Where Was Alexander the Great in 325 BC?
Toward Understanding History Text with a World Model

Yuki Murakami and Yoshimasa Tsuruoka
The University of Tokyo, 3-7-1 Hongo, Bunkyo-ku, Tokyo, Japan
{murakami,tsuruoka}@logos.t.u-tokyo.ac.jp

Abstract

We present a toy world model for interpreting textual descriptions of the movement record of a historical figure such as Genghis Khan or Napoleon. We cast the problem of document understanding as the task of finding episodes that do not violate the soft constraint conditions derived from the document. The model thus allows us to infer his or her locations by finding multiple solutions of an optimization problem. Our experimental results using Wikipedia text on Alexander the Great demonstrate that such inference can indeed be performed with reasonable accuracy. We also show that the information obtained from such inference is useful in solving a hard coreference resolution problem.

1 Introduction

Recent decades have witnessed great strides in data-driven language processing technology, yet there are still many unsolved problems when the machine has to deal with the meaning of a document. Let us consider the following simple question-answering problem.

- Document:
  David left Paris on the 20th of July, driving his favorite Peugeot. He arrived in Athens on the 22nd.
- Question:
  Where was David on the 21st?
  A. London B. Budapest C. Berlin D. New York

A possible answer to this question would be "He was probably in Budapest, although there is a small chance that he was in Berlin". Putting aside the problem of natural language generation, the machine would have to have geographical knowledge and perform some kind of inference about his movement if it is to give a sensible answer to this question.

This paper presents a toy world model that allows us to perform such inference. We test this approach as a first step toward building a computer system that can "understand" documents on world history and answer various questions about historical figures and events. Our aim is to go beyond traditional question-answering frameworks in which the system can only answer the questions about the facts that are explicitly written in the document. We aim to build a system that can simulate what could have happened in the world history using an internal model and give a reasonable answer to any question as long as the answer can be inferred from other pieces of information available in the document.

In this paper, we focus on the much simpler subproblem of modeling the movement record of a historical figure. Our world model is simply an undirected graph with an agent moving on it, and his potential movement histories are obtained as possible solutions to an optimization problem. In experiments, we show that our system can perform inference about his locations with reasonable accuracy and the information obtained from such inference is useful in solving a hard coreference resolution problem.

2 Related Work

There is an increasing body of research on using world knowledge and inference in high-level text processing tasks such as textual entailment, coreference resolution and question answering (Tatu and Moldovan, 2005; Fowler et al., 2005; Rahman and Ng, 2011; Peng et al., 2015; Berant et al., 2015). However, most of the existing approaches use "static" knowledge that is typically expressed as a collection of n-ary relations between entities, and there is little work that attempts to model the dynamics of a world.

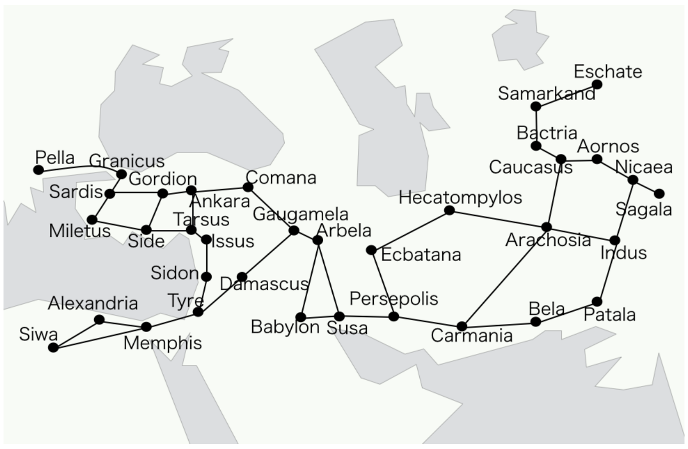

Figure 1: The graph overlayed on the map

Our work is much closer in spirit to SHRDLU (Winograd, 1971), where natural language queries were processed using a world model for toy blocks. More recent research efforts for connecting language with physical world include Logical Semantics with Perception (Krishnamurthy and Kollar, 2013), referential grounding (Liu et al., 2014), 3D scene generation from text (Chang et al., 2014) and generation of QA tasks by simulation (Weston et al., 2015). Our work can be seen as an attempt of grounding textual descriptions in history text to a simulation model for world history.

Our work is also related to previous work on representing structured sequences of actions and events using *scripts* (Schank and Abelson, 1977). Chambers and Jurafsky (2008) proposed a narrative chain model based on scripts. They focused on a particular character, extracted chains of events on his behavior using verbs and their arguments, and sorted them by learning.

3 Model and Inference

3.1 Toy World Model

Our toy world model consists of an agent and an undirected graph $G = (V, E)$, where V is the set of its nodes and E is the set of its edges. Let $h_t \in V$ denote the location of the agent at some discrete time t. The agent starts from the initial node h_0 and, at each time step, either stays at the same node or moves from the current node to its adjacent node. The entire history of his movement, which we hereafter call an *episode*, is thus defined as $< h_0, h_1, ..., h_T >$.

Here, we formulate the problem of understanding a document about an agent as the task of finding an episode that does not contradict the textual descriptions about the agent's locations. In other words, the descriptions in the documents serve as the constraints in finding a possible episode. Note that, in general, there are many episodes that satisfy the constraints, because documents rarely provide the full detail of the movement history of an agent. Once we obtain those episodes, we can use them to resolve questions about the location of the agent at any particular time.

3.2 Alexander's Expeditions

In this work, we create a world model for interpreting documents on *Alexander the Great*, who was a famous king of ancient Macedonia. Figure 1 shows the graph that we have manually created from a map using frequent location names in Wikipedia. It shows the 35 locations names used in our experiments. Note that this graph is a very crude approximation to the real geographical cost and constraints in those days. Ideally, we should incorporate more detailed information such as distance, terrain, and environment into the model, but we leave it for future work.

Constraint conditions are generated from a document. For example, the sentence "Alexander the Great won a battle near Granicus river in May 334 BC." would produce the constraint that his location in May 334 BC is Granicus, which translates into something like h_2 = Granicus in our model. Although sophisticated information extraction techniques could be used to do this, we simply use the co-occurrence of the term "Alexander the Great", time and location expressions within a sentence to generate the constraints. Note

Algorithm 1 Finding feasible episodes

```
function FINDFEASIBLEEPISODES($maxR$,$maxIter$,$\alpha$)
  $feasibleEpisodes \leftarrow \{\}$
  for $round = 1$ to $maxR$ do
    $currentE \leftarrow$ GETRANDOMEPISODE()
    $bestE \leftarrow currentE$
    for $iter = 1$ to $maxIter$ do
      $nextE \leftarrow$ GETNEIGHBOREPISODE($currentE$)
      if $Val(currentE) < Val(nextE)$ then
        $currentE \leftarrow nextE$
        if $Val(nextE) > Val(bestE)$ then
          $bestE \leftarrow nextE$
        end if
      else
        $temperature \leftarrow \frac{1}{30}\alpha^{\frac{iter}{maxIter}}$
                                          ▷ $0 < \alpha < 1$ : $\alpha$ is constant
        if $rand(0,1) \leq e^{\frac{Val(nextE)-Val(currentE)}{temperature}}$ then
          $currentE \leftarrow nextE$
        end if
      end if
    end for
    $feasibleEpisodes.insert(bestE)$
  end for
return $feasibleEpisodes$
```

Algorithm 2 Computing a neighbor episode

```
function GETNEIGHBOREPISODE($currentEpisode$)
  $e \leftarrow currentEpisode$
  if $rand(0,1) < 0.5$ then
    $p1, p2 \leftarrow$ GETCONSECUTIVESAMESTATES($e$)
    $e.remove(p2)$
    $p3 \leftarrow$ GETRANDOMSTATE($e$)
    $e.insert(p3)$                          ▷ at next p3
  end if
  if $rand(0,1) < 0.5$ then
    $p1 \leftarrow$ GETRANDOMSTATE($e$)
    $p2 \leftarrow$ GETADJACENTSTATE($p1$)
    $e.insert(p2, p1)$                      ▷ at next p1
  end if
  if $rand(0,1) < 0.5$ then
    $p1, p2, p3 \leftarrow$ GETDETOUR($e$)            ▷ p1 = p3
    $e.remove(p2, p3)$
  end if
  if $rand(0,1) < 0.5$ then
    $loop \leftarrow$ GETRANDOMLOOP($G$)        ▷ G is the graph
    if loop contains some state in e then
      $e.reverseInLoop(loop)$
    end if
  end if
return $e$                              ▷ as a neighbor episode
```

that this simplistic method can generate erroneous constraints as well, but we will later show that reasonable inference can be performed even with these noisy constraints.

3.3 Calculation of Feasible Episodes

We use simulated annealing (Kirkpatrick et al., 1983) to find the episodes that satisfy the (soft) constraint conditions. Other approaches to optimization such as integer linear programming can be used for this purpose, but we chose simulated annealing due to its generality and easiness of implementation.

Algorithm 1 shows how we calculate feasible episodes. The score of an episode, $Val(e)$, is computed as the proportion of the constraint conditions satisfied by the episode. In this algorithm, we start with a random episode and attempt to find the episode that has the best score. More specifically, at each iteration, we generate a new episode by making a small modification to the current episode. Finally, we add the episode having the best score to the list of the feasible episodes. We repeat this whole process $maxR$ times to obtain multiple episodes.

Algorithm 2 describes the four operations to compute a neighbor episode in Algorithm 1. The first operation changes the time when the agent stays at the same place. For example, $< Ankara \rightarrow Ankara \rightarrow Tarsus \rightarrow Issus >$ is changed to $< Ankara \rightarrow Tarsus \rightarrow Tarsus \rightarrow Issus >$. The second operation adds a detour to the episode. For example, $< Ankara \rightarrow Tarsus >$ is changed to $< Ankara \rightarrow Gordion \rightarrow Ankara \rightarrow Tarsus >$. The third operation removes a detour from the episode. For example, $< Ankara \rightarrow Gordion \rightarrow Ankara \rightarrow Tarsus >$ is changed to $< Ankara \rightarrow Tarsus >$. The fourth operation alters the path from one location to another. For example, $< Caucasus \rightarrow Aornos \rightarrow Nicaea >$ is changed to $< Caucasus \rightarrow Arachosia \rightarrow Indus \rightarrow Nicaea >$. Each of these four operations is performed with 50% probability.

4 Experiments

4.1 Corpus and Settings

We used the English Wikipedia dataset[1] for the experiments. In this data set, there were 482 sentences which include the strings of "Alexander the Great" and "BC". Among them, 87 sentences included a location name in our list, and they were used to generate (noisy) constraint conditions. The constraints which had the same time and location conditions were treated as one constraint, so we did not take into account the frequency of appearance. As a result, 39 (noisy) constraints were generated. We manually checked those 39 constraints and found that 32 of them correctly describe Alexander's location at a particular time.

The simulation setting is as follows:

- The initial place is "Pella", i.e., h_0 = Pella.

[1]downloaded in November 2013.

Time	Place (Ans)
331 BC	Arbela
A century later, the "Men of the Mountain Land," from north of Kabul River, served in the army of Darius III of Persia when he fought against **Alexander the Great** at **Arbela** in **331 BC**.	
330 BC	Persepolis
After invading Persia, **Alexander the Great** sent the main force of his army to **Persepolis** in the year **330 BC** by the Royal Road.	

Table 1: Examples of questions

- One time step corresponds to two months.
- At each time step, the agent (Alexander) either stays at the same node or moves from the current node to one of its adjacent nodes.
- Each episode consists of 72 steps (i.e. $T = 71$), which correspond to Alexander's twelve-year expedition from 334 BC to 323 BC.

The values of α and $maxR$ in Algorithm 1 were set to 0.001 and 1,000 respectively.

4.2 Question Answering

First, we examine how accurately our system can answer questions like "Where was Alexander the Great in 325 BC?", when the answer is not explicitly written in the text. We have created 32 questions from the aforementioned 32 constraints that correctly describe Alexander's locations. Table 1 shows examples of questions with the Wikipedia sentences from which the questions were created.

When the system infers the answer to a question, we make sure that the system has no access to the sentences that convey the information about the correct answer. In other words, we exclude those sentences when generating the constraint conditions for the simulation.

For each question, the system calculates 1,000 episodes by simulated annealing and ranks the places according to how many times they have appeared during the time period specified in the question. The system then returns the top N places as the answer. We consider the answer to be correct if the correct place is included in the top N places.

As a baseline method for comparison, we also calculate the top N places according to their temporal distance to the time specified by the question. For example, we prioritize the mention pair (Tyre, 332 BC) over (Ankara, 333 BC) if the time specified by the question is 331 BC.

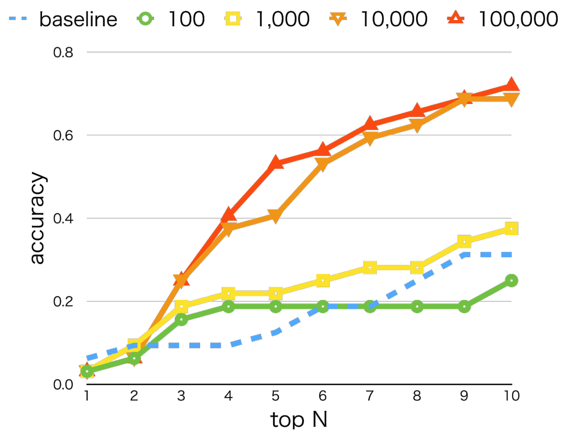

Figure 2: The accuracy of the top-N answers

Figure 2 shows the accuracy of the top-N answers for the 32 questions. The dotted line shows the result of the baseline method and the four solid lines show the results of our inference-based approach when the maximum numbers of iterations (maxIter) in Algorithm 1 are set to 100, 1,000, 10,000 and 100,000. As can be seen, the accuracy rate improves as the number of iterations in simulated annealing increases. The accuracy rates achieved by performing more than 10,000 iterations are significantly higher than those of the baseline. As for the computational cost, it took about half an hour to obtain 1,000 episodes (with maxIter = 100,000) for each question using eight cores of Xeon X5680.

4.3 Coreference Resolution

We show an example of coreference resolution using our world model. Table 2 shows a paragraph created from the Wikipedia text[2], where the phrase "the area" in the last sentence could refer to any of the four difference places mentioned in the sentences. Since there are few syntactic or lexical clues for disambiguation, it is a difficult coreference resolution problem[3].

When performing the inference for this problem, we did not use the constraints derived from the sentences that contain the candidate places,

[2]We have replaced "It" at the beginning of the original sentence with "Bela".

[3]We tested two publicly available coreference resolution systems (Stanford Core NLP and Illinois coreference System). Neither of them could not identify the correct antecedent.

time	325 BC
anaphor	the area
antecedent	Bela
other candidates	Arachosia, Carmania, Babylon

Bela is directly to the south of the ancient provinces of **Arachosia** and Drangiana, to the east of **Carmania** and due west of the Kingdoms of Ancient India. In **325 BC**, **Alexander the Great** crossed **the area** on his way back to **Babylon** after campaigning in the east.

Table 2: Example of coreference resolution

candidates	maxIter		
	1,000	10,000	100,000
Bela (Ans)	**247/1,000**	**547/1,000**	**745/1,000**
Carmania	210/1,000	454/1,000	640/1,000
Arachosia	154/1,000	404/1,000	651/1,000
Babylon	35/1,000	8/1,000	1/1,000

Table 3: Coreference resolution by inference

since we are interested in the situation where no explicit information is available in the document.

The inference results are shown in Table 4.3. The values in the table show how many times the places appeared in the 1,000 episodes at the times corresponding to 325 BC. The correct antecedent, Bela, has the highest values, and the infeasible antecedent, Babylon, has very low values, which demonstrate the usefulness of the inference in coreference resolution.

4.4 Error Analysis

We discuss the constraint conditions which could never be satisfied by any resulting episodes. Two examples are shown below.

- Constraint: 334 BC, Alexandria
- Sentence: The port of **Alexandria**, founded by **Alexander the Great** in **334 BC**, was a hub for Mediterranean trade for centuries.

- Constraint: 323 BC, Memphis
- Sentence: Arrhidaeus, one of **Alexander the Great**'s generals, was entrusted with the conduct of Alexander's funeral to **Egypt** in **323 BC**.

The first constraint is problematic because, in actual history, Alexander the Great was not in Egypt in 334 BC. This seemingly erroneous constraint was created by the ambiguity of the word "Alexandria", because it can refer to many other cities having the same name. The sentence of the second constraint does not describe Alexander the Great—it describes Arrhidaeus, who was one of his generals. However, our simplistic co-occurrence-based method wrongly created a constraint from it. These results suggest that our world model could help us to detect and suppress wrong interpretations of text since the constraints derived from wrong interpretations are unlikely to be satisfied in the simulation.

5 Conclusion

We have presented a toy world model that allows us to simulate the movement history of a historical figure and perform inference about his locations. Experimental results using Wikipedia text demonstrate its inference ability and potential usefulness in high-level NLP applications such as question-answering and coreference resolution.

In future work, we plan to develop a more robust environment on which we can quantitatively evaluate the level of document understanding by using a world model. We aim to build an evaluation method for comparing different approaches.

Our future work should also encompass extending the toy world model. Currently, the agent only moves on the graph, and thus the historical events that can be represented by the model is limited. Increasing the variety of actions that the agent can perform and the number of historical figures should be an interesting direction of future work.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions.

References

Jonathan Berant, Noga Alon, Ido Dagan, and Jacob Goldberger. 2015. Efficient global learning of entailment graphs. *Computational Linguistics*, 41(2):221–264.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08*, pages 789–797.

Angel X Chang, Manolis Savva, and Christopher D Manning. 2014. Learning spatial knowledge for text to 3D scene generation. In *Proceedings of Empirical Methods in Natural Language Processing, EMNLP*, pages 2028–2038.

Abraham Fowler, Bob Hauser, Daniel Hodges, Ian Niles, Adrian Novischi, and Jens Stephan. 2005. Applying cogex to recognize textual entailment. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 69–72.

Scott Kirkpatrick, MP Vecchi, et al. 1983. Optimization by simmulated annealing. *Science*, 220(4598):671–680.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206.

Changsong Liu, Lanbo She, Rui Fang, and Joyce Y Chai. 2014. Probabilistic labeling for efficient referential grounding based on collaborative discourse. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 13–18.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 809–819.

Altaf Rahman and Vincent Ng. 2011. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 814–824.

Roger Schank and Robert P Abelson. 1977. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures.* Lawrence Erlbaum.

Marta Tatu and Dan Moldovan. 2005. A semantic approach to recognizing textual entailment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 371–378.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698.*

Terry Winograd. 1971. *Procedures as a representation for data in a computer program for understanding natural languages*. Ph.D. thesis, Massachusetts Institute of Technology, Project Mac.