**Building a pipeline for NLP in Hebrew**

SCE – college of engineering

Software Engineering Department

Be'er Sheva

Project number: [BS-SE-21-93]

Authors: Tshobruzki Michal

Alfi Tal

Academic Facilitator: Dr. Schwartz-Chassidim Hadas, Dr. Litvak Marina

Professional Facilitator: Dr. Schwartz-Chassidim Hadas, Dr. Litvak Marina

Table of contents

[Abstract 4](#_Toc60513827)

[1. Literature review 5](#_Toc60513828)

[1.1. Web accessibility 5](#_Toc60513829)

[1.1.1. Cognitive accessibility 5](#_Toc60513830)

[1.1.2. Web Content Accessibility 5](#_Toc60513831)

[1.2. Hebrew Text Preprocessing 6](#_Toc60513832)

[1.2.1. Multi-word expressions 7](#_Toc60513833)

[1.2.2. The Linguistic Challenge 7](#_Toc60513834)

[1.2.3. The Architectural Design 7](#_Toc60513835)

[1.2.4. The Annotation Scheme of YAP 8](#_Toc60513836)

[1.2.5. Named *Entity* Recognition (NER) 9](#_Toc60513837)

[1.2.5.1. The difficulties with NER in Hebrew 9](#_Toc60513838)

[1.2.6. Coreference Resolution 9](#_Toc60513839)

[1.3. References 10](#_Toc60513840)

[2. Project specification 11](#_Toc60513841)

[2.1. General 11](#_Toc60513842)

[2.2. Major customers 11](#_Toc60513843)

[2.3. Goals and objectives 11](#_Toc60513844)

[2.4. Problems 11](#_Toc60513845)

[2.5. Annual work plan 12](#_Toc60513846)

[Staff meetings 12](#_Toc60513847)

[2.6. Feasibility and cost / Utility 12](#_Toc60513848)

[System feasibility 12](#_Toc60513849)

[2.7. Outcomes 12](#_Toc60513850)

[2.8. Gantt - work plan 13](#_Toc60513851)

[3. Implementation 13](#_Toc60513852)

[3.1. Existing systems 13](#_Toc60513853)

[3.2. The new system 13](#_Toc60513854)

[3.3. Method 13](#_Toc60513855)

[3.4. System architecture 14](#_Toc60513856)

[3.4.1. Architecture diagram 15](#_Toc60513857)

[3.5. Technologies 15](#_Toc60513858)

[3.6. SRS- Software Requirements Specification 15](#_Toc60513859)

[3.7. Processes Michal 16](#_Toc60513860)

[3.8. Reports 16](#_Toc60513861)

[3.9. Volumes, loads and performance 16](#_Toc60513862)

[3.10. Glossary of Terms 16](#_Toc60513863)

[3.11. Charts 17](#_Toc60513864)

[4. Technology and infrastructure 17](#_Toc60513865)

[5. Manual testing on the too 17](#_Toc60513866)

[6. Risk management Michal 18](#_Toc60513867)

# Abstract

For languages with simple morphology, such as English, automatic annotation pipelines such as spaCy or Stanford’s CoreNLP successfully serve artificial intelligence (AI) or data science (DS) projects in academia and the industry. For many morphologically rich languages (MRLs), similar pipelines show suboptimal performance that limits their applicability for text analysis in research and commercial use. The suboptimal performance is mainly due to errors in early morphological disambiguation decisions, which cannot be recovered later in the pipeline, yielding incoherent annotations overall.

Our project deals with finding a solution for NLP in Hebrew to make texts accessible to populations with cognitive disabilities.

We will achieve this goal by creating a Pipeline that parses text and executes file for the next step.

Today, except for the ONLP tool, there is no other tools that perform multiple stages of text preprocessing and morphological analysis of Hebrew texts.

In our project we plan to use ONLP for text preprocessing and early stages of morphological analysis, which will serve us in further stages of the text simplification pipeline.

# Literature review

## Web accessibility

Access to information in digital media enables efficient consumption of information on the Internet in accordance with the user's digital literacy capability. Digital literacy is now defined as the ability to access and operate a wide range of activities and sources of information through digital tools. Digital literacy relies on skills such as the ability to navigate in cyberspace, dealing with a wide range of information from diverse sources, the ability to decipher and process information presented in various ways (textual, visual, auditory (and by various means) text, image, video, etc., branching thinking, information processing Multi-channel and critical thinking, meaning that digital literacy relies heavily on advanced cognitive abilities and complex thought processes. (Kennedy, 2011)

### Cognitive accessibility

Cognitive accessibility refers to the simplification and adaptation of the physical, virtual, and human environment so that a clear, simple, and unambiguous environment is created. Accessible environment (clear and understandable) includes easy structure for navigation and orientation, understandable and clear information, easy-to-use technology, and simple and understandable processes. For certain populations, including people with cognitive disabilities, this accessibility is an essential condition for participation in the various living environments in which we all take part. And from the accessibility of language and content in particular, people who belong to other populations, such as new immigrants, immigrants, the elderly, children, and people who have difficulty reading and writing for other reasons may also benefit. (Richardson, 2006)

### Web Content Accessibility

As of October 25, 2015 - any application or new content that will be posted on a website (old or new) - must be Immediately accessible from the moment of investment to use the public.

The population of people with disabilities is defined as the largest minority group in the world. Its numbers over a billion people and makes up between 10% and 15% of the world's population. According to the Central Bureau of Statistics, there are currently more than 1,600,000 people in Israel with disabilities (physical, sensory, mental, emotional and disability as a result of chronic illness), who are over 20% of the total population in the country.

**The Equal Rights for Persons with Disabilities[[1]](#footnote-1)** Act, 1998 (hereinafter: the "Law"), is intended to anchor the right of a person with a disability to equal and active participation in society in all areas of life.

## Hebrew Text Preprocessing

The process of linguistic accessibility on websites is carried out as follows: extracting the text from the website, performing preprocessing on the text, and finally performing a linguistic simplification to the processed text. Our project focuses on the pre-processing stage of the text, which is done in the following order:

Sentence splitting

Morphological Analysis

POS tagging

Tokenization

NER

Syntactic Parsing

Co-reference Resolution

Text Simplification

Execution flow

Simplified Text

Pass I:

Text Summarization,

NER replacement, etc.

Pass II:

Recommendations for Sentence Compression, Passive voice replacement, etc.

Figure1: The execution flow for the process of Text preprocessing

As shown in figure 1, the next stage of the process is text simplification. Text simplification is the process of reducing the grammatical complexity of a text, while retaining its information content and meaning. The aim of text simplification is to make text easier to comprehend for human readers, or process by programs. (Siddharthan, 2006)

During the execution flow we use some tools which are already available, such as ONLP and "Yet Another Parser" (YAP) that allows morpho-syntactic parser for morphological and syntactic analysis of Hebrew Texts.

Most of the work of linguistic analysis assumes that the lexical items (i.e. the linguistic expressions) are fully observed and correspond to the analysis tree that describes them. An example is shown below in Figure 2.

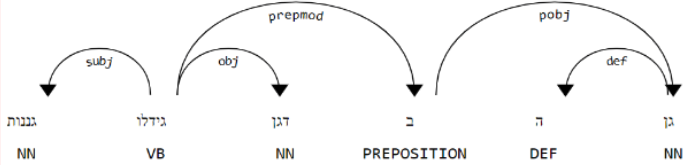


Figure 2 : Given a sentence, there is a parsing tree that represents its linguistic context as follows

In the English the case is really such for the most part, since given a sentence, its decomposition to give strings (i.e. the process of tokenization) is simpler and more convenient (less important why). In Semitic language this is not the case, for example in Hebrew, there are many words whose function are prepositions and therefore there is a challenge when building a surgical tree. Example: in the word 'בצל'. Is it meant to be a 'בצל'? Or is the intention in 'ב-צל'. (Goldberg, 2011, June)

The tools that have been developed to date use Pipeline which allows word processing in languages like English. The main reason for this sub-optimal performance on Semitic languages is that the pipeline design inherent in these frameworks is inappropriate for languages that exhibit extreme morphological ambiguity in their input stream. (Tsarfaty, 2019)

The difficulties in the Hebrew language can be classified into several categories:

### Multi-word expressions

MWEs are lexical units that consist of more than one word. They tend to be semantically idiosyncratic. MWEs are challenging for grammatical theories and grammar development, but as they account for approximately half of the entries in the lexicon, incorporating them into grammars is important. Moreover, identifying MWEs is important for natural language processing applications – if MWEs are not identified as such, that will probably cause problems further down the processing pipeline. (Sheinfux, 2015)

### The Linguistic Challenge

Hebrew poses computational processing challenges typical of Semitic languages. Hebrew orthography uses optional diacritics, and its morphology uses both root-pattern and affixational mechanisms. Hebrew inflects for gender, number, person, state, tense, and definiteness. Furthermore, Hebrew has a set of attachable clitics that are typically separate words in English, e.g., conjunctions (such as +ו w+ ‘and’), prepositions (such as ב b ‘in’), the definite article (ה h ‘the’), or pronouns (such as הם hm ‘their’). (Singh, 2012)

Tool called YAP allows to perform the following stages of the text pre-processing in Hebrew.

### The Architectural Design

The core of ONLP is YAP (Yet Another Parser), a morpho-syntactic parser for morphological and syntactic analysis of Hebrew Texts. YAP reimplements and extends the structure-prediction framework of (Zhang, 2011).

Each lattice arc is a tuple specifying the start-index, end-index, the form of the segment, its part-of-speech, lemma, features, and the index of the raw token the arc originated from. An arc in the lattice can serve as a node in a syntactic dependency tree. Each contiguous path in the lattice presents one valid morphological segmentation of the sentence, for which a dependency tree can be assigned, as in Figure 3. For each path in the lattice, there is an exponential number of dependency trees that are potentially applicable.

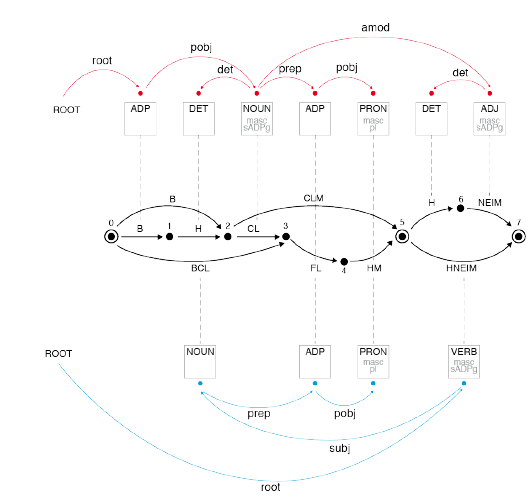


Figure 3: The Joint Morpho-Syntactic Search-Space. Lattice paths vary in length. Each lattice path can be assigned an exponential number of dependency trees. (Tsarfaty, 2019)

We refer to the task of selecting the most likely lattice-path as Morphological Disambiguation (MD), and to the task of selecting the most likely dependency tree for a given path as Dependency Parsing (DEP). For an input sentence x, our goal is to jointly predict a single pair of MD(x) and DEP(x) that are consistent with one another, and form the most-likely analysis of the sentence. The MD component is the transition-based morphological parser of (More, 2016), which is formally based on the structure prediction framework of (Zhang, 2011). MD accepts a sentence lattice MA(x) as input and delivers a selected sequence of arcs (morphemes) MD(x) as output. (Tsarfaty, 2019)

### The Annotation Scheme of YAP

The process performed on each word classifies it into several categories:

* Morphological Segmentation
* Part-of-Speech (POS) Tags - [List of all the used POS](https://nlp.biu.ac.il/~rtsarfaty/onlp/hebrew/postags)
* Morphological Features
* Lemmas
* Dependency Tree
* Lattices

Figure 4 presents an example of the Lattice Representation and Parsing. The lattice for the two-token sentence “בצלם הנעים”. Each lattice arc corresponds to a lexical item. (Goldberg, 2011, June)

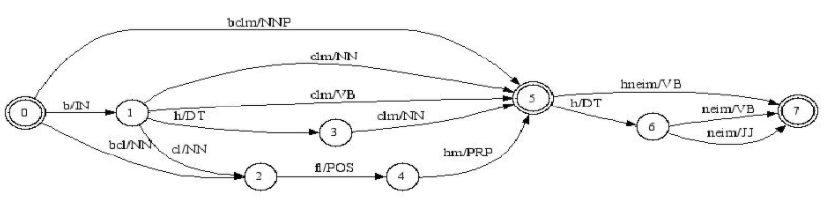


Figure 4: Lattice representation of the sentence "בצלם הנעים". (Goldberg, 2011, June)

### Named Entity Recognition (NER)

NER is a fundamental NLP task, commonly formulated as classification over a sequence of tokens. Morphologically-Rich Languages (MRLs) pose a challenge to this basic formulation, as the boundaries of Named Entities do not coincide with token boundaries, rather, they respect morphological boundaries. To address NER in MRLs we then need to answer two fundamental modeling questions: (1) What should be the basic units to be identified and labeled, are they token-based or morpheme-based? and (2) How can morphological units be encoded and accurately obtained in realistic (non-gold) scenarios? (Bareket, 2020)

#### The difficulties with NER in Hebrew

In MRLs (Morphologically-Rich Languages), words are internally complex, and word boundaries do not generally coincide with the boundaries of prominent syntactic and semantic units. This fact has critical ramifications for sequence labeling tasks, and for NER in MRLs in particular.

While morphological decomposition is required to obtain accurate NE boundaries. the morphological decomposition of surface token in some MRLs is subject to extreme morphological ambiguity. This means that the sequence of morphemes composing a token is not deterministically recoverable from its character sequence and is not known in advance. This ambiguity gets further magnified by the fact that MRLs such as Hebrew, Arabic, and other Semitic languages lack capitalization altogether, and the script of these languages also suppresses vowels (diacritics). This means that for every surface space-delimited token, there are many conceivable readings which impose different sets of NE boundaries. (Bareket, 2020)

### Coreference Resolution

The literature on coreference resolution has traditionally divided the task into twodifferent settings, addressing the task at either the Within-document (WD) or Cross document (CD) level. Each setting has presented different challenges, model design choices, and historically different evaluation practices. In CD coreference resolution, the instances consist of multiple documents, each authored independently, without any inherent linear ordering between them. As a result, co-referring expressions across documents are often lexically divergent while lexically similar expressions may refer to different concepts. Table 1 shows example documents discussing similar, yet distinct, events (two different nominations of a US Surgeon General) with overlapping participants (“President Barack Obama”) and event triggers (“name”). Leveraging accurate CD coreference models seems particularly appealing for applications that merge information across texts, which have been gaining growing attention recently, such as multi-document summarization and multi-hop question answering. (Cattan, 2020)

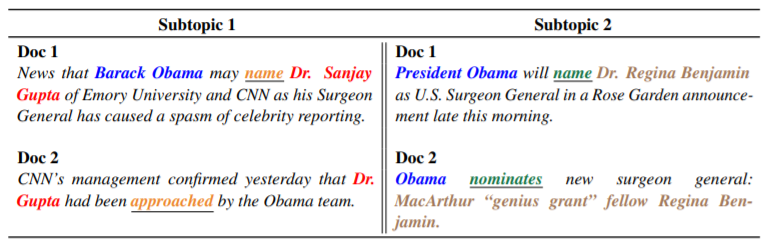


Table 1: The underlined words represent events, same color represents a coreference cluster. Different documents describe the same event using different words (e.g. name, approached). The two subtopics present a challenging case of ambiguity between the two different nominations. (Cattan, 2020)

## References

Bareket, D' &'. (2020). Neural Modeling for Named Entities and Morphology. *arXiv preprint arXiv:2007.15620v1*, 1-22.

Cattan, A' E'. (2020). Streamlining Cross-Document Coreference Resolution: Evaluation and Modeling. *arXiv preprint arXiv:2009.11032*, 1-12.

Goldberg, Y' &'. (2011, June). Joint Hebrew segmentation and parsing using a PCFGLA lattice parser. *Joint Hebrew segmentation and parsing using a PCFGLA lattice parser. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* , 704-709.

Kennedy, H' E'. (2011). Can the web be made accessible for people with intellectual disabilities. *The Information Society*, 29-39.

More, A' &'. (2016). Data-driven morphological analysis and disambiguation for morphologically rich languages and universal dependencies. *In Proceedings of COLING*, 337-348.

Richardson, H' A'. (2006). Enabling or disabling technologies? A critical approach to web accessibility. *Information Technology & People*, 203-218.

Sheinfux, L' H'. (2015). Hebrew verbal multi-word expressions. *In Proceedings of the 22nd International Conference on Head-Driven Phrase Structure Grammar, Nanyang Technological University (NTU)*, 122-135.

Siddharthan, A'. (2006). Syntactic simplification and text cohesion. *Research on Language and Computation*, 77-109.

Singh, N' &'. (2012). Hebrew morphological preprocessing for statistical machine translation‏. *In Proceedings of the 16th European Association for Machine Translation Conference*, 43-50.

Tsarfaty, R' S'. (2019). What’s Wrong with Hebrew NLP? *EMNLP-IJCNLP*, 259-264.

Zhang, Y' &'. (2011). Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 105-151.

# Project specification

## General

Today, for the Hebrew language, there is no tool that allows for linguistic simplification because of the complexity of the language.

The Hebrew language is a Semitic language and therefore it is difficult to develop a linguistic simplifier that will receive a text and simplify it for people with cognitive disabilities. The difficulty stems from the fact that the Hebrew language has many rules unlike Latin languages and therefore tools that are suitable for languages like English are not suitable for the Hebrew language.

## Major customers

A team of programmers will use artificial intelligence as part of their final project and perform a linguistic simplification of the product of our project. At the end, they will open a plug-in for the Chrome browser in order to allow people with cognitive disabilities to receive linguistically simplified texts that are easy to read on websites so that they can be independent on the web.

## Goals and objectives

The main goal of our project is to get text in Hebrew and pass in a pipeline of preprocessing that includes: Morphological Segmentation, Part-of-Speech (POS) Tags, Morphological Features, Lemmas, Dependency Tree, Lattices. After those we will take the text to tool that makes Named Entity Recognition (NER) and get as output named entity mentions.

## Problems

|  |  |  |
| --- | --- | --- |
| Problem | Cause | Effect |
| Recognize complex words | In Hebrew there are words that can be simplified to a preposition and a basic word or a complex word. For example the word "מענק" can be interpreted as grant and can be interpreted as a pair of words "מ"-"ענק" from-huge | changes the essence of the sentence |
| The result of the pre-processing is not accurate | Insufficient in-depth analysis of difficult words and words in foreign languages written in Hebrew. | Inaccurate results |
| The NER sub-system will not be developed at specified time | The complexity of building such system might take a lot of time and resources. | Impairment of the linguistic simplification process |

## Annual work plan

Connection to YAP API and manual F score testing

Creating a gentile interface

Creating database of difficult word to simplification

Get text and replace difficult word before the preprocessing.

Add NER process

Create a final product that contains all the results of preprocessing

### Staff meetings

* Once a week with the development team

## Feasibility and cost / Utility

### System feasibility

In the Hebrew language, there is currently no tool that can perform linguistic simplification as in Latin languages. Our project will advance the theme and be a significant step in promoting tool creation.

## Outcomes

End of the first development phase – until March 2021.

End of the second development phase – until June 2021.

Operate the system – July 2021.

## Gantt - work plan

# Implementation

## Existing systems

Currently, there are no linguistic simplification systems in Hebrew.

The linguistic simplification that exists today is done manually by experts, it is an expensive process that takes a long time, one of the reasons is: there are not many people engaged in this craft.

## The new system

Creating a system that receives texts in the Hebrew language, performs a preliminary process on it and transfers the data to the next stage in which machine learning will be performed for the purpose of linguistic simplification.

## Method

To perform the linguistic simplification, the following process is required to be implemented:

1. Receive text from a website.
2. Decomposition of the text into sentences.
3. Replacing complex words, with simpler words from data set.
4. Sending the text to a YAP tool (which parses sentences in Hebrew and tags words).
5. Obtaining an answer in the form of JSON in which each word is classified according to the following criteria: morphological decipherment, parts of speech, morphological features, die, hanging tree, Lattices, NER.
6. Displays the data conveniently to the user with the GUI opened as well.
7. Transfer the data to the next tool, which will perform the linguistic simplification itself.

Tokenization

Sentence splitting

POS tagging

Morphological

NER

Syntactic Parsing

Co-reference Resolution

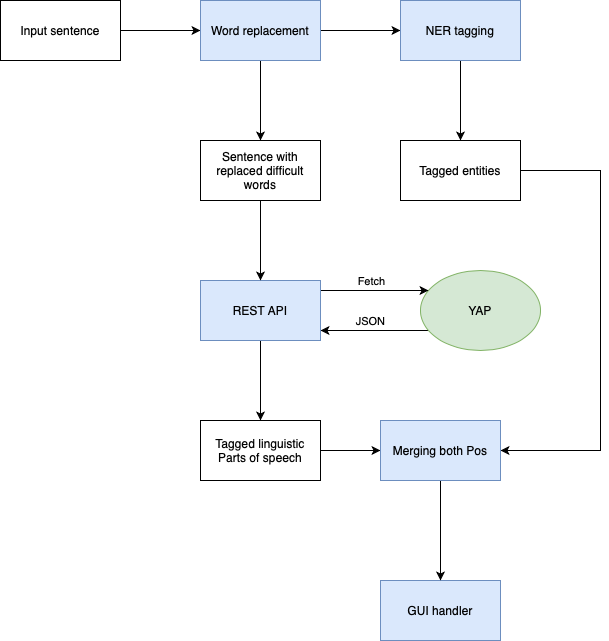
Text Simplification

## System architecture

The system consists of several components, each component performs a different action on the source text:

* Word replace - exchanges words that are complex to identify in the YAP tool.
* Request handler - Sending the sentence to the YAP tool.
* Name Entity Recognition – tagging words that are entities.
* GUI handler - Displays some of the data retrieved from the server clearly to the user.
* Merging results – Merge the results from both Name Entity Recognition and YAP.
* File writer - Creates a csv file that contains the rest of the information: Lattice, dependency tree.

### Architecture diagram



## Technologies

* Developed language- Python3, OOP methodology
* GUI- Python3, Tkinter library
* YAP - Developed in Go as open source
* Development environment- PyCharm

## SRS- Software Requirements Specification

* Creating a graphical user interface for the preliminary process.
* Creating data base of complicated words in Hebrew.
* Replace complex words in the text before transferring them to the YAP tool.
* YAP does not recognize all the words correct. To make a reliable preprocessing, we will create data base of words as: "און-ליין" (in YAP will be: "און ל ה יין" in our data base: "בזמן הגלישה באתר") and "ותיק" (in YAP will be: "ו תיק" in our data base: "מבוגר").
* Obtaining an answer in the form of JSON in which each word is classified according to the following criteria: morphological decipherment, parts of speech, morphological features, die, hanging tree, Lattices, NER.
* Name Entity Recognition- obtaining an answer in the form of JSON in which each word is classified according to its name entity.

## Processes Michal

|  |  |
| --- | --- |
| Process | Description |
| Obtain file in which every word is classified | Replace difficult words in entered text, split into sentences, transfer to YAP pipeline. Output- JSON file |
| Name Entity Recognition | Enter text into NER program and get the words with their name entity. |

## Reports

After sending a request to the server, the YAP processes the information received and returns a JSON object containing the following information:

* Tokenized text - the original sentence after being broken down into words.
* Segmented text - the sentence after division into significant units, such as words, prepositions
* Lemmas - the words that make up the sentence, but in their basic speech form, without verbal inflections, in the singular plural.
* Dependency Tree - A tree represented by a table, indicating which word is linguistically dependent on which word.
* Lattice - Tagging all possible combinations for each word (along with its possible inflections) in the categories - POS, tense, lemma, etc.

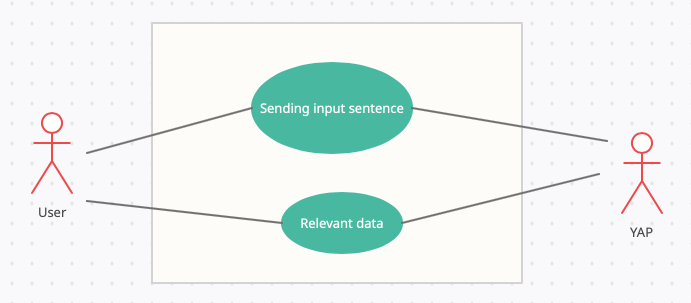
## Volumes, loads and performance

Our system uses few interfaces with REST API. Switching data between different interfaces can take a long time and depends on the network.

## Glossary of Terms

* Pipeline - A set of data processing elements connected in series, where the output of one element is the input of the next one. In the
* Preprocessing - A preliminary processing of data to prepare it for the primary processing or for further analysis. The term can be applied to any first or preparatory processing stage when there are several steps required to prepare data for the user.
* YAP – A system that provides a complete automatic, morphological, and syntactic annotation for Hebrew texts.
* NER - named-entity recognition, is a subtask of [information extraction](https://en.wikipedia.org/wiki/Information_extraction) that seeks to locate and classify [named entities](https://en.wikipedia.org/wiki/Named_entity) mentioned in [a](https://en.wikipedia.org/wiki/Unstructured_data) text, into pre-defined categories , such as person names, organizations, locations, [medical codes](https://en.wikipedia.org/wiki/Medical_classification), time expressions, quantities, monetary values, percentages, etc.
* Hard to tag words ('difficult words') – Words collected by us due to observations made during the work with YAP. It also contains words that are: slang, Proverbs in high language, Phrases in English (such as 'on-line' which is written in Hebrew און-ליין).

## Charts

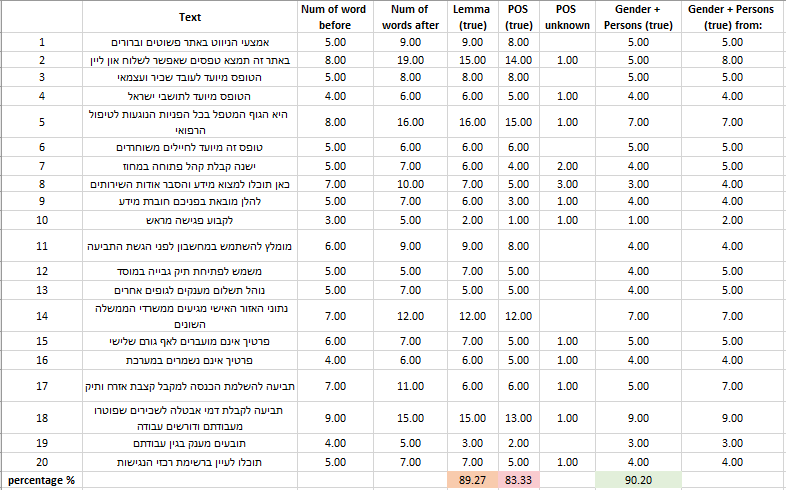


# Technology and infrastructure

In our project we receive from the user a sentence that we want to break down into different parts of the speech and labeling them according to their linguistic context.

To achieve this goal, we built a GUI where the user enters a text. The GUI then pass the text through a pipeline that performs the following actions:

Replacing complex words with simple words to label, split the text into sentences. Then, with the help of the REST API that we build, the GUI sends a request to the YAP tool and the NER system. We then merge the results with the merge tool that we develop as well. The result will be export into a csv file.

1. Manual testing on the too

# Risk management Michal

|  |  |  |  |
| --- | --- | --- | --- |
| Probability | Damage | Cause | Risk |
| 60% | 5 | To perform the full pipeline, we will use different interfaces that written in different languages | Inconsistency between interfaces used |
| 5% | 5 | Network glitches, or intentional removal of the service from the website | The YAP API, may be disconnected from the web. |
| 26% | 4 | * Insufficient in-depth analysis of difficult words. * words in foreign languages written in Hebrew. | The result of the pre-processing is not accurate |
| 60% | 3 | Switching data between different interfaces can take a long time and depends on the network | Long running times |
| 50% | 3 | The complexity of building such system might take a lot of time and resources. | The NER sub-system will not be ready in time. |
| 100% | 2 | The system uses Rest API, which requires a network connection and adds external dependency to the system | Network dependency |

1. Refers to regulations for web content accessibility (WCAG) of Israel [↑](#footnote-ref-1)