Personalized Sequential Language Vocabulary Learning Recommender System

Peteris Nikiforovs[1] and Laura Bledaite[2]

[1] Free University of Bozen-Bolzano,
peteris.nikiforovs@stud-inf.unibz.it
[2] Free University of Bozen-Bolzano,
laura.bledaite@stud-inf.unibz.it

Abstract. This report describes a personalized sequential language vocabulary learning recommender system based on users' language level, known words, the preferred text difficulty and document length.
We calculate a personalized score for each document and recommend the documents with the highest scores.

1 Introduction

When learning a foreign language, one of the most important aspects is building the vocabulary. While the majority would agree on this, it is still not clear what vocabulary, i.e., what specific terms are most needed and/or most important for a particular person in a particular stage of the learning process.

Even if there exist various recommender systems offering the user texts that might enrich their vocabulary, we haven't come across one. Typically, such systems would make content-based recommendations, i.e., use the information about the text and the user to find the most appropriate ones. However, we argue that the learning process is highly sequential. More clearly, it is more efficient to learn easier, more frequent words from texts with not too sophisticated sentence structure first and to increase the difficuty with the knowledge you gained when you read those easier items before. Therefore, the typical static recommender systems could be significantly improved by adopting a model that makes sequences of recommendations.

The sequential recommendation process has already been analyzed before.([1], [2], [3]) However, these particular types of recommendations have been mostly exploited for a music domain. Nevertheless, there are motivating examples in other domains as well. For instance, it is natural to suggest the user to buy a book which has a sequel even if it has a slightly lower probability of being bought, because the overall expected profit might turn out to be much higher in the former case. Similarly, when helping to learn a language it is sensible to start from an easy book/article and step by step propose to read more sophisticated pieces.

The recommender system that we are planning to build would make sequential recommendations based on the users' personal information and the texts

they've read before. The text in a collection would be described by the bag of words they contain and the linguistic sentence complexity. The idea is to recommend the user such documents that contain a lot of words that he already knows plus a bunch of new ones, also taking into account the appropriate text complexity. In this way, the system would not allow the learner to forget the words which they have just learned and force them to add a couple of new ones, plus would strive to offer the texts of suitable difficulty.

Clearly, a sequence may have many different paths. For example, after reading one document, a person may have a choice of 5 other documents, and after reading each of those, they may choose between 5 more documents in every case. This suggests a decision tree approach, which would allow modeling different paths of choices following the learning process.

Further, we organize our project report as follows. First, we describe the related work done in this area. Then we propose a way to assign personalized scores to documents with a view to recommend the ones with the highest scores. After that we describe what the user interface of such a system would look like and how it is implemented in our prototype. We also discuss how the system may be evaluated and finish with conclusions and ideas of possible improvements of the system.

2 Related Work

As far as we know, the area of sequential recommendations with applications to language learning is a new research area. Probably the most popular domain for sequential recommendations is music. [1] presents a knowledge-intensive Case-Based Reasoning system to generate a sequence of songs customized for a community of listeners. [2] presents and evaluates heuristics to adapt playlists automatically given a song to start with (seed song) and immediate user feedback. [3] describes a recommender system using a particular MDP model, its initialization using a predictive model, the solution and update algorithm, and its actual performance on a commercial site.

Taking into account these quoted studies, we have been focusing on language learning. Therefore, most of the theory used for building it comes from the state of the art of the information search and retrieval.

The more specific area of studies related to our project is a task of natural language processing, namely, sentence difficulty evaluation. A simple measure of difficulty could be the length of it. However, there are many more sophisticated metrics. We will mention several other approaches. [4] proposes that the dependency relations among words in sentences can be represented as directed acyclic graphs (DAGs) and argues that the centrality measures acquired from DAG sentence representation is better than a number of other approaches which include Minimal Terminable Units (T-Units, originally proposed in [6]) and D-Level Scale ([7], [8]). Dependency Locality Theory ([9]) suggests that the syntactic complexity of sentences increases in proportion to the length of syntactic dependency. [5] introduces the possibility of using the average dependency

distances (ADDs) of a sentence as one of the possible measures to indicate its complexity. In our project, we use ADD metric ([5]) which is the improvement of [4] using Dependency Locality Theory ([9]).

3 System Description

3.1 Technical Details

First of all, let us clarify the text characteristics that can influence their suitability for the user at a certain time. We assume that an important characteristic is the number of the words that are not known for the user. Furthermore, we argue that the occurrence frequency of the unknown words is influential. Moreover, we think that the text sentences' linguistic complexity plays a role as well. Finally, the overall text length might also be important.

In order to address the first and the second characteristics, we build an index for all the vocabulary terms. Each record in this index consists of a term, the number of occurrences of this term in all the documents in a collection and the document frequency, i.e. the number of documents that contain the term. Each user is represented by a list consisting of all the terms of all the documents that he already read.

For instance, if the vocabulary of a collection contains only five words and there are 100 documents in this collection, a possible vocabulary index could be as shown in Table 1.

Term	Occurrences	Documents
the	450	100
and	250	100
but	120	90
let	30	30
knife	2	1

Table 1. Example of Vocabulary Index

For each unread document two personalized scores: *new-words-score* (*nws*) and *terms-novelty-score* (*tns*) are then calculated as follows. Let T_u denote a set of terms that a user already read at some point in the learning process, T_d - the set of terms contained in the document d and T_d^n - the set of terms in the document d still new for the user. T_d^n is a set difference of T_d and T_u calculated as follows:

$$T_d^n = T_d \setminus T_u,$$

where $S \setminus T$ is the set which contains all the elements which occur in S but not in T . For example, $\{1, 2, 3\} \setminus \{3, 4, 5\} = \{1, 2\}$.

The *new-words-score* reflects the closeness of the real percentage of new words to the preferred percentage of new words in a document. Then:

$$nws(d) = \frac{1}{y+1},$$

or

$$nws(d) = exp(-y),$$

where

$$y = |\frac{|T_d^n|}{|T_d|} - \alpha|$$

and α is the prefered percentage of new words in the suggested documents. α is tuned by the user using the corresponding slider.

The *terms-novelty-score* for each document is the average of the logarithms of the occurrences of each term in T_d^n:

$$tns(d) = \frac{\sum_{t \in T_d^n} \log_{10} freq(t)}{|T_d^n|},$$

where $freq(t)$ is the overall number of occurrences of term t in a collection. This score favors the documents containing more novel frequent words rather than rarer ones.

The third characteristic mentioned above that might influence the text suitability for the user is the linguistic sentence complexity of the text. We use the average dependency distances (ADDs) of a sentence as a measure to indicate its complexity. It is just one of the many possibilities and we do not argue about its advantages and/or disadvantages in the scope of this paper. The measure was introduced in [5] and the discussion can be found there.

Imagine the sentence *"Ann read the article silently and understood it properly."*. Figure 2 shows the DAG respresentation of it.

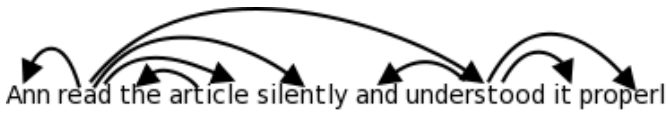

Fig. 1. The DAG representation of a sentence "Ann read the article silently and understood it properly.", preserving the word order.

The ADD of a particular sentence is calculated as follows:

1. Parse the text by Stanford Parser,
2. Calculate the ADD.

The Stanford Parser [10] is used because of its ability to output the dependencies among the words of a sentence together with the distances of those dependencies. The output for the sentence given above (Figure 2) is:

nsubj(read-2, Ann-1)
det(article-4, the-3)
dobj(read-2, article-4)
advmod(read-2, silently-5)
cc(read-2, and-6)
conj(read-2, understood-7)
dobj(understood-7, it-8)
advmod(understood-7, properly-9)

The output of the Stanford Parser is quite self-explanatory. *nsubj*, *det*, *dobj*, *advmod*, *cc* and *conj* represent the types of the dependencies between the words in parentheses. The dependency distance (DD) of a particular dependency is the absolute difference of the involved words in the sentence (the numbers in parentheses).

$$DD(dep, s) = |pos_1^{dep}(s) - pos_2^{dep}(s)|,$$

where $pos_i^{dep}(s), i \in 1, 2$ denotes the position of i-th word of the dependency *dep* in the sentence s. Therefore, the dependency distance of *nsub(read-2, Ann-1)* is 1 as explained below.

$$DD(nsub(read-2, Ann-1), s^*) = |2-1| = 1,$$

where s^* is the sentence being analyzed (*"Ann read the article silently and understood it properly."*).

Then, the ADD of a sentence is the average of all the dependency distances of a sentence. Let D_s denote the set of dependencies of a sentence s. Then the ADD of a sentence is:

$$ADD(s) = \frac{\sum_{dep \in D_s} DD(dep, s)}{|D_s|},$$

where $|X|$ denotes the number of elements in a set X.

Therefore the ADD of the sentence being analyzed is equal 2.375.

$$ADD(s^*) = \frac{1+1+2+3+4+5+1+2}{8} = \frac{19}{8} = 2.375$$

The third characteristec mentioned - *sentence-complexity-score* (*scs*) is calculated as an average of all individual ADDs of the sentences of a text.

$$scs(d) = \frac{\sum_{s \in d} ADD(s)}{sent(d)},$$

where $sent(d)$ is the number of sentences in document d.

Because the scale of the scs score is different from the scale of the nws and tns scores, it is normalized.

$$scs^{norm}(d) = \frac{scs(d)}{\max_{d' \in D} scs(d')}$$

The overall score would be tuned by the parameters set by the user and calculated as follows:

$$score(d) = nws(d)(\alpha) + \beta \cdot tns(d) + \gamma \cdot (1 - scs^{norm}(d)),$$

where α is the preferred percentage of new words, β is the level of the novelty of new terms (the higher the β, the more frequent the new words) and γ is the level of sentence simplicity. All of these parameters are tuned by the user.

The last charasteristic mentioned - the document length - is addressed by classifying the documents into the length classes and letting the user choose the prefered classes. The overall score would then be calculated only for the documents that satisfy the preferred class.

3.2 User Interface

The main goal of the recommender system is to give reading recommendations to the user to better extend her vocabulary with the most frequently used words in the fastest way possible. That is why the recommendations take up most of the space in the user interface. To prove the effectiveness of the system and also enable the user to measure her progress, we offer some statistics that will allow the user to compare her vocabulary before and after reading the recommended documents. The main user interface is split into three main parts.

The first section at the top represents the current user's vocabulary and the third section, functionally and visually identical to the first section, at the bottom represents the user's vocabulary after reading all the provided recommendations. Statistics shown include the number of known words (also as a percentage of the total words in the corpus), the number of unknown words, the number of known top 1000 most frequent words etc. In the prototype, we also show all the known words with the frequency index (e.g. the most frequent word in the corpus is ranked 1) next to it.

The main part of the system, displayed between the vocabulary before and the vocabulary after, is the recommendations. Since they are sequential and meant to be read in the presented order, they are numbered. Below each recommendation some basic statistical information is presented to the user such as the number of known and new words in the document, how much the user knowledge will improve after reading the recommendation etc. In the prototype, we also show all the new words with their frequency rank. Next to the recommendation is an option to read the document on the screen and then mark it as read.

If the user follows the recommendations and reads them in the presented order, her vocabulary, shown in the first section, will be exactly the same as shown in the third section below the recommendations. If the user has no recorded vocabulary, the recommendations will be presented in such an order that the user learns most of the most frequent words first.

The user has the option to personalize recommendations even more by choosing the percentage of desired new words in a document, how much we should favor documents containing more novel frequent words rather than rare ones and the user can also choose the text difficulty. Finally, the user can also filter documents by length (short, medium, long).

In the prototype, there is no option to create a new account but at the top there is a choice to become a user that has read a certain number of documents in the collection.

3.3 System Behavior

We chose to build the recommender system as a website. It has a backend which computes the recommendations and generates the user interface and a frontend which displays the recommendations and allows the user to filter them. In the prototype, the backend which is written in C#/.NET, stores the entire document collection and all indexes in memory and contains a simple web server. The frontend is written in HTML and Javascript and uses AJAX to load the data from the backend.

When the system is used for the first time, for every document its sentence difficulty score is calculated, the document text is tokenized and all tokens and terms are stored, and a word frequency index is then calculated and stored. In the prototype, it's done at every startup. When new documents are added to the collection, the indexes only need to be updated, not regenerated. For every user there's a list of the terms known to the user.

When recommendations are generated for a user, the *new-words-score* and the *terms-novelty-score* are computed for all documents in the collection because these scores are always different for every user as they depend on each user's individual accumulated knowledge. The *sentence-complexity-score* is not user dependent and therefore can be calculated before calculating the final score, which is tuned by the paramenters set by the user. Documents are then sorted by the score in a descending order and the best recommendations are returned.

Because for every document we need to find the difference of terms known to the user and the terms in the document, it takes a long time for a large collection of documents and/or when the user knowledge is large. Our implementation's performance is acceptable when there are thousands of documents in the collection but it takes much longer when there are tens of thousands of documents in the collection which is a more realistic number.

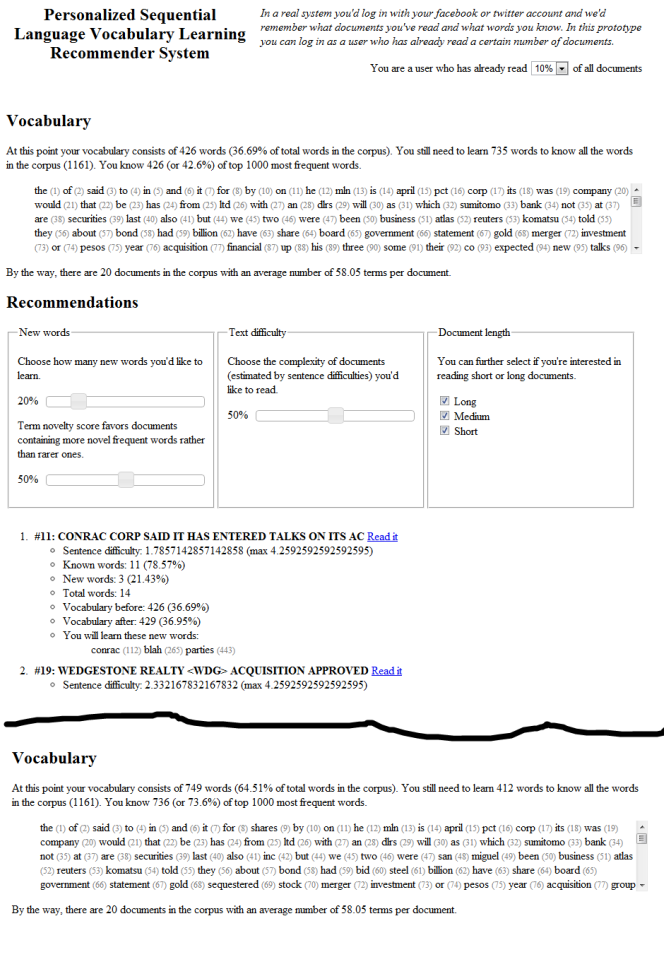

Fig. 2. A screenshot of the user interface of the prototype.

4 Evaluation Strategy

For the system evaluation we chose to use a modification of a standard *explicit feedback* where the user has to mark her opinion about the items suggested. The reason is that the system's objective is not only to suggest a user the interesting pieces to read, but also to guide her properly in the learning process. In order to include the latter aim in the system evaluation, we gather the *indirect explicit feedback*.

After a chosen number of steps in a learning process, i.e. after reading a chosen number of items, the user is presented a test, automatically composed of a number of words that were new for her before she read the recommended items. A test might consist of a list of definitions plus a list of words, where a user should have to join them in an appropraite way. It could also be a list of sentences with gaps to fill in plus a list of words, where the user has to write the given words in the gaps. After completing such a test, the user $u \in U$ is given a mark, which represents a deduced percentage of the new words she actually learned out of those which she saw for the first time. All the users, who completed the test, are then divided into four categories: A, B, C and D. According to the number of users who ended up in a particular category, the system can understand whether it is working successfully.

5 Conclusions and Future Work

Obviously, the system described is not perfect. Even though the main functions are working as can be seen from the prototype provided, many improvements can be done.

First of all, the scalability issues must be dealt with. With a thousand documents the system is still working well. However, such size of collection is not realistic in a real recommender system. Therefore, some modifications are unavoidable.

Secondly, the theoretical part needs improvement as well. At the moment the scores in charge of the percentage of new terms and the novelty of those terms are pretty simple. The logarithm in tns reduces the impact of the increasing number of occurrences while it is growing. However, the more advanced score may perform better leaving space for further improvement.

Thirdly, the area of sentence difficulty evaluation is being researched constantly. Therefore, even if the ADD were the good measure at the time of writing this report, it may not be among the good measures in a couple of years.

Finally, the user interface might also be modified based on the user feedback.

References

1. Baccigalupo, C., Plaza, E.: A Case-Based Song Scheduler for Group Customised Radio. ICCBR '07 Proceedings of the 7th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development (2007) 433 – 448

2. Pampalk, E., Pohle, T., Widmer, G.: Dynamic Playlist Generation Based on Skipping Behaviour. Proceedings of the 6th ISMIR Conference (2005)
3. Shani, G., Heckerman, D., Brafman, R.I.: An MDP-Based Recommender System. Journal of Machine Learning Research **6** (2005) 1265–1295
4. Oya, M.: Directed Acyclic Graph Representation of Grammatical Knowledge and its Application for Calculating Sentence Complexity. Proceedings of The 15th Conference of Pan-P cific Association of Applied Linguistics (2010)
5. Oya, M.: Syntactic Dependency Distance as Sentence Complexity Measure. Proceedings of The 16th Conference of Pan-P cific Association of Applied Linguistics (2011)
6. Hunt, K.W.: Grammatical Structures Written at Three Grade Levels. Champaign, IL: National Council of Teachers of English (1965)
7. Rosenberg, S., Abbeduto, L.: Indicators of Linguistic Competence in the Peer Group Conversational Behavior of Mildly Retarded Adults. Applied Psycholinguistics **8** (1987) 19–32
8. Covington, M.A., He, C., Brown, C., Naci, L., Brown, J.: How Complex is that Sentence? a Proposed Revision of the Rosenberg and Abbeduto D-Level Scale. CASPR Research Report 2006-01 (2006)
9. Gibson, E.: The Dependency Locality Theory: a Distance-Based Theory of Linguistic Complexity. In A.Marantz, Y.Miyashita, W.O'Neil (Eds.), Image, Language, Brain (2000) 95–126. Cambridge, MA: MIT Press
10. De Marneffe, M.C., Manning, C.: The Stanford Typed Dependencies Representation. COLING Workshop on Cross-framework and Cross-domain Parser Evaluation. Retrieved from http://www-nlp.stanford.edu/software/lex-parser.shtml