Vocabby: Vocabulary learning with in the context of user domain

Haemanth Santhi Ponnusamy
Department of computational linguistics,
Eberhard Karls University of Tuebingen,
hameanthsp@gmail.com

Himanshu Bansal
Department of computational linguistics,
Eberhard Karls University of Tuebingen,
email@domain

Abstract

This is an vocabulary learning application for a targeted set of user who already posses some basic skills of the English language and bored of the traditional method of learning new word in the language. This application provides the ability to upload text of user preference such as a story book, research writing etc... All activities generated for vocabulary learning and their distractor are closed to the chosen content.

1 Introduction

The vocabulary learning is a never ending task. Since the languages are vast and continuously evolving. Majority of the tools for vocabulary learning takes the user through a pre defined set of word groups which is constructed into a hierarchy of words in the language.(motivation) The main drawback of this approach are, it takes too long for someone to get a feel of progress, it might not be the order in which the user wants to learn words and the words might not be introduced in the context that the user is not interested in. All the above reasons makes the user feel boring and finally quit the habit of learning in few days or weeks. (content creation) In order to keeps the user of different kind motivated, the developer has to manually/automatically generate lot of content to show the examples to all the words in the language in multiple context. We attempt to overcome these issues of creating tool for vocabulary learning by allowing the user to choose content their interest and we use that to generate the content for the vocabulary learning activities. By this way one could choose to improve vocabularies used in a specific domain with specific context. They could move on to a new area once they complete that. (Indefinite space) So the problem of vocabulary learning scales down from an indefinite space to a finite space.

2 Related works

Details of some of the existing tools goes here.

3 Our application

3.1 Building vocabulary

The vocabulary is built from the user text.

3.1.1 Candidates

All the words in the user text doesn't need to be addressed. For examples function words (the, of, in, on etc...), rare/very less frequent words and improperly parsed words. So the words occurring below the frequency of 10 and the stop words(https://github.com/explosion/spaCy/blob/master/spacy/lang/en/stop_words.py) are eliminated.

Also in-order to differentiate between the same words occurring in the different parts of speech. The words are represented as a tuple of word and its POS tag.

(word, POS tag)

3.1.2 Occurrence map

All the sentences in which the candidate words occur are cleaned and mapped to the corresponding words. So each candidate word is mapped to all the sentences in which it occur and also each sentence is mapped to all the all the candidate words it contains.

3.1.3 Complexity

There are lot of methods to measure the complexity of a word. We choose frequency. In a language, we consider the words that are less frequently used

are as more complex and the words that occurs more frequent are as easier or well known. We use the SUBTLEX-US (**?**), a database of 50 million words containing word frequencies based on English-US movies and TV series subtitles.

$$C_w = \frac{1}{\log_{10}(freq_w)} \tag{1}$$

Where the $freq_w$ in from this database is the average frequency of words per million words.

3.2 Creating structure

3.2.1 Family

We group the words into families, similar to (**?**) work on word families but instead of seven subgroups we form a single group for all types. This is based on the intuition that the user can extrapolate their knowledge of inflections of a word to understand/predict all the forms of a words family by knowing only one or few of them.

3.2.2 Network

Now the families has to co-exist in the space of language(conditioned by the book) as a network like a society. It is a fully connected network with each family with a different affinity to another. The affinity is a measure of contextual similarity between the families. We compute cosine similarity between the mean of word vectors of all the members of the family to similar mean vector representation of another family.

$$V_{F_i} = \frac{1}{n}\sum_{k}^{n} V_{w_k} \tag{2}$$

where n is the number of elements of the family F_i.

$$S_{ij} = \frac{V_{F_i} * V_{F_j}}{\|V_{F_i}\|\|V_{F_j}\|} \tag{3}$$

where, S_{ij} is the cosine similarity between the vector representations of families F_i and F_j.

By this way we create more structure in the space of vocabulary. Which come handy in many situations like activity creation, updating mastery of each vocabulary, analysis etc... This structure helps in reducing the search space by allowing the model to get a better inference about learners level with relatively very less and effective interactions compared to a method of tracking each word in the vocabulary individually.

3.3 Content organization

3.3.1 Books

All the processed data such as vocabulary, families, network, sample sentences are packed into a book instance. This also tracks the meta information such as Title, Author, Genre, Year and Publisher. This help any future user to select the processed book directly. This also could help in comparing the performance of different users on the same book.

3.3.2 Bookshelf

All such processed book are organized in multiple bookshelves specific to each domain similar to the gutenberg project (http://www.gutenberg.org).

3.4 Model

In this work we maintain multiple models to track and update different aspect of the application.

3.4.1 Learner

The learner instance maintain the personal information of the user and tracks the list of instance of books the user has choose to improve vocabulary and the progress in them. This also could maintain overall vocabulary knowledge of the user and customize the activity type, feedback and book suggestions based on individual needs.

3.4.2 Tutor

For each book the user selects, a tutor instance will be created. The main activities of the tutor are to design a learning session, evaluate the performance and track the mastery level of the user w.r.t all the vocabulary in the book. Then again generate a new session based on the updated mastery levels.

Mastery Score: The value ranges between 0 and 1. Initially it is assigned to 0.5 to indicate the uncertainty. Based on the performance of the user it is either increased or decreased by a factor. This approach implicitly capture the un-visited nodes in the network.

Update rule: As we have built a network of families capturing the contextual similarity. We can incorporate this into our update rule to update the mastery scores. When we get some outcome for an activity involving a member from the family F_i.

$$M_j = M_j * (1 + (\alpha * sign * S_{ij})) \tag{4}$$

Where M_i is the mastery of the family i. α is a tunable parameter for the magnitude of an update. $sign \epsilon \{-1, +1\}$ is the direction of the update. It depends on the correctness of user response to the corresponding activity. And S_{ij} is the measure of contextual similarity between the two families.

3.4.3 Session

An instance of session is created by the tutor. A session is a list of word families to be learned and evaluated in a session. The key functions of this model are to deciding the interaction type (teaching, testing, feedback), compose an interaction with all required data and handles the flow and closure of the session.

3.5 Activity

We create two type of activities.

4 Implementation

Acknowledgments