Autoregressive Reasoning over Chains of Facts with Transformers

Ruben Cartuyvels, Graham Spinks, Marie-Francine Moens
LIIR lab, KU Leuven, Belgium
{first}.{last}@kuleuven.be, sien.moens@kuleuven.be

Abstract

This paper proposes an iterative inference algorithm for multi-hop explanation regeneration, that retrieves relevant factual evidence in the form of text snippets, given a natural language question and its answer. Combining multiple sources of evidence or facts for multi-hop reasoning becomes increasingly hard when the number of sources needed to make an inference grows. Our algorithm copes with this by decomposing the selection of facts from a corpus autoregressively, conditioning the next iteration on previously selected facts. This allows us to use a pairwise learning-to-rank loss.We validate our method on datasets of the TextGraphs 2019 and 2020 Shared Tasks for explanation regeneration. Existing work on this task either evaluates facts in isolation or artificially limits the possible chains of facts, thus limiting multi-hop inference. We demonstrate that our algorithm, when used with a pre-trained transformer model, outperforms the previous state-of-the-art in terms of precision, training time and inference efficiency.

1 Introduction

The task of multi-hop explanation generation has recently received interest as it could be a stepping-stone towards general multi-hop inference over language. Multi-hop reasoning requires algorithms to combine multiple sources of evidence. This becomes increasingly hard when the number of required facts for an inference grows, because of the exploding number of combinations and phenomena such as semantic drift (Fried et al., 2015; Jansen, 2018). The WorldTree dataset was designed specifically for (>2)-fact inference (Jansen et al., 2018; Xie et al., 2020): it consists of elementary science exam questions that can be explained by an average of 6 facts from a complementary dataset of textual facts. The explanation regeneration task as in the TextGraphs Shared Tasks (Jansen and Ustalov, 2019; Jansen and Ustalov, 2020) asks participants to retrieve and rank relevant facts (given one of these natural language questions and its answer as input[1]) such that the top-ranked facts explain the answer to the question. An example is shown in the upper left part of fig. 1 (and more in appendix A).

As each question-answer pair potentially has many supporting facts, evaluating all combinations of facts is computationally prohibitive. Previous work remedies this by computing scores for facts in isolation, or by severely limiting the number of combinations of facts (Das et al., 2019; Banerjee, 2019; Chia et al., 2019). The latter is done by considering combinations of few facts only and/or by reranking combinations of only the top retrieved facts by a simpler method. Both approaches limit multi-hop reasoning as facts are not combined or too many facts are ignored by the simple first-stage retriever.

In this paper we propose a method to retrieve facts that does build long chains of facts, while being efficient. During training, a pre-trained neural language model encodes the question-answer pair as well as a randomly selected combination of ground-truth facts, before evaluating candidate facts. For efficiency,

[1]Systems that do not require the answer, but that retrieve facts based on a question only, could be of great utility: the retrieved facts could be used to infer the answer to the question. However in this paper we follow the task definition of (Jansen and Ustalov, 2019), who define explanation regeneration as a stepping-stone task for multi-hop inference. The method we propose is equally applicable when the answer is not given, but evaluating this setting is left for future work.

Proceedings of the 28th International Conference on Computational Linguistics, pages 6916–6930
Barcelona, Spain (Online), December 8-13, 2020

only a set of neighborhood facts (which we call 'visible' facts) is considered. This set is obtained by selecting the nearest facts by tf-idf distance to the question, answer and set of ground truth facts. During inference, facts are ranked iteratively: in each iteration, the highest scoring fact is selected and encoded together with previously selected facts and the question-answer pair, so the next ranking is always conditioned on the current set of chosen facts. At each inference step, the set of visible facts consists of the nearest facts to the already selected facts and question-answer pair. This autoregressive formulation and the definition of neighborhoods together enable the use of losses that incorporate interactions between different facts, like the pairwise RankNet loss from the learning-to-rank literature (Burges et al., 2005).

Most methods in earlier work use some form of rank-rerank set-up, but none dynamically expand the set of reranked facts, and the reranked set is usually too small (Das et al., 2019; Chia et al., 2019; Banerjee, 2019). Banerjee (2019) and Chia et al. (2019) use iterative schemes, but the former only reconsiders the top 15 initially retrieved facts and the latter uses a term frequency based approach. Das et al. (2019) consider chains of facts during training and inference, but only up to length 2, because of the explosive number of combinations. All of the above systems use pointwise losses only. While components of our method are inspired by previous work, we are the first to put them together in a principled way, so that they enable each other. Our neighborhoods enable both our iterative inference procedure to efficiently build chains of up to 10 facts, and the use of more informative losses. Our training is designed to support iterative inference and to close the gap between training and inference as much as possible.

Contributions. (1) By defining dynamically growing neighborhoods of facts for our model to operate in, we limit computational cost without severely limiting the range of our method, (2) We define a new autoregressive training and inference method to evaluate facts within the context of other facts; (3) We apply a learning-to-rank loss that successfully exploits interactions between facts, leading to an improvement in MAP score over previous baselines.

2 Related Work

Explanation regeneration was promoted as a TextGraphs 2019 Shared Task (Jansen and Ustalov, 2019). We briefly summarize systems proposed in 2019. Most methods finetuned a pre-trained transformer like BERT (Devlin et al., 2019) in a learning-to-rank set-up, as reranker with the question and answer as query and facts as documents, like Nogueira and Cho (2019). Das et al. (2019) use BERT to classify chains of 2 facts, drawn from initially retrieved facts by a tf-idf retriever and facts with words in common with those facts. Their idea is similar to ours, but due to the high number of combinations, they are limited to explanations of only 2 facts. As a second approach, they use BERT like Nogueira and Cho (2019) to rank single facts: but Das et al. simply rank all facts instead of the top-T initially retrieved ones, which is computationally expensive for the larger 2020 dataset.

Chia et al. (2019) use an iterative scheme where the tf-idf representation of the question is aggregated with the tf-idf representations of previously selected facts for retrieval. They compare this to a BERT based approach like described above. In contrast, selected facts and the question in our method are encoded as one paragraph with BERT in a trainable way, which allows for more complex relations to be learnt. Banerjee (2019) adds gold facts as context during training and scores facts individually with BERT during inference. The top facts are reranked by iterative rescoring, which only uses the originally computed scores and precomputed (hence untrainable) sentence embeddings. D'Souza et al. (2019) propose a pair-wise learning-to-rank approach with SVMs.

Das et al. (2018) propose a retriever-reader system that iteratively retrieves and 'reads' relevant paragraphs from a large text corpus for open-domain QA. High level analogies can be drawn between their system and ours, but their architecture involves separate query and document encoders, a recurrent reasoner and a specialized reading comprehension model. Their system is trained end-to-end for QA.

In conclusion, our work has similarity to the widely used retrieval-reranking paradigm, but the initially retrieved set of facts is dynamically extended based on selected facts. Our training and inference method to evaluate facts within the context of other facts successfully models interactions between different facts, framing the task more as a reading comprehension task.

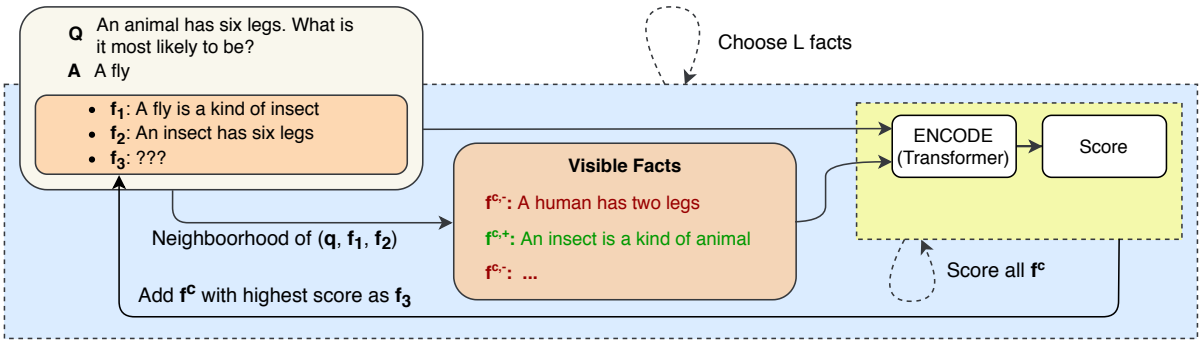


Figure 1: An overview of ARCF during inference. The computed score represents $P(\boldsymbol{f}_3 \mid \boldsymbol{f}_{1,2}, \boldsymbol{q})$.

3 Proposed Approach

This section describes our proposed method, which we call 'Autoregressive Reasoning over Chains of Facts' (ARCF). ARCF consists of an initial retrieval component described in §3.3, a learning-to-rank training scheme and an iterative inference procedure (both in §3.4, fig. 1 shows the latter schematically).

3.1 Task Description

Given is a dataset $\mathcal{D}_f$ of facts in text form (size D_f). Each fact $\boldsymbol{f}$ consists of word tokens: $\boldsymbol{f} = [w_1, w_2, ..., w_{Z_f}]$. Further given is a training, validation and test dataset $\mathcal{D}_{qa}$ containing multiple choice questions (size D_{qa}). Each question is concatenated with answer options: 4 (or so) multiple choice answers, with the correct one marked. Question and answer(s) together form a query $\boldsymbol{q} = [w_1, w_2, ..., w_{Z_q}]$. All queries in $\mathcal{D}_{qa}$ can be explained by 1 to 22 gold facts in $\mathcal{D}_f$. Gold facts for a question are marked with 'grounding', 'central' or 'lexical glue', depending on their role in explaining the question. Central facts are key in explaining the question, lexical glue links facts together (e.g., by synonymy or taxonomy relations) and grounding facts connect facts to the question[2].

The task for a given $\boldsymbol{q} \in \mathcal{D}_{qa}$ is to rank all facts in $\mathcal{D}_f$, with gold facts ranked higher than irrelevant facts. The *answer* in this context is the answer to a question, always encoded together with the question in $\boldsymbol{q}$, and not the output target of the task. When we write 'query', we mean an instance of $\boldsymbol{q}$, i.e., a question concatenated with its answer. The output target is the set of gold facts $\boldsymbol{f}^*_{1,...,G}$ for a $\boldsymbol{q}$. The mean average precision (MAP) of the gold facts in the computed ranking is calculated as evaluation metric. We use the notation $\boldsymbol{f}_{1,...,N}$ for an intermediate set of facts, $\boldsymbol{f}_{1,...,M}$ for a completed set and $\boldsymbol{f}^*_{1,...,G}$ for the gold set for a $\boldsymbol{q}$, with G the number of gold facts. We call a concatenation of a query with a number of facts a prefix: $\boldsymbol{p} = [\boldsymbol{q} \mid \boldsymbol{f}_{1,...,N}]$. Appendix A shows example $\boldsymbol{q}$'s and gold facts.

3.2 Model

We use a neural language encoder to compute a function $f_\theta : \mathcal{V}^Z \mapsto \mathbb{R}$. The input is the concatenation of a query and a number of facts $[\boldsymbol{q} \mid \boldsymbol{f}_{1,...,N}]$. The output is a scalar score s, indicating how well the last of the concatenated facts (the candidate fact) fits in the explanation for the query. The process is iterated: a chosen fact is appended to $\boldsymbol{f}_{1,...,N}$ and a new score is computed. We use pre-trained transformer models, like BERT or RoBERTa (Devlin et al., 2019; Liu et al., 2019; Vaswani et al., 2017), because the task dataset is relatively small, and this allows for the reasoning to incorporate external knowledge.

3.3 Fact - Question Neighborhoods

Neighborhoods. The first step of the method consists of computing neighborhoods of visible facts, for each question and corresponding set of facts, denoted by $vis_k : \mathcal{D}_f \cup \mathcal{D}_{qa} \mapsto \mathcal{D}_f^K$. To retain tractability, facts from neighborhoods $vis_k(\cdot)$ will later be ranked, while facts outside $vis_k(\cdot)$ are out of consideration. In contrast to classic rank–(neural) rerank approaches, our neighborhood (initially retrieved set that will be reranked) will expand dynamically. Pairwise distances between all facts and between questions and

[2]We refer the reader to (Jansen et al., 2018; Xie et al., 2020) for more information.

facts are precomputed. A fact $\boldsymbol{f}^c$ is visible from $[\boldsymbol{q} \mid \boldsymbol{f}_{1,\ldots,N}]$ if it is one of the k nearest facts either to $\boldsymbol{q}$, or to one of the $\boldsymbol{f}_{1,\ldots,N}$ (denoted as $nearest_k$). We use $K_N \leq (N+1) \cdot k$ to refer to the cardinality[3] of $vis_k(\cdot)$. Formally: $vis_k(\boldsymbol{q}, \boldsymbol{f}_{1,\ldots,N}) = \bigcup_{\boldsymbol{x} \in \{\boldsymbol{q}, \boldsymbol{f}_{1,\ldots,N}\}} nearest_k(\boldsymbol{x})$.

For a given question and (possibly empty) set of facts, the algorithm should be able to retrieve the set of visible facts, but is agnostic to how this set or the distances are computed. Given the limited size of $\mathcal{D}_f$ and $\mathcal{D}_{qa}$, if we use an inexpensive distance metric, the overhead of computing all pairwise distances is small. For larger datasets or for distances that are expensive to compute, the overhead might not be negligible. Approximate nearest-neighbor methods could then be used.

k	Fraction
90	0.90
130	0.95
180	0.97
290	0.99

Table 1: Mean fraction of gold facts reachable.

Distances. We tried computing the distances as distance between *tf-idf* vectors, as Word Mover's Distance (Kusner et al., 2015), as the distance between sentence embeddings computed by a pre-trained BERT or Sentence-BERT (Reimers and Gurevych, 2019), and as the reciprocal number of words in common (lexical overlap). We compared the distance metrics by the fraction of gold facts, for a given k, that could be reached in an unlimited number of 'hops'[4] via *gold* facts from $vis_k(\cdot)$, starting from a $\boldsymbol{q} \in \mathcal{D}_{qa}$. We only computed the fractions on the training data to prevent test set leakage. We found tf-idf to work best for all k: table 1 shows some indicative ratios (mean over $\boldsymbol{q} \in \mathcal{D}_{qa}$). The fact that tf-idf works best here can be explained by the fact that the dataset is well curated and terminology is uniform across facts.

Das et al. (2019) construct a connectivity graph between facts, which they use to extend the set of initially retrieved facts by tf-idf. They use lexical overlap as criterion for being connected, resulting in a dense graph and leading to an explosive number of chains. In contrast, parameter k in our definition of vis_k allows for easy and precise control of the size of neighborhoods.

3.4 Autoregressive Ranking of Candidate Facts

We propose to decompose the conditional probability distribution over rankings of facts autoregressively:

$$P(\boldsymbol{f}_{1,\ldots,M} \mid \boldsymbol{q}) = \prod_{i=1}^{M} P(\boldsymbol{f}_i \mid \boldsymbol{f}_{1,\ldots,i-1}, \boldsymbol{q}), \tag{1}$$

where $\boldsymbol{f}_1$ is the highest ranked fact. At each iteration, we compute $P(\boldsymbol{f}_i \mid \boldsymbol{f}_{1,\ldots,i-1}, \boldsymbol{q})$ for all visible candidate facts $\boldsymbol{f}_i$, and we select the fact with the highest probability to be ranked at position i. The unnormalized probability for $P(\boldsymbol{f}_i \mid \boldsymbol{f}_{1,\ldots,i-1}, \boldsymbol{q})$ is computed by our scoring function as $s = f_\theta([\boldsymbol{q} \mid \boldsymbol{f}_{1,\ldots,i-1} \mid \boldsymbol{f}_i])$. For the task at hand, only the interclass order in the produced ranking is relevant: relevant facts should be ranked higher than irrelevant facts. The intraclass order, i.e., the relative order of relevant facts and of irrelevant facts, does not affect the MAP (§3.1). Eq. 1 can thus be understood as the decomposition of selecting a set of facts jointly into selecting one fact at a time (conditioned on previously selected facts). Scoring all combinations of facts is definitely infeasible, while scoring facts independently is too simplifying. The decomposition aims to strike a balance between the two.

Conditioning the selection of facts on previously selected facts brings several advantages (compared to scoring facts independently as Nogueira and Cho (2019)). First, it enables the incremental building of chains of reasoning. The role of many facts in explaining a question is not immediately apparent when they are looked at in isolation, and only becomes more evident when they are considered as part of a larger explanation. Consider the question: "*George warms his hands by rubbing them. Which skin surface produces the most heat? (A) dry palms*". The relevancy of "*1: as moisture of an object decreases, the friction of that object against another object increases*" is clearer when "*2: friction causes the temperature of an object to increase*" is also known. Without the latter, someone (without world knowledge) might, for instance, regard facts about any other physical property that varies with moisture level as equally relevant to the question as the former, while they are not.

[3] $N+1$ for N facts and 1 query, $\leq$ because the neighborhoods might overlap.

[4] A hop is defined as an imagined 'move' from either a $\boldsymbol{q}$ or $\boldsymbol{f}$ to another visible fact.

Second, by processing multiple facts at once, we can leverage BERT as a reading comprehension model, rather than as a retrieval model. The task requires not merely retrieving facts that seem relevant, like a search-engine would, but gathering a set of facts from which the answer to a simple science question can be inferred. That clearly requires more reasoning than a traditional retrieval task. Research has shown that pre-trained transformers are able to infer knowledge from paragraphs of text, which is why they are more suited to handle this formulation of the task (Liu et al., 2019; Clark et al., 2019).

Since we only need to be able to retrieve the facts with the highest probabilities, we can avoid computing normalized probabilities and instead compute scores (i.e., unnormalized probabilities). During training (next §) we do compute probabilities, in order to compute losses, but only over subsets of facts.

Training

Samples. The training input is encoded as $\boldsymbol{x} = [\boldsymbol{q}|\boldsymbol{f}^*_{1,...,N}|\boldsymbol{f}^c]$, with $\boldsymbol{q}$ a query (question and answer), $\boldsymbol{f}^*_{1,...,N}$ a set of gold facts, and $\boldsymbol{f}^c \in \mathit{vis}_k(\boldsymbol{q}_i, \boldsymbol{f}^*_{1,...,N})$ a candidate fact, which can be positive or negative, from the visible neighborhood of $[\boldsymbol{q} \mid \boldsymbol{f}^*_{1,...,N}]$. We train our scoring function f_θ with stochastic gradient descent, to output high scores for positive candidate facts and low scores for negative candidate facts.

Training samples for one $\boldsymbol{q}$ are constructed by first uniformly sampling a number $N \leq G$ of gold facts $\boldsymbol{f}^*_{1,...,N}$ from $\boldsymbol{q}$'s full set of gold facts. N is itself uniformly sampled: $N \sim \mathcal{U}(0, G)$. The query and gold facts are concatenated into a prefix $\boldsymbol{p} = [\boldsymbol{q}|\boldsymbol{f}^*_{1,...,N}]$. For one given prefix, positive training samples $\boldsymbol{x}_p$ are constructed by concatenating $\boldsymbol{p}$ with all remaining visible gold facts. Negative training samples $\boldsymbol{x}_n$ are constructed by concatenating the same $\boldsymbol{p}$ with a number of uniformly sampled visible negative facts[5]. We either use all visible negatives, or sample a number until the minibatch is full. The process is repeated: multiple prefixes are constructed for every query in $\mathcal{D}_{qa}$, and every prefix is appended with multiple visible, gold and negative facts. We construct multiple prefixes per $\boldsymbol{q}$ so that we have multiple training samples per $\boldsymbol{q}$, and so that the model is trained with different explanation lengths.

The prefix itself serves as a sample as well: the model is trained to score $\boldsymbol{p}$ highest when no more visible gold facts remain, i.e., when $\boldsymbol{f}^*_{1,...,N}$ contains all gold facts or when the remaining gold facts are not visible. During inference, $\boldsymbol{p}$ getting the highest score in an iteration is a stopping condition: the gathered set of facts is then considered complete.

Losses. Because classical maximum likelihood training for eq. 1 would allow to backpropagate a loss only after all candidate facts have been considered, i.e., after K_N forward passes, we resort to different loss functions. A simple loss that can be used is the pointwise binary cross-entropy loss (*bXENT*), which considers each input example $\boldsymbol{x}$ individually and trains to correctly classify the candidate $\boldsymbol{f}^c$ as relevant or irrelevant. We propose to use the pairwise *RankNet* loss (Burges et al., 2005):

$$L(\boldsymbol{x}_p, \boldsymbol{x}_n; \theta) = -\log(\sigma(f_\theta(\boldsymbol{x}_p)) - \sigma(f_\theta(\boldsymbol{x}_n))) \,, \tag{2}$$

Where σ is the logistic sigmoid function, and $\boldsymbol{x}_p$ and $\boldsymbol{x}_n$ are samples in which $\boldsymbol{f}^c$ is a positive and negative fact, respectively. This loss is shown by Chen et al. (2009) to maximize a lower bound on the MAP. To further amplify between-fact interactions in the gradient, we also use the conditional ranking variant of *Noise-Contrastive Estimation* (NCE), which covers >2 facts at once (Ma and Collins, 2018; Gutmann and Hyvärinen, 2010):

$$L(\boldsymbol{x}_{1,...,B}; \theta) = -\log \frac{\exp(f_\theta(\boldsymbol{x}_1) - \log P_n(\boldsymbol{f}^c))}{\sum_{j=1}^{B} \exp(f_\theta(\boldsymbol{x}_j) - \log P_n(\boldsymbol{f}^c))} \,, \tag{3}$$

Where $\boldsymbol{x}_1$ is positive and $\boldsymbol{x}_{>1}$ are negative, B is the batch size, and P_n is a negative sampling distribution over candidates: a uniform distribution over $vis_k(\boldsymbol{p})$. This loss has been used for training word embeddings as a more efficient approximation to the negative log-likelihood loss (Mnih and Kavukcuoglu, 2013; Mikolov et al., 2013). When training with NCE and RankNet, samples in one batch share a common prefix $\boldsymbol{p}$ and only differ in their candidate fact $\boldsymbol{f}^c$, so that all samples $\boldsymbol{x}$ in one loss term $L(\cdot, \theta)$ (eqs. 2-3) only differ in $\boldsymbol{f}^c$ and hence the model is trained to score candidate facts and not prefixes.

[5]Note that $\boldsymbol{f}^*_{1,...,N}$ in the prefix of both positive and negative training samples consists only of gold facts, and that all candidate facts $\boldsymbol{f}^{*,c}$, gold or not, come from the visible neighborhood $\mathit{vis}_k(\boldsymbol{p})$.

The proposed training scheme – training a model to predict the next gold element conditioned on previous gold elements – is reminiscent of training text generation models with teacher forcing. A known weakness of teacher forcing is *exposure bias* (Ranzato et al., 2016); models are conditioned on ground-truth data during training, as opposed to on their own outputs during inference. ARCF exhibits this discrepancy as well, which is why, during training, we try replacing a uniformly sampled amount of gold facts $\boldsymbol{f}^*_{1,...,N}$ (in the prefix) with uniformly sampled negative facts from $vis_k(\boldsymbol{p})$. This feature is called 'CN' later. Hence the model is trained to be more robust to mistakes it makes during inference. A similar technique was already proposed for text generation as *scheduled sampling* (Bengio et al., 2015).

Inference

At inference, we incrementally build an explanation, i.e., a set of facts. The input follows the same encoding format: $\boldsymbol{x} = [\boldsymbol{q}|\boldsymbol{f}_{1,...,N}|\boldsymbol{f}^c]$ where $\boldsymbol{f}_{1,...,N}$ are previously selected facts (and not gold facts like in training). At each iteration, we use the query $\boldsymbol{q}$ concatenated with already selected facts $\boldsymbol{f}_{1,...,N}$ as updated retrieval query, and rank all other visible facts $\boldsymbol{f}^c \in vis_k(\boldsymbol{q}, \boldsymbol{f}_{1,...,N})$. The highest scored fact is appended to the query for next iteration. Note that the set of visible facts is extended with the neighborhood of the selected fact in each iteration. The set of selected facts is considered to be complete when its cardinality N equals L or when the highest scored sample is the sample without candidate appended (see prev. §). Multiple rankings are made; one with each intermediate set of facts and the question as retrieval query. Algorithm 1 shows the procedure in pseudocode.

When the stopping condition is met, we end up with an explanation $\hat{\boldsymbol{p}} = [\boldsymbol{q}|\boldsymbol{f}_{1,...,M}]$. To convert the result to a ranking, $\boldsymbol{f}_{1,...,M}$ are ranked highest. Facts that were considered as candidate facts but not selected are ranked next; their relative order is determined by their scores in the last iteration. Next, all facts that were never considered are ranked by a simple metric like tf-idf distance from the computed explanation $\hat{\boldsymbol{p}}$. We experimented with beam search procedures, where the beams were intermediate sets of facts. This did not improve performance on the validation set, so we do not consider it further.

Algorithm 1 Inference procedure for one $\boldsymbol{q}$

Input: $\boldsymbol{q} \in \mathcal{D}_{qa}$, $\boldsymbol{f}_j \in \mathcal{D}_f$, neighborhoods vis_k, scoring function f_θ, max length L
$facts \leftarrow \emptyset$, $allscores \leftarrow \emptyset$,
$candidates \leftarrow vis_k(\boldsymbol{q})$, $l \leftarrow 0$
while $l < L$ **and** not stopping condition **do**
  $scores \leftarrow \emptyset$
  for $j = 1...|candidates|$ **do**
    $scores[j] \leftarrow f_\theta([\boldsymbol{q} \,|\, facts \,|\, candidates[j]])$
  end for
  $facts \leftarrow facts + [\boldsymbol{f}_{\arg\max(scores)}]$
  $candidates \leftarrow vis_k(\boldsymbol{q}; facts)$
  $allscores[l] \leftarrow scores$, $l \leftarrow l + 1$
end while
return produce ranking($facts, allscores$)

This iterative fact selection bears similarity with how token-per-token text generation is usually performed with neural networks. Instead of computing a probability distribution over the vocabulary in one forward pass, our procedure requires a forward pass per score, i.e., per fact considered. To keep required resources for inference reasonable, only neighborhoods of facts are considered, instead of all facts, reducing the number of forward passes in one iteration from $D_f = 9707$ to K_N. Inference for a single $\boldsymbol{q}$ requires $L + \sum_{l=1}^{L} K_{l-1} = \mathcal{O}(L^2 k)$ forward passes[6], with L a chosen maximum number of iterations and k the neighborhood size. Parameter k controls the trade-off between completeness and efficiency.

4 Experiments

4.1 Data & Preprocessing

In the 2020 version of the task $\mathcal{D}_f$ contains 9727 facts, and $\mathcal{D}^{train}_{qa}$, $\mathcal{D}^{val}_{qa}$, $\mathcal{D}^{test}_{qa}$ contain 2206, 496 and 1664 questions respectively (Xie et al., 2020). The dataset has been extended w.r.t. the 2019 version of the task. For completeness, we also include results obtained with baselines and our models on the 2019 data. The 2019 data includes 902, 214 and 541 questions for training, validation, and testing respectively, along with 4950 facts (Jansen et al., 2018). We remove incorrect answers from $\boldsymbol{q}$ (like Das et al. (2019)),

[6] $L + \sum_{l=1}^{L} K_{l-1} \leq L + \sum_{l=1}^{L} l \cdot k = L + \frac{L}{2}(k + Lk) = \mathcal{O}(L^2 k)$, with L the max. number of iterations and K defined in §3.3: $K_l \leq (l+1) \cdot k$.

mark the correct answer with "(answer)" and the start of the gold facts with "(explanation)"[7]. An example of a tokenized input sample could be "[START] *When does water start boiling? (answer) At 100°C. (explanation) This is a gold fact. This is another gold fact.* [SEP] *This is a candidate fact that is gold or not* [SEP]". Examples of $\boldsymbol{q} \in \mathcal{D}_{qa}$ and their explaining facts are shown in Appendix A.

4.2 Baselines

As a simple baseline, we include a *tf-idf* vector retrieval model, that ranks facts by cosine similarity between their and the $\boldsymbol{q}$'s tf-idf representation. We stem facts and $\boldsymbol{q}$, and remove stopwords before computing tf-idf vectors. Baseline *single-fact* concatenates a $\boldsymbol{q}$ and a single candidate fact, and computes a relevance score for all $\boldsymbol{f}^c \in \mathcal{D}_f$ by encoding the concatenation $[\boldsymbol{q} \mid \boldsymbol{f}^c]$ with BERT and projecting the final layer's CLS-token embedding to a scalar with a linear layer (Das et al., 2019; Nogueira and Cho, 2019). The model is trained with binary cross-entropy (relevant or not).

The highest score in the 2019 competition was obtained by an ensemble of baselines *single-fact* and *path-rerank* (Das et al., 2019). Model *path-rerank* ranks facts for a $\boldsymbol{q}$ by first retrieving the top-T facts with the *tf-idf* retriever from above. This initial top-T set is extended with all facts that have $\geq$1 words in common with one of the top-T facts. Next, all combinations of up to $C = 2$ facts are taken from this extended set. A relevance score is computed for all combinations (*chains*), in the same way as in *single-fact* or our models: concatenate $\boldsymbol{q}$ and the C facts, encode the concatenation with BERT and project the final CLS-token embedding to a score s with a linear layer. A fact's relevance score is the maximum score of any chain it appears in. The binary cross-entropy loss is used for training the model. We use the implementation of Das et al. (2019) for the *single-fact* and *path-rerank* baselines[8].

Complexities. Das et al. (2019) used *single-fact* and *path-rerank* for the smaller 2019 dataset, with fewer facts and fewer $\boldsymbol{q}$. They already noted that *single-fact* is not scalable to a large corpus of facts: for the 2020 data, $D_f \approx$ 10K forward passes are required to solve a single $\boldsymbol{q}$ during inference. One epoch trained with bXENT consists of 21M samples ($D_{qa}^{train} \cdot D_f$, trained 3 epochs). The *path-rerank* model uses $T = 25$ for training, which generates 7k chains of facts per $\boldsymbol{q}$, resulting in 16M training samples (trained 1 epoch). Using $T = 50$ during inference results in 16k chains (hence forward passes) per $\boldsymbol{q}$. This number can be controlled by setting T, but setting T too small would leave too many relevant facts out of consideration. In contrast, the neighborhood in our method is less restrictive as it depends on selected facts and thus expands progressively as more facts are selected.

4.3 Experiments

We implemented our algorithm and baselines using PyTorch and the Transformers library[9] (Paszke et al., 2019; Wolf et al., 2019). The *tf-idf* baseline was implemented with SciKit Learn (Pedregosa et al., 2011). To keep comparisons fair, all results on 2019 data (baselines and ARCF) are obtained by finetuning the publicly available pre-trained *bert-base-uncased* (since this model is used in Das et al. (2019)). To reduce resource usage, all models on 2020 data were finetuned from the smaller pre-trained *distilroberta-base*[10]. It can reasonably be expected that using bigger or more advanced pre-trained models would further improve results. We ran experiments on 1 16GB Nvidia Tesla P100 GPU. We used the Adam optimizer (Kingma and Ba, 2015), with learning rate 2e$-$5 and linear LR decay. We append samples to minibatches until they reach 5000 tokens. An overview of used hyperparameters can be found in appendix B. For training we set neighborhood size $k = 180$ (L only impacts inference), for inference we set the maximum and minimum number of iterations $L = 9$, $L_{min} = 3$, and $k = 290$. Some hyperparameters were taken from Das et al. (2019), while others were tuned manually.

[7] It is worth noting that technically, ARCF can perfectly run without the correct answer marked and without incorrect answers removed, although a performance drop on the explanation regeneration task can be expected.

[8] Code available at `https://github.com/ameyagodbole/multihop_inference_explanation_regeneration`.

[9] Our code and trained models are publicly available at `https://github.com/rubencart/LIIR-TextGraphs-14`.

[10] When we finetuned *bert-base-uncased* on the 2020 data or *distilroberta-base* on the 2019 data we obtained similar results.

Algorithm	Loss	MAP 2020	MAP 2019
TF-IDF		0.3743	0.3870
SF	bXENT	0.4992	0.5574
PR	bXENT	0.4629	0.5313
ENSEMBLE	bXENT	0.5081	0.5625
ARCF	RankNet	**0.5815**	**0.5707**
with CN	RankNet	0.5810	0.5575
ARCF	NCE	0.5728	0.5634
with CN	NCE	0.5759	0.5691

(a)

Algorithm	Loss	test 2020	val 2020
ARCF	RankNet	0.5815	0.5931
w bXENT	bXENT	−0.0082	−0.0060
w SF train	bXENT	−0.102	−0.101
w SF inf	RankNet	−0.037	−0.041
w/o prefix, neighb.	RankNet	−0.053	−0.054
w/o R3	RankNet	−0.0004	−0.0001
w/o R3, S2	RankNet	−0.072	−0.079

(b)

Algorithm	train T (H)	inf T (s/$\boldsymbol{x}$)
SF	56.3	18.4
PR	16.2	31.8
ARCF	5.7	9.6

(c)

k	90	130	180	210	290
MAP	.567	.577	.581	.581	.581
T (s/$\boldsymbol{x}$)	2.5	3.6	4.9	5.7	7.8

(d)

L	2	4	6	8	10	12	14
MAP	.561	.576	.581	.581	.581	.581	.582
T (s/$\boldsymbol{x}$)	0.43	1.4	2.9	4.9	7.6	11.0	15.1

(e)

Table 2: (a) Mean average precision (MAP) of baselines (upper) and ARCF (bottom) on the 2020 and 2019 test sets. (b) Ablations on the 2020 test and validation sets. (c) Training time in hours and inference time in seconds/sample. (d)-(e) Impact of k (with $L = 8$) and L (with $k = 180$) on 2020 test MAP and inference time.

In the remainder of this section, **ARCF** denotes our proposed method, **SF** refers to the *single-fact* baseline, **PR** is the *path-rerank* baseline, **CN** means 'conditioned on negatives', **S2** means 'rank scored but not selected facts 2nd', **R3** means 'rank rest 3rd' (as opposed to omitting them altogether, see §3.4).

4.4 Results and Discussion

Test set results. Tables 2a,c show results on the hidden 2019 and 2020 test sets[11], total training time and inference time per sample, for the baselines and ARCF. The test scores are obtained by models that got the highest validation score of 5 training runs with different random seeds, while the training times are averaged over these 5 runs. As can be seen, ARCF outperforms the baselines both in terms of obtained MAP and efficiency[12]. The highest 2020 MAP is 0.5815, which put us at the second place in the online competition. Appendix C shows examples of validation set questions and predicted facts.

On the 2020 test set, all our models obtain a higher MAP than all baselines. This is not the case on the 2019 data, which suggests that ARCF benefits more from additional training data. Including >2 facts in one loss term with NCE shows no benefit compared to the pairwise RankNet loss. Conditioning on negatives has no significant impact. Five 2020 test set evaluations of ARCF and SF show that the difference in scores is statistically significant: ARCF with RankNet scores *higher* on average than SF (1-tailed indep. t-test, $p < 0.001$).

Ablation study. As ARCF consists of several components, we perform an ablation study on the 2020 test and validation sets. Results are shown in table 2b. First, ARCF was trained with the pointwise **bXENT** loss (but still with prefixes and neighborhoods). The MAP drops, showing that pairwise information in the gradients improves learning w.r.t. pointwise information. Second, ARCF is compared to **ARCF w SF train**, which is trained like SF and uses algorithm 1 for inference. The large drop in MAP

[11]Leaderboard of 2020 competition: `https://competitions.codalab.org/competitions/23615` (26/10/2020), 2019 competition: `https://competitions.codalab.org/competitions/20150` (26/10/2020).

[12]Xie et al. (2020) report a MAP score of 0.52 instead of 0.4992 on the 2020 test set with a *single-fact* BERT baseline, but since the publication of their paper the 2020 dataset (incl. the test set) has been updated. They also use BERT instead of RoBERTa, and possibly different hyperparameters.

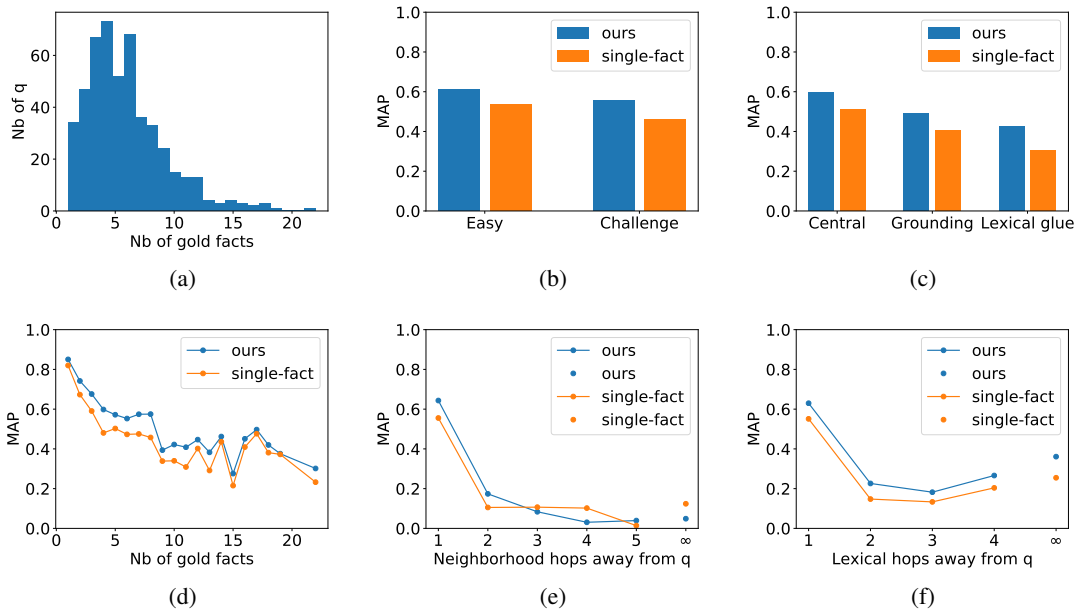


Figure 2: MAP of ARCF vs. *single-fact* on subsets of facts.

indicates that algorithm 1 only works for inference if the model is trained accordingly. Next, **ARCF w SF inf** is trained like ARCF and evaluated like SF. This result shows that ARCF training still improves performance w.r.t. SF even when facts are scored individually, and it shows that algorithm 1 improves performance. **ARCF w/o prefix, neighb** is trained with the pairwise RankNet loss but without prefixes and neighborhoods. Minibatches each contain 1 positive and $B-1$ negatives that are uniformly sampled from all $D_f - G$ negative facts. Evaluation is carried out like for SF. The score drops almost 10%, which emphasizes both the need for informative negatives when training with a pairwise loss, which our neighborhoods provide by returning nearby facts, and the importance of training with prefixes. It also affirms the gain of algorithm 1 for inference. Leaving facts that were never scored out of the ranking (**w/o R3**) has negligible impact. Ignoring scored but unselected facts too (**w/o R3, S2**) cuts performance by $>10\%$: including them is thus essential. Visible facts are scored anyway, so including them all in the ranking comes at no additional cost, except for once sorting them based on their score. We also trained a randomly initialized version of *distilroberta-base* (instead of pre-trained). Although we did not extensively tune hyperparameters, the maximum obtained test MAP was about 25% of pre-trained models.

Impact of neighborhood size k and maximum explanation length L. As tables 2d-e show, MAP increases with k and L, before flattening around $k = 180, L = 8$. Inference time per sample increases approximately quadratically with L and linearly with k, which is in line with the number of FW passes for ARCF inference growing with $\mathcal{O}(L^2k)$. Fig. 3 in appendix D also shows this. Table 2d shows that we could have taken $k = 180$ for inference, with virtually no loss in MAP and almost 40% speedup, but the $k = 290$ we used showed better validation results (likewise for $L = 6$).

Performance on subsets of facts. Since the leaderboards only return a scalar test MAP, we run a number of experiments on the smaller validation set. The results are therefore only indicative. Fig. 2a shows the number of $\boldsymbol{q} \in \mathcal{D}_{qa}^{val}$ for different numbers of corresponding gold facts (G). Fig. 2b-c show MAP scores of ARCF (RankNet) vs. the SF baseline on subsets of facts as marked in the dataset. ARCF increases MAP on the Challenge subset with an absolute %10[13]. It significantly improves retrieval of all roles, but most of lexical glue and grounding facts. These are facts that support central facts, which might be easier to detect when using the context of other facts as our model does. This is a useful improvement, as reasoning of explanations that contain lexical glue and grounding facts is easier to understand[14].

[13] All questions in the dataset are marked by annotators with either 'Easy' or 'Challenge' (all $\boldsymbol{q}$ are one of the two).
[14] Examples in appendices A and C.

Figs. 2e-f show the MAP for facts that need an increasing number of hops to reach (from $\boldsymbol{q}$ or a fact to another fact, only via gold facts). In fig. 2e the hops are always to a fact in the current neighborhood $vis_k(\cdot)$, in fig. 2f to facts with lexical overlap as seen in Das et al. (2019). The '∞' on the x-axis shows the precision for facts that cannot be reached from $\boldsymbol{q}$ in any number of hops (they can still be correctly retrieved by our method, but only via a negative fact). Surprisingly, the MAP of ARCF drops below that of SF for facts that are $3-4$ neighborhood hops away. When lexical overlap hops are considered, ARCF performs better than the baseline for 'farther' facts. The precision values for figure 2c-f are computed as by Jansen and Ustalov (2019), by first removing gold facts that have another role (or are not exactly h hops away) both from the gold set and from the predicted ranking, and then computing the MAP of the remaining predicted ranking w.r.t. the remaining gold set[15].

Fig. 2d shows the MAP for $\boldsymbol{q}$'s with different numbers of gold facts G (the point at $x = 2$ shows the MAP on those $\boldsymbol{q}$ that have a total of $G = 2$ gold facts). ARCF gives a consistent improvement over the baseline for all G.

5 Conclusion

Future work. Future work might expand on our approach by finding alternative methods to evaluate a fact w.r.t. other facts in an iterative inference procedure, or by designing better fact-question neighborhood methods. Additionally, the performance of our method when the correct answer to a question is not given could be evaluated. Future work could then infer the answer from retrieved facts in a downstream QA setting. Finally, one could improve the method by considering to *remove* earlier chosen facts from the intermediate set of selected facts.

We have proposed a new method to retrieve relevant facts for an explanation regeneration task by iteratively evaluating candidate facts with respect to previously selected facts using a learning-to-rank approach. We have successfully evaluated our method on the Textgraphs 2019 and 2020 datasets and have performed several ablation experiments. We have analyzed time complexity of our method and the performance on different subsets of facts. By selecting the nearest facts by similarity between tf-idf vectors, considering not just the question but also already selected facts, only a subset of facts are considered at each step, and ARCF outperforms previous state-of-the-art methods at a higher efficiency.

Acknowledgements

The research leading to this paper received funding from the Research Foundation Flanders (FWO) under Grant Agreement No. G078618N and from the European Research Council (ERC) under Grant Agreement No. 788506. The Flemish Supercomputer Center (VSC) provided hardware and GPUs.

[15] E.g., to compute the MAP for central facts, gold grounding and lexical glue facts are removed from the predicted ranking, so they do not influence the central fact MAP. The MAP of the remaining predictions w.r.t. the gold central facts is then computed.

References

Pratyay Banerjee. 2019. ASU at TextGraphs 2019 shared task: Explanation ReGeneration using language models and iterative re-ranking. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 78–84. Association for Computational Linguistics.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Christopher JC Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 89–96.

Wei Chen, Tie-Yan Liu, Yanyan Lan, Zhi-Ming Ma, and Hang Li. 2009. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems*, pages 315–323.

Yew Ken Chia, Sam Witteveen, and Martin Andrews. 2019. Red dragon ai at textgraphs 2019 shared task: Language model assisted explanation generation. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 85–89.

Peter Clark, Oren Etzioni, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, Sumithra Bhakthavatsalam, et al. 2019. From'f'to'a'on the ny regents science exams: An overview of the aristo project. *arXiv preprint arXiv:1909.01958*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2018. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*.

Rajarshi Das, Ameya Godbole, Manzil Zaheer, Shehzaad Dhuliawala, and Andrew McCallum. 2019. Chains-of-reasoning at textgraphs 2019 shared task: Reasoning over chains of facts for explainable multi-hop inference. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 101–117.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jennifer D'Souza, Isaiah Onando Mulang', and Sören Auer. 2019. Team SVMrank: Leveraging feature-rich support vector machines for ranking explanations to elementary science questions. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 90–100. Association for Computational Linguistics.

Daniel Fried, Peter Jansen, Gustave Hahn-Powell, Mihai Surdeanu, and Peter Clark. 2015. Higher-order lexical semantic models for non-factoid answer reranking. *Transactions of the Association for Computational Linguistics*, 3:197–210.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Peter Jansen and Dmitry Ustalov. 2019. Textgraphs 2019 shared task on multi-hop inference for explanation regeneration. In *Proceedings of the Thirteenth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-13)*, pages 63–77.

Peter Jansen and Dmitry Ustalov. 2020. TextGraphs 2020 Shared Task on Multi-Hop Inference for Explanation Regeneration. In *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*. Association for Computational Linguistics.

Peter Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton Morrison. 2018. Worldtree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Peter Jansen. 2018. Multi-hop inference for sentence-level textgraphs: How challenging is meaningfully combining information for science question answering? In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pages 12–17.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 957–966.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zhuang Ma and Michael Collins. 2018. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3698–3707. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *4th International Conference on Learning Representations*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhengnan Xie, Sebastian Thiem, Jaycie Martin, Elizabeth Wainwright, Steven Marmorstein, and Peter Jansen. 2020. Worldtree v2: A corpus of science-domain structured explanations and inference patterns supporting multi-hop inference. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5456–5473.

Appendix A Examples of TextGraphs-2020 data

Question 1	The influence of the Moon on the tides on Earth is greater than that of the Sun. Which best explains this? (answer) The Moon is closer to Earth than the Sun.
Fact 0 - Role central	the gravitational pull of the Moon on Earth's oceans causes the tides
Fact 1 - Role central	as distance from an object decreases , the pull of gravity on that object increases
Fact 2 - Role grounding	closer means lower; less; a decrease in distance
Fact 3 - Role grounding	a moon is a kind of celestial object; body
Fact 4 - Role central	the moon is the celestial object that is closest to the Earth
Fact 5 - Role grounding	the Sun is a kind of star
Fact 6 - Role grounding	a star is a kind of celestial object; celestial body
Fact 7 - Role central	the Moon is the celestial object that is closer to the Earth than the Sun
Fact 8 - Role grounding	Earth is a kind of planet
Fact 9 - Role grounding	a planet is a kind of celestial object; body
Fact 10 - Role lexglue	cause is similar to influence
Fact 11 - Role lexglue	gravity means gravitational pull; gravitational energy; gravitational force; gravitational attraction

Question 2	A student placed an ice cube on a plate in the sun. Ten minutes later, only water was on the plate. Which process caused the ice cube to change to water? (answer) melting.
Fact 0 - Role central	melting means matter; a substance changes from a solid into a liquid by increasing heat energy
Fact 1 - Role grounding	an ice cube is a kind of solid
Fact 2 - Role grounding	water is a kind of liquid at room temperature
Fact 3 - Role central	water is in the solid state , called ice , for temperatures between 0; -459; -273 and 273; 32; 0 K; F; C
Fact 4 - Role lexglue	heat means heat energy
Fact 5 - Role lexglue	adding heat means increasing temperature
Fact 6 - Role central	if an object; a substance; a location absorbs solar energy then that object; that substance will increase in temperature
Fact 7 - Role central	if an object; something is in the sunlight then that object; that something will absorb solar energy
Fact 8 - Role central	the sun is a source of light; light energy called sunlight
Fact 9 - Role lexglue	to be in the sun means to be in the sunlight
Fact 10 - Role central	melting is a kind of process

Table 3: Example $\boldsymbol{q} \in \mathcal{D}_{qa}^{val}$ with explaining gold facts and their roles. The wrong multiple choice answer options have already been removed.

Appendix B Hyperparameters

All ARCF models were trained with an L2 weight decay coefficient of 0.01. We tried training baselines *single-fact* and *path-rerank* with the same weight decay, but validation and test set results were lower than without weight decay.

Tables 4a and 4b show the hyperparameters we used for ARCF training and inference. Hyperparameters for training with different loss functions are largely the same, only the number of epochs trained might differ with 1 or 2. When conditioning on negatives (CN) was used for training, we first trained for 2 epochs as normal and then started replacing a uniformly sampled proportion between 0.0 and 0.3 of gold facts in the prefix $\boldsymbol{f}^*_{1,\dots,N}$ by uniformly sampled negatives from the visible neighborhood of the prefix as it was before the replacement $vis_k(\boldsymbol{p})$. Parameter L is only relevant for inference, as it is only used by algorithm 1.

Hyperparam	value
LR	2e−5
Epochs	4
L2 weight decay	0.01
ADAM ϵ	1e−8
ADAM β_1	0.9
ADAM β_2	0.999
k	180
Tokens in minibatch	5000

(a)

Hyperparam	value
k	290
L	9
L_{min}	3
Tokens in minibatch	24k
R3	tf-idf
S2	true

(b)

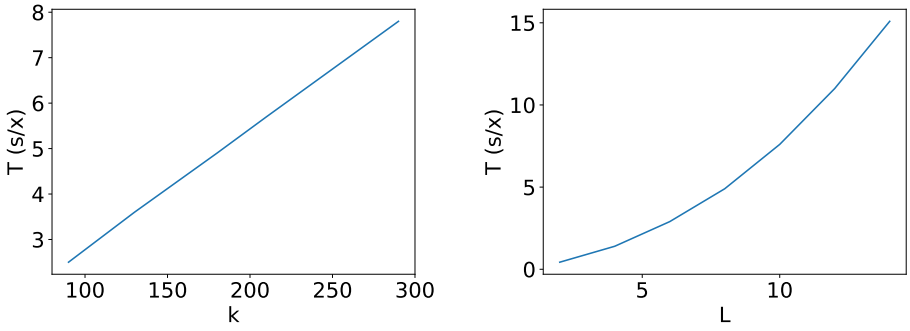
Table 4: Hyperparameters used for ARCF training (with RankNet) and inference.

Appendix C Examples of validation set predictions

Table 5 shows the 15 highest ranked facts by ARCF (RankNet) for the two validation questions in table 3. The first column of each row for gold facts (which are correctly ranked high) is colored blue. As can be seen in the predictions for Question 1, some wrongly predicted facts are clearly related to the question but not necessary for explaining the answer (e.g. Fact 0). While others (like Fact 2) could actually be used for explaining the question as well. Fact 2 for question 1 in table 5 could reasonably take the place of gold Fact 7 for the same question in table 3.

Appendix D Extra plots

See figure 3.



(a) (b)

Figure 3: Inference time in seconds per sample of ARCF for k and L, see also tables 2d-e. Time grows approximately quadratically in L and linearly in k.

Question 1	The influence of the Moon on the tides on Earth is greater than that of the Sun. Which best explains this? (answer) The Moon is closer to Earth than the Sun.
Fact 0	cause is similar to influence.
Fact 1	as the gravitational pull of the moon on the Earth decreases , the size of the tides on Earth decrease.
Fact 2	the Moon is closer to the Earth than the Sun.
Fact 3	closer means lower; less; a decrease in distance.
Fact 4	as the gravitational pull of the moon on the Earth decreases , the size of the tides on Earth decrease.
Fact 5	gravity means gravitational pull; gravitational energy; gravitational force; gravitational attraction.
Fact 6	as distance from an object decreases , the pull of gravity on that object increases.
Fact 7	as distance from an object increases , the pull of gravity on that object decrease.
Fact 8	an increase is the opposite of a decrease.
Fact 9	as the distance from an object increases , the force of gravity on that object will decrease.
Fact 10	a moon is a kind of celestial object; body.
Fact 11	the gravitational pull of the Moon on Earth's oceans causes the tides.
Fact 12	the gravitational pull of the Sun on Earth's oceans causes the tides.
Fact 13	less is similar to decrease.
Fact 14	to lower means to decrease.

Question 2	A student placed an ice cube on a plate in the sun. Ten minutes later, only water was on the plate. Which process caused the ice cube to change to water? (answer) melting.
Fact 0	melting is a kind of process.
Fact 1	melting means matter; a substance changes from a solid into a liquid by increasing heat energy.
Fact 2	an ice cube is a kind of solid.
Fact 3	water is a kind of substance.
Fact 4	water is a kind of liquid at room temperature.
Fact 5	water is in the solid state , called ice , for temperatures between 0; -459; -273 and 273; 32; 0 K; F; C.
Fact 6	temperature is a measure of heat energy.
Fact 7	heat means heat energy.
Fact 8	water is in the liquid state , called liquid water , for temperatures between 273; 32; 0 and 373; 212; 100 K; F; C.
Fact 9	ice is a kind of solid.
Fact 10	if an object; a substance; a location absorbs solar energy then that object; that substance will increase in temperature.
Fact 11	melting is when solids are heated above their melting point.
Fact 12	adding heat means increasing temperature.
Fact 13	cooling;colder means removing;reducing;decreasing heat;temperature.
Fact 14	heating means adding heat.

Table 5: Example $\boldsymbol{q} \in \mathcal{D}_{qa}^{val}$ with explaining gold facts and their roles. The wrong multiple choice answer options have already been removed.