Towards a Comprehensive Dictionary of Middle Persian

Francisco Mondaca[1], Kianoosh Rezania[2], Slavomír Čéplö[3], Claes Neuefeind[4]

[1] [4]Cologne Center for eHumanities, University of Cologne
[2] [3]Center for Religious Studies, Ruhr University Bochum
E-mail: f.mondaca@uni-koeln.de, kianoosh.rezania@rub.de, slavomir.ceploe@rub.de, c.neuefeind@uni-koeln.de

Abstract

This paper discusses the process of developing a flexible and comprehensive model for a bilingual corpus dictionary with a dead language, in this case, Zoroastrian Middle Persian, as the source language, and a particular focus on accommodating termini technici and multi-word expressions. Advanced search capabilities are achieved through the integration of state-of-the-art technologies, with plans to further enhance the system by implementing advanced natural language processing techniques. The project offers two distinct API solutions to cater to diverse user needs and ensure efficient access to lexical data. One of these is a dedicated API designed specifically for the web application. The other is a REST API, which simplifies data access and promotes scalability. The project acknowledges the potential for future integration with large language models, underlining the prospect for future enhancements. This approach encourages collaboration and innovation in historical linguistics, highlighting the crucial role of adaptable and cutting-edge technologies in developing a robust lexicon for historical languages.

Keywords: corpus-based dictionary; middle persian; api; rest; graphql

1. Middle Persian Language and Texts

1.1 Middle Persian

Middle Persian was spoken in the province Persis (Fārs) in the first millennium CE. It served as the official language of the Sasanian Empire (224 - 651 CE), a dominant power beside the Roman Empire during late antiquity. Middle Persian derives from Old Persian, the language spoken in the same area until the third century BCE. In the last centuries of the first millennium CE, Middle Persian has developed into New Persian, the major language of people in today's Iran, Afghanistan, and Tajikistan. With an estimated 800,000 words, the Middle Persian corpus is undoubtedly the most comprehensive among Old and Middle Iranian text corpora.

In the vast territory of the multi-lingual and multi-national Sasanian Empire, Middle Persian served as a sort of lingua franca. As a vital linguistic bridge, it significantly contributed to the transcultural communication between diverse regions and civilizations during the first millennium CE. In this role, it enabled the exchange of ideas, knowledge, and cultural practices and fostered a rich and interconnected environment. Middle Persian's prominence within the Sasanian empire highlights its importance in both the administrative and cultural spheres, facilitating effective governance and fostering cultural communication across the empire's vast territories.

One of the cultural fields in which Middle Persian played a prominent role was religion. It was the language of choice for the documentation of numerous religious traditions, including Zoroastrianism, Manichaeism, and Christianity. Whereas Middle Persian texts are transmitted fragmentary to us, religious texts build the largest Middle Persian corpora. Zoroastrian Middle Persian (ZMP) texts account for more than 85 percent of the whole corpus, the Manichean ones to circa five percent. Administrative texts and Sasanian and post-Sasanian inscriptions constitute the rest of the Middle Persian corpus.

Different scripts have been used to write down Middle Persian texts. Sasanian inscriptions are written in the inscriptional Pahlavi script, Manichaean texts in the Manichaean, and most Zoroastrian and administrative texts in the Pahlavi cursive script. In the post-Sasanian period, Zoroastrians also used other scripts, such as Avestan and Perso-Arabic, to document their texts.

Middle Persian texts have been written on a wide range of materials. Inscriptions are engraved in stone and metal coins and seals, administrative texts are transmitted via papyri, parchment, and ostraca. Most extant Zoroastrian and Manichean texts are written on paper whereas the Zoroastrian texts written in the Pahlavi script are transmitted in codices. Some of these texts can be dated only roughly to the last centuries of the Sasanian period, more than half of them have been authored in the early Islamic period, specifically in the ninth and tenth centuries CE. The oldest extant Zoroastrian codices date back to the 14th century (Macuch, 2009; Rezania, 2023).

1.2 Zoroastrian Middle Persian

ZMP texts are of paramount importance in late antiquity, in terms of linguistic history, history of religions, cultural development, and scientific thought. These texts not only transmit and interpret ancient Iranian religious ideas but also showcase intellectual connections with other religions and cultures of late antiquity. However, the current state of scholarly engagement with these texts is uneven. While many unsophisticated texts have been repeatedly edited, some complex texts remained relatively neglected.

The ZMP texts transmitted in codices can be categorized into the following genres:

- Zand texts, i.e. Middle Persian translations of and commentaries on the Avestan texts
- Zand-related literature, i.e. texts based on Zand texts
- Theological works, mostly from the early Islamic period
- Juridical literature
- Moral-didactic literature
- Narrative-historical or mythological works
- Ritual literature, texts dealing with the performance of Zoroastrian rituals

In essence, ZMP literature comprises two related text layers: Zand, which is primarily a legacy of the Sasanian period (with some later additions), and ninth-century theological literature. An intermediate textual layer, the Zand-related literature is closely dependent on Zand. Besides this multi-layered religious textual material, several other significant texts have been preserved. Although these texts belong to the Zoroastrian literary corpus,

they are not strictly religious in the narrow sense. Among them are epic texts with religious motifs that are not part of the Zoroastrian doctrinal literature. Some of these texts were translated into Arabic and New Persian and are frequently cited in Muslim compositions.

2. Zoroastrian Middle Persian: Digital Corpus and Dictionary (MPCD)

A comprehensive digital corpus of Middle Persian literature is a desideratum. There are two attempts to this end: the TITUS project[1] and the Pārsīg Database[2]. Both corpora are neither based on manuscripts, nor provide an extensive annotation, nor a dictionary. The MPCD project aims to address this gap.

To provide a platform for the study of Zoroastrian Middle Persian texts, the MPCD project[3] aims at creating an annotated corpus of ZMP texts written in Pahlavi script and transmitted via codices. It will also provide a comprehensive corpus-based dictionary covering the whole lemmata in the corpus. Our corpus annotation consists of four layers:

- orthographical (transliteration) and phonographical (transcription)
- grammatical (morpho-syntactic)
- semantic
- intertextual (linking Zand texts with their Avestan original)

Within the overall structure of the project, the corpus and dictionary function as closely interrelated analytical tools. For this purpose, the project employs a web-based working environment that facilitates collaborative work on the corpus and dictionary. For the sake of verifiability, all texts in our corpus are linked to the images of the corresponding folios in the codices. In doing so, users have the possibility to switch from the dictionary via the corpus to the folio images and vice versa.

The project intends to create a platform to also be used in the future for the remaining Middle Persian sub-corpora, as well as for the corpora of other Middle Iranian languages. It is thus conducted with a view to the eventual creation of a complete dictionary of Middle Persian (incorporating all of its sub-corpora) as well as to an expansion of the corpus into the domains of other Middle Iranian languages.

3. Middle Persian Lexicography

3.1 Traditional Lexicography

Compared with ZMP, the lexicographical analysis of Manichaean Middle Persian and Middle Persian inscriptions has progressed more significantly. Durkin-Meisterernst (2004) nearly covers the whole Manichaean Middle Persian vocabulary, offering English equivalents, full attestations for each lemma, and bibliographies for scholarly discussions of individual words. This renders older lists obsolete. The vocabulary of Sasanian royal inscriptions is documented in Back (1978), while Gignoux (1972) and Humbach & Skjærvø (1980, 1983)

[1] https://titus.uni-frankfurt.de/indexd.htm
[2] https://www.parsigdatabase.com/
[3] https://www.mpcorpus.org

remain valuable resources. However, revisions and updates are needed to incorporate newly discovered rock inscriptions, seals, and numismatic materials. The status of ZMP literature, the most extensive corpus, remains challenging. Three dictionaries were created in the 20th century: Nyberg (1928), MacKenzie (1971), and Nyberg (1974). These relatively brief works, based on a few non-Zand Pahlavi texts, contain approximately 3,100 lemmata in MacKenzie's dictionary and ca. 3,000 lemmata in Nyberg's Manual of Pahlavi. Although MacKenzie's work is more popular due to its adoption of the currently preferred transcription system, Nyberg's entries provide references and textual quotations. Both works contain some etymological information. For Zand literature whose vocabulary is not included in these dictionaries, researchers must rely on lexicographical works such as Dhabhar (1949) and Kapadia Kapadia (1953). For theological works from the early Islamic period, the largest part of the ZMP corpus, researchers should be satisfied with glossaries accompanying text editions. They are all valuable means, with their limitations, however.

3.2 Digital Lexicography

The advent of digital media in lexicography has transformed the concept of a corpus and its importance for lexicographical work (Granger et al., 2012). A digital corpus is characterized by its interconnection with the lexicon, enabling accurate representation of various levels of primary data, metadata, and structural and linguistic insights that can be utilized in lexicographic analysis. By creating a mesostructure (i.e., establishing internal connections within the lexicon) and assigning (semasiologically organized) lemmata to taxonomical concepts, it becomes possible to implement onomasiological access to the dictionary. The MPCD dictionary is the first attempt in Old and Middle Iranian Linguistics to fulfill this wish.

4. Modeling a Middle Persian-English Dictionary

4.1 A Pragmatic Approach to Dictionary Compilation

To fully grasp the compilation process, it is essential to understand the joint efforts and shared roles between philologists and computational linguists on this project. On the one hand, philologists begin their annotation tasks utilizing computationally pre-annotated texts as a base. On the other hand, computational linguists concentrate on creating a platform that enables the interconnection of corpora and dictionaries. In this process, computational linguists modify a React application developed for Kosh (Mondaca et al., 2019), a framework designed for developing and maintaining APIs for dictionaries. The glossaries and dictionaries added to Kosh are employed by philologists to search for and enrich their annotations and the dictionary. To facilitate collaboration between the two fields and to address potential issues, we jointly developed our data model using RelaxNG Compact[4], a syntax similar to EBNF (Extended Backus-Naur Form). This cooperative methodology allows both teams to effectively communicate their ideas and needs. At the same time, this model aids the computational linguists in the project in devising a Django[5]-based model that relies on PostgreSQL, a relational database management system, as its foundation. It is crucial to account for the scope of the dictionary and

[4] RELAX NG's Compact Syntax
[5] https://www.djangoproject.com

its possible complexities from the beginning, as modeling a dictionary for a relational database requires the establishment of a flexible model. This approach helps prevent intricate refactoring processes during the process of compiling the dictionary, which could be difficult and time-consuming to resolve.

4.2 Dictionary Compilation as Part of the Annotation Process

In the interim period, i.e. before a common interface that integrates both the corpus and dictionary is available to the philologists, they annotate and refine the corpus texts on spreadsheets. A consensus has been reached between computational linguists and philologists to employ an extended version of the CoNLL-U format [6]. The rationale behind this extension is multifaceted, encompassing the provision of lexicographic data as well as details concerning the physical location of tokens within the context of a manuscript. Furthermore, the extension facilitates the conveyance of pertinent information regarding a text's metastructure, including elements such as chapters and sections, thus contributing to a more comprehensive understanding of the text's organization and layout. While modeling the data in RelaxNG provides a conceptual foundation and a depiction of the desired dictionary structure, the actual outcomes in projects are often significantly impacted by the accessible data. As a result, our principal emphasis is to employ a pragmatic strategy. This involves examining the existing data in the CoNLL format and to take into account the philologists' requirements delineated in the RelaxNG schema in order to create a straightforward and potentially flexible data model using the Django framework.

4.3 Extending CoNLL-U for Lexicographic Purposes

In order to address specific lexicographical requirements, two additional columns have been introduced in addition to the standard CoNLL-U "lemma" field: "meaning" (see Figure 1) and "term_tech" (not in the Figure). The "lemma" and "meaning" columns are intended to provide the lemma and meaning associated with each token, while the "term_tech" column contains a selection from a finite set of values used to identify the category of the terminus technicus if the lemma in question is such one, such as "judicial", "religious", and so forth. These technical terms are translated and explained by philologists in a separate shared spreadsheet. This serves as a foundation for the explanation of Zoroastrian technical terms to be applied later in the dictionary. It is crucial to acknowledge that a lemma and its corresponding meaning may not always align perfectly in a one-to-one relationship for a given token. In other words, a single lemma could be associated with multiple meanings. This distinction is significant and should be considered when developing the dictionary model within the Django framework. A notable challenge encountered by the philologists during annotation pertained to the handling of multi-word expressions (MWEs) (Měchura, 2016, 2018). MWEs are not only a crucial aspect of lexicography but also have practical implications in our project. As computational linguists, our responsibility extends beyond merely modeling MWEs; we must also parse this information consistently. To address this issue, we collaborated with the philologists to denote each MWE in the extended CoNLL-U file by marking the corresponding transcription and transliteration fields with an underscore (Figure 2). This marker indicates the presence of an MWE. While processing each sentence, we then search the list of parsed lemmata for the lemmata associated with the MWE and link them accordingly.

[6] https://universaldependencies.org/format.html

LEMMA	MEANING	UPOS	FEATS
pad	in	ADP	AdpType=Prep
nām	name	NOUN	Animacy=Inan
ud	and	CCONJ	_
šnāyišn	honour	NOUN	Animacy=Inan
ī	ezafe	DET	_
wisp-sūd	all-beneficial	ADJ	_
dādār	creator	NOUN	Animacy=Anim
ohrmazd	Ohrmazd	PROPN	NameType=Giv|Animacy=Anim|Transc=Yes

Figure 1: Snippet of the extended CoNLL-U file

huruāxm.	urwāhm	joy
i.	ī	ezafe
və/hą̇.	weh	good person
vašōvə̱t̰.	wišuftan	destroy
/	$	$
u.	ud	and
āštī.	āštīh	peace
bā.	bē	away
barə̱t̰.	burdan	carry
_	bē burdan	drive away
u.	ud	and
anāštī.	anāštīh	discord
aṇ/dar.	andar	in
āβarə̱t̰.	āwardan	bring
_	anāštīh andar āwardan	breed discord

Figure 2: Handling Multi-Word Expressions

5. Modeling Lexical Data with RelaxNG Compact Syntax

5.1 Using RelaxNG Compact Syntax for Effective Scholarly Modeling and Collaboration

The RelaxNG Compact syntax is highly valued in scholarly applications for its concise nature, expressive power, and validation capabilities. This syntax provides a clear and succinct representation of schema specifications, facilitating easy comprehension and modification during development. Its expressiveness allows for the creation of complex models while still maintaining readability. The syntax's adaptability caters to a wide range of modeling requirements and accommodates evolving project needs. Moreover, its compatibility fosters interdisciplinary collaboration by offering a shared language for schema development, enhancing communication between philologists and computational linguists. Lastly, the syntax supports XML document validation against a schema, ensuring data integrity and early identification of potential issues. These benefits make the RelaxNG Compact syntax suitable for scholarly projects requiring efficient collaboration and the creation of intricate, adaptable models.

5.2 Preliminary RelaxNG Model: An Overview and Its Role in the Project

It is key to understand that the existing RelaxNG model primarily serves as a tool for reflection, rather than a definitive schema for our project. The primary purpose of the RelaxNG model is to foster communication between the philologists and the computational linguists, allowing the philologists to convey their vision of the dictionary

```
start = dictionary # dictionary metadata in TEI header

dictionary = element dictionary {taxonomy, entry+} # reference to taxonomy and entries

taxonomy = attribute taxonomy {string} # the taxonomy is defined in taxonomy.rng

entry = element entry {
    attribute entryId {xsd:ID}, # unique ID for the entry
    element lemma {xsd:string}, # the normalized form
    headword?, # If headword is NULL, lemma (and POSs) will be showed as the headword of the entry
            # We will use headword for e.g. verbs with different infinitive forms and present stems
    language,
    # We use crossRef for linking different lemmata to one entry,
    # for example different forms/readings of one lemma
    attribute crossRef {xsd:IDREF}, # if the entry is a cross reference the other elements will be empty

    # TODO: to generally follow TEI, we can put the following information into this structure:
    # form (including orth, forms), gramGrp (pos), def (hierarchizedMeanings = senses)
    morphology, # different components shown in different locations, see infra
    timeline, # shows the frequency of the occurrences of the lemma in different cent.
    element relativeFrequency {xsd:int}, # from corpus, per 100,000

    types, # attested types in the corpus linked to the lemma
    orthographicVariants, # union of transliterations
    occurrences,
    hierarchizedSenses,
    internalReferences?,
    equivalents?,
    etymology?,
    biblioEntry?,
    comment?,

    attribute stage {"inprogress" | "finished" | "published"},
    attribute DOI {xsd:anyURI}
}
```

Figure 3: Entry element in RelaxNG - Current status

and its relationship to the corpus. Although the RelaxNG model is crucial to the overall process, the practical implementation and effectiveness of the schema are achieved at the Django level due to its practical implications. Figure 3 and 4 offer an overview of the "entry" model, specifically focusing on the elements "lemma" and "hierarchizedSenses".[7] A considerable challenge lies in identifying the most appropriate taxonomy to employ, as a consensus has not yet been reached. The continuing discussions and adjustments will aid in the development of a comprehensive and effective model, ensuring a solid foundation for the integration of the dictionary and corpus.

6. Modeling Lexical Data in Django

The MPCD project has opted to employ the Django web framework due to its nature as a proven and mature technology with a large user base and support network. This, as evidenced by its use in a number of major open-source[8] and commercial[9] applications, ensures that the technical foundation of the project can remain supported beyond the life span of the project itself. The framework's built-in tools and ability to accommodate varying levels of complexity contribute to streamlined project advancement and expansion.

[7] The RelaxNG data model for our corpus and dictionary is available on GitHub: https://github.com/middlepersian/relaxng_model

[8] https://github.com/mozilla/pontoon

[9] https://www.instagram.com/

Django's features along with its modular architecture, contribute to the stability and maintainability of the MPCD project. The availability of a supportive community and extensive documentation enhances the integration of various libraries and tools. Furthermore, the implementation of a distinct app for the dictionary has the potential to streamline development and debugging processes, due to improved code organization, maintainability, and reusability. This methodological approach should facilitate subsequent project integrations and augmentations, contributing to the project's long-term viability.

6.1 Base Models

Upon examining the data in the CoNLL-U extended files and the RelaxNG model, it becomes evident that the project must address lemmata, meanings/senses, taxonomic references, and MWEs. Simultaneously, it is essential to develop a flexible model to handle and organize semantic information. By considering these aspects, the resulting model will be better equipped to manage the complexity and nuances inherent in lexical data. It is worth noting that, as of the time of writing, the final model for the dictionary has not been determined. Internal discussions are ongoing to establish the most suitable model. Additionally, we are incorporating insights from our discussions with other scholars in the field during the international workshop, "Towards a Comprehensive Middle Persian Dictionary," held in April 2023 at the University of Cologne. This collaborative approach is set to significantly bolster the development of a robust and comprehensive model for the project.

6.2 Lemma Model

At the heart of the dictionary application lie the "lemma" and "meaning" models. Although the RelaxNG model presents a more intricate structure outlining the appearance of an entry, the "lemma" and "meaning" models in Django are specifically designed for maximum flexibility. These models not only mirror the columns in the CoNLL-U extended files but also represent a graph, facilitating an adaptable and robust foundation for handling linguistic data.

Měchura (2016, 2018) identifies the challenges associated with managing MWEs in tree structures for lexical data and proposes addressing them as "shareable entries". He delves into the implications of using tree structures or graphs, especially considering the human perspective. As humans may struggle with the sometimes intricate nature of graphs, XML trees serve as a compromise between human comprehension and computational processing, despite the enhanced flexibility and ease of handling offered by graphs. A primary concern with MWEs in tree structures lies in their inability to possess multiple parent nodes. Měchura introduces "graph-augmented trees" as a solution, enabling the sharing of MWEs among entries and their subsequent serialization in XML, in compliance with the XLink standard. This approach fosters more efficient and adaptable handling of MWEs in lexicographic data. Měchura's method aligns with a feature of the Lexical Markup Framework (LMF)[10], where multi-word entries can exist independently and be linked to specific senses of other entries through their ID. Our project also adopts this strategy, facilitating the seamless integration and management of MWEs in the lexicographic data

[10] https://www.lexicalmarkupframework.org

```
stem = element stem {
    attribute type {"past" | "present"},
    xsd:string
    }

# In which centuries a lemma is attested can be retrieved from the metadata of the texts,
# in which the texts are authored.
timeline = element timeline {
    attribute century {xsd:int},
    attribute freq {xsd:decimal} # frequency of the tokens linked to the lemma in the century
}

hierarchizedSenses = element hierarchizedSenses { # one per entry
    semantic+
}

semantic = element semantic {
    attribute serialId {xsd:int}, # internal numbering

    # reference to the parent node to produce a flexible hierearchy
    # In MPCD, we will often use only two levels for grouping the senses.
    # Nevertheless, we would like to be able to produce a hierarchy with
    # more levels in case of more complex lemmata.
    node?,
    semanticCore,
    element morphologicalForms {morphologicalForm+},
    element mwes {mwe*} # alphabetic sorted
}

semanticCore = element semanticCore {
    element sense {xsd:string}, # in the case of term. tech. = its definition

    # explanation may include detailed description of the use of the lexeme,
    # reasons for transcription, and prototypical examples
    element explanation {xsd:string}?,

    # grammar may include some grammatical explanations about the lemma
    # including valences
    element grammar {xsd:string}?,

    # reference to the taxonomy
    # We will take the liberty to eventually link a sense to more than one concept
    element semanticDomain {concept+}?,
    termTech?,
```

Figure 4: Selected elements from an entry in RelaxNG - Current status

structure. Currently, our software functions as a relatively basic yet efficient dictionary writing system, and we by now have no immediate or long-term plans to offer XML-based views to users. Nevertheless, to comprehend the process of serializing data in XML format is essential for improving the system's capabilities and to manage the lexicographic data generated by our project effectively. Acquiring this knowledge will facilitate the continuous refinement and expansion of the software, ensuring that it remains a valuable and adaptable resource for both users and researchers.

During the development phase, we have intentionally not implemented an entry model, as its final structure is still evolving. However, based on our current understanding, we have identified the core components that are likely to be included in the entry. In addition to standard elements such as word(form) and language, a lemma may possess

```python
class Lemma(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid_lib.uuid4, editable=False)
    word = models.CharField(max_length=100)
    language = models.CharField(max_length=3, null=True, blank=True)
    categories = ArrayField(
        models.CharField(max_length=50, blank=True, null=True), null=True, blank=True
    )
    multiword_expression = models.BooleanField(default=False)
    related_lemmas = models.ManyToManyField(
        "self", blank=True, related_name="lemma_related_lemmas", through="LemmaRelation"
    )
    related_meanings = models.ManyToManyField(
        "Meaning",
        blank=True,
        related_name="lemma_related_meanings",
        through="LemmaMeaning",
    )
    created_at = models.DateTimeField(auto_now_add=True)

    history = HistoricalRecords()

    class Meta:
        constraints = [
            models.UniqueConstraint(
                fields=["word", "language"], name="word_language_lemma"
            )
        ]
        indexes = [
            models.Index(fields=["word", "language"]),
        ]
        ordering = ["word"]
```

Figure 5: Lemma Model in Django - Current status

related objects of the same type that can be categorized into a specific classification (e.g., term_tech) or classified as a MWE. Furthermore, it may have associated meanings. This preliminary understanding informs our current approach, allowing for greater flexibility and adaptability as the project evolves. The inherent simplicity of our system enables a lemma to be associated with multiple lemmata and meanings, even across different languages. Although the primary objective of this project is to develop a Middle Persian-English dictionary, we have designed the model with the potential to create multiple dictionaries with the existing lemmata, rather than exclusively focusing on a single language pair. This flexibility allows for broader applications and adaptability in various linguistic contexts.

6.3 Meaning Model

The meaning model (Figure 6) is simple, yet effective. It comprises the meaning itself, an associated language, and a boolean flag indicating whether the meaning is related to a lemma. This design choice stems from the existence of another model, section, which can have a meaning. Sections are especially useful for sentences, as we consider them as ranges of tokens. By incorporating this boolean field, we can more efficiently filter meanings without the need to create additional models for sentence translations or other potential objects that may require translations in the future. This streamlined approach promotes versatility and adaptability for a variety of translation requirements.

```python
class Meaning(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid_lib.uuid4, editable=False)
    meaning = models.TextField(null=True, blank=True, db_index=True)
    lemma_related = models.BooleanField(default=True)
    language = models.CharField(max_length=10, blank=True, null=True, db_index=True)
    related_meanings = models.ManyToManyField('self', blank=True, related_name='meaning_related_meanings')
    created_at = models.DateTimeField(auto_now_add=True)


    def related_lemmas(self):
        return self.lemma_related_meanings.all()

    history = HistoricalRecords()
    class Meta:
        constraints = [
            models.UniqueConstraint(
                fields=['meaning', 'language'], name='meaning_language_meaning'
            )]
        ordering = ['meaning']
        indexes = [
            models.Index(fields=['meaning', 'language']),
```

Figure 6: Meaning Model in Django - Current status

6.4 Token Model

At the heart of the platform lies the token model, which has been designed to connect with one or more lemmata and meanings. This design allows lemmata and meanings to relate to each other, with the relationship originating from the lemma. Although the relationship is symmetrical, enabling access to the lemma from the meaning, it must be initiated from the lemma. By parsing data from the extended CoNLL-U files, we can create lemmata along with their associated meanings and independently link lemmata and meanings to a token. This approach enables us to access tokens within meanings, providing us with increased flexibility and adaptability in managing the corpus data.

7. Exploring Middle Persian Lexical Resources

7.1 APIs for Efficient and Collaborative Historical Linguistics Research

APIs function as vital intermediaries that enable communication and data exchange between disparate software components, playing a critical role in modern software development. As highlighted by Amundsen (2020), they offer several key advantages, including reduced computational time and cost, facilitated ease of computations, and the ability to tackle previously unresolved issues. Furthermore, APIs contribute to standardization by providing a consistent and structured method for software components to interact, simplifying the development process. They also promote modularity, allowing developers to create and modify individual components without disrupting the entire system, thus enhancing maintainability and flexibility. APIs facilitate extensibility, as they enable software to be easily expanded or integrated with new features and services. Additionally, they improve security by enabling controlled access to specific functionalities, ensuring that sensitive data remains protected. APIs hold significant relevance for lexical data in historical languages such as Middle Persian, primarily due to their capacity to facilitate access, promote interoperability, and support data enrichment. By offering standardized methods for interaction between software applications, APIs enable researchers and developers to access linguistic information without in-depth knowledge of the underlying data structure. This accessibility encourages collaboration and allows for integration of lexical data from

```python
class Token(models.Model):
    id = models.UUIDField(primary_key=True, default=uuid_lib.uuid4, editable=False)
    number = models.FloatField(null=True, blank=True)
    number_in_sentence = models.FloatField(blank=True, null=True)

    root = models.BooleanField(default=False)
    word_token = models.BooleanField(default=True)
    visible = models.BooleanField(default=True)

    text = models.ForeignKey('Text', on_delete=models.CASCADE, null=True, blank=True, related_name='token_text')
    image = models.ForeignKey('images.Image', on_delete=models.CASCADE, null=True, blank=True, related_name='token_image')

    language = models.CharField(max_length=3, null=True, blank=True)
    transcription = models.CharField(max_length=50)
    transliteration = models.CharField(max_length=50, blank=True)
    lemmas = models.ManyToManyField('dict.Lemma', blank=True, through='TokenLemma', related_name='token_lemmas')
    meanings = models.ManyToManyField('dict.Meaning', blank=True, through='TokenMeaning', related_name='token_meanings')

    avestan = models.TextField(null=True, blank=True)

    previous = models.OneToOneField('self',
                                    related_name='next',
                                    blank=True,
                                    null=True,
                                    on_delete=models.SET_NULL, db_index=True)

    gloss = models.TextField(blank=True, null=True)

    multiword_token = models.BooleanField(default=False)
    multiword_token_number = ArrayField(models.FloatField(blank=True, null=True), null=True, blank=True)
    related_tokens = models.ManyToManyField('self', blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

    history = HistoricalRecords()
```

Figure 7: Token Model in Django - Current status

various sources, fostering innovative research approaches and insights. Furthermore, APIs enable efficient updating and customization, ensuring that users have access to the most current and accurate information tailored to their specific research needs. In the context of the project, two distinct APIs have been developed: a REST (Representational State Transfer) API (Fielding, 2000) and a GraphQL API (GraphQL, 2021). Each offers unique advantages that cater to different user requirements and preferences, enhancing efficiency, flexibility, and collaboration across various platforms and systems. The MPCD project has been designed with an API-centric approach. Through its web application, it offers access to APIs for both the dictionary and corpus apps, which are supported by Django and specific API libraries. Furthermore, the web application displays the most recent publications related to the project through the Zotero API. A team of philologists diligently curates a Zotero catalog featuring a comprehensive bibliography of Middle Persian literature and publications associated with the project.

7.2 REST API

The REST API follows a well-defined architectural style, which simplifies the development process due to its standardized and straightforward nature. It is designed to support scalability, thereby ensuring seamless access to data and services as the user base grows. Moreover, the REST API enables caching of responses, reducing the load on servers and enhancing performance for frequently accessed data. To further facilitate the development process and improve accessibility for developers, we have integrated a comprehensive and interactive documentation interface for the API. This integration allows developers to explore the API's endpoints, understand the data structures, and test API requests and responses directly within the documentation. By providing this interface, we aim to foster

a user-friendly environment for developers to interact with and utilize our lexical data in a more efficient and effective manner.

7.2.1 Integration with Large Language Models

The adoption of a REST API offers significant advantages, such as the potential for interoperability with large language models. This approach enables seamless integration of our lexical data with advanced natural language processing systems, supporting the development of plugins to enhance user experience and expand the applications of historical language data. The flexibility of this project promotes collaboration and innovation within the field of historical linguistics, aligning it with the rapidly evolving sphere of artificial intelligence and language modeling research. Using APIs has been proven to be effective when applied to models with fewer parameters than large-scale language models, indicating promising future developments in this area (Schick et al., 2023).

7.3 GraphQL API

The GraphQL API offers flexible querying, allowing clients to request precisely the data they need, which results in enhanced efficiency by reducing the amount of unnecessary information transferred. The GraphQL API uses a single endpoint for all data requests, simplifying the management of multiple resources and streamlining the development process. One of the reasons GraphQL has become so popular among frontend developers is its ability to enhance productivity and improve the overall development experience. With its strong typing system, developers can easily discover the available data and understand the structure of the API, which leads to fewer errors and better maintainability.

7.4 React Web Application

In our web application, which is built using the React JavaScript library and the Relay data-fetching framework, we have opted to employ the GraphQL API. This choice offers several advantages in the context of a React-based application. GraphQL's flexibility in querying allows the React components to request precisely the data they need, ensuring an efficient data transfer and minimizing over-fetching. This feature is particularly beneficial in the context of a dynamic web application, where various components might have specific data requirements. Furthermore, the combination of GraphQL with Relay allows for seamless integration with the React library, enabling efficient and scalable data-fetching operations. Relay manages the GraphQL requests and optimizes data fetching, reducing the complexity of managing data within the application and promoting a more maintainable codebase.

8. Search

One of the most pertinent features a dictionary should possess is a well-organized macrostructure to facilitate effective information retrieval. Digital dictionaries hold a distinct advantage over their printed counterparts in terms of accessing efficient search modalities, as opposed to the wordlists commonly found in printed dictionaries. Although

traditional databases, including both NoSQL and SQL types, offer full-text search capabilities, they are not specifically designed for search purposes like dedicated search engines. To address this need, we employ Elasticsearch[11] as a search engine, enhancing the search experience and improving the overall usability of the digital dictionary.

8.1 Main Search Modi

Prefix, regular expression (regexp), match, and wildcard searches are advantageous for retrieving lexical data due to their ability to accommodate diverse query patterns and efficiently filter relevant information from extensive datasets. These search techniques enable users to explore linguistic data with greater precision and flexibility. Prefix searches allow users to identify words or phrases beginning with a specified sequence of characters, which is particularly helpful when investigating lexical data for morphological patterns or etymological connections. By focusing on the initial characters, prefix searches facilitate the discovery of related terms and enable researchers to gain insights into language structure and development. Regular expression searches, or regexp searches, offer a powerful method for retrieving lexical data based on complex patterns. By employing a combination of symbols and characters, users can create highly specific search criteria that match a wide variety of linguistic patterns. This flexibility allows researchers to uncover intricate relationships between words or phrases and investigate language phenomena that may otherwise be difficult to discern. Match searches provide a more straightforward approach to lexical data retrieval, locating exact or partial matches within the dataset. This method proves advantageous in cases where users are searching for specific terms or phrases, as it ensures that only the most relevant results are returned. Match searches contribute to the efficiency and accuracy of information retrieval in linguistic research. Wildcard searches introduce an additional layer of flexibility by allowing users to substitute one or more characters within a search query with a wildcard symbol. This functionality enables researchers to locate words or phrases with varying character combinations, accommodating uncertain or incomplete search criteria.

8.2 Implementing Semantic Search

Elasticsearch enables the incorporation of vector embeddings into the search process, an approach commonly referred to as semantic search. The advantage of this method lies in its ability to retrieve documents, lemmata, or meanings that are related to the search term or the terms found, based on their appearance in a corpus or their relationships within a language model. While numerous embedding models are available for English, resources for Middle Persian are lacking. Consequently, the initial implementation of vector search will focus on representing Middle Persian meanings in English. For this purpose we will develop a word-embedding model for Middle Persian. This approach aims to facilitate the discovery of meaningful connections and insights in Middle Persian linguistic data, thereby enhancing the overall search experience and supporting more comprehensive research.

9. Future Enhancements

Integrating a dictionary with a corpus requires technical development and effective communication between philologists and computational linguists. As we continue to refine

[11] https://www.elastic.co/

and expand this integration, we are actively seeking areas for improvement and further development.

In an effort to facilitate access to our data for as many researchers as possible, we will provide our corpus data in both TEI and CoNLL formats. Moreover, we might be able to make our lexical data available in both TEI-Lex0 (Tasovac et al., 2018) and Ontolex (McCrae et al., 2017) formats. Additionally, we aim to integrate our data with large language models, which are undergoing rapid development at the time of writing.

10. References

Amundsen, M. (2020). *Design and build great web APIs: robust, reliable, and resilient*. The pragmatic programmers. Raleigh, North Carolina: The Pragmatic Bookshelf.

Back, M. (1978). *Die sassanidischen Staatsinschriften. Studien zur Orthographie und Phonologie des Mittelpersischen der Inschriften zusammen mit einem etymologischen Index des mittelpersischen Wortgutes und einem Textcorpus der behandelten Inschriften*. Number 18 in Acta Iranica. Leiden: Brill.

Dhabhar, B.N. (1949). *Pahlavi Yasna and Visperad*. Bombay: The Trustees of the Parsee Punchayet Funds and Properties.

Durkin-Meisterernst, D. (2004). *Dictionary of Manichean Middle Persian and Parthian (Dictionary of Manichaean Texts. Vol. III Texts from Central Asia and China, Part 1)*. Turnhout: Brepols.

Fielding, R.T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. thesis, University of California, Irvine. URL https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

Gignoux, P. (1972). *Glossaire des inscriptions pehlevies et parthes*. Number 1 in Corpus Inscriptionum Iranicarum, Supplementary series. London: Lund Humphries.

Granger, S., Paquot, M., Granger, S. & Paquot, M. (eds.) (2012). *Electronic Lexicography*. Oxford, New York: Oxford University Press.

GraphQL (2021). GraphQL Specification. https://spec.graphql.org. Accessed: April 18, 2023.

Humbach, H. & Skjærvø, P.O. (1980). *The Sassanian inscription of Paikuli. Part 2: Synoptic Tables*. Wiesbaden: Dr. Ludwig Reichert.

Humbach, H. & Skjærvø, P.O. (1983). *The Sassanian inscription of Paikuli. Part 3.1: Restored text and translation. Part 3.2: Commentary*. Wiesbaden: Dr. Ludwig Reichert.

Kapadia, D.D. (1953). *Glossary of Pahlavi Vendidad*. Bombay.

MacKenzie, D.N. (1971). *A concise Pahlavi dictionary*. London: Oxford University Press.

Macuch, M. (2009). Pahlavi literature. In R.E. Emmerick & M. Macuch (eds.) *The literature of pre-Islamic Iran. Companion volume I to 'A history of Persian literature'*, number 17 in A history of Persian literature. London: I.B. Tauris, pp. 116–196.

McCrae, J.P., Gil, J., Gràcia, J., Bitelaar, P. & Cimiano, P. (2017). The OntoLex-Lemon Model: Development and Applications. In *Electronic lexicography in the 21st century. Proceedings of eLex 2017 conference*. URL https://elex.link/elex2017/wp-content/uploads/2017/09/paper36.pdf.

Mondaca, F., Schildkamp, P. & Rau, F. (2019). Introducing Kosh, a Framework for Creating and Maintaining APIs for Lexical Data. In *Electronic Lexicography in the 21st Century. Proceedings of the eLex 2019 Conference, Sintra, Portugal*. Brno: Lexical Computing CZ, pp. 907–21. URL https://elex.link/elex2019/wp-content/uploads/2019/09/eLex_2019_51.pdf.

Měchura, M. (2016). Data Structures in Lexicography: from Trees to Graphs. In *The 10th Workshop on Recent Advances in Slavonic Natural Languages Processing, RASLAN 2016, Karlova Studanka, Czech Republic, December 2-4, 2016*. pp. 97–104. URL http://nlp.fi.muni.cz/raslan/2016/paper04-Mechura.pdf.

Měchura, M. (2018). Shareable Subentries in Lexonomy as a Solution to the Problem of Multiword Item Placement. In J. Čibej, V. Gorjanc, I. Kosem & S. Krek (eds.) *Proceedings of the XVIII EURALEX International Congress: Lexicography in Global Contexts*. Ljubljana, Slovenia: Ljubljana University Press, Faculty of Arts, pp. 223–232.

Nyberg, H. (1928). *Hilfsbuch des Pehlevi*. Uppsala: Almquist & Wiksell. {:issued:1928/1931}.

Nyberg, H. (1974). *A manual of Pahlavi. Part II: Glossary*. Wiesbaden: Harrassowitz.

Rezania, K. (2023). Zoroastrianism in Early Islamic Period: its participation in ᶜAbbāsid theological-philosophical discourse and its absence in the transmission of Sasanian culture. In S. Heidemann & K. Mewes (eds.) *The reach of empire – the Early Islamic Empire at work*, volume 2 of *Studies in the History and Culture of the Middle East*. Berlin: De Gruyter.

Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N. & Scialom, T. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. URL http://arxiv.org/abs/2302.04761. ArXiv:2302.04761 [cs].

Tasovac, T., Romary, L., Banski, P., Bowers, J., de Does, J., Depuydt, K., Erjavec, T., Geyken, A., Herold, A., Hildenbrandt, V., Khemakhem, M., Lehečka, B., Petrović, S., Salgado, A. & Witt, A. (2018). TEI Lex-0: A baseline encoding for lexicographic data. https://dariah-eric.github.io/lexicalresources/pages/TEILex0/TEILex0.html.