Using graphs to represent Japanese words for serious games

Tristan Tribes[1], Virgil Rouquette-Campredon[1], Samuel Helye[1], and Madalina Croitoru[2]

[1] Computer Science Department, Faculty of Science, University of Montpellier
[2] Boreal, LIRMM, INRIA, CNRS, University of Montpellier

Abstract. The aim of this paper is to explore the representation problem of how Japanese words are constructed from Kanjis (i.e. symbols for concepts). We discuss the various implications of such representations and demonstrate a fully functional game built upon a Neo4J implementation.

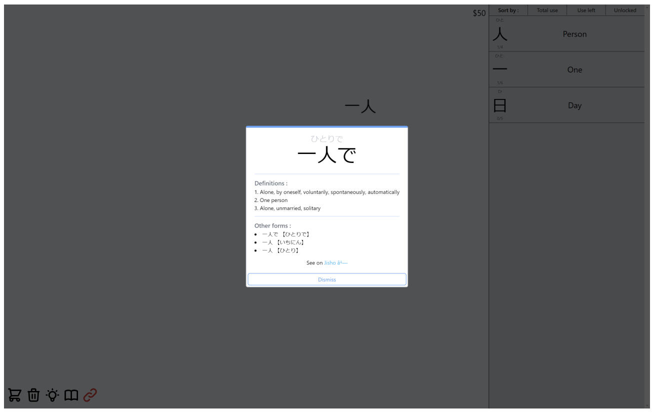
1 Introduction

The main contribution of this paper is to discuss the graph based knowledge representation and reasoning aspects behind the implementation of a serious online Japanese learning game: Lost My Pieces. Graph based knowledge representation and reasoning techniques have been historically investigated in the International Conference on Conceptual Structures (ICCS) community [10]. Starting with work on Conceptual Graphs [14] [2], Concept Graphs [5], Existential Graphs [8] or, more recently, Argumentation Graphs [6] the main idea behind such graph representations is simple. Using a graph based representation of the problem and by translating the reasoning task, in a sound and complete manner, into a graph operation, one can benefit from combinatorial optimisations from the graph theoretical world for practical reasoning task [3]. For example, by having positive existentially closed deduction of logical formulae translated into labelled graph homomorphism, one can unveil query structures that yield excellent results in practice [4].

In this paper we place ourselves in the context of graph based knowledge representation and reasoning and answer the following research questions: "How to represent kanji combination for facilitating the learning of the Japanese language?". In order to endorse the feasibility of the solution we also demonstrate a fully functional serious game [1][15][11][13][7] that allows users to learn Japanese online and discuss various paths for future developments.

2 Motivating example

Imagine a student that is keen to learn Japanese. The Japanese language, notoriously difficult, is composed of three writing systems. The most difficult one is called kanji, where each ideogram represent a concept or an idea, and can be by itself a whole word. There are about 2000 kanjis to learn in order to be able to simply carry on a day to day task such as reading a newspaper. To further the difficulty of the learning, combining kanjis can give birth to new words. For instance combining the kanji for woman (女) with the kanji for day (日) gives the word Sun goddess (日女). Similarly, the kanji for one (一) together with the kanji for person (人) gives the word alone (一人).



ひとりで
一人で

Definitions :
1. Alone, by oneself, voluntarily, spontaneously, automatically
2. One person
3. Alone, unmarried, solitary

Other forms :
- 一人で 【ひとりで】
- 一人 【いちにん】
- 一人 【ひとり】

See on Jisho â†—

Dismiss

Fig. 1: Screenshot of Lost My Pieces

Despite its beauty, it is complicated to keep motivation up when learning Japanese. To counter this problem we propose a novel way of learning by means of an interactive platform. The intuition of the interaction is inspired by the game Little Alchemy[3][9] that allows the combination of pictograms in order to make new images.

In our platform, called Lost My Pieces [4], the user is presented with a set of basic kanjis. The user can drag and drop the kanjis onto the central pane (called whiteboard) and combine them to discover new words. In Figure1 a screenshot of the combination of the two kanjis person and one is depicted. Please notice three important aspects:

– First, albeit impossible to render on a picture, the stroke order required for writing the kanji is also depicted when the kanji is selected, facilitating the learning of writing.
– Second, in an attempt to render the platform pleasant to use we established a point system allowing the user to unlock kanjis according to their budget. This is visible in Figure2 that shows all possible kanjis to unlock (and thus to further combine).

[3] `https://littlealchemy.com/`
[4] `https://lostmypieces.com/`

– Third, when searching for a new kanji to unlock, one can directly see how many new words they will be able to create with this newly acquired kanji using their already existing ones. However, such functionality will render the representation task more difficult as explained in Section 3.

In order to code the platform we have used a common client-server architecture. On the client side we have used the JavaScript library React. On the server side we have used Node.js®, a Javascript run-time environment and the Fastify framework. For storing data we have used Neo4j. In the following section we will detail the implementation solution. Then, in Section 3 we will explain the data storage problem and our contribution.

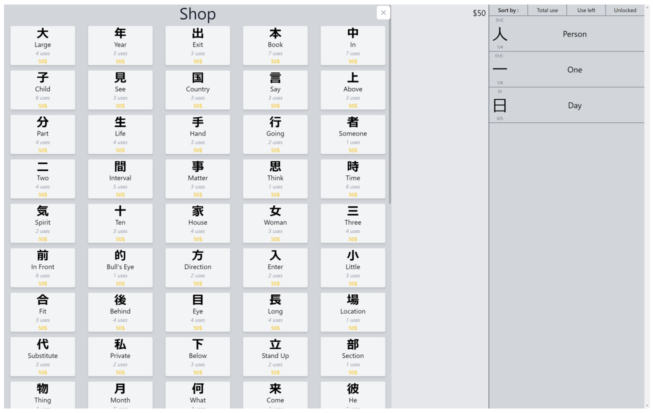


Fig. 2: Screenshot of Lost My Pieces Shop

The technologies used for developing the platform as as follows:

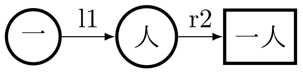
– React (front end) : Allows for faster UI development, with reusable components and seamless page update and interactions.
– Fastify (back end, REST API) : Fast API framework to serve database requests to the end-user.
– NodeJS (back end) : Javascript run-time to handle all server related tasks.
– NGINX (server) : Handles all traffic towards the server.

We have purposely not described the Neo4j component as knowledge representation and reasoning issues are detailed in the next section.

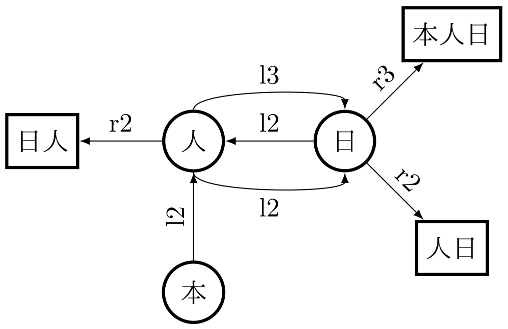
3 Representation problems

The representation problem we address in this paper is the following. We consider a list of kanjis $\mathcal{K}$. For each kanji $k \in \mathcal{K}$ I need to store the meaning of the kanji and, for elements of $\mathcal{K} \setminus \{k\}$ their combined meaning. For example, we can consider A and B two kanjis. The word AB respects the order of its components (i.e. A is first, B is second) and its meaning is different from BA. To represent this using a database we explored several possibilities.

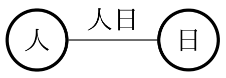
First we considered using a graph where the nodes could either be kanjis or words and the edges are labelled using the length of the corresponding word. We exploit paths as follows:



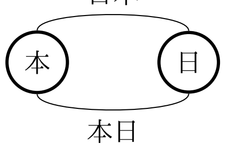
While using this method we could exploit paths (i.e. represent words) of arbitrary length. Unfortunately, the main problem is that manipulating this structure (for our purpose) was too computationally expensive. For example, for a word of length 3 we should test 9 combinations of all possible orders (i.e. paths). For illustration, in the image below we show the representation for a word of size three.



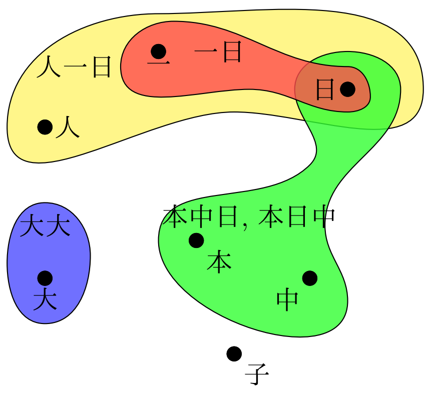
A second solution consisted of labelling the edges in the graph as words (with nodes only representing kanjis). This solution is depicted below.



The advantage of this representation is that we can represent all orders by simply checking if an edge label exists.

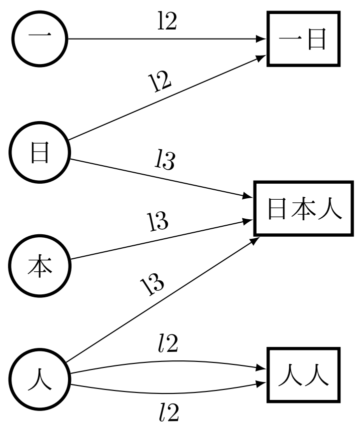


However, the words of length 3 or more require a hypergraph support. Furthermore, it requires to have self cycles (for words that contain repeating kanjis). The hypergraph depiction is shown below.



The main problem with the hypergraph representation was the fact that a word could have several definitions and different variations. For example the word obtained by combining the kanji large and book has two definitions: root and foundation. Furthermore it has two different reading possibilities depending on the context. If this word is represented as an hyperedge we will need further combinatorial elements to add the information above.

Our solution it to represent the hypergraph as a bipartite graph as follows.



In this final solution (the one that we decided to implement using Neo4j) the bipartite graph[16][12] contains two classes of nodes. On one hand the kanjis (represented as round nodes, on the left) and, on the other hand, the words, represented as rectangles, on the right part of the graph. The edges are labelled with the length of the word they lead to. This allows for querying, by kanji, the words of a given length. The different forms or definitions of a word could be represented as attributes on the words, or additional nodes connected to their associated words.

This combinatorial structure is implemented in Neo4j. In Figure 3, in orange we depict three kanjis for day, one and person. In blue we depict the words constructed with these kanjis. For instance the top most blue node represents men day with three different readings depicted in red.

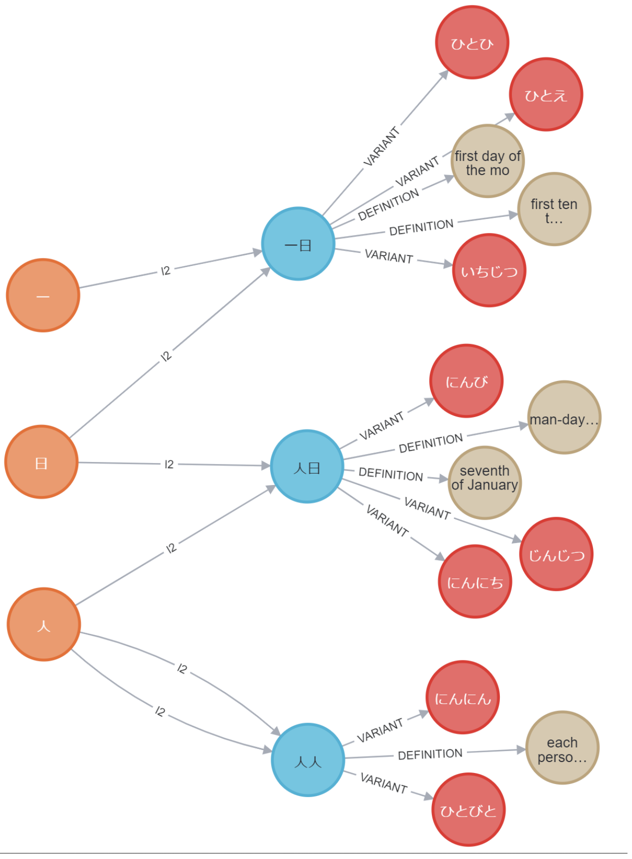


Fig. 3: Screenshot of Neo4j implementation

Overall, in our platform LostMyPieces we have encoded the 101 most frequent kanjis resulting in a JSON file of size 1 545 KB. With these kanjis we can make 2711 words. The size of the resulting graph is of 10436 nodes and 6563 edges (as follows l2 : 3618 / l3 : 2064 / l4 : 756 / l5 : 125).

To read a newspaper we should consider 2136 kanjis (that are taught at school and allow to form 100694 words). We encoded this in Neo4J resulting in a JSON file of size : 51 813 KB. The resulting graph consists of 319 591 nodes and 295 095 edges, with words made of up to 18 kanjis[5]. Both graphs are available upon request.

4 Evaluation and discussion

In this paper we have demonstrated how using a graph based knowledge representation solution can be practically used in the context of a serious game allowing users to learn Japanese. For far our evaluation has been purely functionality oriented. We have held several game sessions with various level users that have then returned an evaluation sheet deeming themselves satisfied by our game. Ideally we would like to test our game with two classes of users, having similar Japanese language skills. they will carry on learning the language over a period of time and then we could assess the effectiveness of the plateform. Such effort is, for now, impossible to put in practice. However we are quite optimistic with respect to the effectiveness of our game because it allows the user to understand how the construction of the word takes place (as opposed to just simply memorising it). We thus hypothesise that such approach will yield better learning results.

Several future work directions are currently considered. First, several questions regarding the development of the game could be considered. Can we add more information on words? How they are linked, do they have similar structure? Can we use N5 to N1 Japanese levels to categorise our words? Can we create new imaginary words from this whole database? Instead of using all Japanese words, could we instead use a list of important words from Japanese learning methods? Can we use the shop to see what people would usually buy first ? Can we randomise the shop to alleviate the order bias (buying the first one, or buying the one with most uses)?

From a representation point of view, the Neo4j database is not local and having delays on every requests is painful. Can we develop our own local solution for graph database that works inside browser? How suitable is an RDF representation of the structure and could RDF/S rules bring new information relevant to the game?

5 Acknowledgements

The authors acknowledge the participants of Imagine Master class at University of Montpellier for the provided user data. We also thank M. Lafourcade for his feedback and giving us the opportunity to work together.

[5] 公共土木施設災害復旧事業費国庫負担法: National Government Defrayment Act for Reconstruction of Disaster-Stricken Public Facilities

References

1. C. C. Abt. *Serious games.* University press of America, 1987.
2. M. Chein and M.-L. Mugnier. *Graph-based knowledge representation: computational foundations of conceptual graphs.* Springer Science & Business Media, 2008.
3. M. Chein, M.-L. Mugnier, and M. Croitoru. Visual reasoning with graph-based mechanisms: the good, the better and the best. *The knowledge engineering review*, 28(3):249–271, 2013.
4. M. Croitoru and E. Compatangelo. A tree decomposition algorithm for conceptual graph projection. In *KR*, pages 271–276, 2006.
5. F. Dau. Concept graphs and predicate logic. In *International Conference on Conceptual Structures*, pages 72–86. Springer, 2001.
6. P. M. Dung. An argumentation-theoretic foundation for logic programming. *The Journal of logic programming*, 22(2):151–177, 1995.
7. W. L. Johnson, H. H. Vilhjálmsson, and S. Marsella. Serious games for language learning: How much game, how much ai? In *AIED*, volume 125, pages 306–313, 2005.
8. K. L. Ketner. Pierce’ s existential graphs as the basis for an introduction to logic: Semiosis in the logic classroom. In *Semiotics 1980*, pages 231–239. Springer, 1982.
9. O. T. Leino. Stone+ life= egg–little alchemy as a limit-idea for thinking about knowledge and discovery in computer games. In *Proceedings of Philosophy of Computer Games Conference*, 2016.
10. G. W. Mineau and B. Moulin. *Conceptual Graphs for Knowledge Representation: First International Conference on Conceptual Structures, ICCS'93, Quebec City, Canada, August 4-7, 1993. Proceedings*, volume 699. Springer Science & Business Media, 1993.
11. U. Ritterfeld, M. Cody, and P. Vorderer. *Serious games: Mechanisms and effects.* Routledge, 2009.
12. F. Serratosa. Fast computation of bipartite graph matching. *Pattern Recognition Letters*, 45:244–250, 2014.
13. B. H. Sørensen and B. Meyer. Serious games in language learning and teaching-a theoretical perspective. In *DiGRA Conference*, pages 559–566, 2007.
14. J. F. Sowa. Conceptual graphs for a data base interface. *IBM Journal of Research and Development*, 20(4):336–357, 1976.
15. T. Susi, M. Johannesson, and P. Backlund. Serious games: An overview. 2007.
16. H. Zha, X. He, C. Ding, H. Simon, and M. Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 25–32, 2001.