Focused Reading: Reinforcement Learning for What Documents to Read

Enrique Noriega-Atala [1] **Marco A. Valenzuela-Escarcega** [1] **Clayton T. Morrison** [1] **Mihai Surdeanu** [1]

Abstract

Recent efforts in bioinformatics have achieved tremendous progress in the machine reading of biomedical literature and the assembly of the extracted biochemical interactions into large-scale models such as protein signaling pathways. However, batch machine reading of literature at today's scale (PubMed alone indexes over 1 million papers per year) is infeasible due to both cost and processing overhead. In this work we propose *focused reading* as an alternative. Focused reading casts machine reading as a search process where queries are defined by pairs of entities, e.g., proteins, to be connected through the models extracted from the literature. Our approach casts the task as an incremental search over the graph of biochemical interactions, where each focused reading step makes a choice between widening the search space (exploration), or focusing on the most relevant documents (exploitation). We learn strategies that distinguish between exploration and exploitation using reinforcement learning (RL), and demonstrate that an RL-learned strategy is capable of answering more queries while reading fewer papers than a strong deterministic baseline.

1. Introduction

The millions of academic papers in the biomedical domain contain a vast amount of information that may lead to new hypotheses for disease treatment. However, scientists are faced with a problem of "undiscovered public knowledge," as they struggle to read and assimilate all of this information (Swanson, 1986). Furthermore, the literature is growing at an exponential rate (Pautasso, 2012); PubMed[1] has been adding more than a million papers per year since 2011. We have surpassed our ability to keep up with and integrate these findings through manual reading alone.

Large ongoing efforts, such as the BioNLP task community (Nédellec et al., 2013; Kim et al., 2012; 2009) and the DARPA Big Mechanism Program (Cohen, 2015), are making progress in advancing methods for machine reading and assembly of extracted biochemical interactions into large-scale models. However, to date, these methods rely on either manual retrieval of relevant documents by humans, or processing large batches of documents that may or may not be relevant to the model being constructed.

Batch machine reading of literature at this scale poses a new, growing set of problems. First, access to some documents is costly. The PubMedCentral (PMC) Open Access Subset[2] (OA) is estimated[3] to comprise 20%[4] of the total literature; the remaining full-text documents are only available through paid access. Second, while there have been great advances in quality, machine reading is still not solved. Updates to our readers requires reprocessing the documents. For large document corpora, this quickly becomes the chief bottleneck in information extraction for model construction and analysis. Finally, even if we could cache all reading results, the search for connections between concepts within the extracted results should not be done blindly. At least in the biology domain, the many connections between biological entities and processes leads to a very high branching factor, making blind search for paths intractable.

To effectively read at this scale, we need to incorporate methods for *focused reading*: develop the ability to pose queries about concepts of interest and perform targeted, incremental search through the literature for connections between concepts while minimizing reading documents that are likely irrelevant.

For example, suppose a biologist is interested in what the literature has to say about how protein Pi3k affects protein KTF. Figure 1 gives a schematic representation of what the focused reading search might look like: given the initial seed query entities, Pi3k and KTF, focused reading re-

[1]The University of Arizona, Tucson, Arizona, USA. Correspondence to: Enrique Noriega-Atala <enoriega@email.arizona.edu>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

[1]http://www.ncbi.nlm.nih.gov/pubmed

[2]https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

[3]https://tinyurl.com/bachman-oa

[4]This includes 5% from PMC author manuscripts.

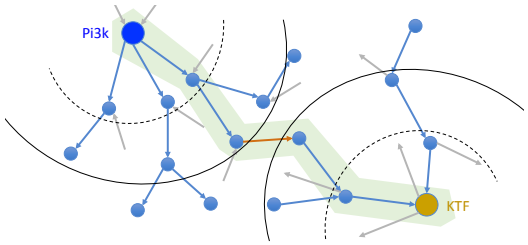

Figure 1. Evolution of search for the directed path that connects two proteins during focused reading.

trieves a set of papers that mention both, extracts interactions that connect additional entities to the seed entities, and adds them to an expanding graph of connected entities. The first pass adds new entities around Pi3k and KTF (within the dashed circles in Fig. 1), and a second pass expands the graph further (to include all entities within the solid circle boundaries). Eventually two entities on the periphery of the expanding subgraphs are linked (the orange direct edge in the figure) and focused reading returns the complete path from Pi3k to KTF. The key research challenge is then: how can focused reading search in a way that finds these paths while minimizing the number of papers that need to be read?

In this paper we present what we believe is the first algorithm for focused reading. We make the following contributions:
(1) Present a general framework for a family of possible focused reading algorithms along with a baseline instance.
(2) Cast the design of focused reading algorithms in a reinforcement learning (RL) setting, where the machine decides if it should explore (i.e., cast a wider net) or exploit (i.e., focus reading on a specific topic).
(3) Evaluate our focused reading policies in terms of search efficiency and quality of information extracted. The evaluation demonstrates the effectiveness of the RL method: this approach found more information than the baseline we propose, while reading fewer documents.

2. Related Work

The past few years have seen a large body of work on information extraction (IE), particularly in the biomedical domain. This work is too vast to be comprehensively discussed here. We refer the interested reader to the BioNLP community (Nédellec et al., 2013; Kim et al., 2012; 2009) for a starting point. However, most of this work focuses on entity/relation/event extraction given a document, not on *what documents to read* given a goal. To our knowledge, we are the first to focus on the latter task.

Reinforcement learning has been used to achieve state of the art performance in several natural language processing (NLP) and information retrieval (IR) tasks. For example, RL has been used to guide IR and filter irrelevant web content (Seo & Zhang, 2000; Zhang & Seo, 2001). More recently, RL has been combined with deep learning with great success, e.g., for improving coreference resolution (Clark & Manning, 2016). Finally, RL has been used to improve the efficiency of IE by learning how to incrementally reconcile new information and help choose what to look for next (Narasimhan et al., 2016), a task close to ours. This serves as an inspiration for the work we present here, but with a critical difference: Narasimhan et al. (2016) focus on slot filling using a pre-existing template. This makes both the information integration and stopping criteria well-defined. On the other hand, in our focused reading domain there is no well-defined template: we do not know ahead of time which new pieces of information are necessarily relevant and must consider this in the context of our search.

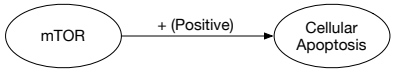

Figure 2. Graph edge encoding the relation extracted from: *mTOR triggers cellular apoptosis*.

3. Focused Reading

In this paper we consider focused reading for the biomedical domain, and we focus on binary promotion/inhibition interactions between bio entities. Thus, the IE component constructs a directed graph, where vertices represent entities participating in an interaction (protein, gene, gene product, or other biological process), and edges represent directed activation interactions. Edge labels indicate whether the controller entity has a *positive* (promoting) or *negative* (inhibitory) influence on the controlled entity. Figure 2 shows an example edge in this graph. Importantly, this graph is constructed on the fly, as the IE system is incrementally exposed to more papers.

We use REACH[5], an open source IE system (Valenzuela-Escárcega et al., 2015), to extract interactions from unstructured biomedical text. We couple this IE system with a Lucene[6] index of biomedical publications to retrieve papers based on queries about entity mentions in the text (as discussed below).

Importantly, we essentially use IE as a black box[7], and focus on strategies that guide *what* the IE system reads for a complex information need. In particular, we consider the scenario where a biologist (or other model-building process) queries the literature about how one entity (source)

[5] https://github.com/clulab/reach
[6] https://lucene.apache.org
[7] Thus, our method could potentially work with any IE system.

Algorithm 1 Focused Reading

```
1:  procedure FocusedReading(S, D)
2:      G ← {{S, D}, ∅}
3:      repeat
4:          Σ ← EndpointStrategy(G)
5:          (A, B) ← ChooseEndPoints(Σ, G)
6:          Q ← ChooseQuery(A, B, G)
7:          (V, E) ← Lucene+Reach(Q)
8:          Expand(V, E, G)
9:      until IsConnected(S, D) OR StopConditionMet(G)
10: end procedure
```

affects another (destination), where the connection is typically indirect (as in the example in Figure 1).

Algorithm 1 outlines the general focused reading algorithm for this task. In the algorithm, S, D, A, and B represent individual entities, where S and D are the **s**ource and **d**estination entities in the user query. G is the graph of interactions that is iteratively constructed during the focused reading procedure, with V being the set of vertices (entities), and E the set of edges (connecting entities participating in an interaction). Σ is a strategy for selecting entities, while Q is a Lucene query constructed in each iteration to retrieve new papers to read.

The algorithm initializes the search graph to contain the two unconnected entities as vertices: $\{S, D\}$ (line 2). The algorithm then enters into its central loop (lines 3 through 9). The loop terminates when one or more directed paths connecting S to D are found, or when a stopping condition is met: either G has not changed since the previous run through the loop, or after exceeding some number of iterations through the loop (in this work, ten).

At each pass through the loop the algorithm grows the search graph as follows:

1. The graph G is initialized with two nodes, the source (S) and destination (D) in the user's query, and no edges (because we have not read any papers yet to understand what interactions exist).

2. Given the current graph, choose a strategy, Σ, for selecting which entities to query next: *exploration* or *exploitation* (line 4). In general, exploration aims to widen the search space by adding many more nodes to the graph, whereas exploitation aims to narrow the search by focusing on entities in a specific region of the graph.

3. Using strategy Σ, choose the next entities to attempt to link: (A, B) (line 5).

4. Choose a query, Q: again, *exploration* or *exploitation*, following the same intuition as with the entity choice strategy (line 6). Here exploration queries retrieve a wider range of documents, while exploitation queries are more restrictive.

5. Run the Lucene query to retrieve papers and process the papers using the IE system. The result of this call is a set of interactions, similar to the interaction in Figure 2 (line 7).

6. Add the new interaction participant entities (vertices V) and directed influences (edges E) to the search graph (line 8).

7. If the source and destination entities are connected in G, stop: the user's query has been satisfied. Stop with a failure if G has not changed since the last run through the loop, or we've gone through the loop 10 times. Otherwise, continue from step 2.

The central loop performs a bidirectional search in which each iteration expands the search horizon outward from S and D (as depicted in Figure 1). Algorithm 1 represents a family of possible focused reading algorithms, differentiated by how each of the functions in the main loop are implemented. In this work, IsConnected stops after a single path is found, but a variant could consider finding multiple paths, paths of some length, or incorporate other criteria about the properties of the path. We next consider particular choices for the inner loop functions.

4. Baseline Algorithm and Evaluation

The main functions that affect the search behavior of Algorithm 1 are EndpointStrategy and ChooseQuery. Here we describe a *baseline* focused reading implementation in which EndpointStrategy and ChooseQuery are designed to attempt to find any directed path from S to D as quickly as possible, that is, in as few passes through the inner loop, which in turn should tend to minimize the number of papers that must be read.

There are a variety of approaches to consider for how EndpointStrategy can implement exploration versus exploitation strategies. For the baseline, we consider the following interpretations: *exploration* will involve selecting entities that are more connected to other entities, and therefore more likely to select documents that introduce more entities, while *exploit* will focus on entities that have been introduced more recently to the graph under the intuition that searching for what they are connected to may be more likely to reveal a path between the fringes of the S and D subgraphs. Either strategy has potential advantages for finding a connecting path: exploration will retrieve more potential paths, but as the cost of reading more papers, while exploiting may more quickly connect the fringes, but only if the current fringes are sufficiently close.

	Baseline	Best RL Query Policy	
# IR queries	573	433	25% decrease
Unique papers read	26,197	19,883	24% decrease
# Paths recovered (out of 289)	189 (65%)	198 (68%)	3% increase

Table 1. Results of the baseline and RL Query Policy for the focused reading of biomedical literature.

For the baseline, we fix the ENDPOINTSTRATEGY to always *explore*, under the intuition that search will introduce more entities earlier, and therefore be more likely to connect quickly; although this might be at the cost of potentially reading more than needed, it is still better than exhaustively reading the entire corpus. Under this strategy, CHOOSEENDPOINTS chooses entities (A, B) that currently have the most inward and outgoing edges (i.e., highest vertex degree) in the current state of G (disallowing choosing an entity pair used in a previous query).

Now that we have our candidate entities (A, B), our next step is to formulate how we will use these entities to retrieve new papers. Here we consider two classes of queries: (1) we restrict our query to only retrieve papers that simultaneously mention both A and B, and therefore is more likely to retrieve a paper with a direct link between A and B (*exploit*), or (2) we retrieve papers that mention either A or B, therefore generally retrieving more papers that will introduce more new entities (*explore*). For our baseline, where we are trying to find a path between S and D as quickly as possible, we implement a greedy CHOOSEQUERY: first try the conjunctive exploitation query; if no documents are retrieved, then "relax" the search to the disjunctive exploration query.

4.1. Data Set

To evaluate the baseline, we constructed a data set based on a collection of papers seeded by a set of 132 entities that come from the University of Pittsburgh's **Dy**namic **C**ell **E**nvironment (DyCE) model, a biomolecular model of pancreatic cancer (Telmer et al., 2017). These entities are known to participate in protein signaling pathways that drive pancreatic cancer. Using these entities, we retrieved 70,719 papers that mention them. We processed all papers using REACH, extracting all of the interactions mentioned, and converted them into a single graph. The resulting graph consisted of approximately 80,000 vertices, 115,000 edges, and had an average (undirected) vertex degree of 24. We will refer to this graph as the *REACH graph*, as it represents what *can* be retrieved by REACH from the set of 70K papers. It is important to note that our focused reader has access to this graph only during training, and *not* at evaluation time. During testing, the focused reader must dynamically reconstruct the subset of the graph necessary to answer the given information need.

We then conducted an exhaustive shortest path search for directed paths between all the entity pairs that are known to be connected in a signaling pathway (according to the cancer researchers) and found a total of 789 entity pairs connected by a directed path in the REACH graph. We selected 289 of these entity pairs perform an initial evaluation of our baseline algorithm, described in the next section. (In the more extensive evaluation in Section 5 we partition all 789 entity pairs into three parts to perform cross validation.)

4.2. Baseline Results

We ran this baseline focused reading algorithm on each of the 289 pairs of entities, in each case attempting to recover a directed path from one to the other. The results are summarized in the middle column of Table 1: by issuing a total of 573 queries, the baseline read 26,197 papers out of the total 70,719 papers (37% of the corpus), in order to recover 189 of the paths (65% of the paths that could be found using this corpus).

4.3. Baseline Error Analysis

Although the baseline recovers paths between nearly two thirds of the test pairs, we would like to understand the conditions under which it failed to find the remaining paths that we know exist within the REACH graph (Section 4.1). We performed an error analysis on a random subsample of 82 of the test pairs where the baseline failed to find a path – in all of these cases, the baseline algorithm reached the limit of 10 iterations through the inner search loop without finding a path. Table 2 presents a summary of the types of errors that caused the baseline implementation to fail.

Error cause	Frequency
NER error	12
Ungrounded ID	19
Empty query result	20
Premature finish	4
QUERYCHOICE exploiting too early	5
Poor choice of endpoint entities	21

Table 2. Baseline error causes

We detail these error types next:

NER error and *Ungrounded ID* represent cases where either the reader made a mistake in labeling an entity, or the entity grounding component, which attempts to link each

extracted named entity to a knowledge base of known entities, e.g., Uniprot for protein names,[8] does not currently cover the entity name.[9] These kinds of errors can be addressed as part of improvements to the IE component but are out of the scope of focused reading.

When the algorithm *finished prematurely*, nothing inherently incorrect happened during the search process, but the search reached the maximum number of iterations allowed. We suspect that letting the search continue for a few more iterations would have found an answer.

QUERYCHOICE *exploiting too early* occurs when, in the cascading strategy, the most restrictive (i.e., exploitation) IR query succeeds but returns very few documents in the first couple iterations and the information contained in the retrieved documents already existed in the graph. In this case, the exploitative search was too restrictive and resulted in no new information being added. This could have been mitigated if the algorithm chose to do a less restrictive query (explore) early on the process to retrieve a larger set of documents and therefore added more information.

Poor choice of endpoint entities is the problem in which both endpoint entities are correctly grounded, but the retrieved set of documents does not add new information to G. In this case, the ENDPOINTSTRATEGY and CHOOSEENDPOINTS procedures should have selected different entities to try to connect.

In summary, the baseline error analysis suggests that, while focused reading is indeed possible, choosing when to explore versus when to exploit in QUERYCHOICE and selecting endpoint entities (a function of ENDPOINTSTRATEGY and CHOOSEENDPOINTS) is not trivial and leaves room for improvement.

5. Reinforcement Learning for Focussed Reading

From the above analysis, we found that a significant number of the failures might have been avoided had the algorithm used a different strategy for ENDPOINTSTRATEGY and/or CHOOSEQUERY. Informally, the baseline model chose to exploit when it should have explored, or viceversa. The conditions for making different choices depend on the current state of G, and earlier query behavior can affect later query opportunities, making this an iterative decision making problem and a natural fit for a RL formulation.

Inspired by this observation, we consider RL for finding a better policy for ENDPOINTSTRATEGY and CHOOSEQUERY. We consider a space of four possible actions; each "action" must include both an endpoint entity selection and a query choice, but there are two (explore/exploit) options for each:

- In the context of choosing the endpoints, either *exploit*, which restricts the choices only to the most recently added vertices, or *explore*, which considers the highest ranked vertices across the whole graph. For example, suppose focused reading is in iteration 8, there is an entity, α, that was introduced in iteration 1 and is connected to 5 other entities in the current search graph, and another entity, β, that was introduced in the previous iteration (iteration 7) and is connected to 2 other entities. In this case exploit would select β, while explore would select α.
- In the context of querying, *exploit* produces a conjunctive query, whereas *explore* uses a wider, disjunctive query. For example, if building a query to retrieve documents about entities α and β, exploit will require that the documents mention both α AND β in the same document, while explore will retrieve any documents that mention either α OR β.

Altogether, the four possible actions are then: (1) Endpoint Explore + Query Explore, (2) Endpoint Exploit + Query Explore, (3) Endpoint Explore + Query Exploit, and (4) Endpoint Exploit + Query Exploit.

Table 3 summaries the features that are used to describe the *state* of the search. Note that the features include representation of the history of the search (e.g., how many times the algorithm used a particular entity to search), as well as the state of the graph (e.g., how well connected two entities are).

With the goal of recovering paths as quickly as possible, we provide a reward of $+1$ if the algorithm successfully finds a path, a reward of -1 if the search fails to find a path, and assess a "living reward" of -0.05 for each step during the search, to encourage trying to finish the search as quickly as possible.

Based on the above framing of focused reading as a RL problem, we evaluated different configurations of RL-based focused reading: (1) *RL All Actions* learns a policy for choosing among all four actions combinations; (2) *RL Endpoint Only* fixes the query strategy to be identical to the baseline but learns a policy for selecting between endpoint explore and exploit; and (3) *RL Query Only* fixes the endpoint selection strategy to be identical to the baseline but learns a policy for selecting between query choice exploration and exploitation.

Since our interest is ultimately in improving the efficiency

[8] http://www.uniprot.org

[9] This happens in the current reader because it uses a conditional random field model for named entity recognition, which sometimes introduces false positives that do not exist in the relevant knowledge bases.

Feature Name	Description
Iteration	Current iteration # of the search
PA query log count	How many time has PA been used in previous queries
PB query log count	How many time has PB been used in previous queries
Same component	Are PA and PB in the same connected component of G?
Iteration of introduction for PA	Which iteration first introduced PA
Iteration of introduction for PB	Which iteration first introduced PB
PA Rank	PA's rank by total degree in G
PB Rank	PB's rank by total degree in G

Table 3. Features that describe the state of search in the RL-based focused reading algorithm. "PA" stands for "participant A", i.e., the latest entity chosen from the source subgraph. "PB" stands for "participant B", i.e., the latest entity chosen from the destination subgraph. "Iteration" indicates the number of times focused reading has gone through the central loop in Algorithm 1.

	Fold 1				Fold 2				Fold 3			
	Paths	Queries	Papers	Score	Paths	Queries	Papers	Score	Paths	Queries	Papers	Score
RL Query Only	189	398	20,151	**0.93**	182	392	19,682	**0.92**	207	358	18,107	**1.14**
RL All Actions	189	412	20,791	0.90	183	415	19,884	**0.92**	208	388	19,355	1.07
Baseline	180	459	25,550	0.70	176	500	26,270	0.66	192	462	25,250	0.76

Table 4. 3-fold cross validation results comparing two RL focused reading models against the baseline.

of focused reading search, that is, recovering the most paths while minimizing the number of papers read, we define the performance *score* as the ratio $\frac{\text{\# paths}}{\text{\# papers}}$; a higher scores is better as it means we are generally being finding more relevant paths in the papers we retrieved.

We trained the three different configurations of RL focused reading using SARSA and Q-Learning (Sutton & Barto, 1998). As the number of unique states is large, we used a linear approximation of the q-function. Once the policy converged during training, we fixed the linear q-function estimate and used this as the fixed policy for selecting queries at evaluation time.

We evaluated the variations of RL Policies on the same data set of entity pairs used to evaluate the baseline as well as performed a three-fold cross validation where two folds were used for training and one for testing; each fold contained exactly 263 entity pairs for which a path can be found in the paper corpus. Table 4 reports the results of the most intersting RL policies and baseline for the three-fold cross validation. These results show that the RL Query only policy, which fixes the Endpoint selection strategy to *exploration* while learning the query strategy, generally performs the best. (We did consider two variants of the RL Endpoint Only policy: one that chooses query exploit only, which was found to do worse than the baseline, and query explore only, which does better than the baseline but worse than the other two policies.) Table 1 compares the results for the baseline under the initial evaluation set (described in Sec. 4.1) against the RL Query Only, and here we see that compared to the baseline, the RL Query Policy resulted in a 25% reduction in the number of queries that were run, leading to a 24% reduction in the number of papers that were read, while at the same time *increasing* the number of paths recovered by 3%.

We tested the statistical significance of the difference in results between the baseline and the best RL policy on the testing dataset by performing a bootstrap resampling test. Our hypotheses were that the policy reads fewer papers, makes fewer queries and finds more paths. The resulting estimated p-values for fewer papers and fewer queries were found to be near 0, and $p < 0.003$ for finding more paths.

5.1. Ablation Test for RL State Features

We performed a feature ablation study to assess what contribution features make to learning efficacy. The results are summarized in Table 5. The features are clustered into five different groups. We measured the impact of removing one feature group at a time. The table highlights that removing the *Entity introduction* feature (keeping track of which iteration a biological entity is first introduced to the graph during search) has a small penalty in the amount of paths recalled, but achieves the highest ratio of paths found to papers read score. Individually removing most of the other features had a negative impact, indicating that each is indeed important to model both the state of the graph and the history of the search. All in all, using all the features achieves a good balance across the three metrics reported in the table. This suggests that the design of relevant features for RL-based focused reading is not trivial and deserves further attention. We leave this as future work.

	All features	− Same component	− Ranks	− Iteration number	− Query counts	− Entity introduction
Paths found	198	201	**202**	199	200	196
Papers read	19,883	20,531	20,463	19,893	20,918	**17,936**
Queries made	433	467	469	487	484	**403**
Score	99.58	97.90	98.71	100.04	95.61	**109.28**

Table 5. Ablation test over the features that encode the state of the RL policies.

Error cause	Freq.
Empty query result	12
Ungrounded participant	4
Low yield from IE	2

Table 6. Error analysis for the best RL-based focused reading policy.

5.2. Error Analysis of the RL Policy

Finally, we analyzed the execution trace of eighteen (20% of the errors) of the searches that failed to find a path under RL. The results are summarized in Table 6. The table shows that the main source of failures is receiving *no results* from the information retrieval query, i.e., when the IR system returns zero documents for the chosen query. This is typically caused by over-constrained queries. The second most common source of failures was *ungrounded participants*, i.e., when at least one of the selected participants that form the query could not be linked to our protein knowledge base. This is generally caused by mistakes in our NER sequence model, and also tends to yield no results from the IR component. Finally, the *low yield from IE* situation appears when the the information produced through machine reading in one iteration is scarce and adds no new components to the interaction graph, again resulting in a stop condition.

6. Conclusions and Future Work

We introduced a framework for the focused reading of biomedical literature, which is necessary to handle the data overload that plagues even machine reading approaches. We have presented a general focused reading algorithm family, an intuitive baseline algorithm instance from that family, and formulated a reinforcement learning approach to search for better policies for choosing which entities to use and how to construct a query given entities. Informally, the RL approach learns when focused reading should explore (widen its search) or exploit (narrow the search). We demonstrated that the RL-based focused reading is more efficient than the baseline (e.g., reading 24% fewer papers), while successfully finding 3% more target paths.

There are many exciting directions to take this work. First, there are many more potential EndpointsStrategy and QueryChoice actions to explore, along with additional search state information that can be incorporated. Second, we plan to be more flexible with the query actions. Instead of facing a binary action choice (explore vs. exploit), we will use a battery of choices where the queries allow flexibility in the number of words between the entities of interest and on the document retrieval limits, which could lead us to further improvements to the number of processed papers. Third, we will expand focused reading to efficiently search for multiple paths between the source entity (S) and destination (D). Finally, we must incorporate additional constraints into the search itself; biologists are interested in paths that occur in particular biological contexts (e.g., healthy versus cancer pancreatic cells), and paths that help explain specific kinds of influence. These constraints should be incorporated into the ranking of candidate search strategies.

Acknowledgements

This work was partially funded by the Defense Advanced Research Projects Agency (DARPA) Big Mechanism program under ARO contract W911NF-14-1-0395.

Dr. Mihai Surdeanu discloses a financial interest in Lum.ai. This interest has been disclosed to the University of Arizona Institutional Review Committee and is being managed in accordance with its conflict of interest policies.

References

Clark, Kevin and Manning, Christopher D. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*, 2016.

Cohen, Paul R. DARPA's Big Mechanism program. *Physical Biology*, 12(4):045008, 2015.

Kim, Jin-Dong, Ohta, Tomoko, Pyysalo, Sampo, Kano, Yoshinobu, and Tsujii, Jun'ichi. Overview of bionlp'09 shared task on event extraction. In *Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing: Shared Task*, pp. 1–9. Association for Computational Linguistics, 2009.

Kim, Jin-Dong, Nguyen, Ngan, Wang, Yue, Tsujii, Jun'ichi, Takagi, Toshihisa, and Yonezawa, Akinori. The genia event and protein coreference tasks of the bionlp shared task 2011. *BMC bioinformatics*, 13(11):1, 2012.

Narasimhan, Karthik, Yala, Adam, and Barzilay, Regina. Improving information extraction by acquiring external evidence with reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2016)*, 2016.

Nédellec, Claire, Bossy, Robert, Kim, Jin-Dong, Kim, Jung-Jae, Ohta, Tomoko, Pyysalo, Sampo, and Zweigenbaum, Pierre. Overview of bionlp shared task 2013. In *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 1–7, 2013.

Pautasso, Marco. Publication growth in biological subfields: patterns, predictability and sustainability. *Sustainability*, 4(12):3234–3247, 2012.

Seo, Young-Woo and Zhang, Byoung-Tak. A reinforcement learning agent for personalized information filtering. In *Proceedings of the 5th international conference on Intelligent user interfaces*, pp. 248–251. ACM, 2000.

Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

Swanson, Don R. Undiscovered public knowledge. *The Library Quarterly*, 56(2):103–118, 1986.

Telmer, C. A., Sayed, K., Butchy, A. A., Kaltenmeir, Lotze, Michael, and Miskov-Zivanov, N. Manuscript in preparation. 2017.

Valenzuela-Escárcega, Marco A., Hahn-Powell, Gustave, Hicks, Thomas, and Surdeanu, Mihai. A domain-independent rule-based framework for event extraction. 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 Assian Federation of Natural Language Processing: Software Demonstrations (ACL-IJCNLP)*, 2015.

Zhang, Byoung-Tak and Seo, Young-Woo. Personalized web-document filtering using reinforcement learning. *Applied Artificial Intelligence*, 15(7):665–685, 2001.