KLPT – Kurdish Language Processing Toolkit

Sina Ahmadi
Insight Centre for Data Analytics
National University of Ireland Galway
ahmadi.sina@outlook.com

Abstract

Despite the recent advances in applying language-independent approaches to various natural language processing tasks thanks to artificial intelligence, some language-specific tools are still essential to process a language in a viable manner. Kurdish language is a less-resourced language with a notable diversity in dialects and scripts and lacks basic language processing tools. To address this issue, we introduce a language processing toolkit to handle such a diversity in an efficient way. Our toolkit is composed of fundamental components such as text preprocessing, stemming, tokenization, lemmatization and transliteration and is able to get further extended by future developers. This project is publicly available.

1 Introduction

Language technology is an increasingly important field in our information era which is dependent on our knowledge of the human language and computational methods to process it. Unlike the latter which undergoes constant progress with new methods and more efficient techniques being invented, the processability of human languages does not evolve with the same pace. This is particularly the case of languages with scarce resources and limited grammars, also known as less-resourced languages.

Various natural language processing (NLP) tasks are of pipeline architecture; that is, to address a specific task, a few other language processing tasks may be initially required (Manning et al., 2014). With the current advances in the open-source movements, more researchers and industrial developers are encouraged to share their knowledge in an open-source manner, accessible under certain conditions (Ljungberg, 2000). Therefore, the development of underlying tasks in NLP for a specific language will potentially pave the way for further contributions to the field, by either improving the current tools or further progress in new tasks. For instance, tokenization as a fundamental task is widely required in many other applications such as part-of-speech tagging, machine translation and syntactic analysis. Once addressed, future researchers can build upon it for more advanced tasks or eventually improve it.

Despite a plethora of performant tools and specific frameworks for NLP, such as NLTK (Loper and Bird, 2002), Stanza (Qi et al., 2020), Teanga (Ziad et al., 2018) and spaCy, the progress with respect to less-resourced languages is often hindered by not only the lack of basic tools and resources but also the accessibility of the previous studies under an open-source licence. This is particularly the case of Kurdish, a less-resourced Indo-European language that is the focus of the current paper. As an example, although the task of spell-checking and stemming for Kurdish have been addressed by many previous studies, (Jaf and Ramsay, 2014; Salavati and Ahmadi, 2018; Mustafa and Rashid, 2018; Saeed et al., 2018a; Hawezi et al., 2019) to mention but a few, none of them provides an implementation of their tool under any licence.

On the other hand, some previous studies use specific frameworks that are hardly integrable and inter-operable. For instance, (Walther and Sagot, 2010) and (Walther et al., 2010) describe their efforts in developing a large-scale morphological lexicon and a part-of-speech tagger for Kurdish within the Alexina framework under the LGPL-LR licence. Despite the valuable impact of this study in the field, for example in (Cotterell et al., 2017) and (Gökirmak and Tyers, 2017), the tool does not

1https://github.com/sinaahmadi/klpt

2https://github.com/explosion/spaCy
Table 1: A comparison of the Kurdish alphabets. Variations are specified with "/".

<table>
<thead>
<tr>
<th>IPA</th>
<th>a</th>
<th>e</th>
<th>ë</th>
<th>i</th>
<th>ï</th>
<th>o</th>
<th>ù</th>
<th>u</th>
<th>ù</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latin</td>
<td>a</td>
<td>e</td>
<td>ê</td>
<td>i</td>
<td>ï</td>
<td>o</td>
<td>ù</td>
<td>u</td>
<td>ü</td>
</tr>
<tr>
<td>Arabic</td>
<td>ә</td>
<td>ә</td>
<td>ә</td>
<td>ә</td>
<td>ә</td>
<td>ә</td>
<td>ә</td>
<td>ә</td>
<td>ә</td>
</tr>
</tbody>
</table>

(a) Consonants

| IPA | b | tʃ | ʃ | d | f | ɡ | h | ʒ | k | l | m | n | p | q | t | v | w | x | ʒ | ɣ |
| Latin | b | ç | c | d | f | ɡ | h | j | k | l | ll | m | n | p | q | r | f | r | r | s | t | v |
| Arabic | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә |

(b) Vowels

Kurdish and Laki, respectively with kmr Kurmanji), Central Kurdish (or Sorani), Southern
The division of Kurdish into Northern Kurdish (or diaspora around the world (Ahmadi et al., 2019).
Iran and Syria and also, among the Kurdish di-
Kurdish belongs to the Northwestern branch of the
seem to be widely used in the subsequent projects. As such, projects such as (Jaf and Ramsay, 2014) and (Ahmadi and Hassani, 2020a) tackle the very same topic from scratch.
Language-specific toolkits have been previously designed for various languages, such as IceNLP for Icelandic (Lofthsson and Rögnvalds-
Kurdish has been historically written in various scripts, namely Cyrillic, Armenian, Latin and Arab-
Kurdish people that those are in fact different dia-
Kurdish language is a highly inflectional lan-

2 Kurdish Language

Kurdish belongs to the Northwestern branch of the
The division of Kurdish into Northern Kurdish (or Kurmanji), Central Kurdish (or Sorani), Southern Kurdish and Laki, respectively with kmr, ckb, sdb and 1ki ISO 639-3 language codes, has been widely studied previously (Edmonds, 2013). Based on the structural differences between these, some scholars believe that they are distinct languages and therefore, refer to them as Kurdish languages (Kreyenbroek, 2005). On the other hand, it is also commonly believed by both scholars and

| IPA | b | tʃ | ʃ | d | f | ɡ | h | ʒ | k | l | m | n | p | q | t | v | w | x | ʒ | ɣ |
| Latin | b | ç | c | d | f | ɡ | h | j | k | l | ll | m | n | p | q | r | f | r | r | s | t | v |
| Arabic | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә | ә |

 seems to be widely used in the subsequent projects. As such, projects such as (Jaf and Ramsay, 2014) and (Ahmadi and Hassani, 2020a) tackle the very same topic from scratch.
Language-specific toolkits have been previously designed for various languages, such as IceNLP for Icelandic (Lofthsson and Rögnvalds-
Kurdish has been historically written in various scripts, namely Cyrillic, Armenian, Latin and Arab-
Kurdish people that those are in fact different dia-
Kurdish language is a highly inflectional lan-

2 Kurdish Language

Kurdish belongs to the Northwestern branch of the
The division of Kurdish into Northern Kurdish (or Kurmanji), Central Kurdish (or Sorani), Southern Kurdish and Laki, respectively with kmr, ckb, sdb and 1ki ISO 639-3 language codes, has been widely studied previously (Edmonds, 2013). Based on the structural differences between these, some scholars believe that they are distinct languages and therefore, refer to them as Kurdish languages (Kreyenbroek, 2005). On the other hand, it is also commonly believed by both scholars and
verbs, Kurdish has a few number of around 300 single-word verbs (Walther and Sagot, 2010), e.g. kirdin/kirin “to do”, which are inflected based on person (1,2,3, SG, PL), tense (past, present, future), aspect (indefinite, perfect, progressive, imperfective) and mood (indicative, subjunctive, conditional). Unlike Kurmanji, Sorani Kurdish does not have future tense and uses adverbs for this purpose. However, Kurdish extensively takes use of compound constructions for creating new verb forms, particularly with (Noun + Verb), (Adjective + Verb) and (Preposition + Verb) forms (Traida, 2007). For instance, sitlav ‘hi (n), pîroz ‘holy’ (adj) and het (verbal particle denoting ‘up’) with the single-word verb kirdin can respectively form compound verbs sitlav kirdin “to greet”, pîroz kirdin “to congratulate” and het kirdin “to turn on”. The stringing characteristic of the Arabic-based script of Kurdish further adds to this morphological complexity in such a way that several word forms may be concatenated together (Ahmadi, 2020b).

Regarding syntax, Kurdish has a subject–object–verb word order and is a null-subject (or pro-drop) language. The presence of grammatical markers for nominative and oblique cases varies within dialects and subdialects. For instance, in the Sorani subdialects of Sulaymaniyah and Erbil, respectively categorized as Southern Sorani and Northern Sorani by (Matras, 2017), the oblique case is marked differently. Another particularity of the Kurdish language is its morphosyntactic alignment in the past tense of transitive verbs. In such tenses, an ergative–absolutive alignment occurs where the subject of intransitive verbs behaves like the patient of the transitive verb in the past (Haig, 1998; Karimi, 2014). Unlike Kurmanji which uses oblique cases for this purpose, Sorani only uses different pronominal markers to specify ergativity, therefore it is called split-ergative (Esmaili and Salavati, 2013). Except the past tenses, a nominative-accusative alignment is observed in other tenses.

Not being equally documented and used, Kurdish dialects have different levels of linguistic resourcefulness. In comparison to Sorani and Kurmanji which are widely used by the media and press, Southern Kurdish and Laki are underdocumented and lack basic language resources such as electronic dictionaries and corpora (Fattah, 2000; Ahmadi et al., 2019; Ahmadi, 2020c).

3 Current State of Kurdish Language Processing

The earliest works in the field of Kurdish language processing date back to 2009. Our literature review indicates that some of these contributions fail to provide open-source solutions. Despite financial and scientific constraints in Kurdish language processing, the Kurdish Language Processing Project (KLPP) (Esmaili et al., 2013) in 2012 and Kurdish Basic Language Resource Kit (Kurdish BLARK) (Hassani, 2018) in 2014 have succeeded to promote an open-source vision based on research volunteering within the Kurdish scientific communities. However, the outcomes of these projects are mostly released in an unorganized manner for individual tasks.

In order to understand the current state of the Kurdish language in the realm of NLP and com-
Computational linguistics, we reviewed the scientific publications that directly address an issue in those fields. A total number of 53 publications are collected from the widely-used academic databases and search engines such as Google Scholar, and then classified based on their discussed sub-fields which are illustrated in Figure 1. The Kurdish dialects are not evenly discussed in the previous studies, with Sorani making up a predominant proportion of almost 90%. Although a smaller proportion represents the Kurmanji dialect, no publication is found with respect to processing of the Southern Kurdish or Laki dialects. Regarding the research focus of the previous works, a range of NLP sub-fields has been addressed, particularly in text mining, morphological and syntactic analysis and, creation of lexical resources. We exceptionally included optical character recognition as it is of importance for converting printed material to electronic forms (Ahmadi et al., 2019). The full list of the surveyed papers can be found in Appendix A.2.

More importantly, we analyze previous publications from the following two perspectives:

• **Open-source:** Does the paper provide the discussed resource or tool under an open-source license? To this end, we verified the content of the papers and also, checked the Web, particularly major distributed version control systems such as GitHub, GitLab and BitBucket.

• **Applicability:** Does the paper, implicitly or explicitly, propose an approach or methodology that can be applied to solve the same problem in the other dialects of Kurdish? For the choice of the word, we were inspired by (Årdal et al., 2011) where the possibility of applying common practices of software development for drug discovery are investigated. For instance, (Ahmadi et al., 2019) is deemed an applicable contribution where lexicographical resources can be created for other dialects. On the other hand, (Ahmadi, 2019) is not applicable to other dialects due to its ad-hoc solution for transliterating Sorani texts according to its phonological and phonetic rules.

Figure 2 provides the number of previous publications in the Kurdish language processing field per year, and specifies their open-source status and their applicability. Although most of these publications are applicable to other dialects, only 18 out of 53 of them provide their resources or tools under an open-source license. Among the open-source ones, 11 are outcomes of volunteering projects, KLPP and Kurdish-BLARK. Given the small number of non-scientific contributions, we did not include them in this survey. A few notable examples of such contributions are Kurdînûs, Vejin Dictionaries and VejinBooks which mostly focus on Sorani Kurdish and script conversion tasks.

---

5https://scholar.google.com
6https://github.com
7https://gitlab.com
8https://bitbucket.org
9https://github.com/aso-mehmudi/kurdinus
10https://lex.vejinbooks.com
11https://books.vejin.net
4 KLPT Architecture

KLPT is implemented in Python and is composed of four core modules with specific tasks. Although we were inspired by the functionality of relevant NLP toolkits, particularly NLTK and spaCy, no external library is used in this toolkit. Regarding the toolkit design, we followed the rules of scientific software development suggested by (Prlić and Procter, 2012) along with common practices in Python programming language. Figure 3 provides the structure of the toolkit. In order to facilitate the integration of variations specific to dialects and scripts and more importantly, to avoid hard-coding, required files are provided in the data folder. For instance, the data required for the preprocess module is imported from preprocess.json. In addition, third-party programs can be provided in bin. test and docs respectively contain test cases and project documentation. Regarding the latter, we use Sphinx documentation generator⁷².

It is worth noting that each module within the klpt package has been previously studied and evaluated separately. Our goal is to introduce the functionality of the modules within the toolkit in this section.

4.1 Preprocess

Many keyboard layouts are specifically designed for Kurdish where different character encoding are assigned to visually-similar graphemes. In addition to the usage of non-Kurdish keyboards, such as Arabic, Turkish and Persian keyboards, such diversity creates abnormality across texts in Kurdish writing. For instance, the grapheme ð (î) can be represented as ð (U+064A), ð (U+0649), ð (U+PEF2), ð (U+PEF1) and ð (U+06CC), among which only the latter should be used in the Arabic-based script of Kurdish. Moreover, various writing conventions are used for each dialect and script. For instance, in Kurmanji, when dates are affixed with a morpheme, the suffix may be separated by ', - or without any marker as in 2020’an, 2020-an and 2020an.

To remedy such issues in an automatic and structured manner, the preprocess module provides two main functions: normalize() for normalizing encoding abnormalities by unifying characters in such a way that only one specific encoding is used for each grapheme and, standardize() which applies orthographic conventions to the text. For example, when ðèvî ‘hope’ is suffixed with the vowel a (Izafá, meaning ‘of’), a semi-vowel y appears between the two vowels and is usually written as ðèvîya or ðèvîya ‘hope of’. As the latter form is considered less ambiguous, this function converts the first form accordingly. Although defining a universal orthography for Kurdish is out of scope of our project, we believe that writing conventions and orthographies should be addressed to some extent. Therefore, in this initial version, we follow the writing conventions proposed by (Aydoğan, 2012) for Kurmanji and (Hashemi, 2016) for Sorani.

In addition to these two functions, unify_numeral() is provided to convert numerals, namely in Farsi (۰۱۲۳۴۵۶۷۸۹), Eastern Arabic (٠١٢٣۴۵۶۷۸۹) and Western Arabic (0123456789). Although we set the latter as default for all scripts, users will have the

---

⁷²https://www.sphinx-doc.org

![Figure 3: Structure of KLPT](image-url)
choice to modify the numerals according to the administrations in the Kurdish regions. All these three functions are then evoked within `preprocess()` function which normalizes, standardizes and unifies the text according to the given arguments.

The general procedure followed in this module can be summarized as string replacement. For this purpose, we define regular expressions for each dialect and script. The regular expressions along with the character mappings are provided in `preprocess.json` in such an order that the intended normalization and standardization are carried out correctly. Although this module is not explicitly evoked within other modules, except in the `transliterate` module, it is recommended that the output of the preprocessing module be used as the input of other modules by the user.

4.2 Transliterate

Given the diversity of the alphabets used in Kurdish, transliteration is a necessity to facilitate the communication between speakers and is also beneficial to various NLP tasks, such as named-entity recognition and machine translation. Although Kurdish orthographies are phonemic, i.e. each grapheme is supposed to represent a single phoneme, transliterating characters within the alphabets is more challenging than it appears. This is particularly due to \( \text{و} \) \( (U+0648) \) and \( \text{ی} \) \( (U+06CC) \) in the Arabic-based alphabet which can be respectively mapped to ‘\( \text{w} \)\’ and ‘\( \text{y} \)’. For instance, in مکورت پیور and بیور ‘axe’ and کرت ‘short’, respectively. Moreover, there is no grapheme for the vowel \( i \), also known as Bizrobe “the little furtive”, in the Arabic-based script which creates further challenges in the morphological analysis of the language (Ahmadi, 2019).

In this module, we focus on transliterating Arabic-based and Latin-based scripts of Kurdish using WERGOR transliterator\(^\text{13}\) (Ahmadi, 2019). This tool uses a rule-based approach based on the phonological and syllabic characteristics of Kurdish for distinguishing double-usage characters, i.e. \( \text{و} \) and \( \text{ی} \), and predicting the placement of \( i \). Although the algorithm efficiently transliterates double-usage characters, it has been evaluated to detect \( i \) with a low accuracy of 39%.

4.3 Stem

Although the task of stemming has been previously addressed in the literature, no open-source viable solution was available for Kurdish. Therefore, we developed morphological rules containing combinations of Kurdish morphemes in Sorani and Kurmanji, and also an annotated lexicon containing lemmas with specific flags such as part-of-speech tags and stems. The morphological rules and the lexicons are then used to develop a morphological analyzer and spell-checker in HUNSPELL (Ooms, 2017) for Kurdish, where they are respectively known as affixes \( (\text{.aff}) \) and dictionary \( (\text{.dic}) \). Thanks to the wide usage of HUNSPELL in open-source text editors such as Apache OpenOffice, our development will be also beneficial for general purposes such as spell-checking in text editors. More importantly, we integrate HUNSPELL in KLPT for this module using a wrapper program\(^\text{14}\).

The Stem module comes with two classes: Stem and Spellcheck. Although these two classes focus on two different tasks, they are provided in the same module as they are both based on the same implementation in Hunspell. Given a word, the Stem class provides four main functions, namely `stem()` for retrieving word-form stem, e.g. \( \text{kirdin/kirin} \) \( (\text{do.INF}) \rightarrow \text{kir} \), `lemmatize()` for lemmatization, e.g. \( \text{kirdbüm} \) \( (\text{do.1SG.PST.PFV}) \rightarrow \text{kirdin} \), `analyze()` for morphological analysis which returns a dictionary containing the flags according to HUNSPELL such as part-of-speech, terminal suffixes and inflectional suffixes and finally, `suffix_suggest()` which returns all the possible suffixes that can appear with a given lexeme. In addition to these, `generate()` will also be added to the module which generates a word-form given morphemes.

On the other hand, the Spellcheck class provides `check_spelling()` and `correct_spelling()` which are respectively used for spell checking (Boolean output) and spell correction. For instance, given خواردوومانه \( (\text{xwardûmate}) \), `check_spelling()` detects that it is incorrectly written and a few suggestions are provided by `correct_spelling()`, among which خواردوومانه \( (\text{xwardûmane}) \) “(we) have eaten”. The performance of the tool is further described in (Ahmadi, 2020d,a).

\(^{13}\)https://github.com/sinaahmadi/wergor

\(^{14}\)https://github.com/MSeal/cython_hunspell
4.4 Tokenize

Although both Arabic-based and Latin-based alphabets use spaces to delimit word boundaries, not all words correspond to a token in Kurdish. This is particularly due to the complex morphology, e.g. article marking suffixes, and the writing traditions. In the Arabic-based alphabet, there is a tendency to concatenate clitics, affixes and words together which results many tokens being written as one single word-form without any space as in ١٩٥٠ (it) is also their hope which is composed of four tokens, noun ﻩو، endoclitic =±, pronominal enclitic -yan and present copula e. The Latin-based script, particularly when used for writing Kurmanji, respects word boundaries in a better way. For instance, the same phrase is written as “hêvîya wan ji ew e”.

In this module, we use the tokenization approach proposed by (Ahmadi, 2020b). This approach uses an annotated lexicon with a morphological analyzer to tokenize words in Sorani and Kurmanji. Given the wide usage of compound forms in word formation in Kurdish, a lexicon is also provided for multi-word expressions (MWEs) and their possible forms, with and without space. That way, the inconsistencies in writing compound words is tackled efficiently. In addition to mwe_tokenize() and word_tokenize() which are respectively provided for the tokenization of words and MWEs, sent_tokenize() is a third function which tokenizes a given text into sentences based on punctuation marks. It is worth mentioning that words and MWEs are respectively separated by and by default which can be customized by the user.

4.5 Configuration

Given the combination of scripts and dialects of the input data, verification of the several configurations of each class can be complex. Therefore, we provide the configuration module which is used internally within the modules when an object of a class is initialized. This way, the class constructors validate the arguments by evoking this module and the error-handling is carried out only in the Configuration class.

For further clarification on the interaction of the individual modules within the KLPT package, Figure A.5 shows its package and class diagrams in the Unified Modeling Language (UML).

5 Usages

In this section, we provide basic usages of the application programming interface (API) of the KLPT package. The package is available on the Python Package Index (PyPI), \(^{15}\) in Python 3.5 and later and, can be installed as follows:

```bash
pip install klpt
```

The installation of the package comes with the data files, i.e. data, and requirements which are also installed. Once the package is installed, each module can be imported and used as described above. Figure 4 provides an example on how to work with various modules of the package.

```python
>>> from klpt.preprocess import Preprocess
>>> from klpt.transliterator import Transliterate
>>> from klpt.tokenize import Tokenize
>>> from klpt.stem import Stem

# Preprocess module
>>> preprocessor = Preprocess("Sorani", "Arabic",
numeral="Latin")
>>> preprocessor.normalize("(لا مساعدة) 1950\(ذد سالم\)\(1950\)ا\)"
راسته لە وۇردوادر\)
>>> preprocessor.standardize("(لا مساعدة) 1950\(ذد سالم\)\(1950\)ا\)"
راسته لە وۇردوادر\)

# Transliterate module
>>> transliterator = Transliterate("Kurmanji", "Latin",
target_script="Arabic")
>>> transliterator.transliterate("روژھﻼتا ناڤین\")

# Stem module
>>> stemmer = Stem("Sorani", "Arabic")
>>> stemmer.check_spelling("سۆتوانپەدۆت")
False
>>> stemmer.correct_spelling("سۆتوانپەدۆت")
"سۆتوانپەدۆت"
>>> stemmer.stem("سۆتوانپەدۆت")
"سۆتوانپەد" (سۆتوانپەد) (سۆتوانپەد)
>>> stemmer.analyze("سۆتوانپەد")
‘سۆتوانپەد’, ‘stem’: ‘سۆتوانپەد’}

# Tokenize module
>>> tokenizer = Tokenize("Kurmanji", "Latin")
>>> tokenizer.tokenize("endamên encûmena wezîrên")
[‘endam_ên’, ‘encûmen_ên’, ‘wezîr_ên’]
```

Figure 4: Basic usage of the KLPT package for the Sorani and Kurmanji dialects

6 Conclusion and Future Work

In this paper, we present KLPT, an open-source toolkit developed in Python and composed of core modules, namely Preprocess, Stem,

\(^{15}\)https://pypi.org
Tokenize and Transliterate for processing the Sorani and Kurmanji dialects of Kurdish. In addition to the provided modules, the toolkit enables future researchers to contribute their work by extending the modules for more advanced tasks and other dialects. We believe that recognizing every single contribution to the toolkit is encouraging for researchers and also, beneficial to help Kurdish to pass over its less-resourced status.

As a future work, we would like to extend the current version to include syntactic and semantic parsing for Sorani and Kurmanji. Given the scarcity of resources regarding computational linguistics and natural language processing, we believe that the KLPT package will create a new field of interest for Kurdish linguists as well. Therefore, we are aiming at creating educational content to introduce the field to non-expert public too.

Acknowledgments

The author would like to thank his two colleagues, Dr. Kyumars Sheykh Esmaili and Dr. Hossein Hassani who respectively initiated the Kurdish Language Processing Project and Kurdish-BLARK. Despite the lack of financial support of Kurdish-related projects, these initiatives have made huge contributions thanks to volunteer researchers. Similarly, the constructive comments of the three anonymous reviewers were very useful and are much appreciated.

References


Roojwan Sc Hawezi, Muhammed Y Azeez, and Ahmed A Qadir. 2019. Spell checking algorithm for agglutinative languages “Central Kurdish as an example”. In 2019 International Engineering Conference (IEC), pages 142–146. IEEE.


### Table A.2: Classification of the publications in the field of Kurdish language processing

<table>
<thead>
<tr>
<th>Reference</th>
<th>Year</th>
<th>Field</th>
<th>open-source</th>
<th>applicable</th>
<th>dialects</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Mohammed et al., 2012)</td>
<td>2012</td>
<td>Dialectology</td>
<td>no</td>
<td>no</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Esmaili and Salavati, 2013)</td>
<td>2013</td>
<td>Dialectology</td>
<td>no</td>
<td>yes</td>
<td>Sorani, Kurmanji</td>
</tr>
<tr>
<td>(Hassani and Medjedovic, 2016)</td>
<td>2016</td>
<td>Dialectology</td>
<td>no</td>
<td>yes</td>
<td>Sorani, Kurmanji</td>
</tr>
<tr>
<td>(Malmasi, 2016)</td>
<td>2016</td>
<td>Dialectology</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Al-Talabani et al., 2017)</td>
<td>2017</td>
<td>Dialectology</td>
<td>no</td>
<td>yes</td>
<td>Sorani, Kurmanji, Gorani</td>
</tr>
<tr>
<td>(Littel et al., 2016)</td>
<td>2016</td>
<td>Information retrieval and Text mining</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Hassani, 2017b)</td>
<td>2017</td>
<td>Information retrieval and Text mining</td>
<td>yes</td>
<td>no</td>
<td>Sorani, Kurmanji</td>
</tr>
<tr>
<td>(Esmaili, 2012)</td>
<td>2012</td>
<td>Information retrieval and Text mining</td>
<td>no</td>
<td>no</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Esmaili et al., 2014)</td>
<td>2014</td>
<td>Information retrieval and Text mining</td>
<td>yes</td>
<td>yes</td>
<td>Sorani, Kurmanji</td>
</tr>
<tr>
<td>(Jal, 2016)</td>
<td>2016</td>
<td>Information retrieval and Text mining</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Rashid et al., 2017a)</td>
<td>2017</td>
<td>Information retrieval and Text mining</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Rashid et al., 2017b)</td>
<td>2017</td>
<td>Information retrieval and Text mining</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Ahmadi, 2019)</td>
<td>2019</td>
<td>Information retrieval and Text mining</td>
<td>yes</td>
<td>no</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Saeed et al., 2018)</td>
<td>2018</td>
<td>Information retrieval and Text mining</td>
<td>no</td>
<td>no</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Mustafa and Rashid, 2018)</td>
<td>2018</td>
<td>Information retrieval and Text mining</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Saeed et al., 2018a)</td>
<td>2018</td>
<td>Information retrieval and Text mining</td>
<td>no</td>
<td>no</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Ahmadi et al., 2020)</td>
<td>2020</td>
<td>Lexical resources</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Esmaili et al., 2013)</td>
<td>2013</td>
<td>Lexical resources</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Alizabadi et al., 2014)</td>
<td>2014</td>
<td>Lexical resources</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Alizabadi, 2014)</td>
<td>2014</td>
<td>Lexical resources</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Yaseen et al., 2020)</td>
<td>2020</td>
<td>Lexical resources</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Ahmadi et al., 2019)</td>
<td>2019</td>
<td>Lexical resources</td>
<td>yes</td>
<td>yes</td>
<td>Sorani, Kurmanji, Gorani</td>
</tr>
<tr>
<td>(Abdulrahman et al., 2019)</td>
<td>2019</td>
<td>Lexical resources</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Abdulrahman and Hassani, 2020)</td>
<td>2020</td>
<td>Lexical resources</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Ataman, 2018)</td>
<td>2018</td>
<td>Lexical resources</td>
<td>yes</td>
<td>yes</td>
<td>Kurmanji</td>
</tr>
<tr>
<td>(Hassani, 2017a)</td>
<td>2017</td>
<td>Machine Translation</td>
<td>no</td>
<td>yes</td>
<td>Sorani, Kurmanji</td>
</tr>
<tr>
<td>(Kaka-Khan, 2018)</td>
<td>2018</td>
<td>Machine Translation</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Wallther and Sagot, 2010)</td>
<td>2010</td>
<td>Morphological and syntactic analysis</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Wallther et al., 2010)</td>
<td>2010</td>
<td>Morphological and syntactic analysis</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Salavati et al., 2013)</td>
<td>2013</td>
<td>Morphological and syntactic analysis</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Jal and Ramsay, 2014)</td>
<td>2014</td>
<td>Morphological and syntactic analysis</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Jal and Ramsay, 2016)</td>
<td>2016</td>
<td>Morphological and syntactic analysis</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Gokturk and Tyers, 2017)</td>
<td>2017</td>
<td>Morphological and syntactic analysis</td>
<td>yes</td>
<td>yes</td>
<td>Kurmanji</td>
</tr>
<tr>
<td>(Salavati and Ahmadi, 2018)</td>
<td>2018</td>
<td>Morphological and syntactic analysis</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Mustafa and Rashid, 2018)</td>
<td>2018</td>
<td>Morphological and syntactic analysis</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Ahmadi and Hassani, 2020a)</td>
<td>2020</td>
<td>Morphological and syntactic analysis</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Mohammed, 2012)</td>
<td>2012</td>
<td>Optical character recognition</td>
<td>no</td>
<td>no</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Mohammed, 2013)</td>
<td>2013</td>
<td>Optical character recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Shaltungi and Hama, 2016)</td>
<td>2016</td>
<td>Optical character recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Zarro and Anwer, 2017)</td>
<td>2017</td>
<td>Optical character recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Yaseen and Hassani, 2018)</td>
<td>2018</td>
<td>Optical character recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Dimler and Aydin, 2018b)</td>
<td>2018</td>
<td>Optical character recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Kaka-Khan, 2017)</td>
<td>2017</td>
<td>Other</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Hashim and Alizadeh, 2018)</td>
<td>2018</td>
<td>Sign language recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Kamal and Hassani, 2020)</td>
<td>2020</td>
<td>Sign language recognition</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Daneshfar et al., 2009)</td>
<td>2009</td>
<td>Speech recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Barkhoda et al., 2009)</td>
<td>2009</td>
<td>Speech recognition</td>
<td>no</td>
<td>no</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Shahrampour et al., 2009)</td>
<td>2009</td>
<td>Speech recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Hassani and Karimnezhad, 2011)</td>
<td>2011</td>
<td>Speech recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani</td>
</tr>
<tr>
<td>(Dimler and Karamber, 2017)</td>
<td>2017</td>
<td>Speech recognition</td>
<td>no</td>
<td>no</td>
<td>Kurmanji</td>
</tr>
<tr>
<td>(Dimler and Aydin, 2018a)</td>
<td>2018</td>
<td>Speech recognition</td>
<td>no</td>
<td>yes</td>
<td>Sorani, Kurmanji</td>
</tr>
<tr>
<td>(Qader and Hassani, 2019)</td>
<td>2019</td>
<td>Speech recognition</td>
<td>yes</td>
<td>yes</td>
<td>Sorani</td>
</tr>
</tbody>
</table>
Figure A.5: The Package and class models of KLPT in the Unified Modeling Language (UML)