Key-Value Memory Networks for Directly Reading Documents

Alexander H. Miller[1] **Adam Fisch**[1] **Jesse Dodge**[1,2] **Amir-Hossein Karimi**[1]
Antoine Bordes[1] **Jason Weston**[1]
[1]Facebook AI Research, 770 Broadway, New York, NY, USA
[2]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
{ahm,afisch,jessedodge,ahkarimi,abordes,jase}@fb.com

Abstract

Directly reading documents and being able to answer questions from them is an unsolved challenge. To avoid its inherent difficulty, question answering (QA) has been directed towards using Knowledge Bases (KBs) instead, which has proven effective. Unfortunately KBs often suffer from being too restrictive, as the schema cannot support certain types of answers, and too sparse, e.g. Wikipedia contains much more information than Freebase. In this work we introduce a new method, Key-Value Memory Networks, that makes reading documents more viable by utilizing different encodings in the addressing and output stages of the memory read operation. To compare using KBs, information extraction or Wikipedia documents directly in a single framework we construct an analysis tool, WIKIMOVIES, a QA dataset that contains raw text alongside a preprocessed KB, in the domain of movies. Our method reduces the gap between all three settings. It also achieves state-of-the-art results on the existing WIKIQA benchmark.

1 Introduction

Question answering (QA) has been a long standing research problem in natural language processing, with the first systems attempting to answer questions by directly reading documents (Voorhees and Tice, 2000). The development of large-scale Knowledge Bases (KBs) such as Freebase (Bollacker *et al.*, 2008) helped organize information into structured forms, prompting recent progress to focus on answering questions by converting them into logical forms that can be used to query such databases (Berant *et al.*, 2013; Kwiatkowski *et al.*, 2013; Fader *et al.*, 2014).

Unfortunately, KBs have intrinsic limitations such as their inevitable incompleteness and fixed schemas that cannot support all varieties of answers. Since information extraction (IE) (Craven *et al.*, 2000), intended to fill in missing information in KBs, is neither accurate nor reliable enough, collections of raw textual resources and documents such as Wikipedia will always contain more information. As a result, even if KBs can be satisfactory for closed-domain problems, they are unlikely to scale up to answer general questions on any topic. Starting from this observation, in this work we study the problem of answering by directly reading documents.

Retrieving answers directly from text is harder than from KBs because information is far less structured, is indirectly and ambiguously expressed, and is usually scattered across multiple documents. This explains why using a satisfactory KB—typically only available in closed domains—is preferred over raw text. We postulate that before trying to provide answers that are not in KBs, document-based QA systems should first reach KB-based systems' performance in such closed domains, where clear comparison and evaluation is possible. To this end, this paper introduces WIKIMOVIES, a new analysis tool that allows for measuring the performance of QA systems when the knowledge source is switched from a KB to unstructured documents. WIKIMOVIES contains ∼100k questions in the movie domain, and was designed to be answerable by using either a perfect KB (based on OMDb[1]), Wikipedia pages or an imper-

[1]http://www.omdbapi.com

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1400–1409,
Austin, Texas, November 1-5, 2016.

fect KB obtained through running an engineered IE pipeline on those pages.

To bridge the gap between using a KB and reading documents directly, we still lack appropriate machine learning algorithms. In this work we propose the Key-Value Memory Network (KV-MemNN), a new neural network architecture that generalizes the original Memory Network (Sukhbaatar *et al.*, 2015) and can work with either knowledge source. The KV-MemNN performs QA by first storing facts in a key-value structured memory before reasoning on them in order to predict an answer. The memory is designed so that the model learns to use keys to address relevant memories with respect to the question, whose corresponding values are subsequently returned. This structure allows the model to encode prior knowledge for the considered task and to leverage possibly complex transforms between keys and values, while still being trained using standard backpropagation via stochastic gradient descent.

Our experiments on WIKIMOVIES indicate that, thanks to its key-value memory, the KV-MemNN consistently outperforms the original Memory Network, and reduces the gap between answering from a human-annotated KB, from an automatically extracted KB or from directly reading Wikipedia. We confirm our findings on WIKIQA (Yang *et al.*, 2015), another Wikipedia-based QA benchmark where no KB is available, where we demonstrate that KV-MemNN can reach state-of-the-art results—surpassing the most recent attention-based neural network models.

2 Related Work

Early QA systems were based on information retrieval and were designed to return snippets of text containing an answer (Voorhees and Tice, 2000; Banko *et al.*, 2002), with limitations in terms of question complexity and response coverage. The creation of large-scale KBs (Auer *et al.*, 2007; Bollacker *et al.*, 2008) have led to the development of a new class of QA methods based on semantic parsing (Berant *et al.*, 2013; Kwiatkowski *et al.*, 2013; Fader *et al.*, 2014; Yih *et al.*, 2015) that can return precise answers to complicated compositional questions. Due to the sparsity of KB data, however, the main challenge shifts from finding answers to developing efficient information extraction methods to populate KBs automatically (Craven *et al.*, 2000; Carlson *et al.*, 2010)—not an easy problem.

For this reason, recent initiatives are returning to the original setting of directly answering from text using datasets like TRECQA (Wang *et al.*, 2007), which is based on classical TREC resources (Voorhees *et al.*, 1999), and WIKIQA (Yang *et al.*, 2015), which is extracted from Wikipedia. Both benchmarks are organized around the task of answer sentence selection, where a system must identify the sentence containing the correct answer in a collection of documents, but need not return the actual answer as a KB-based system would do. Unfortunately, these datasets are very small (hundreds of examples) and, because of their answer selection setting, do not offer the option to directly compare answering from a KB against answering from pure text. Using similar resources as the dialog dataset of Dodge *et al.* (2016), our new benchmark WIKIMOVIES addresses both deficiencies by providing a substantial corpus of question-answer pairs that can be answered by either using a KB or a corresponding set of documents.

Even though standard pipeline QA systems like AskMR (Banko *et al.*, 2002) have been recently revisited (Tsai *et al.*, 2015), the best published results on TRECQA and WIKIQA have been obtained by either convolutional neural networks (Santos *et al.*, 2016; Yin and Schütze, 2015; Wang *et al.*, 2016) or recurrent neural networks (Miao *et al.*, 2015)—both usually with attention mechanisms inspired by (Bahdanau *et al.*, 2015). In this work, we introduce KV-MemNNs, a Memory Network model that operates a symbolic memory structured as $(key, value)$ pairs. Such structured memory is not employed in any existing attention-based neural network architecture for QA. As we will show, it gives the model greater flexibility for encoding knowledge sources and helps shrink the gap between directly reading documents and answering from a KB.

3 Key-Value Memory Networks

The Key-Value Memory Network model is based on the Memory Network (MemNNs) model (Weston *et al.*, 2015; Sukhbaatar *et al.*, 2015) which has proven useful for a variety of document reading and question answering tasks: for reading children's books and answering questions about them (Hill *et al.*, 2016), for complex reasoning over sim-

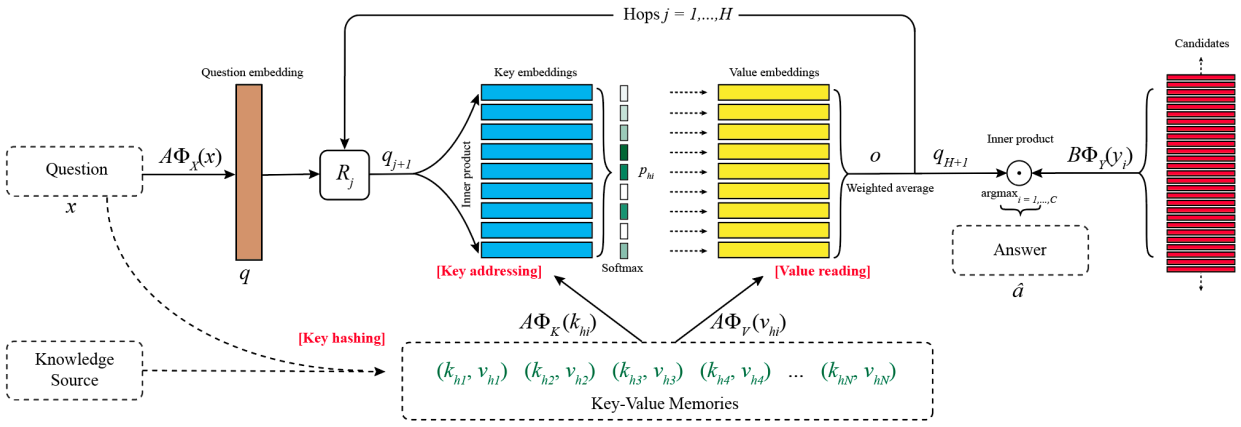

Figure 1: The Key-Value Memory Network model for question answering. See Section 3 for details.

ulated stories (Weston *et al.*, 2016) and for utilizing KBs to answer questions (Bordes *et al.*, 2015).

Key-value paired memories are a generalization of the way context (e.g. knowledge bases or documents to be read) are stored in memory. The lookup (addressing) stage is based on the key memory while the reading stage (giving the returned result) uses the value memory. This gives both (i) greater flexibility for the practitioner to encode prior knowledge about their task; and (ii) more effective power in the model via nontrivial transforms between key and value. The key should be designed with features to help match it to the question, while the value should be designed with features to help match it to the response (answer). An important property of the model is that the entire model can be trained with key-value transforms while still using standard backpropagation via stochastic gradient descent.

3.1 Model Description

Our model is based on the end-to-end Memory Network architecture of Sukhbaatar *et al.* (2015). A high-level view of both models is as follows: one defines a memory, which is a possibly very large array of slots which can encode both long-term and short-term context. At test time one is given a query (e.g. the question in QA tasks), which is used to iteratively address and read from the memory (these iterations are also referred to as "hops") looking for relevant information to answer the question. At each step, the collected information from the memory is cumulatively added to the original query to build context for the next round. At the last iteration, the final retrieved context and the most recent query are combined as features to predict a response from a list of candidates.

Figure 1 illustrates the KV-MemNN model architecture.

In KV-MemNNs we define the memory slots as pairs of vectors $(k_1, v_1) \ldots, (k_M, v_M)$ and denote the question x. The addressing and reading of the memory involves three steps:

- **Key Hashing:** the question can be used to preselect a small subset of the possibly large array. This is done using an inverted index that finds a subset $(k_{h_1}, v_{h_1}), \ldots, (k_{h_N}, v_{h_N})$ of memories of size N where the key shares at least one word with the question with frequency $< F = 1000$ (to ignore stop words), following Dodge *et al.* (2016). More sophisticated retrieval schemes could be used here, see e.g. Manning *et al.* (2008),

- **Key Addressing:** during addressing, each candidate memory is assigned a relevance probability by comparing the question to each key:

$$p_{h_i} = \text{Softmax}(A\Phi_X(x) \cdot A\Phi_K(k_{h_i}))$$

  where $\Phi_\cdot$ are feature maps of dimension D, A is a $d \times D$ matrix and $\text{Softmax}(z_i) = e^{z_i} / \sum_j e^{z_j}$. We discuss choices of feature map in Sec. 3.2.

- **Value Reading:** in the final reading step, the values of the memories are read by taking their weighted sum using the addressing probabilities,

and the vector o is returned:

$$o = \sum_i p_{h_i} A\Phi_V(v_{h_i}) \ .$$

The memory access process is conducted by the "controller" neural network using $q = A\Phi_X(x)$ as the query. After receiving the result o, the query is updated with $q_2 = R_1(q + o)$ where R is a $d \times d$ matrix. The memory access is then repeated (specifically, only the addressing and reading steps, but not the hashing), using a different matrix R_j on each hop, j. The key addressing equation is transformed accordingly to use the updated query:

$$p_{h_i} = \text{Softmax}(q_{j+1}^\top A\Phi_K(k_{h_i})) \ .$$

The motivation for this is that new evidence can be combined into the query to focus on and retrieve more pertinent information in subsequent accesses. Finally, after a fixed number H hops, the resulting state of the controller is used to compute a final prediction over the possible outputs:

$$\hat{a} = \text{argmax}_{i=1,\ldots,C}\text{Softmax}(q_{H+1}^\top B\Phi_Y(y_i))$$

where y_i are the possible candidate outputs, e.g. all the entities in the KB, or all possible candidate answer sentences in the case of a dataset like WIKIQA (see Sec. 5.2). The $d \times D$ matrix B can also be constrained to be identical to A. The whole network is trained end-to-end, and the model learns to perform the iterative accesses to output the desired target a by minimizing a standard cross-entropy loss between $\hat{a}$ and the correct answer a. Backpropagation and stochastic gradient descent are thus used to learn the matrices A, B and $R_1, \ldots, R_H$.

To obtain the standard End-To-End Memory Network of Sukhbaatar *et al.* (2015) one can simply set the key and value to be the same for all memories. Hashing was not used in that paper, but is important for computational efficiency for large memory sizes, as already shown in Dodge *et al.* (2016). We will now go on to describe specific applications of key-value memories for the task of reading KBs or documents.

3.2 Key-Value Memories

There are a variety of ways to employ key-value memories that can have important effects on overall performance. The ability to encode prior knowledge in this way is an important component of KV-MemNNs, and we are free to define Φ_X, Φ_Y, Φ_K and Φ_V for the query, answer, keys and values respectively. We now describe several possible variants of Φ_K and Φ_V that we tried in our experiments, for simplicity we kept Φ_X and Φ_Y fixed as bag-of-words representations.

KB Triple Knowledge base entries have a structure of triple "subject *relation* object" (see Table 1 for examples). The representation we consider is simple: the key is composed of the left-hand side entity (subject) and the relation, and the value is the right-hand side entity (object). We double the KB and consider the reversed relation as well (e.g. we now have two triples "Blade Runner *directed_by* Ridley Scott" and "Ridley Scott *!directed_by* Blade Runner" where *!directed_by* is a different entry in the dictionary than *directed_by*). Having the entry both ways round is important for answering different kinds of questions ("Who directed Blade Runner?" vs. "What did Ridley Scott direct?"). For a standard MemNN that does not have key-value pairs the whole triple has to be encoded into the same memory slot.

Sentence Level For representing a document, one can split it up into sentences, with each memory slot encoding one sentence. Both the key and the value encode the entire sentence as a bag-of-words. As the key and value are the same in this case, this is identical to a standard MemNN and this approach has been used in several papers (Weston *et al.*, 2016; Dodge *et al.*, 2016).

Window Level Documents are split up into windows of W words; in our tasks we only include windows where the center word is an entity. Windows are represented using bag-of-words. Window representations for MemNNs have been shown to work well previously (Hill *et al.*, 2016). However, in Key-Value MemNNs we encode the key as the entire window, and the value as only the center word, which is not possible in the MemNN architecture. This makes sense because the entire window is more likely to be pertinent as a match for the question (as the key), whereas the entity at the center is more pertinent as a match for the answer (as the value). We will compare these approaches in our experiments.

Window + Center Encoding Instead of representing the window as a pure bag-of-words, thus mixing

the window center with the rest of the window, we can also encode them with different features. Here, we double the size, D, of the dictionary and encode the center of the window and the value using the second dictionary. This should help the model pick out the relevance of the window center (more related to the answer) as compared to the words either side of it (more related to the question).

Window + Title The title of a document is commonly the answer to a question that relates to the text it contains. For example "What did Harrison Ford star in?" can be (partially) answered by the Wikipedia document with the title "Blade Runner". For this reason, we also consider a representation where the key is the word window as before, but the value is the document title. We also keep all the standard (window, center) key-value pairs from the window-level representation as well, thus doubling the number of memory slots in comparison. To differentiate the two keys with different values we add an extra feature "_window_" or "_title_" to the key, depending on the value. The "_title_" version also includes the actual movie title in the key. This representation can be combined with center encoding. Note that this representation is inherently specific to datasets in which there is an apparent or meaningful title for each document.

4 The WikiMovies Benchmark

The WIKIMOVIES benchmark consists of question-answer pairs in the domain of movies. It was built with the following goals in mind: (i) machine learning techniques should have ample training examples for learning; and (ii) one can analyze easily the performance of different representations of knowledge and break down the results by question type. The dataset can be downloaded from `http://fb.ai/babi`.

4.1 Knowledge Representations

We construct three forms of knowledge representation: (i) Doc: raw Wikipedia documents consisting of the pages of the movies mentioned; (ii) KB: a classical graph-based KB consisting of entities and relations created from the Open Movie Database (OMDb) and MovieLens; and (iii) IE: information extraction performed on the Wikipedia pages to build a KB in a similar form as (ii). We take care to construct QA pairs such that they are all potentially answerable from either the KB from (ii) or the original Wikipedia documents from (i) to eliminate data sparsity issues. However, it should be noted that the advantage of working from raw documents in real applications is that data sparsity is less of a concern than for a KB, while on the other hand the KB has the information already parsed in a form amenable to manipulation by machines. This dataset can help analyze what methods we need to close the gap between all three settings, and in particular what are the best methods for reading documents when a KB is not available. A sample of the dataset is shown in Table 1.

Doc: Wikipedia Article for Blade Runner (partially shown)
Blade Runner is a 1982 American neo-noir dystopian science fiction film directed by Ridley Scott and starring Harrison Ford, Rutger Hauer, Sean Young, and Edward James Olmos. The screenplay, written by Hampton Fancher and David Peoples, is a modified film adaptation of the 1968 novel "Do Androids Dream of Electric Sheep?" by Philip K. Dick. The film depicts a dystopian Los Angeles in November 2019 in which genetically engineered replicants, which are visually indistinguishable from adult humans, are manufactured by the powerful Tyrell Corporation as well as by other "mega-corporations" around the world. Their use on Earth is banned and replicants are exclusively used for dangerous, menial, or leisure work on off-world colonies. Replicants who defy the ban and return to Earth are hunted down and "retired" by special police operatives known as "Blade Runners". ...
KB entries for Blade Runner (subset)
Blade Runner *directed_by* Ridley Scott Blade Runner *written_by* Philip K. Dick, Hampton Fancher Blade Runner *starred_actors* Harrison Ford, Sean Young, ... Blade Runner *release_year* 1982 Blade Runner *has_tags* dystopian, noir, police, androids, ...
IE entries for Blade Runner (subset)
Blade Runner, Ridley Scott *directed* dystopian, science fiction, film Hampton Fancher *written* Blade Runner Blade Runner *starred* Harrison Ford, Rutger Hauer, Sean Young... Blade Runner *labelled* 1982 neo noir special police, Blade *retired* Blade Runner Blade Runner, special police *known* Blade
Questions for Blade Runner (subset)
Ridley Scott directed which films? What year was the movie Blade Runner released? Who is the writer of the film Blade Runner? Which films can be described by dystopian? Which movies was Philip K. Dick the writer of? Can you describe movie Blade Runner in a few words?

Table 1: WIKIMOVIES: Questions, Doc, KB and IE sources.

Doc We selected a set of Wikipedia articles about movies by identifying a set of movies from OMDb[2] that had an associated article by title match. We keep the title and the first section (before the contents box) for each article. This gives ∼17k documents (movies) which comprise the set of documents our models will read from in order to answer questions.

[2] `http://beforethecode.com/projects/omdb/download.aspx`

KB Our set of movies were also matched to the MovieLens dataset[3]. We built a KB using OMDb and MovieLens metadata with entries for each movie and nine different relation types: director, writer, actor, release year, language, genre, tags, IMDb rating and IMDb votes, with ∼10k related actors, ∼6k directors and ∼43k entities in total. The KB is stored as triples; see Table 1 for examples. IMDb ratings and votes are originally real-valued but are binned and converted to text ("unheard of", "unknown", "well known", "highly watched", "famous"). We finally only retain KB triples where the entities also appear in the Wikipedia articles[4] to try to guarantee that all QA pairs will be equally answerable by either the KB or Wikipedia document sources.

IE As an alternative to directly reading documents, we explore leveraging information extraction techniques to transform documents into a KB format. An IE-KB representation has attractive properties such as more precise and compact expressions of facts and logical key-value pairings based on subject-verb-object groupings. This can come at the cost of lower recall due to malformed or completely missing triplets. For IE we use standard open-source software followed by some task-specific engineering to improve the results. We first employ coreference resolution via the Stanford NLP Toolkit (Manning *et al.*, 2014) to reduce ambiguity by replacing pronominal ("he", "it") and nominal ("the film") references with their representative entities. Next we use the SENNA semantic role labeling tool (Collobert *et al.*, 2011) to uncover the grammatical structure of each sentence and pair verbs with their arguments. Each triplet is cleaned of words that are not recognized entities, and lemmatization is done to collapse different inflections of important task-specific verbs to one form (e.g. stars, starring, star → starred). Finally, we append the movie title to each triple similar to the "Window + Title" representation of Sec. 3.2, which improved results.

4.2 Question-Answer Pairs

Within the dataset's more than 100,000 question-answer pairs, we distinguish 13 classes of question corresponding to different kinds of edges in our KB. They range in scope from specific—such as *actor to movie*: "What movies did Harrison Ford star in?" and *movie to actors*: "Who starred in Blade Runner?"—to more general, such as *tag to movie*: "Which films can be described by *dystopian*?"; see Table 4 for the full list. For some question there can be multiple correct answers.

Using SimpleQuestions (Bordes *et al.*, 2015), an existing open-domain question answering dataset based on Freebase, we identified the subset of questions posed by human annotators that covered our question types. We created our question set by substituting the entities in those questions with entities from all of our KB triples. For example, if the original question written by an annotator was "What movies did Harrison Ford star in?", we created a pattern "What movies did [@actor] star in?", which we substitute for any other actors in our set, and repeat this for all annotations. We split the questions into disjoint training, development and test sets with ∼96k, 10k and 10k examples, respectively. The same question (even worded differently) cannot appear in both train and test sets. Note that this is much larger than most existing datasets; for example, the WIKIQA dataset (Yang *et al.*, 2015) for which we also conduct experiments in Sec. 5.2 has only ∼1000 training pairs.

Method	KB	IE	Doc
(Bordes *et al.*, 2014) QA system	93.5	56.5	N/A
Supervised Embeddings	54.4	54.4	54.4
Memory Network	78.5	63.4	69.9
Key-Value Memory Network	**93.9**	**68.3**	**76.2**

Table 2: Test results (% hits@1) on WIKIMOVIES, comparing human-annotated KB (KB), information extraction-based KB (IE), and directly reading Wikipedia documents (Doc).

Memory Representation	Doc
Sentence-level	52.4
Window-level	66.8
Window-level + Title	74.1
Window-level + Center Encoding + Title	**76.9**

Table 3: Development set performance (% hits@1) with different document memory representations for KV-MemNNs.

[3] http://grouplens.org/datasets/movielens/

[4] The dataset also includes the slightly larger version without this constraint.

5 Experiments

This section describes our experiments on WIKIMOVIES and WIKIQA.

5.1 WikiMovies

We conducted experiments on the WIKIMOVIES dataset described in Sec. 4. Our main goal is to compare the performance of KB, IE and Wikipedia (Doc) sources when trying varying learning methods. We compare four approaches: (i) the QA system of Bordes *et al.* (2014) that performs well on existing datasets WebQuestions (Berant *et al.*, 2013) and SimpleQuestions (Bordes *et al.*, 2015) that use KBs only; (ii) supervised embeddings that do not make use of a KB at all but learn question-to-answer embeddings directly and hence act as a sanity check (Dodge *et al.*, 2016); (iii) Memory Networks; and (iv) Key-Value Memory Networks. Performance is reported using the accuracy of the top hit (single answer) over all possible answers (all entities), i.e. the hits@1 metric measured in percent. In all cases hyperparameters are optimized on the development set, including the memory representations of Sec. 3.2 for MemNNs and KV-MemNNs. As MemNNs do not support key-value pairs, we concatenate key and value together when they differ instead.

The main results are given in Table 2. The QA system of Bordes *et al.* (2014) outperforms Supervised Embeddings and Memory Networks for KB and IE-based KB representations, but is designed to work with a KB, not with documents (hence the N/A in that column). However, Key-Value Memory Networks outperform all other methods on all three data source types. Reading from Wikipedia documents directly (Doc) outperforms an IE-based KB (IE), which is an encouraging result towards automated machine reading though a gap to a human-annotated KB still remains (93.9 vs. 76.2). The best memory representation for directly reading documents uses "Window-level + Center Encoding + Title" ($W = 7$ and $H = 2$); see Table 3 for a comparison of results for different representation types. Both center encoding and title features help the window-level representation, while sentence-level is inferior.

QA Breakdown A breakdown by question type comparing the different data sources for KV-MemNNs is given in Table 4. IE loses out especially to Doc (and KB) on Writer, Director and Actor to Movie, perhaps because coreference is difficult in these cases – although it has other losses elsewhere too. Note that only 56% of subject-object pairs in IE match the triples in the original KB, so losses are expected. Doc loses out to KB particularly on Tag to Movie, Movie to Tags, Movie to Writer and Movie to Actors. Tag questions are hard because they can reference more or less any word in the entire Wikipedia document; see Table 1. Movie to Writer/Actor are hard because there is likely only one or a few references to the answer across all documents, whereas for Writer/Actor to Movie there are more possible answers to find.

Question Type	KB	IE	Doc
Writer to Movie	97	72	91
Tag to Movie	85	35	49
Movie to Year	95	75	89
Movie to Writer	95	61	64
Movie to Tags	94	47	48
Movie to Language	96	62	84
Movie to IMDb Votes	92	92	92
Movie to IMDb Rating	94	75	92
Movie to Genre	97	84	86
Movie to Director	93	76	79
Movie to Actors	91	64	64
Director to Movie	90	78	91
Actor to Movie	93	66	83

Table 4: Breakdown of test results (% hits@1) on WIKIMOVIES for Key-Value Memory Networks using different knowledge representations.

Knowledge Representation	KV-MemNN
KB	93.9
One Template Sentence	82.9
All Templates Sentences	80.0
One Template + Coreference	76.0
One Template + Conjunctions	74.0
All Templates + Conj. + Coref.	72.5
Wikipedia Documents	76.2

Table 5: Analysis of test set results (% hits@1) for KB vs. Synthetic Docs on WIKIMOVIES.

KB vs. Synthetic Document Analysis To further understand the difference between using a KB versus reading documents directly, we conducted an experiment where we constructed synthetic documents using the KB. For a given movie, we use a simple grammar to construct a synthetic "Wikipedia" doc-

Method	MAP	MRR
Word Cnt	0.4891	0.4924
Wgt Word Cnt	0.5099	0.5132
2-gram CNN (Yang *et al.*, 2015)	0.6520	0.6652
AP-CNN (Santos *et al.*, 2016)	0.6886	0.6957
Attentive LSTM (Miao *et al.*, 2015)	0.6886	0.7069
Attentive CNN (Yin and Schütze, 2015)	0.6921	0.7108
L.D.C. (Wang *et al.*, 2016)	0.7058	0.7226
Memory Network	0.5170	0.5236
Key-Value Memory Network	**0.7069**	**0.7265**

Table 6: Test results on WikiQA.

ument based on the KB triples: for each relation type we have a set of template phrases (100 in total) used to generate the fact, e.g. "Blade Runner came out in 1982" for the entry BLADE RUNNER RELEASE_YEAR 1982. We can then parameterize the complexity of our synthetic documents: (i) using one template, or all of them; (ii) using conjunctions to combine facts into single sentences or not; and (iii) using coreference between sentences where we replace the movie name with "it".[5] The purpose of this experiment is to find which aspects are responsible for the gap in performance to a KB. The results are given in Table 5. They indicate that some of the loss (93.9% for KB to 82.9% for One Template Sentence) in performance is due directly to representing in sentence form, making the subject, relation and object harder to extract. Moving to a larger number of templates does not deteriorate performance much (80%). The remaining performance drop seems to be split roughly equally between conjunctions (74%) and coreference (76%). The hardest synthetic dataset combines these (All Templates + Conj. + Coref.) and is actually harder than using the real Wikipedia documents (72.5% vs. 76.2%). This is possibly because the amount of conjunctions and coreferences we make are artificially too high (50% and 80% of the time, respectively).

5.2 WikiQA

WIKIQA (Yang *et al.*, 2015) is an existing dataset for answer sentence selection using Wikipedia as the knowledge source. The task is, given a question, to select the sentence coming from a Wikipedia document that best answers the question, where performance is measured using mean average precision (MAP) and mean reciprocal rank (MRR) of the ranked set of answers. The dataset uses a pre-built information retrieval step and hence provides a fixed set of candidate sentences per question, so systems do not have to consider ranking all of Wikipedia. In contrast to WIKIMOVIES, the training set size is small ($\sim$1000 examples) while the topic is much more broad (all of Wikipedia, rather than just movies) and the questions can only be answered by reading the documents, so no comparison to the use of KBs can be performed. However, a wide range of methods have already been tried on WIKIQA, thus providing a useful benchmark to test if the same results found on WIKIMOVIES carry across to WIKIQA, in particular the performance of Key-Value Memory Networks.

Due to the size of the training set, following many other works (Yang *et al.*, 2015; Santos *et al.*, 2016; Miao *et al.*, 2015) we pre-trained the word vectors (matrices A and B which are constrained to be identical) before training KV-MemNNs. We employed Supervised Embeddings (Dodge *et al.*, 2016) for that goal, training on all of Wikipedia while treating the input as a random sentence and the target as the subsequent sentence. We then trained KV-MemNNs with dropout regularization: we sample words from the question, memory representations and the answers, choosing the dropout rate using the development set. Finally, again following other successful methods (Yin and Schütze, 2015), we combine our approach with exact matching word features between question and answers. Key hashing was not used as candidates were already pre-selected. To represent the memories, we used the Window-Level representation (the best choice on the dev set was $W = 7$) as the key and the whole sentence as the value, as the value should match the answer which in this case is a sentence. Additionally, in the representation all numbers in the text and the phrase "how many" in the question were replaced with the feature "_number_". The best choice of hops was also $H = 2$ for KV-MemNNs.

The results are given in Table 6. Key-Value Memory Networks outperform a large set of other methods, although the results of the L.D.C. method of (Wang *et al.*, 2016) are very similar. Memory Networks, which cannot easily pair windows to sentences, perform much worse, highlighting the importance of key-value memories.

[5] This data is also part of the WIKIMOVIES benchmark.

6 Conclusion

We studied the problem of directly reading documents in order to answer questions, concentrating our analysis on the gap between such direct methods and using human-annotated or automatically constructed KBs. We presented a new model, Key-Value Memory Networks, which helps bridge this gap, outperforming several other methods across two datasets, WIKIMOVIES and WIKIQA. However, some gap in performance still remains. WIKIMOVIES serves as an analysis tool to shed some light on the causes. Future work should try to close this gap further.

Key-Value Memory Networks are versatile models for reading documents or KBs and answering questions about them—allowing to encode prior knowledge about the task at hand in the key and value memories. These models could be applied to storing and reading memories for other tasks as well, and future work should try them in other domains, such as in a full dialog setting.

References

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *Semantic Web Conference, 2007*.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR, 2015*.

Banko, M., Brill, E., Dumais, S., and Lin, J. (2002). Askmsr: Question answering using the worldwide web. In *AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases, 2002*.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *EMNLP, 2013*.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD International Conference on Management of Data, 2008*.

Bordes, A., Chopra, S., and Weston, J. (2014). Question answering with subgraph embeddings. In *EMNLP, 2014*.

Bordes, A., Usunier, N., Chopra, S., and Weston, J. (2015). Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *AAAI Conference on Artificial Intelligence, 2010*.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, **12**, 2493–2537.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. (2000). Learning to construct knowledge bases from the world wide web. *Artificial intelligence*, **118**, 69–113.

Dodge, J., Gane, A., Zhang, X., Bordes, A., Chopra, S., Miller, A., Szlam, A., and Weston, J. (2016). Evaluating prerequisite qualities for learning end-to-end dialog systems. In *ICLR, 2016*.

Fader, A., Zettlemoyer, L., and Etzioni, O. (2014). Open question answering over curated and extracted knowledge bases. In *KDD, 2014*.

Hill, F., Bordes, A., Chopra, S., and Weston, J. (2016). The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR, 2016*.

Kwiatkowski, T., Choi, E., Artzi, Y., and Zettlemoyer, L. (2013). Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP, 2013*.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The stanford corenlp natural language processing toolkit. In *ACL: System Demonstrations, 2014*.

Miao, Y., Yu, L., and Blunsom, P. (2015). Neural variational inference for text processing. *arXiv preprint arXiv:1511.06038*.

Santos, C. d., Tan, M., Xiang, B., and Zhou, B. (2016). Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In *NIPS, 2015*.

Tsai, C., Yih, W.-t., and Burges, C. (2015). Web-based question answering: Revisiting askmsr. Technical report, Technical Report MSR-TR-2015-20, Microsoft Research.

Voorhees, E. M. *et al.* (1999). The trec-8 question answering track report. In *Trec, 1999*.

Voorhees, E. M. and Tice, D. M. (2000). Building a question answering test collection. In *ACM SIGIR Conference on Research and Development in Information Retrieval, 2000*.

Wang, M., Smith, N. A., and Mitamura, T. (2007). What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL, 2007*.

Wang, Z., Mi, H., and Ittycheriah, A. (2016). Sentence similarity learning by lexical decomposition and composition. *arXiv preprint arXiv:1602.07019*.

Weston, J., Chopra, S., and Bordes, A. (2015). Memory networks. In *ICLR, 2015*.

Weston, J., Bordes, A., Chopra, S., and Mikolov, T. (2016). Towards ai-complete question answering: a set of prerequisite toy tasks. In *ICLR, 2016*.

Yang, Y., Yih, W.-t., and Meek, C. (2015). Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP, 2015*.

Yih, W.-t., Chang, M.-W., He, X., and Gao, J. (2015). Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL, 2015*.

Yin, W. and Schütze, H. (2015). Convolutional neural network for paraphrase identification. In *NACL: Human Language Technologies, 2015*.