Regular Paper

Automatic Japanese Example Extraction for Flashcard-based Foreign Language Learning

Arseny Tolmachev[1,a)] Sadao Kurohashi[2,b)] Daisuke Kawahara[3,c)]

Received: May 27, 2021, Accepted: January 11, 2022

Abstract: Flashcard systems are effective tools for learning words but have their limitations in teaching word usage. To overcome this problem, we suggest that a flashcard system shows a new example sentence on each repetition. This extension requires high-quality example sentences, automatically extracted from a huge corpus. To do this, we use a Determinantal Point Process which scales well to large data and allows us to naturally represent sentence similarity and quality as features. Our human evaluation experiment on the Japanese language indicates that the proposed method successfully extracted high-quality example sentences.

Keywords: Natural Language Processing, Foreign Language Learning, Example Sentences, Example Extraction

1. Introduction

Learning vocabulary is a crucial step in learning foreign languages and it requires substantial time and effort. Word learning is often done using flashcards: a way of organizing information into question-answer pairs. An example of a flashcard for the Japanese word “柿” is shown on **Fig. 1**. Flashcard systems frequently use Spaced Repetition technique to optimize the learning process. The technique is based on the observation that people tend to remember things more effectively if they study in short periods spread over time (*spaced repetition practice*) opposed to *massed practice* (i.e., cramming) [2], [14]. Anki [*1] is one of the most well-known open-source Spaced Repetition System (SRS).

One major drawback of building a vocabulary with flashcards is that most of the time cards look like the one displayed on Fig. 1 (top): flashcards often lack usage context information. A question card is usually *a word alone*, an answer card could contain a *fixed single* example sentence present. The example does not change from repetition to repetition, and as a result, does not show the full spectrum of word usage. However, humans do not use isolated words for communicating. Words are always surrounded by other words, forming word usages. Learning these word usages is as important as learning words themselves.

Instead of showing only a single field like reading or writing of a flashcard in the question card similarly to the Fig. 1 (top), we propose to use example sentences in both types of cards, as shown in Fig. 1 (bottom). Moreover, we want to show a *new* example sentence on the question card for each repetition. This

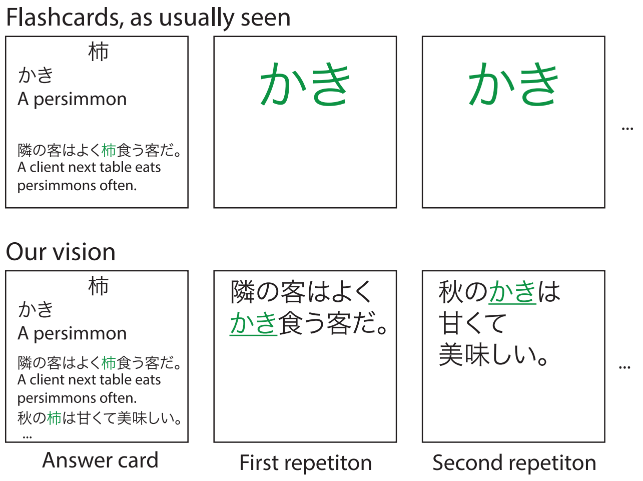

Fig. 1 Flashcards for the word “柿”.

approach allows users to learn correct word usages together with the words themselves. However, implementing it requires a huge number of example sentences.

1.1 Requirements and Overview of Automated Example Extraction Approach

We focus on the automatic extraction of high-quality example sentences that can be used in the question side of flashcards. Collecting an enormous number of high-quality example sentences manually does not scale well. Words can have multiple senses and different usage patterns. A database containing dozens of sentences for each sense of each word would need to contain millions of different sentences. For a set of example sentences, we say that they are of high quality if the sentences have the following properties.

- (Intrinsic) **Value**: Each example sentence should not be bad, for example ungrammatical, a fragment, or unrelated to the

[1] Works Applications Enterprise, Tokushima NLP Laboratory, Tokushima 770–0905, Japan
[2] Kyoto University, Graduate School of Informatics, Kyoto 606–8501, Japan
[3] Waseda University, Shinjuku, Tokyo 169–8555, Japan
[a)] arseny@kotonoha.ws
[b)] kuro@i.kyoto-u.ac.jp
[c)] dkw@waseda.jp

[*1] http://ankisrs.net

target word. Additionally, the sentences should not be too difficult for learners to understand.
- **Diversity**: Inside a set, the sentences should cover different usage patterns and word senses.

In addition, we would like our method to support rare words and rare word senses.

For the task of example extraction, we are given a huge monolingual text corpora and a target word or a phrase to output a set of high-quality example sentences.

1.2 Contributions

We propose a system architecture consisting of two components:
- Candidate filtering, utilizing a specialized search engine, which produces a relatively high number of example sentence candidates from a huge raw corpus,
- Example selection, which takes the list of example sentence candidates and selects high-quality ones from that list, balancing the individual sentence quality and the overall set diversity.

The candidate filtering step is designed such that the selected sentences are syntactically rich near the target word. It is enabled by a search engine that allows querying not only by keywords but by part-of-speech (POS) tags of words and dependency relations. Depending on a target word POS, we use queries that contain typical structures in the parse tree for the target POS with the intention to capture meaningful context for a target word, which is helpful for language learners.

The example selection step utilizes Determinantal Point Process (DPP) mathematical framework, used to model diverse datasets. The DPP allows us to naturally represent data in terms of scalar quality and vector similarity. Additionally, the DPP has several interesting properties. For example, it is possible to compute a marginal probability of drawing a subset of items from a DPP efficiently. Marginal here means a probability of inclusion of a given set in any subset drawn from the DPP. Furthermore, it is proven that this marginal probability measure is submodular. Because of this, it is possible to build a greedy algorithm that selects example sentences one by one, using the marginal probability measure as a weight with reasonable guarantees on the quality of the result. Finally, the DPP is computationally and memory efficient. The computation of marginal probabilities can be performed linearly with respect to the number of sentence candidates. This makes it possible to use the DPP with tens of thousands of candidates in near-realtime scenarios.

We have performed a human evaluation experiment which has shown that our method was preferred by Japanese learners and a teacher compared to two baselines.

2. Example Candidate Filtring with Dependency-aware Search Engine

General search systems like Google or Microsoft Bing are designed for searching documents relevant to a specific query. Such documents are usually long pieces of text, for example, web pages. Language learners and teachers use such systems for acquiring example usages or contexts for words they learn.

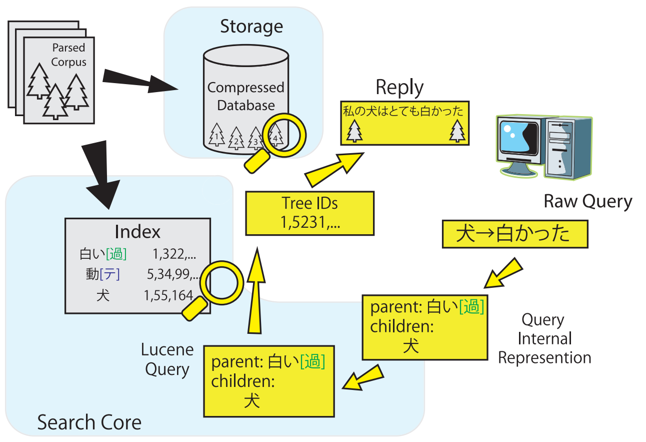

Fig. 2 Overview of a search engine. Input parse trees are preprocessed into inverted index and compressed storage. The search query is converted into internal representation, then is used for the index lookup followed by a fetch of the parse trees from the storage.

However, general search systems are not well-suited for this task. Firstly, users doing such search are not looking for documents, they are looking for sentences. Additionally, conventional search engines usually ignore precise grammatical information when indexing text, although this data is extremely useful for example sentences.

Searching sentences for educational and linguistic usage often requires more features than just querying on terms as general systems do. Usually, we want to find usages of a word in some context. Thus, sentence-level search should support queries not only on the lexical level but on lexical dependencies, part of speech (POS) tags, conjugation forms, and grammatic words as well. In the case of Japanese, those are case markers (が, を) or auxiliary verbs (いる after テ form).

As an example, understanding onomatopoeia is difficult for many Japanese language learners. Example sentences like 肌がピリピリする are mostly useless for learners because they have little specificity — it is easy to replace ピリピリ with another word. Such sentences give a small amount of information on the target word ピリピリ itself. Sentences like 肌がピリピリ痛く感じる are substantially better in this case. We use dependency relations and part of speech tags to select better sentences with rich and meaningful syntactic structure around the target word.

Our goal for the example extraction system is to provide examples even for rare words while keeping rich diversity of the example sentences. We use a huge raw corpus to select example sentence candidates. The final quality of the examples in such an approach will depend on the quality of example sentence candidates. Selecting a small number of documents matching a query is a well-studied process and inverted index-based full-text search engines are used for this task. To select syntactically rich sentences on a scale of a huge corpus, we have developed a distributed Apache Lucene-based search engine [21] which allows querying not only on keywords as most systems do but on dependency relations and grammatical information as well.

The proposed system architecture, shown in **Fig. 2**, consists of two main parts: compressed database and search core. A compressed database is a key-value storage that stores parse trees compressing several trees together. Search core is an Apache

Lucene-based solution for handling queries with dependency relations.

2.1 Compressed Database

Example extraction operates on KNP parse trees as the input data. KNP parse trees contain results of morphological analysis, dependency relations and rich feature information. Most key-value storage databases compress entries individually, however, KNP parse trees share a lot of information, and failing to compress several entries together can significantly increase storage space for huge corpora which are in order of hundred of gigabytes compressed. We use that rich feature information in the further steps of example extraction, so it can not be dropped when storing the trees in the database.

We have developed simple key-value storage which stores multiple parse trees compressed in a single block. The compressed database consists of an index stored as a B-tree using MapDB [*2] and data files. The database index is a mapping from a tree id to a tree compressed pointer and tree size pair. Data files consist of 64-kilobyte blocks. Each block is archived independently of others and saved to disk. There are no inter-block compression dependencies, meaning that blocks can be read in arbitrary order. To extract a tree from a data file, the system needs to read a block from a disk, decompress it, and get a tree from that block. The information about the block and the position inside the block can be stored in a single compressed pointer.

Compressed pointer is a trick taken from the bioinformatics BAM/BGZF [*3] storage format used for storing DNA sequences in compressed files. The compressed pointer consists of two parts: the beginning of a compressed block in a data file and an offset of needed data inside the decompressed block. Block sizes are fixed to 64k and the 64-bit pointer can be formed by making the lower 16 bits to store the uncompressed offset and the remaining 48 bits to store the block start address in the compressed file.

2.2 Search Core

Search core is based on Apache Lucene [*4] with additional components for dependency and POS tag support for indexing and querying. Furthermore, to support fast queries on a huge corpora, the system is implemented in a distributed master-slave-like manner. Distribution was done using an actor programming model and Akka library [*5].

Our goal is to have different possible usages of a target word in the example sentences. For example, verbs should have multiple arguments with different roles and in general, it is better to have the vicinity of a target word syntactically rich. We use dependency information for approximating this information. For accessing syntactic information, we automatically tokenize raw text, extract lemmas, perform POS tagging and parse sentences into dependency trees.

Search engines usually build a reverse index based on tokens, which are computed from the original document. We encode seed tokens for our engine as a concatenation of lemma form and conjugation form tags, which are derived from the original text. For example, the verb 帰った (kaetta – "to leave" in past form) would be represented as "帰る+PAST". Each token also stores the position of its parent.

The next step generates rewritten tokens from the seed tokens until no more new tokens can be created using the rule-based rewriting process. Rewriting is done by replacing content word lexical information with part of speech information or removing some parts of tokens. For example, case markers of nouns are removed for some rules.

This representation allows to easily match the same forms of different words while getting the benefits of the reverse index in terms of performance. A list of created tokens for a raw sentence is shown in **Fig. 3**. This example spawns three tokens for each of its words.

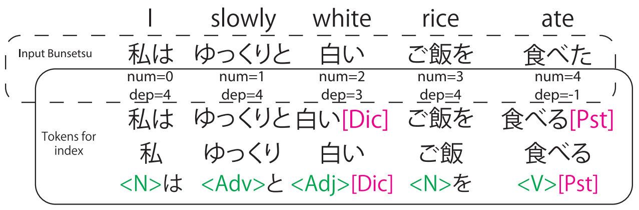
I slowly white rice ate
Input Bunsetsu
私は ゆっくりと 白い ご飯を 食べた
num=0 dep=4
num=1 dep=4
num=2 dep=3
num=3 dep=4
num=4 dep=-1
Tokens for index
私は ゆっくりと 白い[Dic] ご飯を 食べる[Pst]
私 ゆっくり 白い ご飯 食べる
<N>は <Adv>と <Adj>[Dic] <N>を <V>[Pst]

Fig. 3 Word to token conversion for indexing a sentence. Tokens contain lexical information (black), ⟨POS⟩ tags (green) and [conjugation forms] (magenta). Dependency information is common for a set of tokens spawned from a single word. This information consists of word position and dependency position.

2.3 Selecting Example Sentence Candidates

For selecting candidates we use queries that match a target word with up to 3 children or parents. The exact types of parents of children depend on the POS of the target word. The number 3 was chosen to have balance with different arguments and to keep the syntactic vicinity of the target word diverse between the example sentence candidates.

One specific feature of the search system is the ability to limit the number of matching subqueries in a compound query. Usually, a disjunction (OR) query makes its score as a sum of scores of all its subqueries. If the number of processed sentences is large, there will be a large number of sentences that fulfill most of the subqueries for frequent words. By matching all the subqueries an OR query effectively becomes an AND query, however that is not very useful for example sentences. It decreases the total diversity of search results making every result to be more or less the same. To deal with it we implement a special type of compound query that uses only top N scores of its subqueries.

For a word and its part of speech, the system generates a search query that represents frequent patterns of word usage. For this section the following notation is used:

- *word* denotes the *target word*
- ⟨noun⟩, ⟨verb⟩, ⟨adj⟩, ⟨adv⟩ denote any word that have the indicated part of speech
- →denotes a dependency between two terms with right being a parent and left being a child.
- Items in a list form a disjunction, or an "OR" query.
- EOS token means end of sentence. Dependency to it like

[*2] http://www.mapdb.org/
[*3] http://samtools.github.io/hts-specs/SAMv1.pdf
[*4] https://lucene.apache.org/core/
[*5] http://akka.io/

word →EOS means that the word should be the last word in a sentence.

Patterns for individual parts of speech are described below. Each pattern contains an additional search term that decreases the score of a sentence if it contains multiple inclusions of a target word.

Verbs

- ⟨noun⟩-*ga* →word
- ⟨noun⟩-*wo* →word
- ⟨noun⟩-*ni* →word
- ⟨adv⟩ →word
- word →⟨noun⟩
- word →EOS

Patterns for verbs capture that the verb should have some arguments, be the last one or in the middle of the sentence. The limit on matching subqueries is 3. There is a slight bias on the subquery that has ⟨noun⟩-*wo* argument, because object arguments frequently mutate the senses of verbs and are especially important for understanding the example sentences.

Nouns

- word-*ga*
- word-*wo*
- word-*ni*
- word-*de*
- word-*ha* *
- one of:
  - ⟨noun⟩-*no* →word *
  - ⟨noun⟩ →word
  - word-*no* →⟨noun⟩ *
  - word →⟨noun⟩
- word →(⟨verb⟩ OR ⟨adj⟩)
- (⟨verb⟩ OR ⟨adj⟩) →word

The limit on subqueries is 3. The *ha* case marker gives less information about the relation of the word than other cases, so it is given a slight penalty. Dependencies on the nouns with *no* case marker are given a slightly higher weight because *no* could be important for the meaning of nouns.

Adjectives

- ⟨noun⟩-ga →word
- ⟨noun⟩-ha →word
- word →⟨verb⟩
- word →⟨noun⟩
- word →EOS

There is no limit on subqueries. Dependency of a target adjective is given a slightly higher weight if it is a child of a noun, because modifying nouns is a main role of an adjective. Dependency on EOS is given a slightly lower weight.

Adverbs

- (⟨noun⟩-ga AND word) →⟨verb⟩
- (⟨noun⟩-ga AND word) →⟨adj⟩
- (⟨noun⟩-ga AND word) →⟨noun⟩

Adverbs usually modify verbs and interesting usages of them are linked both with the subject of the sentence and the predicate. Patterns try to capture this relationship.

3. Example sentence selection

After we have a relatively large list of example sentence candidates, we select a few of them as example sentences. The outline of the selection part is shown in **Fig. 4**. In this section we describe the ideas behind the DPP and the way we compute individual features.

3.1 Determinantal Point Process

In this section, we provide a very basic explanation of the DPP inner workings. We invite interested readers to refer the original paper [11] which gives a comprehensive overview of the DPP.

Suppose we have a ground set $\mathcal{Y} = \{1...N\}$ of N items (in our case items are example sentence candidates from the search engine). In this stage we want to select a subset $Y \subseteq \mathcal{Y}$ s.t. $|Y| = k$. In its basic form, the DPP defines the probability of drawing a subset Y from a ground set as

$$\mathcal{P}_L(Y) \propto \det(L_Y) \tag{1}$$

Here L_Y denotes the restriction of matrix L to the elements of Y, $L_Y = [L_{i,j}] : i, j \in Y$. L generally can be any semi-positive definite matrix, but for our task we compose it from two types of features: a **quality** scalar q_i and a **similarity** unit vector ϕ_i. Elements of L become a cosine similarity between the similarity features scaled by the quality features

$$L_{i,j} = q_i \phi_i^T \phi_j q_j. \tag{2}$$

The intuition behind the DPP is as follows: because the right part of Eq. (1) contains determinant, when off-diagonal elements of

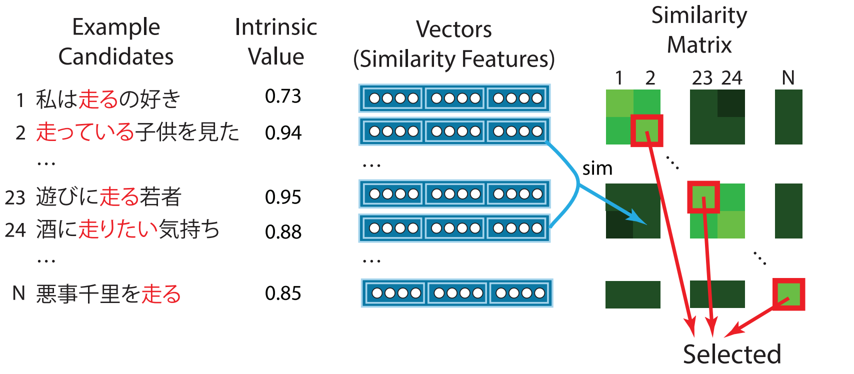

Fig. 4 Example sentence selection. The objective is to select "best" and non-similar example sentences from the input list. The target word is marked red.

L_Y get larger (meaning the cosine similarity of similarity features is large), then the determinant value, or in other words, the probability of drawing Y, gets lower. At the same time, the DPP prefers elements with large values of quality features.

The DPP has a very interesting property. It is easy to compute **marginal** probabilities of inclusion of a set A in all subsets of the ground set $\mathcal{Y}$:

$$\mathcal{P}_L(A \subseteq \mathcal{Y}) = \frac{\sum_{Y:A\subseteq Y\subseteq \mathcal{Y}} \det(L_Y)}{\sum_{Y:Y\subseteq \mathcal{Y}} \det(L_Y)} = \det(K_A).$$

K_A is a restriction of K with the elements of the set A (similar to Eq. (1)). K itself is called *marginal kernel* of the DPP and it can be computed as $K = L(L + I)^{-1}$, where I is an identity matrix.

Selecting diverse items

Because the elements of K can be used to compute the marginal probability of selecting a subset of items from the ground set, it is possible to use the marginal probabilities as a weight for a greedy selection algorithm.

In the beginning we have an empty set $A = \emptyset$. Then we repeatedly add an item i into the set A s.t. $i = \arg\max_i \det(K_{A\cup i})$ until the set A reaches the required size. Please note that this algorithm does not find a MAP answer, that problem is shown to be NP-complete.

Computational complexity

Dealing with L and K directly requires $O(N^3)$ floating-point operations and $O(N^2)$ memory, which can be unwieldy for sufficiently large N.

Fortunately, if L is formulated as Eq. (2), it is possible to work around these requirements. Let B be a feature matrix with rows $B_i = q_i\phi_i$, so $L = B^T B$. Instead of computing $N \times N$ matrix L, we compute a $D \times D$ matrix $C = BB^T$. Note that if we have an eigendecomposition $L = \sum_{n=1}^{N} \lambda_n v_n v_n^T$, we can get the marginal kernel K by rescaling eigenvalues of L:

$$K = \sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + 1} v_n v_n^T.$$

Remember that non-zero eigenvalues of L and C are the same and their eigenvectors are related as well. Namely, the eigendecomposition of L is also

$$\left\{\lambda_n, \frac{1}{\sqrt{\lambda_n}} B^T \hat{v}_n\right\}_{n=1}^{D},$$

where $\hat{v}_n$ are eigenvectors of C. Using this fact, we can compute the elements of marginal kernel K directly from the eigendecomposition of C and the feature matrix B:

$$K_{ij} = \sum_{n=1}^{D} \frac{(B_i^T \hat{v}_n)(B_j^T \hat{v}_n)}{\lambda_n + 1}.$$

Computation of a single element of K takes $O(D^2)$ floating-point operations. For each step of the selection algorithm, we need to compute N new elements of K and compute N determinants of $|A| \times |A|$ size. In addition, we need to compute an eigendecomposition of D. This leads to a total complexity of $O(D^3 + ND^2k + Nk^3)$ for selecting k items using the DPP, which is linear of N.

3.2 Similarity Features

We construct a similarity feature vector as a weighted stacking of three individual feature parts

$$\phi_i = f([w_1 s_i^{\text{lex}} ; w_2 s_i^{\text{synt}} ; w_3 s_i^{\text{sema}} ; r])$$

and a parameter r which makes all sentences similar to each other, following the text summarization task in Ref. [11]. We set $r = 0.7$ in our experiments.

Three similarity feature parts are lexical, syntactic and semantic similarity. Feature weights w_i allow us to prioritize similarity feature components. Lexical and syntactic similarity features are created as count-based vectors and have a large dimensionality. The transformation f here is a compression into a 600-dimensional vector using Gaussian random projections as recommended by Ref. [11] to make the dimensionality of ϕ_i, D, small.

3.2.1 Similarity: Lexical

Lexical similarity features measure word overlap between two sentences, syntactic features measure the structural (POS, grammar, and dependency) similarity between two sentences and semantic features measure the word sense similarity of two sentences. Lexical similarity uses *tf* weighting inside example sentence candidate batch when the inclusion of a content word is given a weight of 1.0; non-content words are given a weight of 0.1.

3.2.2 Similarity: Syntactic

A syntactic similarity for two sentences should be higher if they have similar syntactic structure near the target word, meaning that it was used in a similar syntactic way. In other words, the dependency structure, POS tags, and grammatical words should be similar near the target word. For instance, let's consider the sentences: "He is a fast runner", "She is a slow runner" and "John isn't a good runner". These three sentences have small content word overlap, but have the same syntactic structure.

The idea for the syntactic similarity method is based on an efficient calculation of graph similarity using graphlets. Graphlets are parts of a graph, and it is shown by Ref. [15] that they can be used for the fast approximate computation of graph similarity.

The main idea is to generate subtrees up to a certain size, by growing them from the target word and use those subtrees as features in the vector space. Overall, the syntactic similarity model can be thought of as a bag-of-subtrees model. Dependency trees in Japanese are built of *bunsetsu* – a unit that consists of a lemma with attached functional morphemes. Subtrees are treated as unordered because bunsetsu in Japanese can be moved on the same dependency level.

In the first step, the parse tree is *stripped* from lexical information for open parts of speech by replacing them with part of speech tags. Function words are left as they were.

Secondly, a set of bunsetsu subtrees up to the size of 3 is generated from the stripped tree. The generation starts from the bunsetsu containing the target word and continues until no new subtrees can be created.

Finally, the feature space is expanded by deriving new subtrees. Bunsetsu can contain compound nouns like "参政権" (a right to vote) or "積み上げる" (to place on top of something) which are analyzed to consist of two lexical units. Grammatically, they are not much different from single unit words. This step ensures that sentences containing both several-unit and single-unit words are still going to be structurally similar.

3.2.3 Similarity: Semantic

The semantic similarity score should be higher if the target word is used in the same or a close sense. Other words in the sentence should not directly change the measure.

It is possible to have embeddings (vector representations) for words. These embeddings can be trained from raw text and have interesting properties. For example, the word2vec method [12] gives embeddings that have linguistic regularities. For example, if $d(\text{king})$ is an embedding for the word "king", then it is possible to compute relations between the words using the computed vector representations, for example

$$d(\text{queen}) - d(\text{woman}) + d(\text{man}) \approx d(\text{king}).$$

Prototype Projections

Unfortunately, such methods usually have only a single vector representation for a word, meaning it is impossible to distinguish different senses of a word. **Prototype projections** [22] propses a solution to this problem.

Consider a binary relation $R(a_1, a_2)$ between two words, for example, a relation between a predicate and object with the loss of generality. If we fix a verb in the a_1 slot, then frequently occurring words that can take place in the a_2 slot are going to form a set of **prototype words**. Those prototype words somehow define different senses of the target word. For example, a verb would be "掛ける", then the set of prototype words would contain 声, 電話, 迷惑, 服, 鍵. In Japanese を marks an object for a predicate, and those words usually fill the を case slot in sentences with 掛ける. If we have a parsed corpus, then the set of prototype words $\{w_1, w_2, ..., w_m\}$ can easily be computed.

By using embeddings for the prototype words $d(w_i)$ it is possible to construct a **prototype matrix** C. A matrix is built by simply stacking embedding vectors for the prototype words on top of each other

$$C = [d(w_1), d(w_2), \cdots, d(w_m)]^T.$$

A singular value decomposition (SVD) of matrix $C = U\Sigma V^T$ is going to contain "the most important" directions of the prototype words in the right singular vectors V_i^T corresponding to the largest singular values Σ_i. By dropping the singular vectors corresponding to the smallest singular values, it is possible to get a **prototype space**. It is represented by a matrix $\Sigma_{0:k} V_{0:k}^T$ for k highest singular values. It is possible to create a projection matrix to this subspace in the original space by creating a matrix

$$P_{R,a_1} = (\Sigma_{0:k} V_{0:k}^T)^T (\Sigma_{0:k} V_{0:k}^T).$$

An embedding $d(w_0)$ of a word w_0 projected into a prototype subspace spanned by the word the slot a_1 over the relation R is going to be denoted as

$$w_0|_{a_1}^{\text{R}} = P_{R,a_1} d(w_0).$$

By applying a prototype projection to both ends of the relation, and combining them together by summing them, it is possible to

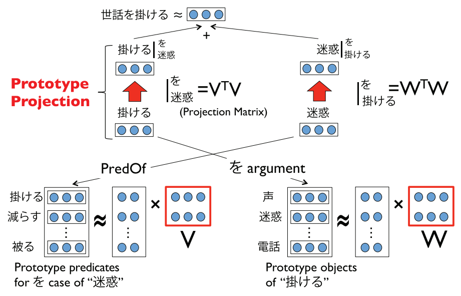

Fig. 5 Prototype Projection for a phrase "迷惑を掛ける".

Table 1 Similarity for Prototype Projection $\bigcirc|_{\text{掛ける}}^{\text{を}} + \text{掛ける}|_{\bigcirc}^{\text{を}}$, $0 : k$, $d_{l\%} = 0.2$.

	世話	迷惑	電話	コート	眼鏡	鍵	声
世話	1.000	0.877	0.734	0.595	0.712	0.686	0.753
迷惑	0.877	1.000	0.716	0.559	0.716	0.667	0.783
電話	0.734	0.716	1.000	0.728	0.825	0.820	0.796
コート	0.595	0.559	0.728	1.000	0.872	0.795	0.694
眼鏡	0.712	0.716	0.825	0.872	1.000	0.883	0.809
鍵	0.686	0.667	0.820	0.795	0.883	1.000	0.775
声	0.753	0.783	0.796	0.694	0.809	0.775	1.000

get a vector representation for the whole relation instance

$$d(R(a_1, a_2)) = a_1|_{a_2}^{\text{R}} + a_2|_{a_1}^{\text{R}}.$$

Instances of this relation with similar meaning will have a high cosine similarity score. Usages in the different meanings will have a low cosine similarity score. The **Fig. 5** displays a construction of a vector representation for a phrase "迷惑を掛ける". It consists of a relation over を case and two prototype projections. The first uses the predicates of "迷惑" over the を case as prototype words and another uses the arguments of "掛ける" over the same case.

Authors have shown that this method is very good for detecting paraphrases like "run a company" versus "operate a company". However, if it can detect semantic information even in distinct words, it should be possible to use it for coarse unsupervised word sense disambiguation as well.

Word embeddings for representations were trained from the web corpus with 0.7B sentences. Lists of frequently used arguments were computed from Japanese case frames [9] by simple aggregation taking the top 200 arguments on each side.

Improving Similarity

Subspaces are creating by performing an SVD decomposition on a matrix composed of stacked word vectors. Words in the matrix are frequently occurring with the target over the relation. The subspace is creating using parts of singular values Σ and right singular vectors V^T for the SVD. The original setting of creating a prototype projection is to drop some percentage of right singular vectors corresponding to lowest $d_{l\%} = 0.2$ (20%) part of all singular values. This means that k in the expression $\Sigma_{0:k} V_{0:k}^T$ is going to be computed so it will include $1 - d_{\%}$ percent of all singular values.

Table 1 shows を case semantic similarity for the かける as a predicate and words 電話, 世話, 迷惑, 負担, コート and 鍵 as arguments. The method seems to work for the definition of working that similarity for the closer meaning should be higher than for distant ones. There seems to be a certain cluster for the words

Table 2 Similarity of Prototype Projection $\bigcirc|^{\text{を}}_{\text{掛ける}} + \text{掛ける}|^{\text{を}}_{\bigcirc}$, $1 : k$, $d_{l\%} = 0.5$.

	世話	迷惑	電話	コート	眼鏡	鍵	声
世話	1.000	0.746	0.415	–0.105	0.052	0.260	0.534
迷惑	0.746	1.000	0.283	–0.348	–0.085	0.024	0.465
電話	0.415	0.283	1.000	0.217	0.452	0.438	0.481
コート	–0.105	–0.348	0.217	1.000	0.539	0.306	–0.076
眼鏡	0.052	–0.085	0.452	0.539	1.000	0.500	0.183
鍵	0.260	0.024	0.438	0.306	0.500	1.000	0.299
声	0.534	0.465	0.481	–0.076	0.183	0.299	1.000

Table 3 Prototype Projection similarity of $\bigcirc|^{\text{を}}_{\text{掛ける}}$, $1 : k$, $d_{l\%} = 0.5$.

	世話	迷惑	電話	コート	眼鏡	鍵	声
世話	1.000	0.473	0.070	–0.470	–0.324	–0.231	0.235
迷惑	0.813	0.532	0.044	–0.550	–0.408	–0.352	0.315
電話	0.070	–0.089	1.000	0.132	0.250	0.181	0.031
コート	–0.470	–0.466	0.132	1.000	0.729	0.239	–0.397
眼鏡	–0.324	–0.367	0.250	0.729	1.000	0.373	–0.317
鍵	–0.231	–0.261	0.181	0.239	0.373	1.000	–0.293
声	0.235	0.184	0.031	–0.397	–0.317	–0.293	1.000

世話, 迷惑, 負担 and their relation for the application to humans. Their similarity with other selected words is lower. On the other hand, 電話 has the highest similarity with 鍵, but 声 is still higher than other words.

For the task of phrase similarity, this model was good. However for the task of assigning a lower similarity for semantically distinct senses and a higher similarity to semantically close words it does not produce a similar result.

SVD is related to eigendecomposition, which means that top vectors are, speaking informally, *main* components of every vector in the matrix. Assuming that the singular vector related to the highest singular value holds the information of the "word itself" and other vectors hold semantic information of its usage in different contexts, it would make sense to discard the *top* singular vector as well. **Table 2** shows the similarity of prototype projections for the same expression when the expression for computing prototype subspace is $\Sigma_{1:k} V^T_{1:k}$, dropping a top singular vector and bottom singular values related to $d_{l\%} = 0.5$ (50%) singular values.

Using prototype projections in this setting is more suitable for detecting different senses, however, parts of projection themselves are interesting as well. Actually, in the setting of $d_{l\%} = 0.2$ projections かける into subspaces spanned by each word かける $|_{\circ}$ yielded almost similar vectors, most having similarity of > 0.9. Summing them to projection $\circ|_{\text{かける}}$ created a situation that almost every similarity is high. For setting $d_{l\%} = 0.5$, $1 : k$, projection かける $|_{\circ}$ had lower similarities, however they were not usable for similarity calculations. In contrast to that, projection $\circ|_{\text{かける}}$ yielded much more "clean" matrix in sense that it clearly separates *semantically close* senses. It is presented in **Table 3**.

Computing Semantic Representations

Prototype projection as a method allows computing a representation for a pair of words and a certain relation like predicate-*wo*-argument. However, it is only a part of a sentence and there could be different arguments modifying the meaning of the word. The resulting semantic representation of a target word in a sentence was combined from multiple single-relation elementary representations. If $v^x_s(t)$ is an elementary semantic vector representation for a target word t over a relation x, then a full representation $v_s(t)$ is going to be a normalized sum of elementary representations

$$v_s(t) = \sum_i w_i v^i_s(t),$$

where w_i is a weight of an elementary representation. Parameters of a prototype projection were $1 : k$ and $d_{l\%} = 0.5$. We compute semantic vector representations differently for each of our four target parts of speech.

Verbs

For verbs, the main information for creating semantic vectors is the predicate-argument structure of a sentence. The verb is going to be a predicate and its arguments can be used to compute a semantic vector. Data for collocations can be aggregated from the case frames. The following list of cases was used for a summation: を, が, に, と, から, まで, 修飾 and で. For all cases except で, weigts w_i were set to 1 and elementary representations were computed as $t|_{\bigcirc} + \bigcirc|_{t}$. For the で case, we use only the projection of a target word $t|_{\bigcirc}$ with the weight $w_{\text{で}} = 0.5$.

Adjectives

Adjectives are treated the same as verbs. Moreover, they usually use only が case and the meaning mostly depends on the argument of が case slot.

Adverbs

Adverbs usually modify a verb and their meaning can depend on the verb's subject or object, especially if a verb is almost auxiliary like する. Adverbs themselves frequently occupy 修飾 slot of a parent verb as well. Because of this, if an adverb is a child of a verb, representation-wise it can be treated as the verb itself, albeit with the reduced number of cases used. Only 修飾 and が cases are used for computing the semantic representation. The adverb's *parent* is used as a target.

If an adverb is a predicate by itself, the algorithm for verbs is used without modifications.

Nouns

Nouns can create a relationship that changes meaning using a dependency with another noun and a の particle. For example, a phrase "私の犬" is certainly about a dog, but a phrase "警察の犬" can have a different meaning. Thus, this data should be used as well. Original case frames operate only on a predicate-argument structure, so this data was collected from a web corpus. Frequent words over this relation were collected to make case frame-like aggregated lists.

If a noun is a predicate of the sentence, then, as in the verb case, predicate-argument analysis information is used in addition to a relation over の if it exists. Otherwise, in addition to a relation over の, a relation between a noun and its parent is used to compute a semantic similarity vector.

Random Projections

Using counting approaches for creating word vectors results in a high dimensionality. Usually, the dimensionality is equal to the dictionary size. However such sizes are unwieldy for DPP dual representation.

Random Projection is a family of methods that allow to decrease the number of dimensions of a data, while keeping some of the original data properties.

Recall that DPP features can be represented as a matrix B.

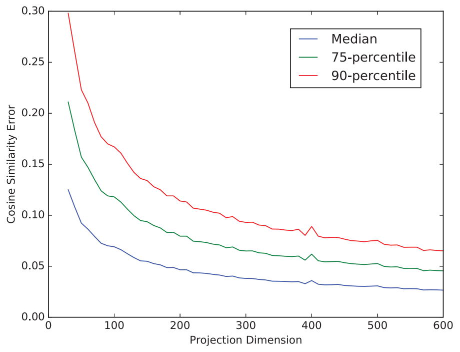

Fig. 6 Cosine similarity error of a random projection.

Rows of this matrix are vectors for each data item. Using lexical and syntactic similarity features, which were described in previous subsections, is going to produce vectors of a large dimensionality M, in hundreds of thousands. However, DPP works efficiently if the number of feature dimensions is relatively low.

Gaussian Random Projection is defined by a matrix P of dimensionality $M \times D$, with the elements of P independently drawn from a normal distribution $\mathcal{N}(0, \frac{1}{M})$, having a zero mean and a variance of $\frac{1}{M}$. A classical result on Random Projections [7] shows that even if $D \propto \log(M)$, with a high probability the distance between original points is approximately equal to the distance between the projected ones. The DPP paper explains the applicability of those results to the DPP case in great detail.

For the example extraction, random projection is used to compress lexical, syntactic, and semantic parts of similarity vectors into a single similarity vector. The size of that vector is chosen experimentally to have $D = 250$. **Figure 6** displays cosine similarity errors of projected vectors compared to non-projected ones. For vectors a and b, and a projection matrix P the cosine similarity error is $|a^T b - (Pa)^T Pb|$.

For this experiment, 1,000 sentences were used. The original dimensions of vectors were 100k, 400k and 300 for lexical, syntactic, and semantic similarities respectively. The semantic similarity vector is small and dense, and after a prototype projection, its effective dimension is smaller than 300. To ease further operations, a concatenated vector of all similarity features was projected into a smaller dimension. For the selected value D of 250, 90% of similarity errors are less than 0.1. Moreover, those errors usually occur when vectors are nearly perpendicular, meaning that their dot product is close to 0. This was not the case for vectors close to each other. Still, when D becomes very small, the approximation error becomes very large.

3.3 Quality Features

Quality features represent an *intrinsic value* of individual sentences as examples of word usage. Our quality feature is defined as a product of four components: semantic centrality (q_i^{cse}), syntactic centrality (q_i^{csy}), difficulty (q_i^{d}) and goodness (q_i^{g}).

$$q_i = q_i^{\mathrm{cse}} q_i^{\mathrm{csy}} q_i^{\mathrm{d}} q_i^{\mathrm{g}}$$

3.3.1 Quality: Centrality

One of main goals of example sentences is to be representative. If the sentence is semantically representative, that means the sentence uses the target word in a frequently used sense. If the sentence is syntactically representative, that means the sentence used the target word in a frequent grammatical pattern. For example, a noun can be frequently used in a certain case with a verb.

This parameter could be treated as a sort of "centrality" of individual usage inside a set of sentences. This centrality could be calculated as a distance to the nearest centroid after an application of a clustering procedure over a fixed similarity measure. For the semantic centrality, the distance measure naturally could be created from a semantic similarity measure. The idea is the same for the syntactical centrality.

For the clustering K-Means++ algorithm was used because it does not require a similarity matrix and is scalable to large datasets. The number of clusters k was set to 30 for semantic centrality. For syntactic centrality, the number of clusters was set to 10 because syntactic diversity is usually lower than semantic one.

3.3.2 Quality: Relative Difficulty

Example sentences should be understandable for learners. Most of the time, the target word should not be "shrouded" by other words. In order to do that, the difficulty of the sentence is adapted as a quality feature. It is done in two steps. First, the difficulty of the sentence is estimated as a single number. Then the difficulty is transformed to a quality coefficient, a number from 0 to 1. This difficulty quality coefficient is multiplied by other quality features.

A sentence difficulty consists of several factors: the grammatical difficulty, lexical difficulty, presence of anaphora, and many other factors. We, however, approximate lexical sentence difficulty d_s of the sentences from the word difficulties using the formula

$$d_s = \left(\sum_{w_i \in s} d_{w_i}^4 \right)^{\frac{1}{4}}.$$

Here d_w is a word difficulty, and w_i are all content words in a sentence. The idea was to make a softmax-like function, that still accumulates difficulty if a sentence contains multiple not very hard words.

A word difficulty is estimated using word frequencies in a corpus and JLPT word lists. First, words are ranked by their frequencies in the corpus. After that, words are assigned a difficulty number based on their rank. Assignments are displayed in the **Table 4**. After the first approximation of the difficulty, the ranked list is merged with JLPT (Japanese Language Proficiency Test) word lists. Some words that have relatively low frequency, however still are present in relatively low JLPT levels and should not be treated as very difficult ones.

Table 4 Word difficulties based on word frequency rank.

Max Word Rank	Word Difficulty
500	0
1000	1
2000	2
5000	3
10000	4
20000	5
50000	6
Rest	7

Table 5 Frequency-based difficulty for words in JLPT lists. Bold entries were moved to lexical difficulty corresponding to the JLPT level.

	Difficulty							
JLPT Level	0	1	2	3	4	5	6	7
N5	182	92	**172**	**84**	**37**	**14**	**5**	**3**
N4	116	68	266	**126**	**50**	**20**	**5**	**1**
N3	167	84	692	492	**187**	**100**	**18**	**3**
N2	49	30	218	388	413	**460**	**207**	**19**
N1	138	73	406	595	648	896	**451**	**42**

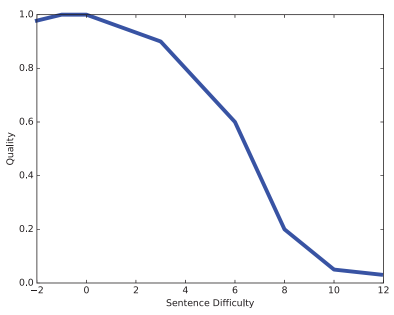

Fig. 7 Sentence difficulty to quality coefficient conversion.

Merging two estimates of word diffuclty is done by decreasing the difficulty of words if JLPT difficulty was estimated higher than frequency-based difficulty. JLPT level N5 was assigned difficulty 1 and JLPT level N1 to the difficulty 5. Frequency-based difficulties for words are shown in the **Table 5**. Bold entries correspond to words that were moved into position so they would have the difficulty specified by JLPT lists.

Here are some examples of words that had frequency difficulty 5 or 6, but a much lower JLPT difficulty. For JLPT N5 ($d_w = 1$) moved words like おまわりさん, 花瓶, 作文 or 上着 which often occur in beginner level textbooks (with hiragana spelling), but do not have a high frequency in the actual language. For N4 ($d_w = 2$), words like 乗り物, 郊外, 押し入れ or 水泳 were moved.

After the computation of the sentence difficulty, the value is transformed into a quality coefficient. It is transformed using a sigmoid-like piecewise linear function $q_d(d_s)$ that is shown on the **Fig. 7**. The idea is to have a slower initial quality decrease when the sentences are not difficult and a rapid decrease when the difficulty becomes larger. The default configuration of this difficulty conversion function stops giving high qualities after the sentence difficulty becomes more than 4, which is about JLPT N3 – intermediate Japanese proficiency level.

Frequency component of word difficulty is computed as $d_w^{\text{freq}} = \lfloor \log_2(1 + w_f/500) \rfloor$. Words which should be known for JLPT N5 were given the difficulty $d_w^{\text{JLPT}} = 1$, words for N1 were assigned $d_w^{\text{JLPT}} = 5$ respectively with other values in between. The final word difficulty score is computed as $d_w = \min(d_w^{\text{freq}}, d_w^{\text{JLPT}})$.

Sentence difficulty is then converted into the quality feature component using a piecewise linear function $q_i^{\text{d}} = T(d_s + \text{bias}_d)$, which is defined as $T = [0, 0.6, 1, 0.9, 0.7, 0.6, 0.2, 0]$ at $[-\infty, -1, 0, 3, 5, 6, 8, \infty]$. The function is rather ad-hoc. It has a maximum of 1 at 0 and decreases to the left and right. We wanted to have positive and negative parts to decrease with the different rates. A bias value bias_d can shift the area of acceptable difficulties for a learner. For example, a bias value of $\text{bias}_d = -3$ would make the quality to be near 1 for the sentences which have the words with the difficulty at most for JLPT N3.

It is possible to modify the difficulty function as $q = q_d(d_s + \text{bias}_d)$, where the bias term would make sentences easier or harder, making it possible to select sentences for learners of other proficiency levels.

3.3.3 Quality: Goodness

The final part of quality features performs soft filtering on the sentences. Web corpus contains a large number of sentences that are actually not full sentences, but only fragments. For example, they could begin with a case marker like "に誰も入ってこない。". Other sentences could not be suitable for example sentences by simple cosmetic criteria. For example, sentences that contain punctuation that hints that the sentence is unfinished are usually not helpful as an example.

Each example sentence is rated by a series of rules and each of the rules output a number from 0 to 1. The product of those numbers is going to be the final "goodness" quality measure.

There are three main types of rules. The first type tries to filter out sentences that will definitely not be good examples, like the ones that begin or end with a case marker. It assigns a low score, near 0.2 or 0.3 to such sentences making the probability of them appear in results extremely low.

The second one ranks things that are completely normal in small quantities, but are bad when numerous. For examples, lots of punctuation in a sentence usually makes it a bad example. Multiple numbers and alphabetic characters can be a part number, useless for an example sentence. However, a single number in a sentence is perfectly fine.

The third type is something in between. For example, entries of the JUMAN analyzer dictionary that automatically were acquired from Wikipedia are not useful in example sentences.

4. Related Work

Several Spaced Repetition Systems are publically available. Anki [*6], open-source software which is probably the most popular SRS available; Mnemosyne [*7], another open-source solution that incorporates research on long-term human memory; and SuperMemo [*8], made by the developer of the original SuperMemo algorithm, which is closed-source and not free. Anki has a plugin that simplifies the process of creating flashcards for the Japanese language. However, none of the above systems automatically provide example sentences. In addition, all of the systems provide only static flashcards without tools to change the content for each repetition. Showing a different example sentence on question cards on each repetition should improve the efficiency of learning vocabulary.

Dictionaries often include example sentences in their articles about words. However, these examples are often very short. 広辞苑 (Kojien) has the following two examples for the second sense of the word 振るう："采配を振るう" and "拳を振るう". They are extremely short and are more like collocations than examples. They do not provide any useful context such as the situations for which this word can be used. Also, the first word in the first example is relatively obscure.

[*6] http://ankisrs.net/
[*7] http://mnemosyne-proj.org/
[*8] http://www.supermemo.com/

There exist dictionaries that consist mostly of example sentences such as Progressive [20] and Wisdom [5]. Example sentences in Progressive are usually full sentences while the example sentences in Wisdom mostly have a fragment-like structure containing only the information needed to understand the word in context. Nevertheless, both of these dictionaries contain very high-quality and useful example sentences. Both of these dictionaries also contain English translations because they explain word usage mostly using example sentences. Still, because the sentences were manually assembled by human editors, their number is limited. Another point to consider is that dictionary content is under copyright protection and cannot be easily used for automated flashcard generation.

There are also freely available example sentence databases. The Tatoeba Project [*9] is a wiki-style database of example sentences maintained by human volunteers. It consists of example sentences for many languages. Thirteen languages have more than 100,000 sentences registered and 39 have more than 10,000. Of note, as of January 22, 2016, there were approximately 570,000 English sentences and 180,000 Japanese sentences. The sentences are interrelated, facilitating translations in different languages in a many-to-many fashion. Still, most of these sentences focus on relatively easy words and many of the sentences are very similar to each other.

Automated extraction of example sentences from a corpora has also been proposed. GDEX [10] describes semi-automated example extraction for the preparation of the electronic version of a Macmillan English Dictionary. The authors select example sentences for English learners and define a suitable example sentence as:

- typical, showing frequent and dispersed patterns of usage,
- informative, helping to educate the definition,
- readable, meaning intelligible to learners, avoiding difficult words, anaphora and other structures that make it difficult to understand a sentence without access to a wider context.

The authors used sentence length, word frequency, information about the presence of pronouns and some other heuristics to judge the quality of sentences. Subsequently, the final example sentences for the dictionary were manually selected by editors. The authors reported problems with garbage and list-like constructions in the raw data that are not useful for example sentences. Still, their approach decreased the time required to construct the dictionary.

There are numerous approaches that approach the problem of selecting example sentences mostly as a word sense disambiguation (WSD) problem [4], [8], [17]. Specifically, de Melo et al. [4] proposed the use of parallel corpora to extract disambiguated sentences from an aligned subtitle database. However, they only examined Spanish-English language pairs and the total number of sentences in their work was small (117,000 sentence pairs). Aligned corpora usually are small or belong to a specific domain, whereas example sentences should be from different domains and cover rare words. One more important feature of that work is a concern about *diversity* of example sentences. It is possible to say, that de Melo work considers mostly lexical diversity and centrality, and does not consider sentence difficulty. Sentence difficulty turns out to be a major factor in the evaluation by language learners.

Shinnou and Sasaki [17] target the Japanese language. They cluster initial sentences into a specified number of clusters. Then, by showing a pair of central sentences to a human operator, they decide whether the clusters should be merged. The authors only consider nouns and measure the similarity between sentences using word overlap and a thesaurus. Their approach is semi-supervised and requires user interaction for system decisions. Furthermore, it is not created for language learners.

Kathuria and Shirai [8] explore the use of disambiguated example sentences in a reading assistant system for Japanese learners. They create a system that assists reading by showing disambiguated example sentences that have the same sense as the word in the text. The senses are defined by the EDICT dictionary [1]. The authors perform WSD based on the similarity between sentences where the similarity consists of a collocation and a sema-syntactic part. The second aspect is based on dependency parse information combined with Goi bunrui hyou [13] prefix matches. Only an aligned corpus is used to extract the example sentences, limiting the number of potential example sentences.

Sentence search tools are related the most to corpus management and exploration tools. However, there are not many tools that use structural information. Jakubicek et al. implements a syntactic corpus search system [6]. This system focused on searching using constituency instead of dependency syntactic structures. Also, the system was not a search engine and query times on sentence structure were in orders of tens of minutes which renders working with huge corpora impossible.

For the Japanese language, dependencies are used in search as well [18], [19]. TSUBAKI search system [18] uses dependency trees for indexing and querying and it is distributed. However, it is a document-level search system and does not allow doing queries using POS information. A system proposed by Takeuchi and Tsujii [19] uses dependency information, but it does not allow the use of grammatical and dependency information. It has more focus on handling paraphrases. Recent versions of Chaki (茶器) corpus management tool [*10] support queries using dependencies, lexical and part of speech information. Chaki, however, can not handle corpora on the scale of hundred million sentences.

5. Evaluation

5.1 Goals of Evaluation

Evaluating the suitability of example sentences for learning a foreign language is difficult. Evaluating the sentences one by one does not determine the diversity of the extracted sentence list.

The automatic evaluation of example sentences is possible if the problem is formulated such that the only criterion is that example sentences should be present for every sense of a word. However, this evaluation does not determine whether the example sentences are actually useful for learners.

We perform an evaluation experiment with learners and a

[*9] http://tatoeba.org/eng/

[*10] https://osdn.jp/projects/chaki/

teacher with two distinct main goals:

(1) To assess the performance of the example extraction system quantitatively;

(2) To validate the assumptions on the meaning of the "quality" of example sentences.

The first goal is achieved by having participants vote on lists of example sentences and select their preferred lists.

For the second goal, the evaluation was performed in the form of an interview. Participants were asked why they have or have not chosen specific lists of example sentences after the initial preference selection.

5.2 Baselines

We used three methods in the evaluation: the proposed one and two baselines. The proposed method is labeled **DPP** in the evaluation results.

The first baseline was a method by de Melo [4], explained in the Section 4. However, because our setting uses only monolingual corpora, only lexical centrality and diversity parts were used from this method. As an additional point, the method did not use raw sentences as input, it was using results of a search engine instead. The method is referred to as **DeMelo**.

The second baseline was a simple uniform random sampling without replacement. The input data, as in **DeMelo**, was a list of example sentence candidates from a search system, not raw examples. Random sampling should have a high diversity in many aspects, however, its results could not be consistent. This method is referred to as **Rand**.

5.3 Data Preparation

For the experiment we have selected the following 14 words:

- verbs: 飛ぶ, 積む, 走る, 取る, 掛ける;
- adjectives: 青い, 汚い, 鋭い;
- adverbs: 全然, バリバリ, サッパリ;
- nouns: 頭, 卵, 足.

Each word has more than one sense and diverse usage patterns. Most of the words are relatively easy and should be familiar to language learners.

For each of the words, we used the top 10k search results from the search engine as example sentence candidates. Each of the words had more than 10k containing sentences. After that, 12 sentences were extracted by each method from each list. That yields a total of $14 \times 12 \times 3$ sentences which were presented to participants of the experiment.

5.4 Learner Evaluation: Protocol

The first part of the evaluation experiment used Japanese language learners as participants. For each word, participants were given three lists of example sentences produced by three methods. The lists were placed side by side in a random order to force participants to read sentence lists in a different order every time. Participants were asked to select a list which was more useful from their point of view for putting sentences on the flashcards. After a participant would select a personally preferable list, anonymized names for the methods were displayed and the participant was asked to explain the reasons behind their choice.

Before the experiment, participants were shown the experiment guidelines, consisting of three main points:

- a brief introduction, explaining about flashcards and usage of example sentences in flashcards;
- explanation of the experiment;
- data handling policy.

Experiment explanation itself was exactly the following text:

- You are going to see automatically-collected example sentences.
- Sentences are going to be created for 14 words: 5 verbs, 3 adjectives, 3 adverbs, and 3 nouns.
- Sentences will be grouped into 3 lists.
- You should select a list which you would prefer to use for creating flashcards for a word.

No explicit criteria for selecting the best list of example sentences was given. Instead of that, participants were asked to explain the selection in an interview-like manner to satisfy the second goal of the experiment.

5.5 Learner Evaluation: Quantitive Evaluation

In total, evaluation and interviews were performed with 23 learners. The first stage of evaluation had 11 participants (1–11) and the additional evaluation had 12 participants (12–23). The evaluation took about 1.5 hours per learner on average. In the first evaluation, there were two participants with relatively low levels. They mostly preferred sentences extracted by DPP because sentence difficulty was used as a quality feature. Participant #1 still had problems with understanding sentences because of the low Japanese proficiency, however, participant #9 could understand most of them. Interview results for the evaluation by learners are discussed in the following subsection. In the second evaluation, we focused on learners with intermediate (N3–N2) levels, for which we had insufficient coverage in the first evaluation.

Vote counts for users and aggregated counts are shown in **Table 6**. DPP gets about half of all votes, which is a good result for the proposed method. It also gets a majority for every participant who had the experience of using flashcards or spaced repetition systems. This gives hope that example sentences are going to be useful inside the flashcards.

Table 7 shows 95% confidence intervals for learners' votes over the example sentence lists. Confidence intervals (CI) were computed by bootstrap resampling: we resampled individual learner votes from a categorical distribution using the collected data as a vote distribution for a learner. Then we aggregated the results in the same way as in the main experiment. We used 100,000 samples in the bootstrap resampling and show lower and upper bounds as the computed percentage at 2.5 and 97.5 levels.

Based on the bootstrap resampling, N3 learners' vote for DPP is larger with a statistical significance than the majority of votes (50%) with the lower CI bound of 64.3. Voting for all learners and N2 learners is larger with a statistical significance than a random choice (33%). On the other hand, the voting of N1 learners for DPP is not larger with a statistical significance than a random choice.

Table 6 Learners' votes on the best example lists. Bold numbers are the majority for a person. FC means the participant has the experience of using flashcards. The level is approximate JLPT-style Japanese language proficiency from N5 (lowest) to N1 (highest). + and − near a level means that the participant is higher or lower than the specified level, and the specified level is the closest one.

#	FC	Level	Rand	DeMelo	DPP
1	*	N4+	3	1	**10**
2	*	N2+	5	3	**6**
3		N2	4	**6**	4
4		N2	5	2	**7**
5		N1	**7**	4	3
6	*	N2−	3	4	**7**
7		N1−	**8**	0	6
8		N1−	4	**7**	3
9	*	N3−	0	1	**13**
10	*	N1−	2	3	**9**
11	*	N1−	3	2	**9**
12	*	N2	0	1	**12**
13		N2	4	**5**	**5**
14	*	N2	3	**6**	5
15		N2	6	1	**7**
16	*	N3	3	2	**9**
17		N1	3	**6**	5
18		N2	**8**	0	6
19		N1	**7**	3	4
20	*	N2	2	2	**10**
21		N3	2	2	**10**
22	*	N3	4	0	**10**
23		N2	0	2	**12**
Total			86	63	173
Percentage			26.7%	19.6%	53.7%

Table 7 Learner vote ratios for DPP and the corresponding 95% confidence intervals, computed with bootstrap resampling.

Level	Votes, %	Lower Bound	Upper Bound
All	53.7	48.8	58.6
N3	75.0	64.3	85.7
N2	53.2	46.1	60.4
N1	39.8	30.6	49.0

5.6 Evaluation by a Native Teacher of Japanese

The second part experiment was performed by showing the same example sentence lists to a native Japanese language teacher. In addition to selecting the best list, a teacher was asked to rank from 1 to 5 how appropriate the list was for students of approximately N3 and N2 JLPT levels. Similar to the learners' case, no explicit criteria were given. Unfortunately, because of time limitations, only one teacher has participated in the second part of the evaluation.

The reason that only teacher was asked to evaluate multiple levels was because we assumed that the learners are not qualified to answer this question. While they can answer whether the sentences are suitable for themselves, learners do not have much exposure to many other learners and can not correctly judge the sentence suitability for different levels. Additionally, high-level learners seem to underestimate the difficulty of the sentences for the stundents of the lower levels.

For the initial selection, the teacher commented that the best list was selected as if examples were for learners of N3 level. The votes on the initial selection were 0, 4, 10 for Rand, DeMelo, and DPP respectively. Average ranks for lists were 3.36, 3.79, 4.64 and 3.86, 4.21, 4.36 for N3 and N2 learner level respectively.

Results of the evaluation by the teacher also assign DPP system is the best for N3 learners both by vote numbers and by average rank. For N2 learners a score for DPP was lower, at the same time the score for DeMelo has raised. The score for Rand was the lowest.

Table 8 Sense coverage of the extracted example sentences. Gold is the number of senses in the monolingual dictionary. Rand, DeMelo and DPP is the number of senses in the extracted example sentences.

Word	Gold	Rand	DeMelo	DPP
飛ぶ	13	4	6	5
積む	6	4	3	3
走る	12	4	4	3
取る	75	11	11	10
掛ける	48	9	10	5
青い	3	2	2	3
汚い	6	3	3	3
鋭い	5	4	3	3
全然	3	2	2	1
バリバリ	3	2	2	2
さっぱり	5	2	3	1
頭	10	3	3	3
卵	4	2	2	4
足	6	1	2	2

Criteria for selection were the following. Non-target words in a sentence should not be too difficult. A sentence should not depend on the outer context like if it was inside the conversation or about current affairs. The sentences should be short and the usages of the target words should be common. These criteria are strongly aligned with the quality features DPP uses for selecting example sentences, which seems to be the reason for its high appraisal by the teacher.

If examples would be selected for N2-like learners, a sentence should include more different structures and usages. However, if usages are too non-usual, in contrary they are more difficult to use, albeit interesting. However, some high-level students had a different point of view.

5.7 Evaluating Semantic Diversity

To check the semantic diversity we have performed word sense disambiguation manually, with senses defined by Super Daijirin Japanese Dictionary. **Table 8** shows the number of senses in the extracted sentences and the total number of senses in a word, as defined by the dictionary. Generally, all methods have comparable performance. However, DPP has significantly lower semantic diversity with 掛ける, and could not produce more than one sense for 全然 and さっぱり. On the other hand, it has better semantic diversity for 青い and 卵. To conclude, DPP keeps the diversity comparable to other methods, while producing overall easier to read sentences and cleaner sentences.

6. Discussion

All evaluations were performed in an interview manner. Participants were asked to explain their choices about lists and criteria they were using. We show frequently discussed positive and negative feedback in **Table 9**.

Generally, list diversity was regarded as one of the main criteria for voting and this criterion was independently developed by most of the learners. Semantic and lexical diversity was the mainly referred part. However, grammatical diversity was named as well. By grammatical diversity participants usually meant, for example, usage of a verb in different grammatical forms. Other themes that frequently came into the criteria for the selection were

Table 9 Comparison of Qualitative Evaluation.

Methods		
DPP	**DeMelo**	**Rand**
Positive Feedback		
○ Short ○ Contain different usages ○ Simple and easy to understand ○ No useless words ○ Possible to guess word meaning from context ○ Look “good” ○ Closer to daily life	○ Contain different usages ○ Contain different grammar	○ Contain different usages ○ Interesting usages ○ Long and lots of context
Negative Feedback		
○ Sentences are short and bland ○ Compound verbs (e.g. 飛び込む) seem not the same as plain verbs (飛ぶ) ○ Similar sentences	○ Many sentence fragments ○ Used difficult grammatical constructions ○ Sentences look too informal ○ More difficult to read fast ○ Similar sentences ○ Lack of punctuation	○ Too much katakana-words ○ Very long sentences ○ Contain difficult words ○ Have words that are not useful for examples ○ More difficult to read fast ○ Emoticons

sentence difficulty and *how interesting* were the sentences. Each of the points is discussed in greater detail below.

6.1 Similarity and Diversity

Diversity was the main hypothesis behind this work and it was validated by the answers of the participants. Most of them have stated that the non-similarity of sentences in a list was one of the main criteria for the selection.

All three used methods were specialized to produce non-similar sentences. DeMelo explicitly tries to select sentences with frequent words and penalize such words in the next selections. Random selection is going to select different words with the high probability if the ground set contained the diverse sense in approximately equal proportions.

For the DPP features were explicitly crafted to deal with semantic and syntactic similarity in addition to lexical similarity. Based on the results, there were cases where DPP was better in terms of diversity and the cases where it was worse.

One example of good performance in this regard was the word “卵”. In addition to the usual meaning of an egg in a sentence like “それには多くの卵を割る必要があります”, the DPP have also displayed several sentences for the usage like “医師の卵に期待が集まっている” with the meaning of “future profession”. Other methods did not produce example sentences with this sense.

For the word “頭” DPP also had produces a sentence with a sense not covered by other methods, but had slight problems with variety. Namely there were 6 sentences with the regular meaning of the word as “head” like “彼女は僕の頭に手をかける”. However, the other 6 had the meaning of beginning of a time period like in the sentence “今年の頭に撮った写真です”. For nouns, most semantic similarity comes from the relation over の particle and the rest from the relation with the noun’s parent if there is one.

The worst-rated DPP selection was for the word “取る”. It had several almost same sentences, for example, “自分の行動に責任を取れ” and “自分の行動に責任をとる”. General sense diversity was not good as well. There could be several reasons for this.

The first reason is that writings of 取る in example sentences were different, as using kanji – “取れ” in the first sentence against hiragana-only “とる” in the second sentence. Hiragana only writing can have multiple ambiguous kanji writings – at least (取る, 執る, 捕る, 採る, 摂る, 撮る, 盗る). A tool could correctly disambiguate them and select the correct one – 取る, but it could not do it in this case and simply produced a list of all candidates. Results of predicate-argument structure analysis contained the non-disambigulated mixture of different writings of とる. In result, the semantic representation vectors become non-similar for usages of 取る compared to とる.

In the case of 掛ける while there is a lack of sense diversity, which is mostly defined by arguments in を case, DPP selects sentences with differing arguments in other slots, e.g., 遊んでいる時に声をかけてみました。(time case present with 時 as argument) and 何かあったら声をかけてね。(no additional cases).

The second reason probably lies in model parameters. There is no training data at the moment to tune model parameters to produce the best example sentences, the greedy selection algorithm with DPP ranking requires tuning in terms of similarity features, quality features, and making every item more or less similar. Applications of DPP described in the Kulesza et al. [11] used training of the quality features and tuning of similarity parameter r, although in this application there was no training data to do so.

The last probable reason lies in the DPP method itself. It is possible not only to select from it greedily, as we do in this work but also to sample from DPP as a distribution, where more diverse subsets are assigned a higher probability [11]. One of its useful properties is that an expected size of a sample $E[n]$ from a DPP is

$$E[n] = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_i + 1},$$

where λ_i are the eigenvalues of C or L matrices. Expected sample sizes are shown in the **Table 10**. Most of the expected sample sizes are very large, much larger than the 12 sentences that were extracted. Large expected sample sizes could mean that one needs to select a large number of items to get a good diverse subset. It seems that similarity features are the reason for high expected sample sizes. Lexical features could give large variation to the shape of vectors and form many distinct eigenvectors. We believe that tuning parameters of similarity feature mixing may help to resolve this problem as well.

Table 10 Expected DPP sample sizes.

Word	飛ぶ	積む	走る	取る	掛ける	青い	汚い
$E[n]$	58.27	67.67	74.53	69.78	75.24	74.32	77.60
Word	鋭い	全然	バリバリ	サッパリ	頭	卵	足
$E[n]$	62.31	110.51	32.80	101.41	110.28	77.59	113.40

As a side note, examples for ばりばり, which had the lowest expected sample size, were good and it was a clear victory of the DPP algorithm. The votes were 2, 1, 8 for Random, DeMelo, and DPP respectively.

6.2 Difficulty

Sentence difficulty also one of the main criteria learners have used for the list selection. The initial assumption for the creation of the system is that example sentences should be easy to understand and as short as possible. One reason for this is because if example sentences are shown as a question on each flashcard, a learner has to read many sentences and overly long sentences create too much cognitive load.

Some learners agree with the initial vision on example sentence difficulty: "an example sentence should not contain words harder than its target". However, there was another point of view as well. If learners thought that the sentences were *for the reference*, like those shown with the definition of a word in a dictionary, then they selected the sentences which were readable, but not *too easy* compared to others. Examples by DPP were *too easy* for those learners. Mostly the people who did not have experience of using flashcards selected sentences in this manner.

There was another small group of learners who wanted to see really difficult sentences. Probably, the difficulty of example sentences should be customizable and one-size-fits-all type of solution is not going to work.

One more "low hanging fruit" that was not done for the sentence difficulty is using kanji for estimation. Learners of lower levels from countries that do not use Kanji simply could not read sentences containing kanji unknown to them. This point should be included in the improved version of the difficulty estimation.

Hypotheses

Optimal for a learner relative difficulty of an example sentence for a target word is a function having at least the following three parameters:

- example usage mode,
- level of a learner,
- learner's familiarity with a target word.

The difficulty of kanji, sentence non-target words, and grammar are included in the function as well, still, we would like to discuss the parameters mentioned in the bullet list in more detail.

Example usage mode is whether example sentences are used for reference or a review. *Reference* usage occurs in a dictionary-like setting where a list of example sentences is presented for a learner to compare between word usage in different senses or situations. In this setting, a learner has other sentences to serve as a sort of anchor to focus on a target word. Because of this, and a need to provide a way to compare sentences from each other, example sentences could have a higher difficulty when they make up a reference list. In contrast to that, if an example sentence is shown as a flashcard question, there is no such "anchor" to compare a single sentence to others. *Review* sentences should be slightly easier to understand because they have fewer hints for a learner in general.

The level of a learner is another major factor whether an example sentence is going to be useful or not. Learners of higher levels are going to understand more difficult sentences and easy ones become boring to them. However beginners and intermediate level learners can find it difficult (and sometimes even impossible without an additional explanation) to understand sentences that advanced learners find interesting.

The last point is the learner's familiarity with the example sentence target word. It strongly relates to the difficulty of the non-target word in a sentence. If a learner is not familiar with the target word, then the sentence itself should be easier. Other words should serve mostly as an explanation for the target word's meaning. If a learner is generally familiar with the word, that context given by an example sentence helps the learner to learn and remember the usage situations of the target. Sentences in this period of familiarity could be a bit harder than the average. However when a learner is completely familiar with the target word, then even usage situations can be inferred by a simple collocation. Collocation is going to help with disambiguating word senses and nothing else.

It seems that we should talk not about good example sentences in general, but about good example sentences *for a learner at some point in a learning process*. Static example lists are not going to solve this problem efficiently, but an educational tool like SRS can. It has access not only to the learner's general knowledge level but to the learning process data for individual words as well. Using it, example extraction system can provide the best examples learner needs at that point of time.

6.3 Sentence Content

Another criterion that was used by learners for selecting sentences was if the sentences were *interesting*. There were 3 main types of such sentences:

- Sentence has a story.
  "画像が汚いのは、携帯カメラで撮ったからです、今度綺麗な写真でも撮っておきましょう" vs. "画像が汚かったりしたら買う気しませんからね"
- Sentence displaying a vivid image.
  "旧ソ連の宇宙飛行士ガガーリンの有人宇宙飛行「地球は青かった」"
- Sentence is funny or unusual for the participant.
  "「中天」とは死者が修行を積むような場所"

At the beginning of the evaluation, there were cases where a single *interesting* sentence gave a reason for a participant to select a list of sentences even if it contained sentences of generally lower quality like complete fragments. After that case, participants were additionally instructed to try judge lists on a whole and not focus only on a single sentence.

Still, such content usually occur only in more or less lengthy sentences containing many different words. DPP was heavily biased against such sentences.

At the same time, lengthy sentence content does not usually interact with the target words directly. Thus such cases should be treated specially.

At present, it is very hard to automatically judge if a sentence is going to be funny, unusual or generally interesting for a learner. However, it should be possible to automatically get sentences that have some kind of story. A simple story is two events with a cause-effect relationship. By exploiting information about events and event relations developed by Shibata [16] it should be possible to collect sentences with frequent events. These story-like sentences could be more helpful for remembering the connotations of a word and its usage.

6.4 Problems with Tooling

Recall that example sentence candidates were pre-selected by a search engine that tries to find good patterns of target word usage, while keeping a sentence length not very long. This kind of selection has actually helped to sidestep most problems related to errors in automatic syntactic analysis (because the sentences were generally short) and predicate-argument structure analysis (because the sentences were selected to have patterns useful for that analysis). However, there were several classes of problems with external tooling that have decreased the performance of the example extraction system in general.

One of those problem classes was discussed in the Section 6.1. Ambiguity in a canonical representation makes it difficult to compute good semantic similarity vectors for such words. Most of the other problems manifested in invalid search results providing bad example sentences to the pre-selection lists.

These problems were mostly caused by the mistakes of morphological analyzer. Sometimes, because of that a completely different word could appear in search instead of a target. For example a sentence 相手が居なけりゃ自分でやりゃあええ。was found when searching for the word "青い". It seems that a morphological analyzer have segmented (やりゃ|あええ) incorrectly (instead of やりゃあ|ええ) and because of that 青い[*11] have "appeared" in the sentence. Incorrect words in sentences because of segmentation errors is a bad problem because such sentences do not actually contain the target word and are completely useless as examples.

Another problem related to analysis arises from character sequences that were unexpected for the analyzer like typos or dialects. For example, a sentence "そんなことを言うとりました" is probably a dialect. The analyzer thinks that "とりました" is a verb 取る and again an incorrect sentence appears in the search result.

6.5 Support of Other Languages

An example extraction system is developed for Japanese, however underlying methods have very few Japanese-specific parts. The system itself is unsupervised and has only a tokenizer, morphologic analyzer, and dependency parser as software dependencies. All other data can be created from a raw corpus analyzed by these three tools.

For example, English could be supported by using a dependency parser that performs labeled dependency analysis like a Stanford Core NLP parser [3]. Parsing a corpus will produce information sufficient to build a database of relations with prototypes, needed to create semantic representation. Creating word embeddings does not require even that information.

7. Conclusion

We proposed an automated example sentence extraction system for language learning using automated tools like spaced repetition systems. The propsed example extraction system focuses on extracting diverse and good sentences and consists of a syntactically aware distributed search system and determinantal point process-based extraction part. We evaluated the system on the Japanese language with learners of different levels and a native teacher of Japanese as a second language. In both settings, the participants preferred our approach to baselines. The proposed method was especially preferred by intermediate (N3) learners, followed by upper-intermediate (N2), with the results being statistically significant in both cases. Advanced level learners (N1) found sentences produced by our method to be short and bland, but still slightly preferred our method to other methods.

Acknowledgments We thank Government of Japan and Ministry of Education, Culture, Sports, Science and Technology for the scholarship support which made this work possible in the first place. We also thank Kyoto University Design School for financial support in this research. The first author additionally wants to thank Maruhashi Koji, Okamoto Seiji and Anai Hirokazu of Fujitsu Laboratories; Takaoka Kazuma and Uchida Yoshitaka of Works Applications Enterprise with the support for this work. Finally, we want to express our gratitude to volunteer student and teacher evaluators.

References

[1] Breen, J.: Proceedings of the Workshop on Multilingual Linguistic Resources, pp.65–72 (2004) (online), available from ⟨http://aclweb.org/anthology/W04-2209⟩.

[2] Cepeda, N.J., Pashler, H., Vul, E., Wixted, J.T. and Rohrer, D.: Distributed practice in verbal recall tasks: A review and quantitative synthesis, *Psychological Bulletin*, Vol.132, No.3, pp.354–380 (online), DOI: 10.1037/0033-2909.132.3.354 (2006).

[3] Chen, D. and Manning, C.: A Fast and Accurate Dependency Parser using Neural Networks, *Proc. 2014 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pp.740–750, Association for Computational Linguistics (online), DOI: 10.3115/v1/D14-1082 (2014).

[4] de Melo, G. and Weikum, G.: Extracting sense-disambiguated example sentences from parallel corpora, *Proc. 1st Workshop on Definition Extraction, WDE '09*, Association for Computational Linguistics, pp.40–46 (2009) (online), available from ⟨http://dl.acm.org/citation.cfm?id=1859765.1859772⟩.

[5] Inoue, N. and Akano, I.: *The Wisdom English-Japanese Dictionary* (*in Japanese*), Senseido (2007).

[6] Jakubicek, M., Kilgarriff, A., McCarthy, D. and Rychly, P.: Fast Syntactic Searching in Very Large Corpora for Many Languages, *Proc. 24th PACLIC*, pp.741–747, Tohoku University (2010).

[7] Johnson, W.B. and Lindenstrauss, J.: Extensions of Lipschitz mappings into a Hilbert space, *Contemporary Mathematics*, Vol.26, No.189-206, p.1 (1984).

[8] Kathuria, P. and Shirai, K.: Word Sense Disambiguation Based on Example Sentences in Dictionary and Automatically Acquired from

[*11] あええ can be a phonetic change from 青い or 淡い

Parallel Corpus, *Advances in Natural Language Processing*, Lecture Notes in Computer Science, No.7614, pp.210–221 (2012).
[9] Kawahara, D. and Kurohashi, S.: Case Frame Compilation from the Web using High-Performance Computing, *Proc. 5th International Conference on Language Resources and Evaluation* (*LREC'06*) (2006) (online), available from ⟨http://aclweb.org/anthology/L06-1203⟩.
[10] Kilgarriff, A., Husak, M., McAdam, K., Rundell, M. and Rychlý, P.: GDEX: Automatically Finding Good Dictionary Examples in a Corpus, *Proc. 13th EURALEX International Congress*, Elisenda Bernal, J.D. (Ed.), Barcelona, Spain, Institut Universitari de Linguistica Aplicada, Universitat Pompeu Fabra, pp.425–432 (2008).
[11] Kulesza, A. and Taskar, B.: Determinantal Point Processes for Machine Learning, *Foundations and Trends in Machine Learning*, Vol.5, No.2–3, pp.123–286 (online), DOI: 10.1561/2200000044 (2012).
[12] Mikolov, T., Yih, W.-T. and Zweig, G.: Linguistic regularities in continuous space word representations, *Proc. NAACL-HLT*, pp.746–751 (2013).
[13] Natural Institute for Japanese Language and Linguistics (Ed.): *Bunrui Goi Hyo*, Dainippon Tosho (2004).
[14] Pavlik, P.I. and Anderson, J.R.: Using a model to compute the optimal schedule of practice, *Journal of Experimental Psychology Applied*, Vol.14, No.2, pp.101–117 (online), DOI: 10.1037/1076-898X.14.2.101 (2008).
[15] Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K. and Borgwardt, K.: Efficient graphlet kernels for large graph comparison, *Proc. 12th International Conference on Artificial Intelligence and Statistics*, Vol.5, pp.488–495, PMLR (2009).
[16] Shibata, T., Kohama, S. and Kurohashi, S.: A Large Scale Database of Strongly-related Events in Japanese, *Proc. 9th International Conference on Language Resources and Evaluation* (*LREC'14*), pp.3283–3288, European Language Resources Association (ELRA) (2014) (online), available from ⟨http://www.lrec-conf.org/proceedings/lrec2014/pdf/1107_Paper.pdf⟩.
[17] Shinnou, H. and Sasaki, M.: Division of Example Sentences Based on the Meaning of a Target Word Using Semi-Supervised Clustering, *LREC 2008* (2008) (online), available from ⟨http://aclweb.org/anthology/L08-1339⟩.
[18] Shinzato, K., Shibata, T., Kawahara, D. and Kurohashi, S.: TSUBAKI: An Open Search Engine Infrastructure for Developing Information Access Methodology, *Journal of Information Processing*, Vol.52, No.12, pp.216–227 (2011).
[19] Takeuchi, J. and Tsujii, J.: A Search System for Japanese Using Dependencies And Paraphrases (in Japanese), *Proc. 11th Annual Meeting of the Association for Natural Language Processing*, pp.568–571 (2005).
[20] Tohno, F., Itabashi, Y. and Althaus, M.E.: *Shogokukan Progressive English - Japanese Dictionary, 4rd Edition*, Shogakukan (2001).
[21] Tolmachev, A., Morita, H. and Kurohashi, S.: A Grammar and Dependency Aware Search System for Japanese Sentences, *Proc. 22nd Annual Meeting of the Association for Natural Language Processing*, pp.593–596 (2016) (online), available from ⟨http://www.anlp.jp/proceedings/annual_meeting/2016/pdf_dir/P15-4.pdf⟩.
[22] Tsubaki, M., Duh, K., Shimbo, M. and Matsumoto, Y.: Modeling and Learning Semantic Co-Compositionality through Prototype Projections and Neural Networks, *Proc. 2013 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), pp.130–140, Association for Computational Linguistics (2013) (online), available from ⟨http://www.aclweb.org/anthology/D13-1014⟩.

Arseny Tolmachev received his B.S. in Ural Federal University, Ekaterinburg, Russia in 2010. He was working as an engineer in the Institute of Engineering Science, RAS (Ural Branch) on high-performance computing applications. From 2013 he became a student of Graduate School of Informatics, Kyoto University and received his M.S. there in 2016. He was a researcher in Fujitsu Laboratories, Ltd. and now a researcher in Works Applications Enterprise. His research interests center on natural language processing, especially low-level analysis and efficient use of computers.

Sadao Kurohashi received his B.S., M.S., and Ph.D. in Electrical Engineering from Kyoto University in 1989, 1991 and 1994, respectively. He is currently a professor of the Graduate School of Informatics at Kyoto University. His current interests include machine translation, information retrieval, and knowledge engineering. He has served as a research supervisor of JST PRESTO: Fundamental Information Technologies toward Innovative Social System Design (2016–2022).

Daisuke Kawahara received his B.S. and M.S. in Electronic Science and Engineering from Kyoto University in 1997 and 1999, respectively. He obtained his Ph.D. in Informatics from Kyoto University in 2005. He is currently a professor of the department of communications and computer engineering, Waseda University. His research interests center on natural language processing, particularly knowledge acquisition and text understanding.