNN}_{\text{relevancy}}(d) = \text{relevant})\} \\
&\texttt{Model:} && o = \{\text{CNN}_{\text{stance}}(d) | d \in C''\}
\end{aligned}
$$

The output of the model is a set of relevant documents with labels regarding the claim. These inferences can be averaged into a single number which describes how likely it is that the claim is factually true.

A.4 Knowledgable Open Dialogue

Finally, we focus on open dialogue with utterance generation dependent on a knowledge base. This improves the insufficiencies of parametric model memory in open-ended dialogue centred around facts. It also increases the explainability of each utterance as it can be traced (with various scores) to the source in the knowledge base. We will focus on the task and model description by Dinan et al. [2018] though there have also been previous works by Dodge et al. [2015] or in the context of a commonsense knowledge base by Young et al. [2018], Wu et al. [2020].

Wizards of Wikipedia In the pipeline proposed by Dinan et al. [2018], the pool of retrieved documents provided by the method of Chen et al. [2017] in the form of a term matrix and a simple inverted index lookup. This pool is fixed so that the results are comparable to their experiments with human annotators. This part could however also be automated and replaced by a more advanced model which could be finetuned. The artefact in this pipeline is the weighted average of retrieved documents based on the dot product between the encoded representation and the encoded topic.

Given the conversation history $(q_1, q_2, \ldots, q_n)$ and topic t the first model proposed by Dinan et al. [2018] can be described as:

$$
\begin{aligned}
&\texttt{Encoder:} && k = \bigcup_{x \in \{q_{n-1}, q_n, t\}} \text{TermVector}(x) \\
&\texttt{Retriever:} && C = \text{InvertedIndexLookup}_7(k) \\
& && C' = \{\text{TransformerEnc}(\text{Title}(d) \frown \text{Paragraphs}(d)_1) | d \in C\} \\
&\texttt{Aggregator} && a = \sum_{d' \in C'} d \times (d \cdot \text{TransformerEnc}(t)) \\
&\texttt{Model:} && o = \text{TransformerDec}(\text{TransformerEnc}(q) + a)
\end{aligned}
$$

The next utterance knowledge source is then dependent on the last two utterances and the topic. The artefact is then added to the context embedding and similarly to the description in Figure 2, it simply moves the current vector in the embedding space, hopefully, in the right direction. This can be considered intermediate fusion since the artefact has yet to be processed by a Transformer model.