Towards Content Accessibility Through Lexical Simplification for Maltese as a Low-Resource Language

Martina Meli
Department of CIS
Faculty of ICT
University of Malta
martina.meli.18@um.edu.mt

Marc Tanti
Institute of Linguistics and Language Technology
Faculty of Media and Knowledge Sciences
University of Malta
marc.tanti@um.edu.mt

Chris Porter
Department of Computer Information Systems
Faculty of ICT
University of Malta
chris.porter@um.edu.mt

Abstract

Natural Language Processing techniques have been developed to assist in simplifying online content while preserving meaning. However, for low-resource languages, like Maltese, there are still numerous challenges and limitations. Lexical Simplification (LS) is a core technique typically adopted to improve content accessibility, and has been widely studied for high-resource languages such as English and French. Motivated by the need to improve access to Maltese content and the limitations in this context, this work set out to develop and evaluate an LS system for Maltese text. An LS pipeline was developed consisting of (1) potential complex word identification, (2) substitute generation, (3) substitute selection, and (4) substitute ranking. An evaluation data set was developed to assess the performance of each step. Results are encouraging and will lead to numerous future work. Finally, a single-blind study was carried out with over 200 participants, where the system's perceived quality in text simplification was evaluated. Results suggest that meaning is retained about 50% of the time, and when meaning is retained, about 70% of system-generated sentences are either perceived as simpler or of equal simplicity to the original. Challenges remain, and this study proposes a number of areas that may benefit from further research.

1 Introduction

Lexical Simplification (LS) is a technique through which complex words are replaced with simpler alternatives while aiming to retain meaning and contextual validity. Although this has been the subject of various studies (Alarcon et al., 2021; Qiang et al., 2021, 2020), it is more often adopted within high-resource languages, such as English or French (Rolin et al., 2021). On the other hand, low-resource languages, such as Maltese, lack sufficient high-quality resources (Hedderich et al., 2020) required for robust natural language processing (NLP).

Reading difficulties are widely acknowledged as a barrier to information access (Mutabazi and Wallenhorst, 2020). For this reason, LS techniques can be adopted at the core of language-based assistive technologies (ATs) to enhance content accessibility (Rolin et al., 2021). Such ATs could benefit different groups of people, from non-native speakers to persons with low literacy levels and individuals with learning difficulties, among others (Alarcon et al., 2021). Unfortunately, limited research exists for low-resource languages, particularly for Maltese. To the best of our knowledge, the state-of-the-art with respect to automated text simplification for Maltese is a 2014 study based on unsupervised lexical substitution (Tanti, 2014). However, Tanti (2014) argues that due to a lack of resource availability as well as choice of techniques, especially at the time, his work produces unsatisfactory results.

With this in mind, the primary objective of this work is to leverage existing NLP techniques as well as arising linguistic resources for Maltese to develop an effective LS system that is capable of simplifying complex words in Maltese news articles, which is an easy domain for collecting high quality data.

This study makes several contributions, including (1) evidence-based insights on the various LS pipeline steps and on the system overall (including perceived quality), (2) an annotated data set based on content derived from online Maltese news portals in collaboration with a Maltese language expert, as well as (3) a framework for implementing an LS system for low-resource languages. An AT in the form of a browser extension was also developed as a reference implementation based on the arising framework; however, this is considered out of scope for this paper.

This paper is based on the primary author's postgraduate research at the Faculty of ICT, University of Malta. This research is in conformity with the University of Malta's Research Code of Practice and Research Ethics Review Procedures.

The rest of the paper is organised as follows. Section 2 presents a summary of related work, followed by a discussion of the data set generated for this study in Section 3. Section 4 describes the developed framework and evaluation of the pipeline, while results are then presented in Section 5. Section 6 presents concluding remarks and limitations.

2 Related Work

Several LS systems exist for various high resource languages, including English (Qiang et al., 2021), Turkish (Uluslu, 2022), French (Rolin et al., 2021), Spanish (Alarcon et al., 2021), and Chinese (Qiang et al., 2020), among others. To our knowledge, the only prior work that performs LS for Maltese is by Tanti (2014), a system that makes use of n-grams and bag of words vectors to determine which words can substitute a target word. The system has some issues such as a small data set that was developed exclusively by the author and a poorly performing system that only produces acceptable substitutes 5% of the time. Since then, new resources became available that would allow us to make a much better system.

Our work is mostly inspired by LSBert (Qiang et al., 2021). This system is adequate for low resource languages as it does not require a training set, only a pre-trained masked language model like BERT (Devlin et al., 2019). It makes use of BERT predictions, token vector similarity, and word frequency to determine which words can substitute a target word. It achieves state-of-the-art results for substitute generation and outperforms baseline systems with commonly-used data sets, attaining the highest accuracy. We adapt this system for the Maltese language and improve upon it in order to achieve better results.

3 Data set

To evaluate and tune the individual modules and the LS system as a whole, a data set[1] was manually compiled for Maltese as the data set made by Tanti (2014) was not satisfactory. To create this data set, we scraped sentences (with permission) from four popular Maltese news portals[2] relating to articles of different news categories. In this way, the data set contains sentences typically viewed by target users of the LS system. This text-scraping approach is commonly used when compiling such a data set (McCarthy and Navigli, 2007; Horn et al., 2014).

The sentences were stratified by news category and number of (word level) tokens. The categories were determined by extracting the top-level category from the news web page and manually determining which category names across different websites were equivalent. Only categories that were common across all websites were used, which gave us five categories across four websites: commerce, sports, lifestyle, politics, and general. Four sentences per category per website were extracted, resulting in 20 sentences per website, or 80 sentences in total. This is suitable for system evaluation, as it has the same size as NNSeval[3], and is a manageable workload for manual annotation.

The chosen sentences also had to meet sentence length requirements to avoid unusually long or short sentences. The lengths of all the sentences in the news websites formed a unimodal distribution with a peak centred between lengths 10 and 25. The sentences sampled from the categories had to fit within this range to ensure that they have a typical sentence length for news articles.

Once the 80 sentences were sampled, they were evenly split into two (stratified by website and category): the dev and test set. The former is used to determine the optimal system hyperparameters and the latter to evaluate the tuned system and report results.

The target words that were used in the data set were selected automatically as described in Section 4 (content words that are not entity names or English words). Since the target words could be either complex (and thus could be simplifiable) or already simple, we refer to them as potentially complex words. This allows the data set to include instances where (i) a target word has substitutes, some of which are simpler, (ii) a target word has substitutes but none of them are simpler, or (iii) a target word does not have any viable substitutes.

[1]https://osf.io/kx5yd/?view_only=f1020fdbb8904eaa96968df7a0f046ca

[2]The websites were https://tvmnews.mt/, https://newsbook.com.mt/, https://one.com.mt/ and https://www.illum.com.mt/.

[3]http://ghpaetzold.github.io/data/NNSeval.zip

This is preferred over other data sets such as NNSeval which only presents complex words and their simpler substitutes since it is more representative of what the system will encounter in practice.

Two annotators worked to manually annotate candidate words for each target word by using a Maltese thesaurus (Serracino-Inglott, 2016) and a Maltese Word2Vec model (`w2v_cc_300d`)[4] to assist in finding candidates. Using a Word2Vec model was favoured over BERT-based models since the latter would produce words that our system would produce which would be a bias in our favour. Annotators were allowed to include candidates that are not suggested by these resources or to not use any candidates at all if necessary (in which case, the system should not substitute the target word).

We recruited a professional proofreader for Maltese to review and edit the manually annotated substitutes. This allows us to be more confident in the accuracy and correctness of the substitutes. The proofreader was asked to only review the substitutes in terms of meaning and context, and not simplicity, as the simplicity component is evaluated in a subsequent stage. Moreover, unlike for the FrenLyS data set (Rolin et al., 2021), hypernyms or hyponyms were not considered as correct candidate substitutes for most cases since these would result in changing the original sentence's meaning. Moreover, the proofreader was instructed to disregard the pro-clitic[5] preceding the target words when checking candidates. Pro-clitics change according to the word they are attached to (e.g. 'the sun', 'the sand' and 'the boy' in Maltese become 'ix-xemx', 'ir-ramel', and 'it-tifel') and so need to be fixed if the latter is substituted.

The next step was selecting which candidates were simpler than the target word. This was a more subjective task, so all annotators were tasked with annotating all the sentences in order to aggregate their annotations and be able to calculate an inter-annotator score. The number of annotators typically recruited varies across studies: some recruit 5 per 50 sentences (Kajiwara and Yamamoto, 2015) and some recruit 50 per sentence (Horn et al., 2014). We recruited 4 native Maltese speakers as annotators of varying backgrounds. We deem the task as a binary annotation task rather than a scoring task, such that annotators had to only mark which candidates they deemed simpler than the target word (or none at all if none are simpler). We then needed to aggregate the annotations to handle disagreements. Some researchers used pairwise agreement (McCarthy and Navigli, 2007; Kajiwara and Yamamoto, 2015) while others used the kappa index (Rolin et al., 2021; Specia et al., 2012). We adopted a simpler approach: every annotation was tallied and normalised by the number of annotators (dividing by 4), generating scores for the candidates and target word. The target word would also get a score according to how many annotators considered none of the candidates to be simpler. Candidates with higher scores than the target word were deemed simpler substitutes. If the highest-scoring candidate and the target word had identical scores, both are listed as simpler substitutes.

An analysis of the dev set revealed that out of 280 target words: 228 included the target word among the simpler substitutes and 52 did not (one of the sentences didn't have a single target word and was ignored). Since the dev set would be used to tune the system, it was important to balance these two cases to avoid biasing the model. The test set doesn't need to be balanced as it is meant to be representative of news content. To balance the dev set, we under-sampled the majority class by randomly sampling 52 target words and discarding the annotation of the rest of them (on the dev set, the system does not attempt to identify target words automatically but only works with what is annotated).

Apart from the Maltese data set, we also wanted to test our system on an English data set that has been used to evaluate other systems in order to compare our performance. For substitute generation, we selected NNSeval for this purpose due to its similar size to our data set and also due to it also being split with a 50:50 ratio, ensuring comparable results. We did not find any lexical simplification systems or data sets that are compatible with the way we select simple substitutes (as a binary classification task that includes the target word itself), so we were not able to evaluate our system on an English data set.

4 LS Framework - Pipeline Design

To meet the study's objectives, the system uses a four-step pipeline: Potential Complex Word Identification (PCWI), Substitute Generation (SG), Substitute Selection (SS), and Substitute Ranking (SR).

Some of these modules have hyperparameters

[4]`https://sparknlp.org/2022/03/16/w2v_cc_300d_mt_3_0.html`

[5]A pro-clitic is a clitic attached to the beginning of another word such as the Maltese determiner 'il-' in 'il-kelb' (the dog).

that needed to be tuned, which are provided in Appendix A. Unless otherwise specified, we performed this tuning using grid search (evaluating on the dev set) on each module separately. Below, we give a description of each module and the respective hyperparameters.

4.1 Potential Complex Word Identification (PCWI)

Typically, CWI is the first pipeline step, but this is a complex and subjective task (Rolin et al., 2021), so we perform CWI implicitly, with potentially complex words, also known as target words, that are deemed generally unsimplifiable being disregarded in the rest of the pipeline steps (Shardlow, 2014). The advantage of having this step is that it reduces computation time and resource usage since fewer words are considered.

We filtered words using POS (part of speech) and NER (named entity recognition) tags, honorifics, and English words. We used *BERTu-uPOS*[6] and *BERTu-NER*[7] for POS and NER tags respectively. Only verbs, adjectives, adverbs, and nouns were considered as potentially complex (Ortiz-Zambrano and Montejo-Ráez, 2021; Finnimore et al., 2019) and entity names were ignored, including honorifics such as 'Mrs.' or 'Dr.'. Moreover, since the system is intended for Maltese, untranslated English words (common in Maltese) are also filtered out. We used *pyenchant*[8] to detect English words. Since some Maltese words have equivalent spelling to their English counterpart (e.g., 'bank'), we resolve such ambiguity by checking if another English word is found next to it (e.g., 'blood bank'), and, if not, assume that it is a Maltese word.

Note that we only use this module to construct the data set and when simplifying sentences at production time. It also does not have any hyperparameters and so is not tuned or evaluated.

4.2 Substitute Generation (SG)

The outputs from the PCWI module are fed as input into the SG module, which outputs the most probable words to replace the target words. For Maltese, we use the Maltese monolingual BERT model *BERTu*[9] (Micallef et al., 2022) similarly to how Qiang et al. (2021) used BERT. For English, we use one of 3 English BERT models, *BERT base model (uncased)*[10], *BERT large model (uncased)*[11], and *BERT large model (uncased) whole word masking*[12]. Note that these masked language models (MLMs) were not fine-tuned and used as-is.

We use these MLMs to predict candidate substitutes by replacing the target word with one or multiple mask tokens. The target word is typically replaced with one mask token in LS systems, but this forces candidate words to be made up of a single sub-word token. Given that Maltese is a language with complex morphology, we consider multi-token prediction. We use a beam search algorithm that is adapted to MLMs to search for the most probable sequence of tokens to fill a sequence of masks, from one mask up to a maximum number of masks. We only tried up to 3 masks since candidates are unlikely to be simple if they contain more tokens. Each number of masks requires a separate beam search. Top candidates in the beam are selected based on their pseudo log-likelihood (PLL) scores by summing the log probabilities of the tokens that replace the masks (these tokens form the whole word) (Salazar et al., 2020). Furthermore, we want to avoid filling multiple masks multiple words instead of one multi-token word. We avoid this by making use of the fact that BERT vocabularies consist of front-of-word and rest-of-word tokens, such as "gidem" (he bit) being split into "gid" (front-of-word) and "##em" (rest-of-word), and simply avoid front-of-word tokens being used anywhere except for the first mask in the sequence (and vice-versa for the first mask). This could force the system to construct non-sense words, since there might not be a longer word starting with a particular token, but the fact that we are using a beam of token sequences helps avoid this. An illustration of the beam search algorithm used is shown in Figure 1. We tested beam sizes between 3 and 5.[13]

Given that pro-clitics need to be fixed after substituting the word they are attached to and given that it would unnecessarily eliminate possible valid substitute tokens when included in the MLM's input (for example, if a mask is preceded by the pro-clitic 'ix-', the masks can only be filled by a noun starting with 'x') we try masking the pro-clitic in front of the target word if there is one. The '-' of the

[6]https://huggingface.co/MLRS/BERTu-upos/
[7]https://huggingface.co/MLRS/BERTu-ner/
[8]https://pypi.org/project/pyenchant/
[9]https://huggingface.co/MLRS/BERTu/

[10]https://huggingface.co/bert-base-uncased/
[11]https://huggingface.co/bert-large-uncased/
[12]https://huggingface.co/bert-large-uncased-whole-word-masking/
[13]Preliminary tests on Maltese indicated that beam sizes smaller than 5 produced inferior results due to the complex morphology and so this was fixed to 5 when tuning for Maltese.

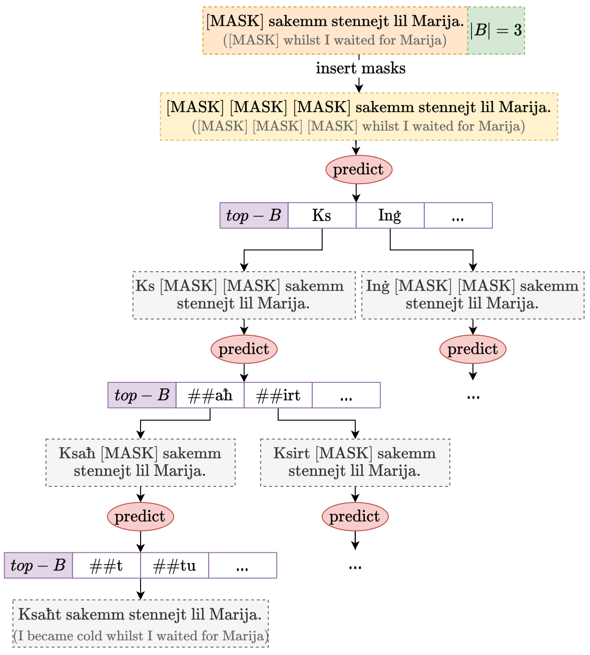
[MASK] sakemm stennejt lil Marija.
([MASK] whilst I waited for Marija)
|B| = 3
insert masks
[MASK] [MASK] [MASK] sakemm stennejt lil Marija.
([MASK] [MASK] [MASK] whilst I waited for Marija)
predict
top – B
Ks
Inġ
...
Ks [MASK] [MASK] sakemm stennejt lil Marija.
Inġ [MASK] [MASK] sakemm stennejt lil Marija.
predict
predict
top – B
##aħ
##irt
...
...
Ksaħ [MASK] sakemm stennejt lil Marija.
Ksirt [MASK] sakemm stennejt lil Marija.
predict
predict
top – B
##t
##tu
...
...
Ksaħt sakemm stennejt lil Marija.
(I became cold whilst I waited for Marija)

Figure 1: Beam Search with beam size $|B|$ and number of masks set to 3, adapted from Nikishina et al. (2022). Note how only front-of-word tokens are considered for the first mask, and only rest-of-word tokens are considered for the other masks.

pro-clitic, which is a separate BERTu token, is left unmasked so that the model is forced to predict a pro-clitic in that particular mask. For example, the phrase 'ix-xita' (the rain) would be masked as '[MASK]-[MASK]'. The pro-clitic mask is always the last mask to be filled by the model in order to allow more freedom in the selection of the actual substitute word.

Given that the beam search fills multiple masks one mask at a time, there was a question about whether these should be filled left-to-right (LTR) or right-to-left (RTL). We decided to leave this as a hyperparameter. We also try using cross-sentence relationship (CSR) where the original sentence (with the target word) is concatenated to the end of the sentence containing the mask tokens, as was done by Qiang et al. (2021).

These hyperparameters were tuned by maximising the F-score on the manually selected candidates in the dev set.

4.3 Substitute Selection (SS)

The candidates produced by the SG module are only valid in terms of fitting the context of the target word. The next step is to filter out the candidates that are semantically different from the target word. We consider two approaches: POS tag filtering and semantic similarity filtering.

POS tag filtering is the simplest. It just checks what the POS tag of the candidate word is after replacing the target word and removing all candidate words that have a different tag from the target word's. Similarity filtering uses a similarity metric to measure the similarity between the MLM's context vector of the candidate word when in the sentence and the context vector of the target word. Candidate words whose similarity is less than a threshold are discarded. As similarity metrics, cosine similarity and word mover's distance (WMD) were attempted. Cosine similarity is widely used for SS (Rolin et al., 2021; Paetzold and Specia, 2017a), but it only calculates the similarity between individual vectors, and thus, when the target or candidate word is a multi-token word, the individual token vectors need to be averaged. On the other hand, WMD gives the distance between two sets of vectors and so can work directly on the multi-token words. Since WMD is a distance function, we first convert it into a similarity function as follows: $\frac{1}{\text{WMD}+1}$. We use the *Word Mover's Distance* library[14] to calculate WMD.

As for the similarity threshold, rather than using a heuristic threshold of 0.5 as used by Rolin et al. (2021), a number was optimised using the dev set as follows. A set of candidates is produced for each target word (using SG), which are sorted by their similarity, which are labelled according to whether the candidate is correct. A threshold is then optimised to maximise the F-score of candidates whose similarity is greater than the threshold (via an exhaustive search among the mid-points between adjacent similarity scores). This threshold was kept fixed once found. We also attempted to scale these similarities such that the similarities of the candidates of each target word would have a mean of 0 and a variance of 1 to make them easier to compare to a single threshold.

As with SG, these hyperparameters were tuned by maximising precision on the manually selected candidates in the dev set. We use precision rather than F-score to focus on the filtering aspect and have more correct substitutes.

4.4 Substitute Ranking (SR)

Having selected the candidates that can replace the target words, the next step is to find which candi-

[14]`https://pypi.org/project/word-mover-distance/`

dates are simpler than their respective target word. We did not make a version of this for the English MLM, only for the Maltese one. The first question to ask is whether the SS filter is necessary or not. Even if it returns a better set of candidates than SG according to the precision score, it could be that this new list excludes simple candidates. For this reason, we include a hyperparameter on whether to use the output of SG or SS as input to SR.

The SR module works by calculating a simplicity score for each candidate. Following literature (Qiang et al., 2021; Uluslu, 2022; Rolin et al., 2021; Qiang et al., 2020), we attempted using the following features to do this: relative frequency, character count, semantic similarity, and MLM probability.

Single-word frequencies are widely used (Qiang et al., 2021; Uluslu, 2022; Rolin et al., 2021; Qiang et al., 2020) since a higher frequency implies simplicity (Rolin et al., 2021). These were generated using Korpus Malti[15] and the Maltese Simplification Corpus[16]. These frequencies were made relative to each corpus (by dividing the word frequency[17] by the total number of words in the respective corpus) to ensure comparable values since the corpora vary in size. Only the Shuffled, Press MT and EU subsets from Korpus Malti were considered, as these encompass words and sentences from various domains, with the latter two consisting of news articles, matching the domain of our data set and system's purpose[18]. Simple and complex texts from the Maltese Simplification Corpus were used, computing relative frequencies using Equation (1). f_s and f_c are the word frequencies in the simple and complex Maltese Simplification Corpus, respectively, whilst t_s and t_c denote the total word counts of the simple and complex corpora.

$$\text{relative frequency} = \frac{\frac{f_s}{t_s}}{\frac{f_s}{t_s} + \frac{f_c}{t_c}} \tag{1}$$

The word character count was chosen to reflect simplicity, as longer words tend to be more complex. We make the character count relative to the data set by dividing a character count by the character count of the longest word in the data set. Semantic similarity, also widely used (Qiang et al., 2021, 2020; Uluslu, 2022), was chosen to reduce the rank of any wrong candidates that make it through the SS/SG module. Similar to similarity filtering in the SS module, it measures the similarity between the target word and the candidate using the MLM context vectors. Moreover, rather than the typically-adopted sentence probability (Uluslu, 2022; Qiang et al., 2021, 2020), the probability of a word fitting into a sentence is applied, using pseudo log-likelihood scores, where the log-probabilities generated by the MLM are summed.

Given that some features have a large range of possible values, we try normalising each feature using L2 normalisation such that the vector formed from a particular feature across all candidates has a magnitude equal to 1. SR generally entails averaging individual scores from candidate word features (Qiang et al., 2021, 2020; Uluslu, 2022), or employing ML models tailored for SR (Rolin et al., 2021). We opted to optimise simple machine learning (ML) models. The classifier models considered were logistic regression, naïve Bayes, XGBoost, and LightGBM, chosen mainly for their ability to handle tabular (Shwartz-Ziv and Armon, 2022) and small data sets (Liang et al., 2020; Sathyaraj and Sevugan, 2015). To train these models, we labelled the candidates in the dev set according to whether they were simpler than the target word. Simpler candidates are labelled with a 1, the rest with a 0. If none of the candidates are simpler than the target word, then they are all labelled 0. The target word is also labelled such that it is only given a 1 when none of the candidates are simpler. The model would then be trained to give a score to the candidate and target words that comes as close as possible to the label.

Hyperparameter tuning was also used in this module, but due to the linear models needing to be tuned as well, which can be numerous (see Appendix B), grid search was used in combination with *Optuna*[19] which uses search space pruning to obtain the best-performing hyperparameters efficiently.

The objective function was set to maximise all the evaluation metrics discussed in Section 5 using multi-objective optimisation. We used the default search algorithm, Tree-structured Parzen Estimator.

[15]`https://mlrs.research.um.edu.mt/`

[16]`https://github.com/mtanti/maltese-simplification-corpus/`

[17]Words are POS tagged when counting their frequency such that it is the frequency of a word-tag pair that is counted.

[18]The news articles found in the Korpus Malti were not the same as the articles used to make our data set, which was made with articles that came out after the corpus was compiled.

[19]`https://optuna.org/`

5 Results and Evaluation

We automatically evaluated each step in the pipeline after tuning the hyperparameters. We also conducted a human evaluation of the full system through a single-blind study.

The automatic evaluation metrics are just different ways of comparing the generated substitutes with the correct substitutes (where correct substitutes are either the set of substitutable words or the set of simple words). The precision metric is the percentage of correctly generated substitutes out of all generated substitutes. The recall metric is the percentage of correctly generated substitutes out of all correct substitutes. The accuracy metric is the percentage of generated substitutes that are correct. The precision@1 metric is the percentage of target words with a correct highest-scoring generated substitute. Finally, the F-score metric is the harmonic mean of precision and recall. Different subsets of these metrics are used to evaluate different modules.

When evaluating the SG and SS modules, we selected the evaluation metrics precision, recall, and F-score. These are the most widely used for SG (Alarcon et al., 2021; Qiang et al., 2020; Paetzold and Specia, 2017a).

Hyperparameter tuning the SG module on the Maltese dev set revealed that it performs best (F-score 0.169) with pro-clitic consideration, right-to-left mask filling, use of CSR, 1 mask, and a beam size of 5. The fact that 1 mask was better is surprising given the complex morphology of Maltese and the small number of generated substitutes (1 mask $\times$ beam size 5 = 5 candidates). This is evidence in favour of BERTu's performance which is suggesting good substitutes with just one token. On English using the English dev set, hyperparameter tuning revealed that the best performing parameters (F-score 0.196) were the same as for Maltese, but using up to 3 masks instead of 1, and using *bert-large-uncased*.

Hyperparameter tuning the SS module on the Maltese dev set revealed that it performs best (precision 0.188) with Cosine similarity filtering, without scaling, using a similarity threshold of 0.85 and no POS tag filtering. Surprisingly, POS tag filtering actually lowered both precision and recall, which probably means that the Maltese POS tagger used could be improved. We opted to just reuse the hyperparameters for English as well rather than performing tuning again since there were no language specific hyperparameters like in the SG module.

The results on the Maltese and English test sets are shown in Table 1, where we quoted the results obtained by Qiang et al. (2021) where they re-implemented a number of LS systems and evaluated them on NNSeval (we only include the results of some top performing models, which include those developed by Paetzold and Specia (2016, 2017b); Gooding and Kochmar (2019)). We can see that the SG module by itself does not beat the system produced by Qiang et al. (2021) but when the additional filtering of the SS module is used, then we double our F-score, which gives us the best results in the table for English. For Maltese we see that SS does not improve our F-score, only the precision.

When evaluating the SR module, we selected the evaluation metrics accuracy, precision, recall, F-score, and precision@1.

Hyperparameter tuning the SR module on the Maltese dev set revealed that it performs best with SG as a source for candidate words, Cosine similarity for similarity scoring, no normalisation, and LightGBM as an ML model. The model gives importance to all features, but mostly to the similarity and pseudo log-likelihood scores, as shown in Table 2. The feature importance scores show that the frequency of the words in the general corpus is twice as important as the frequency of words in the domain-specific corpora, probably because the size of the corpus matters more.

We compared the results of the SR module when using the candidates provided by the SG module with the results of the SR module when using the annotated candidates in the data set. This is to see how the performance of the SR module would change if the SG module was perfect. The results on the Maltese test sets are shown in Table 3. We can see that, while the normal system suggests a correct simpler word as the highest scoring word (precision@1) 72% of the time, a perfect SG module would bump this up to 81%. The difference in performance on the rest of the metrics is not as drastic.

5.1 Human Evaluation

A within-subjects single-blind study was carried out with 207 volunteer participants. Participants were given 16 sentence pairs (i.e., the original version and the system-generated lexically simplified version) selected from a pool of 1 000 sentences. Both sentence selection and pair-wise presentation were randomised to avoid patterns and bias. These sentences were separately scraped from Maltese news portals following the same method outlined in

Data set	System	Precision	Recall	F-score
NNSeval	Paetzold-CA	0.118	0.161	0.136
	Paetzold-NE	0.186	0.136	0.157
	REC-LS	0.103	0.155	0.124
	LSBert	0.194	0.260	0.222
	Our system (SG)	0.218	0.190	0.203
	Our system (SG+SS)	**0.319**	**0.560**	**0.406**
Our Maltese data set	Our system (SG)	0.153	**0.449**	**0.228**
	Our system (SG+SS)	**0.167**	0.340	0.224

Table 1: SG and SS results compared with other systems in literature (best results in bold).

Feature	Importance
Similarity score	3039
PLL score	3031
Shuffled corpus frequency	2057
Character count	1145
Simplification corpus frequency	1082
Press corpus frequency	806

Table 2: Feature importance for simplification score according to the LightGBM model (using split feature importance).

Metric	SG+SR	Gold+SR
Accuracy	**0.886**	0.766
Precision	0.628	**0.768**
Recall	**0.711**	0.709
F-score	0.667	**0.737**
Prec.@1	0.724	**0.814**

Table 3: Our SR results on the Maltese data set (best results in bold). 'Gold' refers to the annotated substitutes in the data set.

Section 3. Participants had to blindly select the sentence they deemed simpler, along with whether the two sentences had the same meaning. Participants were asked about sentences that the system deemed as already in their simplest form - and whether these could be simplified further (and how).

With regards to meaning preservation, participants indicated that the two sentences had the same meaning 44% of the time, and that, of those sentences, 53% thought that the generated sentence was simpler, 29% thought that the original sentence was simpler, and 18% were unsure about which was simpler. Further analysis showed that in most cases where the meaning was changed, this was due to a single substituted word within the sentence.

A demographic analysis showed that younger participants and persons with lower levels of education perceived the system to be more effective. Similar views were provided by individuals whose first language was not Maltese. Of the sentences the system deemed as unsimplifiable, 73% of participants agreed that this was so.

This is an encouraging step for a low-resource language like Maltese, which only required 40 annotated sentences in the dev set to tune the system's hyperparameters.

6 Conclusion

We developed and evaluated an LS pipeline for Maltese, together with the compilation of a Maltese LS data set that was used throughout the process. The various pipeline steps were individually evaluated, with promising results. Our approach also produced significant improvements over the results obtained in the unsupervised lexical substitution system developed for Maltese (Tanti, 2014).

The overall system was also evaluated through a single-blind study with 207 individuals. This was done to determine the overall perceived quality of the system-generated simplified text, and encouraging results were obtained as outlined in Section 5.1. Furthermore, participants generally concurred when presented with sentences that the system determined as already in their simplest form.

Arising from this work, a browser extension was also developed (MaltEasy), acting as a reference implementation of an LS-based AT for Maltese online content accessibility. Although not presented in this paper, MaltEasy provides the team with a first-cut design that motivates the need for further framework improvements and user studies, clearly informing the future of this work.

6.1 Limitations and Future Work

This work presents a promising framework for developing an LS system for Maltese but has some limitations that can be addressed in future work. Despite efforts to compile a new comprehensive data set for the task, the Maltese LS data set used for training and evaluation may benefit from further expansion. Moreover, the system was limited to using BERTu (Micallef et al., 2022), the only available Maltese BERT model, which is only available with a base architecture. The LS system would benefit from using a larger architecture, as shown by the fact that the BERT-large model gave the best results for English. It would also be interesting to determine whether the system developed can be applied to other low-resource languages (after adapting the language specific elements like pro-clitic handling). Furthermore, the proposed LS system focuses on simplification at word level, overlooking multi-word expressions where individual words should not be substituted, a problem that could be solved by a more sophisticated PCWI module the includes multi-word expression detection. Additionally, further filtering might be implemented such that ambiguous sentences are skipped from simplification to avoid unintentionally changing the author's intended meaning.

Acknowledgements

This project was partially funded by the Malta Digital Innovation Authority AI Scholarship.

References

Rodrigo Alarcon, Lourdes Moreno, and Paloma Martínez. 2021. Lexical Simplification System to Improve Web Accessibility. *IEEE Access*, 9:58755–58767.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of the 2019 NAACL-HLT., Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pierre Finnimore, Elisabeth Fritzsch, Daniel King, Alison Sneyd, Aneeq Ur Rehman, Fernando Alva-Manchego, and Andreas Vlachos. 2019. Strong Baselines for Complex Word Identification across Multiple Languages. In *Proc. of the 2019 NAACL-HLT*, pages 970–977, Minneapolis, Minnesota. Association for Computational Linguistics.

Sian Gooding and Ekaterina Kochmar. 2019. Recursive context-aware lexical simplification. In *Proc. of the 2019 Conf. on Empirical Methods in Natural Lang. Proc. and the 9th Int. Joint Conf. on Natural Lang. Proc. (EMNLP-IJCNLP)*, pages 4853–4863, Hong Kong, China. Association for Computational Linguistics.

Michael A. Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. 2020. A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios. *arXiv e-prints*, page arXiv:2010.12309.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using Wikipedia. In *Proc. of the 52nd Annu. Meeting of the Assoc. for Comput. Linguistics (Volume 2: Short Papers)*, pages 458–463, Baltimore, Maryland. Association for Computational Linguistics.

Tomoyuki Kajiwara and Kazuhide Yamamoto. 2015. Evaluation dataset and system for japanese lexical simplification. pages 35–40.

Weizhang Liang, Suizhi Luo, Guoyan Zhao, and Hao Wu. 2020. Predicting hard rock pillar stability using gbdt, xgboost, and lightgbm algorithms. *Mathematics*, 8(5).

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Procs. of the 4th Int. Workshop on Semantic Eval.*, SemEval '07, page 48–53, USA. Association for Computational Linguistics.

Kurt Micallef, Albert Gatt, Marc Tanti, Lonneke van der Plas, and Claudia Borg. 2022. Pre-training Data Quality and Quantity for a Low-Resource Language: New Corpus and BERT Models for Maltese. In *Proc. of the Third Workshop on Deep Learn. for Low-Resour. Natural Lang. Proc.*, pages 90–101, Hybrid. Association for Computational Linguistics.

Eric Mutabazi and Nathanaël Wallenhorst. 2020. Une citoyenneté de seconde classe? n'ayons pas peur des mots! *bildungsforschung*, (1):1–13.

Irina Nikishina, Alsu Vakhitova, Elena Tutubalina, and Alexander Panchenko. 2022. Cross-Modal Contextualized Hidden State Projection Method for Expanding of Taxonomic Graphs. In *Proc. of TextGraphs-16: Graph-based Methods for Natural Lang. Process.*, pages 11–24. Association for Computational Linguistics.

Jenny Ortiz-Zambrano and Arturo Montejo-Ráez. 2021. SINAI at SemEval-2021 Task 1: Complex word identification using Word-level features. In *Proc. of the 15th Int. Workshop on Semant. Eval.*, pages 126–129, Bangkok, Thailand.

G. H. Paetzold and L. Specia. 2016. Unsupervised Lexical Simplification for non-native speakers. In *AAAI'16: Proc. of the Thirtieth AAAI Conf. on AI*, pages 3761–3768.

Gustavo H. Paetzold and Lucia Specia. 2017a. A Survey on Lexical Simplification. *J. Artif. Intell. Res.*, 60:549–593.

Gustavo Henrique Paetzold and Lucia Specia. 2017b. Lexical Simplification with Neural Ranking. In *Proc. of the 15th Conf. of the Eur. Chapter of the Assoc. for Comput. Linguistics: Volume 2, Short Papers*, volume 2, pages 34–40.

Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, Yang Shi, and Xindong Wu. 2021. LSBert: Lexical Simplification Based on BERT. *IEEE/ACM Trans. on Audio Speech and Lang. Process.*, 29:3064–3076.

Jipeng Qiang, Xinyu Lu, Yun Li, Yunhao Yuan, and Xindong Wu. 2020. Chinese Lexical Simplification. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 29:1819–1828.

Eva Rolin, Quentin Langlois, Patrick Watrin, and Thomas François. 2021. FrenLyS: A Tool for the Automatic Simplification of French General Language Texts. In *RANLP 2021*, pages 1196–1205. INCOMA Ltd.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proc. of the 58th Annu. Meeting of the Assoc. for Comput. Linguistics*. Association for Computational Linguistics.

R. Sathyaraj and Prabu Sevugan. 2015. An approach for software fault prediction to measure the quality of different prediction methodologies using software metrics. *Indian J. of Sci. and Tech,*, 8.

Mario Serracino-Inglott. 2016. *Id-Dizzjunarju Malti*, 4 edition. Merlin Publishers.

Matthew Shardlow. 2014. A Survey of Automated Text Simplification. *Int. J. Adv. Comput. Sci. Appl. (IJACSA) Spec. Issue Nat. Lang. Process.*, 4(1).

Ravid Shwartz-Ziv and Amitai Armon. 2022. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90.

Lucia Specia, Sujay Kumar Jauhar, and Rada Mihalcea. 2012. Semeval-2012 task 1: English lexical simplification. In *SEM 2012: The First Joint Conf. on Lexical and Comput. Semantics – Vol. 1: Proc. of the main conf. and the shared task, and Vol. 2: Proc. of the Sixth Int. Workshop on Semantic Eval (SemEval 2012)*, pages 347–355, Montréal, Canada. Association for Computational Linguistics.

M. Tanti. 2014. *Unsupervised lexical substitution for Maltese: steps toward lexical simplification*. Ph.D. thesis, M.S. thesis, Dept. Intellig. Comput. Syst., Univ. of Malta, Msida, Malta.

Ahmet Yavuz Uluslu. 2022. Automatic Lexical Simplification for Turkish.

Parameter	**Values**
SG module	
Max. masks	1, 2, 3
Beam size	3, 4, 5
Pro-clitic mask*	yes, no
Mask fill order	LTR, RTL
Use CSR	yes, no
SS module	
POS tag filtering	yes, no
Similarity filtering	yes, no
Similarity method	cosine, WMD
Scaling	yes, no
SR module	
Candidate words source	SS, SG
Similarity method	cosine, WMD
Feature norm.	yes, no
ML model	logistic reg., XGBoost, LightGBM, Naïve Bayes

Table 4: Hyperparameter search space used when tuning the separate modules.
*Only for the Maltese data set.

A Hypermarameter search space for separate modules

B SR module hyperparameters for ML models

The best hyperparameters of the LightGBM model, which was the best performing model, were max_depth set to 7, n-estimators to 600, and learning_rate to 0.0149.

Parameter	**Values**
Logistic regression	
solver	liblinear, saga
c_value	0.1 - 5 (uniform)
XGBoost	
learning rate	0.01 - 0.3 (uniform)
maximum depth	3 - 9 (integer)
n estimators	100 - 1000 (uniform), with a 100 step
LightGBM	
learning rate	0.01 - 0.3 (uniform)
maximum depth	3 - 9 (integer)
n estimators	100 - 1000, with a 100 step (uniform)
Naïve Bayes	
var_smoothing	1E-12 - 1E-3 (log uniform)

Table 5: Hyperparameters used for the ML models in the SR module.