No Need to *Pay Attention*: Simple Recurrent Neural Networks Work! (for Answering "Simple" Questions)

Ferhan Ture and **Oliver Jojic**
Comcast Labs, Washington, DC 20005
{ferhan_ture, oliver_jojic}@cable.comcast.com

Abstract

First-order factoid question answering assumes that the question can be answered by a single fact in a knowledge base (KB). While this does not seem like a challenging task, many recent attempts that apply either complex linguistic reasoning or deep neural networks achieve 65%–76% accuracy on benchmark sets. Our approach formulates the task as two machine learning problems: detecting the entities in the question, and classifying the question as one of the relation types in the KB. We train a recurrent neural network to solve each problem. On the SimpleQuestions dataset, our approach yields substantial improvements over previously published results — even neural networks based on much more complex architectures. The simplicity of our approach also has practical advantages, such as efficiency and modularity, that are valuable especially in an industry setting. In fact, we present a preliminary analysis of the performance of our model on real queries from Comcast's X1 entertainment platform with millions of users every day.

1 Introduction

First-order factoid question answering (QA) assumes that the question can be answered by a single fact in a knowledge base (KB). For example, "How old is Tom Hanks" is about the [age] of [Tom Hanks]. Also referred to as *simple* questions by Bordes et al. (2015), recent attempts that apply either complex linguistic reasoning or attention-based complex neural network architectures achieve up to 76% accuracy on benchmark sets (Golub and He, 2016; Yin et al., 2016). While it is tempting to study QA systems that can handle more complicated questions, it is hard to reach reasonably high precision for unrestricted questions. For more than a decade, successful industry applications of QA have focused on first-order questions. This bears the question: are users even interested in asking questions beyond first-order (or are these use cases more suitable for interactive dialogue)? Based on voice logs from a major entertainment platform with millions of users every day, Comcast X1, we find that most existing use cases of QA fall into the first-order category.

Our strategy is to tailor our approach to first-order QA by making strong assumptions about the problem structure. In particular, we assume that the answer to a first-order question is a *single* property of a *single* entity in the KB, and decompose the task into two subproblems: (a) detecting entities in the question and (b) classifying the question as one of the relation types in the KB. We simply train a vanilla recurrent neural network (RNN) to solve each subproblem (Elman, 1990). Despite its simplicity, our approach (RNN-QA) achieves the highest reported accuracy on the SimpleQuestions dataset. While recent literature has focused on building more complex neural network architectures with attention mechanisms, attempting to generalize to broader QA, we enforce stricter assumptions on the problem structure, thereby reducing complexity. This also means that our solution is efficient, another critical requirement for real-time QA applications. In fact, we present a performance analysis of RNN-QA on Comcast's X1 entertainment system, used by millions of customers every day.

2 Related work

If knowledge is presented in a structured form (e.g., knowledge base (KB)), the standard ap-

Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2849–2855
Copenhagen, Denmark, September 7–11, 2017.

proach to QA is to transform the question and knowledge into a compatible form, and perform reasoning to determine which fact in the KB answers a given question. Examples of this approach include pattern-based question analyzers (Buscaldi et al., 2010), combination of syntactic parsing and semantic role labeling (Bilotti et al., 2007, 2010), as well as lambda calculus (Berant et al., 2013) and combinatory categorical grammars (CCG) (Reddy et al., 2014). A downside of these approaches is the reliance on linguistic resources/heuristics, making them language- and/or domain-specific. Even though Reddy et al. (2014) claim that their approach requires less supervision than prior work, it still relies on many English-specific heuristics and hand-crafted features. Also, their most accurate model uses a corpus of paraphrases to generalize to linguistic diversity. Linguistic parsers can also be too slow for real-time applications.

In contrast, an RNN can detect entities in the question with high accuracy and low latency. The only required resources are word embeddings and a set of questions with entity words tagged. The former can be easily trained for any language/domain in an unsupervised fashion, given a large text corpus without annotations (Mikolov et al., 2013; Pennington et al., 2014). The latter is a relatively simple annotation task that exists for many languages and domains, and it can also be synthetically generated. Many researchers have explored similar techniques for general NLP tasks (Collobert et al., 2011), such as named entity recognition (Lu et al., 2015; Hammerton, 2003), sequence labeling (Graves, 2008; Chung et al., 2014), part-of-speech tagging (Huang et al., 2015; Wang et al., 2015), chunking (Huang et al., 2015).

Deep learning techniques have been studied extensively for constructing parallel neural networks for modeling a joint probability distribution for question-answer pairs (Hsu et al., 2016; Yang et al., 2014; He et al., 2015; Mueller and Thyagarajan, 2016) and re-ranking answers output by a retrieval engine (Rao et al., 2016; Yang et al., 2016). These more complex approaches might be needed for general-purpose QA and sentence similarity, where one cannot make assumptions about the structure of the input or knowledge. However, as noted in Section 1, first-order factoid questions can be represented by an entity and a relation type, and the answer is usually stored in a structured knowledge base. Dong et al. (2015) similarly assume that the answer to a question is at most two hops away from the target entity. However, they do not propose how to obtain the target entity, since it is provided as part of their dataset. Bordes et al. (2014) take advantage of the KB structure by projecting entities, relations, and subgraphs into the same latent space. In addition to finding the target entity, the other key information to first-order QA is the relation type corresponding to the question. Many researchers have shown that classifying the question into one of the predefined types (e.g., based on patterns (Zhang and Lee, 2003) or support vector machines (Buscaldi et al., 2010)) improves QA accuracy.

3 Approach

(a) From Question to Structured Query. Our approach relies on a *knowledge base*, containing a large set of *facts*, each one representing a binary [subject, relation, object] relationship. Since we assume *first-order questions*, the answer can be retrieved from a single fact. For instance, "How old is Sarah Michelle Gellar?" can be answered by the fact:

```
[Sarah Michelle Gellar,bornOn,4/14/1977]
```

The main idea is to dissect a first-order factoid natural-language question by converting it into a structured query: {entity "Sarah Michelle Gellar", relation: `bornOn`}. The process can be modularized into two machine learning problems, namely ***entity detection*** and ***relation prediction***. In the former, the objective is to *tag* each question word as either entity or not. In the latter, the objective is to *classify* the question into one of the K relation types. We modeled both using an RNN.

We use a standard RNN architecture: Each word in the question passes through an embedding lookup layer E, projecting the one-hot vector into a d-dimensional vector x_t. A recurrent layer combines this *input representation* with the hidden layer representation from the *previous word* and applies a non-linear transformation to compute the hidden layer representation for the *current word*. The hidden representation of the final recurrent layer is projected to the output space of k dimensions and normalized into a probability distribution via soft-max.

In *relation prediction*, the question is classified

into one of the 1837 classes (i.e., relation types in Freebase). In the *entity detection* task, each word is classified as either *entity* or *context* (i.e., $k = 2$).

Given a new question, we run the two RNN models to construct the structured query. Once every question word is classified as entity (denoted by `E`) or context (denoted by `C`), we can extract entity phrase(s) by grouping consecutive entity words. For example, for question "How old is Michelle Gellar", the output of entity detection is [`C C C E E`], from which we can extract a single entity "Michelle Gellar". The output of relation prediction is `bornOn`. The inferred structured query q becomes the following:

{*entityText*: "michelle gellar", *relation*: `bornOn`}

(b) Entity Linking. The textual reference to the entity (entityText in q) needs to be linked to an actual entity node in our KB. In order to achieve that, we build an *inverted index* I_{entity} that maps all n-grams of an entity ($n \in \{1, 2, 3\}$) to the entity's alias text (e.g., name or title), each with an associated $TF\text{-}IDF$ score. We also map the exact text ($n = \infty$) to be able to prioritize exact matches.

Following our running example, let us demonstrate how we construct I_{entity}. Let us assume there is a node e_i in our KB that refers to the actress "Sarah Michelle Gellar". The alias of this entity node is the name, which has three unigrams ("sarah", "michelle", "gellar"), two bigrams ("sarah michelle", "michelle gellar") and a single trigram (i.e., the entire name). Each one of these n-grams gets indexed in I_{entity} with $TF\text{-}IDF$ weights. Here is how the weights would be computed for unigram "sarah" and bigram "michelle gellar" ($\Rightarrow$ denotes mapping):

I_{entity}("sarah") $\Rightarrow$ {node : e_i,
score : $TF\text{-}IDF$("sarah", "sarah michelle gellar")}
I_{entity}("michelle gellar") $\Rightarrow$ {node : e_i,
score : $TF\text{-}IDF$("michelle gellar",
"sarah michelle gellar")}

This is performed for every n-gram ($n \in \{1, 2, 3, \infty\}$) of every entity node in the KB. Assuming there is an entity node, say e_j, for the actress "Sarah Jessica Parker", we would end up creating a second mapping from unigram "sarah":

I_{entity}("sarah") $\Rightarrow$ {node : e_j,
score : $TF\text{-}IDF$("sarah", "sarah jessica parker")}

In other words, "sarah" would be linked to both e_i and e_j, with corresponding $TF\text{-}IDF$ weights.

Once the index I_{entity} is built, we can link *entityText* from the structured query (e.g., "michelle gellar") to the intended entity in the KB (e.g., e_i). Starting with $n = \infty$, we iterate over n-grams of *entityText* and query I_{entity}, which returns all matching entities in the KB with associated $TF\text{-}IDF$ relevance scores. For each n-gram, retrieved entities are appended to the candidate set C. We continue this process with decreasing value of n (i.e., $n \in \{\infty, 3, 2, 1\}$)

Early termination happens if C is non-empty and n is less than or equal to the number of tokens in *entityText*. The latter criterion is to avoid cases where we find an exact match but there are also partial matches that might be more relevant: For "jurassic park", for $n = \infty$, we get an exact match to the original movie "Jurassic Park". But we would also like to retrieve "Jurassic Park II" as a candidate entity, which is only possible if we keep processing until $n = 2$.

(c) Answer Selection. Once we have a list of candidate entities C, we use each candidate node e_{cand} as a starting point to reach candidate answers.

A *graph reachability index* I_{reach} is built for mapping each entity node e to all nodes e' that are reachable, with the associated path $p(e, e')$. For the purpose of the current approach, we limit our search to a single hop away, but this index can be easily expanded to support a wider search.

$I_{reach}(e_i) \Rightarrow$
{node:e_{i_1}, text:The Grudge, path:[`actedIn`]}
{node:e_{i_2}, text:4/14/1977, path:[`bornOn`]}
{node:e_{i_3}, text:F. Prinze, path:[`marriedTo`]}

We use I_{reach} to retrieve all nodes e' that are reachable from e_{cand}, where the path from is consistent with the predicted relation r (i.e., $r \in p(e_{\text{cand}}, e')$). These are added to the candidate answer set A. For example, in the example above, node e_{i_2} would have been added to the answer set A, since the path [`bornOn`] matches the predicted relation in the structured query. After repeating this process for each entity in C, the highest-scored node in A is our best answer to the question.

4 Experimental Setup

Data. Evaluation of RNN-QA was carried out on SimpleQuestions, which uses a subset of Freebase containing 17.8M million facts, 4M unique entities, and 7523 relation types. Indexes I_{entity} and I_{reach} are built based on this knowledge base.

SimpleQuestions was built by (Bordes et al., 2014) to serve as a larger and more diverse factoid QA dataset.[1] Freebase facts are sampled in a way to ensure a diverse set of questions, then given to human annotators to create questions from, and get labeled with corresponding entity and relation type. There are a total of 1837 unique relation types that appear in SimpleQuestions.

Training. We fixed the embedding layer based on the pre-trained 300-dimensional Google News embedding,[2] since the data size is too small for training embeddings. Out-of-vocabulary words were assigned to a random vector (sampled from uniform distribution). Parameters were learned via stochastic gradient descent, using categorical cross-entropy as objective. In order to handle variable-length input, we limit the input to N tokens and prepend a special pad word if input has fewer.[3] We tried a variety of configurations for the RNN: four choices for the type of RNN layer (GRU or LSTM, bidirectional or not); depth from 1 to 3; and drop-out ratio from 0 to 0.5, yielding a total of 48 possible configurations. For each possible setting, we trained the model on the training portion and used the validation portion to avoid over-fitting. After running all 48 experiments, the most optimal setting was selected by micro-averaged F-score of predicted entities (entity detection) or accuracy (relation prediction) on the validation set. We concluded that the optimal model is a 2-layer bidirectional LSTM (BiLSTM2) for entity detection and BiGRU2 for relation prediction. Drop-out was 10% in both cases.

5 Results

End-to-End QA. For evaluation, we apply the relation prediction and entity detection models on each test question, yielding a structured query $q = \{entityText\colon t_e, relation\colon r\}$ (Section 3a). Entity linking gives us a list of candidate entity nodes (Section 3b). For each candidate entity e_{cand}, we can limit our relation choices to the set of unique relation types that some candidate entity e_{cand} is associated with. This helps eliminate the artificial ambiguity due to overlapping relation types as well as the spurious ambiguity due to redundancies in a typical knowledge base. Even though there are 1837 relation types in Freebase, the number of relation types that we need to consider per question (on average) drops to 36. The highest-scored answer node is selected by finding the highest scored entity node e that has an outward edge of type r (Section 3c). We follow Bordes et al. (2015) in comparing the predicted entity-relation pair to the ground truth. A question is counted as correct if and only if the entity we select (i.e., e) and the relation we predict (i.e, r) match the ground truth.

Table 1 summarizes end-to-end experimental results. We use the best models based on validation set accuracy and compare it to three prior approaches: a specialized network architecture that explicitly memorizes facts (Bordes et al., 2015), a network that learns how to convolve sequence of characters in the question (Golub and He, 2016), and a complex network with attention mechanisms to learn most important parts of the question (Yin et al., 2016). Our approach outperforms the state of the art in accuracy (i.e., precision at top 1) by 11.9 points (15.6% relative).

Model	P@1
Memory Network (2015)	63.9
Char-level CNN (2016)	70.9
Attentive max-pooling (2016)	76.4
RNN-QA (best models)	**88.3**
naive ED	58.9
naive RP	4.1
naive ED and RP	3.7

Table 1: Top-1 accuracy on test portion of SimpleQuestions. Ablation study on last three rows.

Last three rows quantify the impact of each component via an ***ablation study***, in which we replace either entity detection (ED) or relation prediction (RP) models with a naive baseline: (i) we assign the relation that appears most frequently in training data (i.e., `bornOn`), and/or (ii) we tag the entire question as an entity (and then perform the n-gram entity linking). Results confirm that RP is absolutely critical, since both datasets include a diverse and well-balanced set of relation types. When we applied the naive ED baseline, our results drop significantly, but they are still comparable to prior results. Given that most prior work do not use the network to detect entities, we can

[1] 75910/10845/21687 question-answer pairs for training/validation/test is an order of magnitude larger than comparable datasets. Vocabulary size is 55K as opposed to around 3K for WebQuestions (Berant et al., 2013).

[2] `word2vec.googlecode.com`

[3] Input length (N) was set to 36, the maximum number of tokens across training and validation splits.

deduce that our RNN-based entity detection is the reason our approach performs so well.

Error Analysis. In order to better understand the weaknesses of our approach, we performed a *blame analysis*: Among 2537 errors in the test set, 15% can be blamed on entity detection — the relation type was correctly predicted, but the detected entity did not match the ground truth. The reverse is true for 48% cases.[4] We manually labeled a sample of 50 instances from each blame scenario. When entity detection is to blame, 20% was due to spelling inconsistencies between question and KB, which can be resolved with better text normalization during indexing (e.g., "la kings" refers to "Los Angeles Kings"). We found 16% of the detected entities to be correct, even though it was not the same as the ground truth (e.g., either "New York" or "New York City" is correct in "what can do in new york?"); 18% are inherently ambiguous and need clarification (e.g., "where bin laden got killed?" might mean "Osama" or "Salem"). When blame is on relation prediction, we found that the predicted relation is reasonable (albeit different than ground truth) 29% of the time (e.g., "what was nikola tesla known for" can be classified as `profession` or `notable_for`).

RNN-QA in Practice. In addition to matching the state of the art in effectiveness, we also claimed that our simple architecture provides an efficient and modular solution. We demonstrate this by applying our model (without any modifications) to the entertainment domain and deploying it to the Comcast X1 platform serving millions of customers every day. Training data was generated synthetically based on an internal entertainment KB. For evaluation, 295 unique question-answer pairs were randomly sampled from real usage logs of the platform.

We can draw two important conclusions from Table 2: First of all, we find that almost all of the user-generated natural-language questions (278/295∼95%) are first-order questions, supporting the significance of first-order QA as a task. Second, we show that even if we simply use an open-sourced deep learning toolkit (`keras.io`) for implementation and limit the computational resources to 2 CPU cores per thread, RNN-QA answers 75% of questions correctly with very reasonable latency.

[4] In remaining 37% incorrect answers, both models fail, so the blame is shared.

Error	Count
Correct	220
Incorrect entity	16
Incorrect relation	42
Not first-order question	17
Total Latency	76±16 ms

Table 2: Evaluation of RNN-QA on real questions from X platform.

6 Conclusions and Future work

We described a simple yet effective approach for QA, focusing primarily on first-order factual questions. Although we understand the benefit of exploring task-agnostic approaches that aim to capture semantics in a more general way (e.g., (Kumar et al., 2015)), it is also important to acknowledge that there is no "one-size-fits-all" solution as of yet.

One of the main novelties of our work is to decompose the task into two subproblems, entity detection and relation prediction, and provide solutions for each in the form of a RNN. In both cases, we have found that bidirectional networks are beneficial, and that two layers are sufficiently deep to balance the model's ability to fit versus its ability to generalize.

While an ablation study revealed the importance of both entity detection and relation prediction, we are hoping to further study the degree of which improvements in either component affect QA accuracy. Drop-out was tuned to 10% based on validation accuracy. While we have not implemented attention directly on our model, we can compare our results side by side on the same benchmark task against prior work with complex attention mechanisms (e.g., (Yin et al., 2016)). Given the proven strength of attention mechanisms, we were surprised to find our simple approach to be clearly superior on SimpleQuestions.

Even though deep learning has opened the potential for more generic solutions, we believe that taking advantage of problem structure yields a more accurate and efficient solution. While first-order QA might seem like a solved problem, there is clearly still room for improvement. By revealing that 95% of real use cases fit into this paradigm, we hope to convince the reader that this is a valuable problem that requires more *attention*.

References

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Matthew W Bilotti, Jonathan Elsas, Jaime Carbonell, and Eric Nyberg. 2010. Rank Learning for Factoid Question Answering with Linguistic and Semantic Constraints. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 459–468, New York, NY, USA. ACM.

Matthew W Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. 2007. Structured Retrieval for Question Answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 351–358, New York, NY, USA. ACM.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Davide Buscaldi, Paolo Rosso, José Manuel Gómez-Soriano, and Emilio Sanchis. 2010. Answering Questions with an N-gram Based Passage Retrieval Engine. *J. Intell. Inf. Syst.*, 34(2):113–134.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 260–269. The Association for Computer Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. *arXiv preprint arXiv:1604.00727*.

Alex Graves. 2008. *Supervised sequence labelling with recurrent neural networks*. Ph.D. thesis, Technical University Munich.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 172–175. Association for Computational Linguistics.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1576–1586. The Association for Computational Linguistics.

Wei-Ning Hsu, Yu Zhang, and James R. Glass. 2016. Recurrent neural network encoder with attention for community question answering. *CoRR*, abs/1603.07044.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *CoRR*, abs/1506.07285.

Zefu Lu, Lei Li, and Wei Xu. 2015. Twisted recurrent network for named entity recognition. In *Bay Area Machine Learning Symposium*.

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.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2786–2792. AAAI Press.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916. ACM.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *TACL*, 2:377–392.

Peilu Wang, Yao Qian, Frank K Soong, Lei He, and Hai Zhao. 2015. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *arXiv preprint arXiv:1510.06168*.

Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. 2016. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 287–296. ACM.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *EMNLP*, pages 645–650.

Wenpeng Yin, Mo Yu, Bing Xiang, Bowen Zhou, and Hinrich Schütze. 2016. Simple question answering by attentive convolutional neural network. *arXiv preprint arXiv:1606.03391*.

Dell Zhang and Wee Sun Lee. 2003. Question Classification Using Support Vector Machines. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 26–32, New York, NY, USA. ACM.