# LT2314 Paper: Automatically Generated Learning Resources for the Web

Jacob Coles

jacob.coles@student.gu.se

*Abstract*— **Generating learning resources by hand takes significant time and expertise. In this paper we investigate a method used for creating a web extension (developed for the Chrome browser), which uses the Sparv API as a remote resource to automatically generate learning exercises from Wikipedia (specifically for prepositions).**

## I. Introduction

It is difficult to find learning resources which interest each individual student. It would be useful to create learning content out of theoretically any text in a target learning language.

In this paper we will investigate the development of automated word- replacement exercises on Wikipedia; a multiple-choice style quis in which the learner may decide which word (out of a list) should fit in a particular slot in each sentence (alongside so-called 'distractors').

There are a few challenges for developing such a system. On the side of language resources, there needs to be a way of correctly identifying part-of-speech (POS) tags for each word, as well as having feasible alternatives to each of these words, with which the learner could mistake the correct word.

Additionally it would be ideal to develop the exercises based on what errors are common for non-native speakers.

## II. Research: Language Resources

If we were to develop this system from scratch, that is, not utilising any premade models or services, it would be necessary to have the appropriate annotated corpora. There are two primary sub-tasks that could require the use of annotated texts.

Firstly for this task, it may be necessary to be able to tokenise and tag texts with POS tags, in order to be able to correctly identify what word-class a word belongs to, and develop learning exercises based on this. Having a token-by-token annotated dataset in the target language is thus required. The two most updated datasets that exist are the Stockholm-Umeå corpus (SUC) [1] and Talbanken [2]. The former contains over a million tokens, whereas the latter contains just under 100 000. There is a difference in that the SUC only contains lemma, PoS and named entity tags, whereas Talbanken contains entire treebanks; it contains dependency tags.

For the scope of developing basic learning exercises (in which a word class needs to be determined so that it can be replaced with an alternative), it may not be necessary to know the dependencies. Dependencies may contain useful information help decide what words are actually feasible replacements for other words. For the scope of this task, it is unlikely that this will be utilised, but it could be a useful consideration.

Secondly, it is useful to be aware of what errors non-native speakers of a language make. Non-native speakers tend to make different errors than native speakers. If we generate distractors (incorrect alternatives in a multiple-choice quiz) which are not likely mistakes for non-native speakers to make, then it is not a very effective learning resource; therefore it is ideal to minimise the kinds of distractors which maximise errors (made by the language learner). Perhaps adapting already-made tools such as Lärka could be promising [3], or even using the data which they have to develop custom tools. It also may be of interest to consider the work in 'A Readable Read: Automatic Assessment of Language Learning Materials based on Linguistic Complexity' [4]. In this paper the COCTAILL corpus is used to evaluate the level of a text (CEFR scale). Using this type of analysis could be a pathway to better evaluate the types of constructions that are appropriate and/or difficult for language learners of any particular level.

## III. Research: Language Technology Resources

It was required to find or develop a resource to parse sentences and extract POS tags (since we are not relying on manually annotated datasets for generating learning resources).

### *A. Sparv REST API*

The Sparv API which is developed by Språkbanken is a web-accessible REST API which enables us to parse text quite simply as a POST request. A request for parsing only requires:

- Target text (in the POST body)
- Options (JSON in the URL)

The major limitation with this system is that it is quite slow, but unpredictably so. Sending a request and returning text from a parsed webpage can take anywhere between 5 seconds and 1 minute, seemingly without reason.
The response object (information retrieved from Sparv) contains an XML format which itself needs to be parsed afterwards to extract the tags, and in this way is perhaps not the most friendly for development.

### *B. NLTK*

NLTK, a package developed at Stanford University, has the ability to create POS tags from sentences. Out-of-the-box, the system only has support for English, so that leaves two options with NLTK.

The first option is performing a word-for-word translation into English (to maintain the word-for-word correspondence), and then running the POS tagger (in English). It is uncertain if it would perform well, as, for example, a preposition such as 'in i' may be phrased as the single word 'inside' in English. Additionally, if the word-for-word correspondence is maintained, it is likely that the word-sense of a word may be incorrectly translated.

A second, perhaps more suitable option, would be to create a model in NLTK from scratch. There exist corpora which could be used for developing such system. Namely the Talbanken developed by Språkbanken is a resource containing 96 346 tokens. Using this set, which contains POS tags for all tokens, it would be possible to create a basic n-gram (statistical) model. Such a model could use a combination of unigrams, bigrams and trigrams to estimate the most likely POS tag for each word.

### *C. Stagger*

Stagger is a POS tagger developed at Stockholm University, and is based on the the Stockholm-Umeå corpus (SUC) [5]. It achieves over 96% accuracy across all word-classes and is a strongly viable option.

#### *iv) Custom Machine Learning Model*

Using one of the aforementioned corpora, it may be possible to create a POS tagger which utilises a machine learning model. A relatively basic model could use an LSTM with word embeddings to predict the class of each word, by looking at the context of each word[6]. The bidirectional-LSTM model has been tested (in English) to have an accuracy of over 96% for word-class classification (POS tagging) over all word classes.

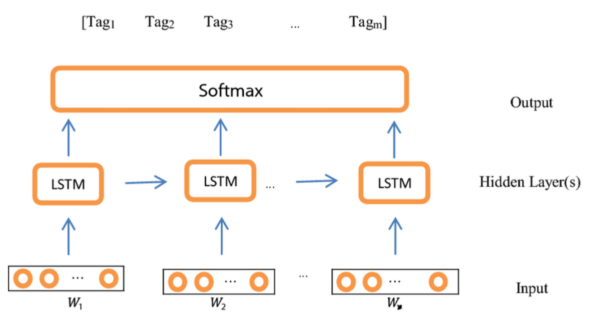


Fig. 1: LSTM seq-to-seq model [7]

There are many so-called 'seq-to-seq' (sequence to sequence) which could be worth consideration. More complex models using autoencoders, transformers or BERT [8] could improve performance even more, though this would be a new research question in itself and goes far beyond the scope of this project.

## IV. Research: Technology Resources

A front-end system which can interface with the back-end resources (POS tagger etc.) is to be developed. Following are the options.

### *A.* VIEW Extension

VIEW is a language learning system designed to create language learning exercises for language learners. This system is currently available to work with English, Russian, German and Spanish, and has several different exercises for different word classes. Working with a system that is largely complete in would be extremely useful.

Unfortunately, it did not seem as though their code was open-source, nor were there any instructions about how to develop for it.

### *B.* Chrome Extension

A chrome extension was seen as a viable alternative to the VIEW extension (though a custom- written browser extension could have been made for any web-browser). This way, the end-to-end system could be developed from the available tools. Presumably, this type of system would be similar to the VIEW tool. In ‘An Approach to the Use and Automatic Generation of Web-Based Learning Materials’ [9] a novel method of deconstructing a website and reconstructing it is presented, which is potentially a useful method here.

## V. Project Outline and Limitations

For the system we have the following design parameters and restrictions to limit and frame the scope of the project:

1) This system was designed to work only on Wikipedia as:
    - The structure is somewhat predictable between different pages
    - There is significant variety in the content available on Wikipedia
2) It will only create learning exercises for prepositions as:
    - The set of common prepositions (especially those relevant to early learners of Swedish) is quite small. This will allow them to be manually coded in.
    - When it comes to other word classes, it would be difficult to determine what words or grammatical forms of words are replaceable with other words. (That is, knowing what words may be commonly mistaken by a language learner.
    - Knowing what ‘language level’ a word belongs to is more of a challenge with other word classes (CEFR levels).

## VI. Project Design

The final design of the system was determined by the time constraints given. A relatively simple system was developed in which the HTML of the target Wikipedia page is sent from a Chrome extension’s javascript-script to a Python web-service (written for this project) over a POST request. The Python web-service then retrieves the POS tags for the text from the Wikipedia article from the Sparv API. The python service then reconstructs the HTML of the original Wikipedia article with the learning exercise inserted, and sends it back to the Chrome extension to be displayed to the user. Following is a slightly more detailed explanation.

The frontend of the system consists of the chrome extension which can be loaded into the browser through the extensions menu. Once it is loaded, it is possible to open it (by clicking the icon in the menu-bar), and then click a big button to generate the learning exercise.

Upon clicking this button, a javascript function is run, which gets the HTML for the body of the article via the `document.querySelector()` function. This (HTML) is then sent via an HTTP POST to an other remote service (written also for this project) written in Python, and built on the Flask web-service. This was done to simulate the design as if it were a remote service, though it would be possible to do the logic directly in the extension itself.

In the python web-service, it is necessary to decompose the HTML into segments of strings so it is ‘friendly’ to the backend service (Sparv) to which the text will be sent.

These strings (text from the Wikipedia article) are sent over HTTP POST to the Sparv REST API. The response of XML format needs to be decomposed by the Python web-service, in which only the relevant data retrieved and stored in ordered lists.

The HTML is recomposed by iterating over these lists. When the prepositions (tagged as ‘PP’) are found, the (single) preposition is replaced by the multi-choice drop-down box, corresponding to a set of randomised prepositions (corresponding to the options/distractors which the user can select from); one of which is the correct answer. The distractors in this case are indeed selected randomly from a manually created list of the most common prepositions in Swedish.

Each of these drop-down box elements also feature a small javascript script to enable the dropdown box to change colour depending on whether the user selected the correct answer or not (red for incorrect, green for correct).

A flow chart showing the design of this is as follows:

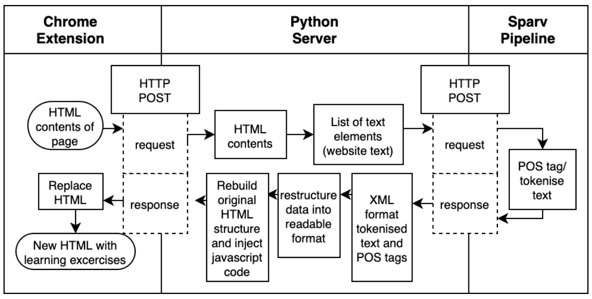
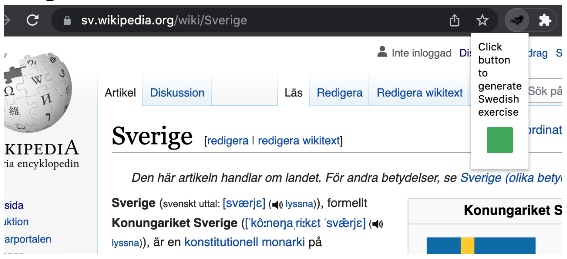

Chrome Extension
Python Server
Sparv Pipeline
HTML contents of page
HTTP POST
request
HTML contents
List of text elements (website text)
HTTP POST
request
POS tag/ tokenise text
Replace HTML
response
Rebuild original HTML structure and inject javascript code
restructure data into readable format
XML format tokenised text and POS tags
response
New HTML with learning excercises

Fig. 2: System design

## VII. Results

When the green box is pressed, it initiates the learning exercise.



sv.wikipedia.org/wiki/Sverige

Inte inloggad

Click button to generate Swedish exercise

Artikel Diskussion Läs Redigera Redigera wikitext Sök på

**Sverige** [redigera | redigera wikitext]

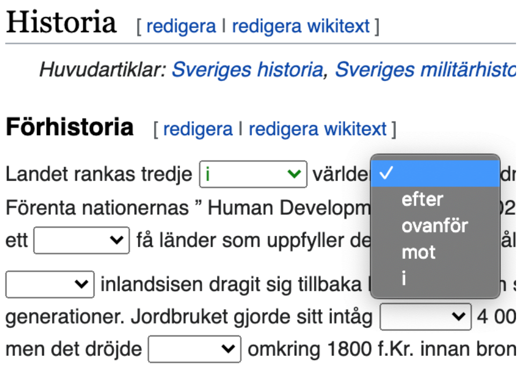
*Den här artikeln handlar om landet. För andra betydelser, se Sverige (olika bety*

**Sverige** (svenskt uttal: [sværjɛ] (lyssna)), formellt **Konungariket Sverige** ([ˈkôːnɛŋaˌriːkɛt ˈsvæ̌rjɛ] (lyssna)), är en konstitutionell monarki på

Konungariket S

Fig. 3: Basic interface of Chrome extension

The exercises consist of dropdown boxes with four options:



**Historia** [ redigera | redigera wikitext ]

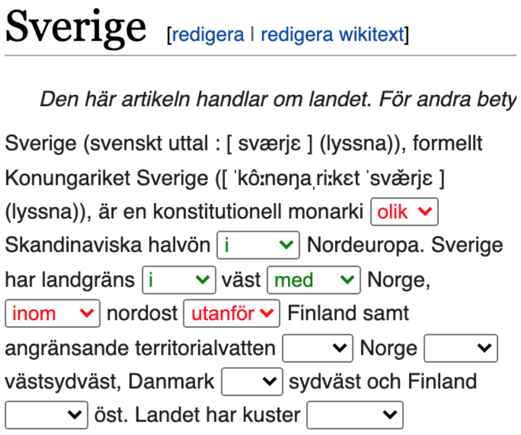
*Huvudartiklar: Sveriges historia, Sveriges militärhistc*

**Förhistoria** [ redigera | redigera wikitext ]

Landet rankas tredje [i] världe
Förenta nationernas ” Human Developm
ett [ ] få länder som uppfyller de
[ ] inlandsisen dragit sig tillbaka
generationer. Jordbruket gjorde sitt intåg [ ] 4 00
men det dröjde [ ] omkring 1800 f.Kr. innan bron

Fig 4. Dropdown box with four alternatives

Here we see that the answers change to green for a correct answer, red for an incorrect answer and are blank for the unanswered slots.



**Sverige** [redigera | redigera wikitext]

*Den här artikeln handlar om landet. För andra bety*

Sverige (svenskt uttal : [ sværjɛ ] (lyssna)), formellt Konungariket Sverige ([ ˈkôːnɛŋaˌriːkɛt ˈsvæ̌rjɛ ] (lyssna)), är en konstitutionell monarki [olik] Skandinaviska halvön [i] Nordeuropa. Sverige har landgräns [i] väst [med] Norge, [inom] nordost [utanför] Finland samt angränsande territorialvatten [ ] Norge [ ] västsydväst, Danmark [ ] sydväst och Finland [ ] öst. Landet har kuster [ ]

Fig. 5: Example of learning exercise

## VIII. Conclusions

The system performs according to the expectations set out at the beginning of the project. A web-app/extension has been developed for the Chrome browser which generates learning exercises on Wikipedia, for learning the correct prepositions. The system utilises the Sparv resource to retrieve POS tags on the target Wikipedia article in order to do this.

### Future Work

The current system only implements multiple-choice options for distractors of prepositions. It may be interesting to develop a system which works with other word classes. In the case of developing (multiple-choice) alternatives for verbs and nouns, it is more relevant to learners to have alternatives which are different word-forms of the same word, instead of options of alternative words. It would be useful for the system to have some kind of ‘awareness’ of the types of errors that could be made for a particular word class, especially within its particular context.

Using resources developed under the Lärka project may be of use, and it may be interesting to continue looking at what research is being done in this field, especially with respect to machine learning.

Developing some other types of exercises which are not multiple choice could be of interest for learners.

Currently the system has a rendering issue which can cause a paragraph (only on some articles) to disappear. In future, it would be useful to design a system which can split the structure of the website in a better, more generalised way, so that no interface elements disappear, and perhaps so that it can work on any website.

## References

[1] Bjursäter, U., 2022. Stockholm—Umeå Corpus (SUC). [online] Ling.su.se. Available at: <https://www.ling.su.se/english/nlp/corpora-and-resources/suc/stockholm-ume%C3%A5-corpus-suc-1.14045> [Accessed 8 January 2022].
[2] Spraakbanken.gu.se. 2022. TalbankenSBX | Språkbanken Text. [online] Available at: <https://spraakbanken.gu.se/en/resources/talbanken> [Accessed 6 January 2022].
[3] Volodina, E., Pilán, I., Borin, L., & Lindström, T. (2014). A flexible language learning platform based on language resources and web services. In Proceedings of LREC 2014, Reykjavik, Iceland (pp. 3973–3978).
[4] Pilán, I., V., S., & Volodina, E. (2015). A Readable Read: Automatic Assessment of Language Learning Materials based on Linguistic Complexity.
[5] Östling, R. Stagger: an Open-Source Part of Speech Tagger for Swedish. (2013). Northern European Journal of Language Technology, 2013, Vol. 3, Article 1. (pp 1–18)
[6] Peilu Wang, Yao Qian, Frank K. Soong, Lei He, & Hai Zhao. (2015). Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network.
[7] Besharati, S., Veisi, H., Darzi, A., & Hosseini Saravani, S. (2021). A hybrid statistical and deep learning based technique for Persian part of speech tagging. Iran Journal of Computer Science, 4.
[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, & Kristina Toutanova. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
[9] Fernández, V., Herranz, S., Pérez-Iglesias, J., Urquiza-Fuentes, J., & Velázquez-Iturbide, J. (2004). An Approach to the Use and Automatic Generation of Web-Based Learning Materials. (pp. 201-208).