




Hengam: An Adversarially Trained Transformer for Persian Temporal Tagging

Sajad Mirzababaei* Amir Hossein Kargaran* Hinrich Schütze Ehsaneddin Asgari

 Center for Information and Language Processing, LMU Munich, Germany

 Computer Engineering Department, Sharif University of Technology, IR

 NLP Expert Center, Data:Lab, Volkswagen AG, Munich, Germany

asgari@berkeley.edu and inquiries@cislmu.org

Abstract

Many NLP main tasks benefit from an accurate understanding of temporal expressions, e.g., text summarization, question answering, and information retrieval. This paper introduces *Hengam*, an adversarially trained transformer for Persian temporal tagging outperforming state-of-the-art approaches on a diverse and manually created dataset. We create *Hengam* in the following concrete steps: (1) we develop *HengamTagger*, an extensible rule-based tool that can extract temporal expressions from a set of diverse language-specific patterns for any language of interest. (2) We apply *HengamTagger* to annotate temporal tags in a large and diverse Persian text collection (covering both formal and informal contexts) to be used as weakly labeled data. (3) We introduce an adversarially trained transformer model on *HengamCorpus* that can generalize over the *HengamTagger*'s rules. We create *HengamGold*, the first high-quality gold standard for Persian temporal tagging. Our trained *adversarial HengamTransformer* not only achieves the best performance in terms of the F1-score (a type F1-Score of 95.42 and a partial F1-Score of 91.60) but also successfully deals with language ambiguities and incorrect spellings. Our code, data, and models are publicly available at <https://github.com/kargaranamir/Hengam>.

1 Introduction

A wide array of natural language processing (NLP) applications relies on accurately identifying of events and their respective occurrence times. Text summarization (Christensen et al., 2013; Aslam et al., 2015; Ghodratinama et al., 2021), question answering (Llorens et al., 2015; Bast and Haussmann, 2015; Jia et al., 2018, 2021), and information retrieval tasks requiring to classify information in a chronological order (Kanhabua and Nejdli, 2013) are all examples of such applications. In order to

address these needs in the last decades, there has been an increased interest in temporal information extraction systems and developing their appropriate corpora and evaluation frameworks. TempEval challenges are, for instance, great examples of such efforts held as a part of SemEval workshops focusing on temporal information extraction (Verhagen et al., 2007, 2010; UzZaman et al., 2013).

The study of temporal expressions in English and other languages has been an ongoing research track in the last decade, spanning renowned rule-based efforts such as HeidelTime (Strötgen and Gertz, 2010) and SUTime (Chang and Manning, 2012) to learning-based approaches, e.g., a transformer-based “BERT got a Date” (Almasian et al., 2021). The majority of efforts in this area have been rule-based, which is suffering from (i) a relatively low recall, as finite rules are usually insufficient to deal with all forms of temporal expressions, and (ii) a relatively low precision, as solely relying on the surface form would lead to a high false positive rate. On the other hand, training on a limited set of examples imposes a challenge for learning-based approaches, as this way, they can hardly see a diverse set of time patterns, even in the presence of large and high-quality datasets (Almasian et al., 2021). Thus, an approach combining the strength of both rule-based approaches and learning-based approaches in temporal tagging would be extremely beneficial.

Similar to many other languages, both rule-based approaches (Mansouri et al., 2018) and learning-based approaches (Mohseni and Tebbifakhr, 2019; Taher et al., 2020; Farahani et al., 2021) are developed for the Persian language. *ParsTime* (Mansouri et al., 2018) is probably the first and the most popular attempt to identify and normalize Persian temporal expressions, which also uses the TimeML scheme (Pustejovsky et al., 2005). *ParsTime* being purely rule-based has several limitations: (i) inability to handle ambiguities in the language, (ii)

* The first two authors contributed equally and their authorships were determined randomly.

incapability to deal with a wide range of temporal terms, and (iii) failing to generalize. The other studies in Persian time tagging have attempted to recognize time and date entities as a subset of named entity recognition (NER) tasks, such as MorphoBERT (Mohseni and Tebbifakhr, 2019), Beheshti-NER (Taher et al., 2020) and ParsBERT (Farahani et al., 2021). These studies all tackle this problem using transfer learning by training a supervised NER model on variations of a pretrained transformer language model, in particular, a BERT (Devlin et al., 2018) model.

Time and date tags are included in Persian NER datasets, such as Peyma (Shahshahani et al., 2018), A’laam (Hosseinnejad et al., 2017), Persian-NER (Text-mining.ir, 2018), and NSURL’19 (Taghizadeh et al., 2019). However, training models based on these datasets do not lead to a high-performance temporal tagging model, as they contain a limited number of temporal tags and do not cover all forms of possible temporal expressions in Persian. For instance, Peyma, which is used in several studies, including MorphoBERT, Beheshti-NER, and ParsBERT, only contains 2126 sentences containing temporal expressions. In addition to the small number of training examples, these datasets are far from being an appropriate temporal dataset that must cover most types of temporal expressions and consider language-specific constraints. Some of the language-specific challenges in Persian are: (i) the difference between formal and informal writing styles, (ii) lexical ambiguity (homographs), and (iii) the use of three calendar systems in Persian: the Gregorian, Hijri, and Jalali calendars, unlike most of languages, referring mostly to only one or two calendars in their texts.

This paper aims to bridge the gap between rule-based and transformer-based approaches by creating an unbiased temporal tagged corpus using a rule-based approach and then adversarial training of a state-of-the-art transformer model. Training begins by fine-tuning a pre-trained model on a created corpus, followed by adversarial fine-tuning with a smaller, strongly labeled corpus using projected gradient descent (PGD). The following are the main contributions of this paper:

(i) We present the *Hengam* rule-based tagger (*HengamTagger*), which is an efficient and extensible rule-based temporal expression identification tool. *HengamTagger* is the only publicly acces-

sible tool capable of extracting Persian temporal expressions.

(ii) We introduce *HengamCorpus*, a sizeable unbiased dataset created by *HengamTagger* covering the majority of formal and informal temporal expressions taking the Persian language constraints into account.

(iii) We developed *HengamTransformer*, a state-of-the-art adversarial transformer-based temporal tagger model trained on the *HengamCorpus*. *HengamTransformer* obtain a *type* F1-Score of 95.42 and a *partial* F1-Score of 91.60 on the evaluation dataset that includes a wide range of temporal patterns in Persian.

2 Related Work

The approaches for the identification of temporal expression fall within two main categories: (i) rule-based and (ii) learning-based methods.

Rule-Based Methods. Rule-based methods identify temporal expressions by constructing deterministic rules. Here we summarize the main instances of such works, namely GUTime (Mani, 2003; Verhagen et al., 2005), HeidelTime (Strötgen and Gertz, 2010), SUTime (Chang and Manning, 2012), and SynTime (Zhong et al., 2017). GUTime is a part of the TARSQI toolkit to enhance question-answering systems in temporally-related queries. GUTime extends TempEx (Mani and Wilson, 2000) with machine-learned rules to resolve temporal expressions based on the TimeML TIMEX 3 standard. HeidelTime employs knowledge resources and linguistic clues to normalize extracted temporal expression rules. SUTime is another renowned system built on TokensRegex (Chang and Manning, 2014) mapping regular expressions defined over text and tokens to semantic objects. SynTime proposes general type-based heuristic rules detecting time mentions based on the similar syntactic behavior of temporal words. SynTime identifies temporal tokens in raw text, searches for other specified types in their surroundings, and then merges these segments into temporal expressions.

ParsTime (Mansouri et al., 2018) is the only previous attempt to develop a rule-based temporal tagger capable of identifying and normalizing Persian temporal expressions. However, some challenges are not addressed by ParsTime, such as homographs, common spelling mistakes, and informal variations in temporal expressions in Persian. Unfortunately, due to a lack of documentation and

feedback from the authors, we were not able to run the ParsTime, but by reviewing the ParsTime code, we ensured that all predefined patterns are reflected in *HengamTagger*. Furthermore, *HengamTagger* resolves some of the ParsTime challenges by defining exclusion patterns and covering a far broader range of temporal expressions described in the §3.1.

Learning-Based Methods. A majority of learning-based methods were introduced at the TempEval challenge of SemEval (Verhagen et al., 2007, 2010; UzZaman et al., 2013). Such models traditionally use textual features, such as characters, words, syntactic, and semantic features. These studies have utilized statistical models such as Conditional Random Fields (CRFs), Markov Logic Networks, and Support Vector Machines (SVMs) to model temporal expressions (UzZaman and Allen, 2010; Filannino et al., 2013; Bethard, 2013). With the recent advances in NLP, models built on top of pre-trained language models, such as BERT (Devlin et al., 2018), are introduced (Chen et al., 2019; Lange et al., 2020; Almasian et al., 2021). These models are trained on several datasets supporting temporal pattern units (Mazur and Dale, 2010; UzZaman et al., 2013; Zhong et al., 2017).

For the Persian language, the learning-based approaches are mainly trained over the general Persian NER datasets, and there is no public annotated dataset in standard time schemes, such as TimeML. Examples of these datasets are Peyma (Shahshahi et al., 2018), Persian-NER (Text-mining.ir, 2018), and NSURL’19 (Taghizadeh et al., 2019). There have also been a couple of studies discussing the creation of a dataset of temporal pattern units. However, we were unable to access their data by contacting the authors (Mansouri et al., 2018; Hosseini et al., 2017). Existing Persian temporal taggers are created using the above-mentioned NER datasets utilizing a variation of BERT transformers (Devlin et al., 2018), such as MorphoBERT (Mohseni and Tebbifakhr, 2019), Beheshti-NER (Taher et al., 2020) and ParsBERT (Farahani et al., 2021). MorphoBERT (using a Persian morphological analyzer combined with BERT) and Beheshti-NER (utilizing a CRF model on top of the BERT network) are NER approaches presented at the NSURL’19 workshop (Taghizadeh et al., 2019) and ranked first and second respectively. Previous studies (Mohseni and Tebbifakhr, 2019; Taher et al., 2020) have noted that, due to the

lack of time and date examples in the NSURL’19 and Peyma datasets, the worst results of the seven different NER classes were associated with time and date categories.

3 Materials and Methods

In this section, we present the workflow of *Hengam* shown in Figure 1. We firstly (i) start with a rule-based tagger (*HengamTagger*), which is then used in (ii) creating a weakly labeled dataset (*HengamCorpus*). (iii) Ultimately, we present our *Hengam* adversarial transformer model (*HengamTransformer*) trained over a strongly labeled dataset. We also describe how we develop a gold standard for this task and evaluate *Hengam* variations against the state-of-the-art approaches.

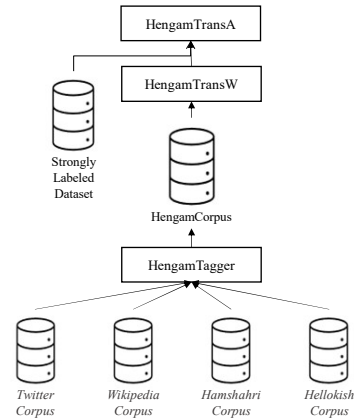


Figure 1: **The overview of Hengam approach.** (i) *HengamTagger* (our rule-based system) identifies the temporal expression from both formal and informal datasets resulting in the automatically annotated *HengamCorpus*. (ii) *HengamCorpus* is then used in a supervised fine-tuning of the *Hengam* Transformer, an XLM-RoBERTa with a CRF layer. Since *HengamCorpus* is considered as a weakly labeled dataset, the transformer model trained solely on *HengamCorpus* is called *Weak Hengam Transformer* or in short *HengamTransW*. (iii) In the next step, we train *Adversarial Hengam Transformer* or in short *HengamTransA* by fine-tuning *HengamTransW* over a strongly labeled dataset using the PGD algorithm.

3.1 HengamTagger

A significant bottleneck in training supervised machine learning models is the preparation of training data, which is time-consuming and, in many cases, expensive when human labeling is required. There are only a few datasets containing both formal and informal Persian temporal labeled data. These datasets are considered too small to be used for training large models with the generalization

ability. In addition, they do not cover a wide diverse set of temporal patterns, and therefore trained models are not able to recognize temporal expressions in many cases. Hence, to overcome both issues, we introduce *HengamTagger*, a rule-based approach designed to automate extracting and labeling temporal expressions using finite predefined patterns.

Tagger Architecture. *HengamTagger* is a rule-based Persian temporal extractor built on top of regular expressions specifying pattern units and patterns that can match temporal expressions. As indicated in the architecture diagram in Figure 2, the temporal patterns of different types are introduced in *HengamTagger* in abstract forms ‘patterns’ and ‘pattern units’ explained in the next part.

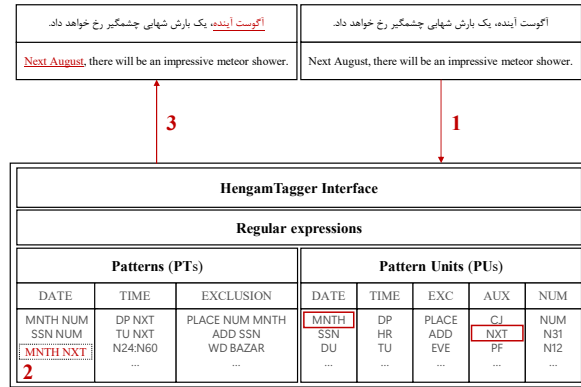


Figure 2: **The Architecture of our Rule-based HengamTagger.** In the rule-based system, the atomic units of temporal expressions are **Pattern Units (PUs)**. These PUs are then combined to generate temporal **Patterns (PTs)**. Our final rules are regular expressions generated from these PTs. For instance, the PUs “MNTH” and “NXT” represent the names of months and the relative temporal terms, respectively. Having the PT rule “MNTH NXT”, meaning a relative temporal term followed by the name of the month, helps *HengamTagger* to detect the example expression آگوست آینده (*august âyande*, “next august”).

Pattern Units (PUs). “Pattern units”, or in a short form *PUs*, are abstract atomic units matching time-related terminologies. *PUs* are then combined to form a more complex but still abstract representation of temporal relations, called “patterns”, shortly *PTs*. We categorize the *PUs* into five groups depending on their usages: (i) date units, (ii) time units, (iii) exclusion units, (iv) auxiliary units, and (v) number units. In the following, we introduce each of these five categories using an example. (i) **Date unit:** date *PUs* represent temporal expressions larger than or equal to 24 hours, such as

days of week, months, seasons, etc. For instance, MNTH *PU* refers to different months in three calendar types including, Gregorian, Hijri (Lunar), and Jalali (Solar) calendars in Persian. (ii) **Time unit:** Time *PUs* represent temporal expressions covering a time less than 24 hours. TU pattern unit is an example referring to different time units, e.g., ساعت (*sâ’at*, “hour”) and ثانیه (*sâniye*, “second”) in Persian. (iii) **Exclusion unit:** these *PUs* represent the building blocks for patterns that can introduce false negatives using homographs to the other *PUs*. For instance, the PLACE *PU* refers to any location may be named after a specific time and date, e.g., مدرسه (*madrese*, “school”) or موزه (*muse*, “museum”). (iv) **Auxiliary unit:** auxiliary *PUs* mainly consist of grammatical terms that help in building *PTs* in combination with other *PUs*. For instance, the NXT pattern unit is a set of words that might come after temporal expressions, e.g., پیشین (*pišîn*, “prior”). (v) **Number unit:** number *PUs* are numbers in digit or in alphabetic format. For instance, N31 represents a number between 1 to 31.

Patterns (PTs). Date, time, and exclusion *PUs* are combined to build three types of date, time, and exclusion *PTs*, respectively. In the following, we introduce each pattern group through an example.

(i) **Date pattern:** date patterns match temporal expressions spanning a time larger than or equal to 24 hours. For example, N31 MNTH pattern matches with temporal expressions, e.g., ۱۶ بهمن (*16 bahman*, “Bahman 16” ≈ “February 4”), ۵ می (*5 mey*, “May 5”), etc. (ii) **Time pattern:** time patterns match the temporal expressions covering a range of hours. For example, TU NXT pattern matches with temporal expressions, e.g., ساعت بعد (*sâ’at ba’d*, “next hour”), دقیقه قبل (*daqiqeh qabl*, “previous minute”), etc. (iii) **Exclusion pattern:** exclusion patterns exclude phrases that are matched by date or time patterns by defining more concise patterns. For example, PLACE N31 MNTH pattern matches with expressions, e.g., ۱۵ خرداد مدرسه‌ی (*madrese-ye 15 xordad*, “15 of khordad school”), بیمارستان ۹ دی (*bimârestân-e 9 dey*, “dey 9th hospital”), etc. Exclusion patterns help to disambiguate the names of persons or places that are homographs with temporal expressions. For Instance, ۱۵ خرداد (*15 xordad*, “15th of Khordad” ≈ “5th of June”) is a temporal expression showing a date but مدرسه‌ی ۱۵ خرداد (*madrese-ye 15 xordad*, “khordad 15th school”) is a place name consisting of a specific date and should not be recognized as a temporal ex-

pression. Another example of the exclusion pattern usages is the continuous verbs having the prefix *mi* (می) in Persian. *mi* is the homograph of “May” which is a 5th month of the year in the Gregorian calendar. This issue is addressed by using multiple exclusion patterns that construct all of the possible verbs that begin with prefix *mi*.

Output Schemes. There are several output schemes supported by *HengamTagger*. Before providing the output, *HengamTagger* merges temporal expressions that have the same tag (Time or Date) and are adjacent to each other. For example, in the temporal expression امروز دوشنبه (*emrooz došanbe*, “today Monday”) there are two phrases, امروز (*emrooz*, “today”) and دوشنبه (*došanbe*, “Monday”) which are dates. *HengamTagger* may match these two expressions separately, but during the post-processing stage, they are merged into one expression. Following are the different output schemes supported by Hengam: (i) **Span Indices**: in this format, the start and end indices of each of the detected temporal patterns are provided separating the “Time”, “Date”, and “DateTime” categories. Note that the “DateTime” is the combination of time and date expressions. (ii) **TimeML**: based on TIMEX 3 standard (Pustejovsky et al., 2005), this format takes four outputs into account. The “Date” tag indicates a calendar time. “Time” tag for temporal expressions less than 1 day (including clock time, daypart, etc.). “Duration” tag for temporal expressions that describe intervals. The “Set” tag is used when the temporal expression refers to recurring events. (iii) **BIO**: in this format, two different tagging schemes are considered, the first one outputs the time and the date as individual entities, i.e., “TIM” and “DAT”. The second one represents the “TMP” entity by combining “Time” and “Date”. These tags are represented in the BIO standard tagging scheme used in the NER tasks (Ramshaw and Marcus, 1999).

3.2 HengamCorpus

We introduce *HengamCorpus* weakly labeled dataset by applying *HengamTagger* (§3.1) over datasets described in §3.2.1. Unlike previous efforts of creating a temporal tagged dataset, empowered by our extensive set of patterns and pattern units, we can consider a wide array of diverse temporal patterns. Furthermore, we introduce a dataset containing strong temporal labeled data and also include challenging sentences to improve *Hengam-*

Transformer training in §3.4.

3.2.1 Raw Text Collections

We chose four popular Persian text collections covering both formal and informal styles: Persian Wikipedia (Fa.wikipedia.org, 2020) and Hamshahri Corpus (Hamshahrionline.ir, 2021) as formal ones, and Twitter (Abdi Khojasteh et al., 2020) and HelloKish dataset (Moradi and Bahrani, 2015) datasets as informal Persian datasets. (i) **PersianWiki**: Persian language collection of Wikipedia articles, the 19th largest edition by the number of articles. As of the data creation date, the dataset contains 739,870 articles with 3,858,609 sentences. (ii) **Hamshahri**: this data is based on the Iranian newspaper Hamshahri, one of Iran’s first Persian language online newspapers. The dataset used for the analysis contains 150,096 news articles resulting in 1,793,147 sentences. (iii) **PersianTwitter**: the data consists of 20,665,964 tweets, mostly in the informal Persian context, which has been further reduced to 9,852,565 tweets after eliminating duplicates. (iv) **HelloKish**: HelloKish is a tourism guidance website that allows people to share their opinions about different places. In total, this dataset spans 2,378 comments constructed from 7,899 sentences.

3.2.2 Training Corpus Creation

Weakly Labeled Dataset. *HengamCorpus* weakly labeled dataset is generated by extracting temporal expressions on the raw text collections § 3.2.1 using *HengamTagger*. We have observed that certain temporal patterns are highly skewed in the datasets in terms of frequency, resulting in a non-uniformity of temporal expression types. We have discussed and visualized this matter in further detail in Appendix §B. The non-uniformity of these temporal patterns introduces a bias in training and evaluation if we ignore these imbalances. To address this issue, we uniformly draw samples from sets of sentences of unique “temporal pattern profile”, presence/absence vector of different temporal patterns within the sentence. The created *HengamCorpus* consists of 313,847 sentences and 12,902,121 tokens covering 1,783,426 date tokens and 195,639 time tokens. *HengamCorpus* differs from other datasets with temporal tags in two ways. First, it includes a wide range of types of temporal expressions without being biased towards any particular pattern. Secondly, all data points are labeled consistently regardless of the context, in both formal

and informal contexts.

Strongly Labeled Dataset. *HengamTagger* does not understand the semantics of words and cannot handle challenges like homographs properly. There are many homographs in Persian that express temporal expressions, on top of having multiple other meanings. For instance, the homograph *mehr* (مهر) can refer to 7th month in the solar calendar, stamp, love, or name of a popular news agency, depending on the context. We need a dataset with correct labels to inform the learning model (*HengamTransformer* described in §3.3) about these differences. Thus, we need to provide a dataset that is strongly labeled. As the labeling process involves a great deal of time and expense, we only make a small portion of strongly labeled instances ($\approx 0.5\% ||HengamCorpus||$), and the rest will be handled by *HengamCorpus* as weakly labeled instances. We collect a set of 1,500 carefully crafted sentences consisting of 2,909 date tokens and 691 time tokens. In the creation of the strong collection, we attempt to include challenging examples (e.g., homographs, polysemous words, etc.) as much as possible. Two annotators participated in the labeling independently, resulting in a kappa agreement score of 0.95. Subsequently, the conflicts were resolved in a joint session.

3.3 HengamTransformer

Similar to any other rule-based approach, the rule-based version of *HengamTagger* has the following disadvantages: (i) it has a relatively low recall because of using a finite set of rules, and (ii) it is incapable of comprehending a complex context to handle challenging cases, e.g., as homographs, which leads to a lower precision. In this step, we introduce *HengamTransformer*, a fine-tuned transformer language model adversarially trained on *HengamCorpus* and a set of strong labels, as a solution to both problems.

HengamTransformer is a neural CRF model consisting of an XLM-RoBERTa transformer model and a linear-chain CRF layer. In this architecture, the transformer neural network component serves as an encoder, which encodes the input sequences of tokens into token embeddings, and subsequently transforms them into token logits. In a sequence labeling model, the RoBERTa model encodes each token into a hidden representation size d , which is then projected onto the tags space determined by the number of classes and the tagging schemes,

i.e. $R^d \mapsto R^{|C|}$, where C indicates the set of tags. Let us consider the input as $X = [x_1, \dots, x_k]$ and their labels as $Y = [y_1, \dots, y_k]$, $y_i \in C$, and the logits generated by the encoder network as $l = [l_1, \dots, l_k]$, $l_i \in R^{|C|}$, where k indicates the length of the sequence. In the next component, the CRF layer employs a label transition function Ψ ($\Psi : R^{|C|*|C|} \rightarrow R$). Using *HengamTransformer*, each possible tag sequence is assigned a score based on the aggregation of emission scores, which is the likelihood of tag y_i given sequence X and transition scores for moving from the tag y_{i-1} to the y_i . Thus, we can assign a score to the sequence of labels, Y , based on the logits and the transition score as the following:

$$score(y, x) = \sum_{i=1}^k l_{i, y_i} + \sum_{i=1}^{k-1} \Psi(y_i, y_{i+1}),$$

where $l_{i,j}$ indicates the j -th entry in logit l_i .

Considering \mathcal{D} as the training set and \mathcal{Y} as the set of all possible tagging schemes, the loss function of the CRF model can be defined as an average of the negative log-likelihoods over the training set:

$$\mathcal{Loss} = -\frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \log \frac{\exp score(y, x)}{\sum_{y' \in \mathcal{Y}} \exp score(y', x)}.$$

Finally, *HengamTransformer* utilizes the Viterbi algorithm (Forney, 1973) to determine the tag sequence with the highest score as the output.

3.4 Adversarial HengamTransformer

First, we use *HengamCorpus* (§3.2) to fine-tune the *HengamTransformer*'s language model. In the next step, we train a complete architecture containing the transformer and the CRF layer jointly in an end-to-end manner. We split the *HengamCorpus*, into train (75%), test (10%) and validation (15%) sets. After reaching the early-stopping point based on the performance of the validation data, we re-train the model on the strong labels (§3.2.2) in an adversarial manner. In many previous works, it has been shown that adversarial training can improve both generalization and robustness (Miyato et al., 2017; Cheng et al., 2019). An adversarial training of *HengamTransformer* contains a min-max optimization process. The max part involves a non-concave maximization problem to find perturbation vectors maximizing the loss for a particular mini-batch. And then in the min step, we deal with a non-convex minimization problem to determine

parameters minimizing the loss function using the Stochastic Gradient Descent (SGD) algorithm.

Suppose that the *HengamTransformer* is defined as a function $f_\theta(X)$, where X is the sub-word embeddings and θ is referring to the trainable parameters. The adversarial training method attempts to find the optimal parameters θ^* minimizing the maximum risk of any adversarial perturbations δ to the embeddings inside a norm ball, which can be written as follows:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{\mathcal{D}} \left[\max_{\|\delta\| \leq \epsilon} L(f_\theta(X + \delta), Y) \right],$$

where \mathcal{D} represents the data distribution, Y represents the label, and L represents the loss function. K -projected gradient descent (K -PGD) adversarial training (Madry et al., 2018), as an effective adversarial training method, is utilized. K -PGD adversarial training, requiring K forward-backward passes through the network, is usually computationally expensive. However, since only a small portion of our data ($< 0.5\%$) is strongly labeled, the adversarial training can be done in an efficient manner.

3.5 Evaluations

Temporal Tags in Persian NER Datasets. There are three public Persian NER datasets that support temporal tags as follows: (i) Peyma dataset contains only 2126 sentences with at least one temporal expression. (ii) NSURL’19 dataset consisted of 1784 temporal sentences (1672 sentences from Peyma dataset as its subset). (iii) Persian-NER which includes approximately one million Wikipedia sentences, including 448,542 sentences with temporal terms. However, this dataset does not support both time and date tags as separate tags and uses the same temporal tag for both.

Exploring NER datasets using *HengamTagger*. Due to incompleteness of annotations in three NER public datasets (Shahshahani et al., 2018; Taghizadeh et al., 2019; Text-mining.ir, 2018), we limit the evaluation to the sentences containing at least one temporal tag. Originally we wanted to evaluate the *HengamTagger* over these datasets. However, the error analysis showed us that the temporal relations, in general, are not consistently and correctly annotated in these cases. Thus, the performance of *HengamTagger* on these datasets can be served as an indication of their quality. Thus, we create the *HengamGold* for a proper evaluation of Persian temporal tagging.

HengamGold Evaluation Dataset. An evaluation of a temporal identifier model requires a dataset that covers a wide range of temporal expression patterns as well as formal and informal contexts. Since there exists no previous such a strongly labeled dataset, we present a small dataset consisting of 200 examples in order to compare our model to other closely related models. To ensure that our HengamGold dataset accurately reflects a real-world situation, we carefully designed 20 parameters, which are specific conditions on the temporal patterns and their interactions with the context. Then we form the evaluation dataset based on these conditions. In Appendix §C, we list the designed conditions along with the number of satisfying sentences in the dataset. Afterward, collected data is annotated independently by two experts with a kappa score of 0.97 which implies high agreement among annotators.

Evaluation Metrics. For the sake of comparison, we report precision, recall, and f1-score. In sequence labeling problem settings, these metrics can be measured in two scenarios: *exact match* and *relaxed match (partial match)* (Segura-Bedmar et al., 2013). The ambiguity of boundaries for the Persian temporal entities encouraged us to choose a relaxed match scenario. *Relaxed match* scenario is evaluated using the following metrics: (i) *Partial evaluation*: comparing the predicted and the true boundaries, regardless of the entity type. (ii) *Type evaluation*: checking whether the predicted type has an overlap with the correct entity type or not. For the calculations we use “nervaluate”, the evaluation toolkit¹ which is developed based on SemEval’13 guidelines (Segura-Bedmar et al., 2013).

Evaluation of *Hengam*. We evaluate the performance of different variants of *Hengam* temporal detectors (rule-based and learning-based) against the *HengamGold* dataset and compare its performance with the state-of-the-art models for Persian temporal tagging, i.e., Beheshti-NER (Taher et al., 2020) and ParsBERT (Farahani et al., 2021). Unfortunately, the MorphoBERT (Mohseni and Tebbifakhr, 2019) and ParsTime (Mansouri et al., 2018) models were not available to be used in this comparison. Here, we utilize two different variations of *HengamTranformer*: (i) *HengamTransformer-weak*: trained on *HengamCorpus* weakly labeled

¹<https://github.com/MantisAI/nervaluate>

data, (ii) HengamTransformer-adversarial: trained on *HengamCorpus* and subsequently adversarially fine-tuned over the strongly labeled data. Furthermore, we also train a version of ParsBERT (ParsBERTHengam) with *HengamCorpus* to investigate the contribution of adversarial training and the CRF layer in the final performance.

Evaluation of Adversarial Training using HengamChallengeSet. For an in-depth comparison of the generalization ability of adversarial Hengam *HengamTransA* over the weakly trained transformer *HengamTransW*, we create another evaluation set of 30 manually annotated challenging examples, called *HengamChallengeSet*. This evaluation set spans examples containing homographs, polysemous cases, and other complex examples to study the effect of *HengamTransW* fine-tuning with strongly labeled dataset.

4 Results

4.1 Temporal Tagging Analysis of Persian NER datasets

A summary of the *HengamTagger* performance on publicly available Persian NER datasets is provided in Table 1. After an extensive error analysis, we concluded that the Persian NER datasets are only partially annotated for the temporal tags, meaning that they cannot be used for a proper evaluation of *HengamTagger*. Therefore, the main mission of Table 1 is (1) to assess the coverage and precision of rules incorporated in *HengamTagger* and (2) compare the time/date tagging quality in different Persian NER datasets. This is the primary reason that we have not included another baseline in Table 1. In addition, we have to indicate that since these NER datasets were used in the training process of “Beheshti-NER” and “ParsBERT”, it did not seem to be the right approach to include these models in the evaluation as well.

Our analysis indicates that the *HengamTagger* gets a high recall on both Peyma and NSURL’19 datasets. In many cases, the source of difference is the inclusion/exclusion of the preposition before the temporal expression as part of the temporal expression. However, there are also some true negatives in these datasets resulting in a lower precision. For instance, in Peyma dataset, the expression در ماه رمضان (*dar mah ramezan*, “In the month of Ramadan”) is not labeled as a temporal expression. In addition, Persian-NER (Text-mining.ir, 2018), for instance, does not distinguish between time and

Dataset	Type			Partial		
	Pr.	Re.	F1	Pr.	Re.	F1
Peyma	72.15	93.81	81.57	69.53	90.41	78.61
NSURL	72.57	94.07	81.93	69.89	90.61	78.91
Persian-NER	89.39	88.30	88.84	58.95	58.23	58.91

Table 1: The performance of *HengamTagger* (Precision, Recall, and F1 scores) on Persian NER datasets containing temporal labels

date tags and uses the same temporal tag for both types. We also found many senseless cases frequently tagged as temporal terms, e.g., سیمی (*simi*, “Wired”), مهدی (*mahdi*, “mahdi, a person name”), and مهمی (*mohemmi*, “an important”). We also found several instances of inconsistency in terms of following the IOB format. In general, *HengamTagger* still gets a relatively high type recall rate on these datasets. The type recall in this dataset increases from 88.30 to 89.25 by simply labeling the three words suggested above with the label “O”. Keeping all of this in mind, although our original plan was to evaluate the *HengamTagger* with the Persian NER datasets, because of the poor quality of temporal tags, the analysis became the other way around. That is the reason we created the *HengamGold* for a proper evaluation of Persian temporal tagging approaches.

4.2 Hengam Evaluation Results

The performance comparison of *HengamTransformer* variations with rule-based *HengamTagger*, Beheshti-NER (Taher et al., 2020), and ParsBERT (Farahani et al., 2021) on *HengamGold* dataset is provided in Table 2. Our results suggest that Hengam variations outperform the state-of-the-art Persian Temporal Tagging approaches Beheshti-NER and ParsBERT. In addition, the *HengamTransformer* variations had superior performance to the rule-based tagger suggesting a better generalization ability of a language-model-based tagging model. *HengamCorpus*’ good quality dataset greatly improved ParsBERT’s performance; nevertheless, the Hengam transformer architecture having a CRF layer on top delivered even better results. Furthermore, the *adversarial HengamTransformer* achieved the best performance in terms of all metrics (precision, recall, and F1) as well as evaluation settings (type evaluation and partial evaluation), among other *HengamTransformers*.

Here we discuss a number of interesting observations we witnessed in the evaluation process of

Hengam temporal taggers: **(i) Style/Spelling error resistance:** *HengamTagger* cannot handle a different style or a severe spelling error. However, *HengamTransformer* is highly resilient to this problem. For instance, the phrases پونزده خرداد (*poonzdah-e xordad*, “*Khordad 15th*”) and سینزده خرداد (*sinzdah-e xordad*, “*Khordad 13th*”) are the informal forms of پانزده خرداد (*pânzdah-e xordad*, “*Khordad 15th*”) and سیزده خرداد (*sizdah-e xordad*, “*Khordad 13th*”) which are successfully recognized by the *HengamTransformer* approach but not the rule-based *HengamTagger*. In several examples, we observed that *HengamTransformer* could resist spelling errors as well. **(ii) Pattern Generalization:** *HengamTagger* is only capable of detecting temporal expression based on predefined rules and cannot detect any new pattern. However, *HengamTransformer* could successfully generalize to detect phrases such as بقیه هفته (*baqie hafte*, “*rest of the week*”), شش هفت ثانیه (*šeš haft sâniye*, “*6-7 seconds*”), and هر ساعت یکبار (*har sâ’at yekbar*, “*every hour*”) without seeing them in advance in the training data. **(iii) Homographs:** There are many temporal markers in Persian involved in homograph relations with other words. Clearly, *HengamTagger* cannot handle this issue without including the context into the pattern. In contrast, the strong labels fed to *HengamTransformer* in the adversarial training helped the model distinguish between the word senses. As an example, both بهمن (*bahman*) and آذر (*azar*) are months of the Persian solar calendar. However, they can also refer to a person’s name or a product. The adversarially trained *HengamTransformer* variation (and interestingly not the *HengamTransformer-weak*) could successfully disambiguate these sentences in phrases سیگار بهمن (*sigar-e bahman*, “*Bahman cigarette*”) and آذر خانم (*azar xânom*, “*Ms. Azar*”).

4.3 Evaluation Results of the Adversarial Training on HengamChallengeSet

Our analysis on the results of *HengamTransA* and *HengamTransW* over the *HengamChallengeSet* shows that the adversarial training (*HengamTransA*) could correctly disambiguate all 30 manually annotated challenging cases, while the weak training *HengamTransW* could only identify 9 out of 30 challenging temporal tags. Detailed results are provided².

²https://github.com/kargaranamir/Hengam/blob/main/data/evaluation/challenge_set/HengamChallengeSet.xlsx

Model	Type			Partial		
	Pr.	Re.	F1	Pr.	Re.	F1
Beheshti-NER	81.67	37.55	51.44	61.25	28.16	38.58
ParsBERT	76.85	31.80	44.99	52.78	21.84	30.89
ParsBERTHengam	89.89	95.40	92.56	83.57	88.69	86.95
HengamTagger	89.93	95.78	92.76	83.99	89.46	86.64
HengamTransW	94.66	95.02	94.84	88.36	88.70	88.53
HengamTransA	95.06	95.78	95.42	91.25	91.95	91.60

Table 2: Comparison of different variations of Hengam temporal detectors, (i) *HengamTagger*: the rule-based tagger, (ii) *HengamTransW*: HengamTransformer trained on *HengamCorpus* weakly labeled data, and (iii) *HengamTransA*: HengamTransformer trained on *HengamCorpus* and subsequently adversarially fine-tuned over the strongly labeled data. The Hengam models are compared with the Beheshti-NER (Taher et al., 2020), ParsBERT (Farahani et al., 2021), and ParsBERT, which is fine-tuned with *HengamCorpus* (ParsBERTHengam) in terms of Precision, Recall, and F1 scores in temporal type-checking and partial evaluations over the *HengamGold* dataset.

5 Conclusions

In this paper, we proposed *Hengam*, an accurate adversarially trained transformer for Persian temporal tagging outperforming state-of-the-art approaches on a diverse and manually created dataset. We achieved this system in the following concrete steps: (1) we developed *HengamTagger*, a fast and extensible rule-based tool that can extract temporal expressions from any language by creating language-specific patterns³. (2) We used *HengamTagger* to annotate a large and diverse Persian text collection (covering both formal and informal contexts) for temporal tags. This way, we made *HengamCorpus* and used it as weakly labeled data for subsequent learning-based temporal tagging. (3) We introduced an adversarially trained transformer model on *HengamCorpus* that can generalize over the *HengamTagger*’s rules evaluated over a set of challenging examples named *HengamChallengeSet*. We studied available Persian temporal datasets and found that the current datasets are inadequate for developing a system to identify temporal expressions. We created the first high-quality gold standard for Persian temporal tagging called *HengamGold*. The *adversarial HengamTransformer* not only achieved the best performance in terms of the F1-score but also successfully dealt with language ambiguities and incorrect spellings.

³Appendix §D gives an example of how to extend the *HengamTagger* for another language.

References

- Hadi Abdi Khojasteh, Ebrahim Ansari, and Mahdi Bohloul. 2020. Lscp: Enhanced large scale colloquial persian language understanding. In *Proceedings of the Twelfth International Conference on Language Resources and Evaluation (LREC 2020)*, pages 6323–6327. European Language Resources Association.
- Satya Almasian, Dennis Aumiller, and Michael Gertz. 2021. Bert got a date: Introducing transformers to temporal tagging. *arXiv preprint arXiv:2109.14927*.
- Javed Aslam, Fernando Diaz, Matthew Ekstrand-Abueg, Richard McCreadie, Virgil Pavlu, and Tetsuya Sakai. 2015. Trec 2014 temporal summarization track overview. Technical report, NATIONAL INST OF STANDARDS AND TECHNOLOGY GAITHERSBURG MD.
- Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1431–1440.
- Steven Bethard. 2013. Cleartk-timeml: A minimalist approach to tempeval 2013. In *Second joint conference on lexical and computational semantics (*SEM), volume 2: proceedings of the seventh international workshop on semantic evaluation (SemEval 2013)*, pages 10–14.
- Angel X Chang and Christopher D Manning. 2012. Suntime: A library for recognizing and normalizing time expressions. In *Lrec*, volume 3735, page 3740.
- Angel X Chang and Christopher D Manning. 2014. Tokensregex: Defining cascaded regular expressions over tokens. *Stanford University Computer Science Technical Reports. CSTR*, 2:2014.
- Sanxing Chen, Guoxin Wang, and Börje Karlsson. 2019. Exploring word representations on time expression recognition. Technical report, Technical report, Microsoft Research Asia.
- Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4324–4333, Florence, Italy. Association for Computational Linguistics.
- Janara Christensen, Stephen Soderland, Oren Etzioni, et al. 2013. Towards coherent multi-document summarization. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1173.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Mehrdad Farahani, Mohammad Gharachorloo, Marzieh Farahani, and Mohammad Manthouri. 2021. Parsbert: Transformer-based model for persian language understanding. *Neural Processing Letters*, 53(6):3831–3847.
- Fa.wikipedia.org. 2020. Persian wikipedia dataset. <https://github.com/miladfa7/Persian-Wikipedia-Dataset>.
- Michele Filannino, Gavin Brown, and Goran Nenadic. 2013. Mantime: Temporal expression identification and normalization in the tempeval-3 challenge. *arXiv preprint arXiv:1304.7942*.
- G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.
- Samira Ghodrathnama, Amin Beheshti, Mehrdad Zakershahrak, and Fariborz Sobhanmanesh. 2021. Intelligent narrative summaries: From indicative to informative summarization. *Big Data Research*, 26:100257.
- Hamshahrionline.ir. 2021. Hamshahri corpus. <https://github.com/armanhm/Hamshahri-Classification-NLP>.
- Shadi Hosseinnejad, Yasser Shekofteh, and Tahereh Emami Azadi. 2017. A’laam corpus: A standard corpus of named entity for persian language. *Signal and Data Processing*, 14(3):127–142.
- Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jan-nik Strötgen, and Gerhard Weikum. 2018. Tequila: Temporal question answering over knowledge bases. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1807–1810.
- Zhen Jia, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. 2021. Complex temporal question answering on knowledge graphs. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 792–802.
- Nattiya Kanhabua and Wolfgang Nejdl. 2013. Understanding the diversity of tweets in the time of outbreaks. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1335–1342.
- Lukas Lange, Anastasiia Iurshina, Heike Adel, and Jan-nik Strötgen. 2020. Adversarial alignment of multilingual models for extracting temporal expressions from text. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 103–109.
- Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James Allen, and James Pustejovsky. 2015. SemEval-2015 task 5: QA TempEval - evaluating temporal information understanding with question answering. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 792–800, Denver, Colorado. Association for Computational Linguistics.

- Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.
- Inderjeet Mani. 2003. Recent developments in temporal information extraction. In *RANLP*, volume 260, pages 45–60.
- Inderjeet Mani and George Wilson. 2000. Robust temporal processing of news. In *Proceedings of the 38th annual meeting of the association for computational linguistics*, pages 69–76.
- Behrooz Mansouri, Mohammad Sadegh Zahedi, Ricardo Campos, Mojgan Farhoodi, and Maseud Rahgozar. 2018. Parstime: Rule-based extraction and normalization of persian temporal expressions. In *European Conference on Information Retrieval*, pages 715–721. Springer.
- Pawel Mazur and Robert Dale. 2010. Wikiwars: A new corpus for research on temporal expressions. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 913–922.
- Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. *ICLR*.
- Mahdi Mohseni and Amirhossein Tebbifakhr. 2019. Morphobert: a persian ner system with bert and morphological analysis. In *Proceedings of The First International Workshop on NLP Solutions for Under Resourced Languages (NSURL 2019) co-located with ICNLSP 2019-Short Papers*, pages 23–30.
- Mehdi Moradi and Mohammad Bahrani. 2015. Automatic gender identification in persian text (persian). *Signal and Data Processing*, 12(4):83–94.
- James Pustejovsky, Robert Ingria, Roser Sauri, José M Castaño, Jessica Littman, Robert J Gaizauskas, Andrea Setzer, Graham Katz, and Inderjeet Mani. 2005. The specification language timeml.
- Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.
- Isabel Segura-Bedmar, Paloma Martínez, and María Herrero-Zazo. 2013. SemEval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (DDIExtraction 2013). In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 341–350, Atlanta, Georgia, USA. Association for Computational Linguistics.
- Mahsa Sadat Shahshahani, Mahdi Mohseni, Azadeh Shakery, and Hesham Faili. 2018. Peyma: A tagged corpus for persian named entities. *arXiv preprint arXiv:1801.09936*.
- Jannik Strötgen and Michael Gertz. 2010. Heildetime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 321–324.
- Nasrin Taghizadeh, Zeinab Borhanifard, Melika Golestani Pour, Mojgan Farhoodi, Maryam Mahmoudi, Masoumeh Azimzadeh, and Hesham Faili. 2019. NSURL-2019 task 7: Named entity recognition for Farsi. In *Proceedings of The First International Workshop on NLP Solutions for Under Resourced Languages (NSURL 2019) co-located with ICNLSP 2019 - Short Papers*, pages 9–15, Trento, Italy. Association for Computational Linguistics.
- Ehsan Taher, Seyed Abbas Hoseini, and Mehrmoush Shamsfard. 2020. Beheshti-ner: Persian named entity recognition using bert. *arXiv preprint arXiv:2003.08875*.
- Text-mining.ir. 2018. Persian-ner dataset. <https://github.com/Text-Mining/Persian-NER>.
- Naushad UzZaman and James Allen. 2010. Trips and trios system for tempeval-2: Extracting temporal information from text. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 276–283.
- Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9.
- Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. Semeval-2007 task 15: Tempeval temporal relation identification. In *Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007)*, pages 75–80.
- Marc Verhagen, Inderjeet Mani, Roser Sauri, Jessica Littman, Robert Knippen, Seok Bae Jang, Anna Rumshisky, Jon Phillips, and James Pustejovsky. 2005. Automating temporal annotation with tarsqi. In *Proceedings of the ACL interactive poster and demonstration sessions*, pages 81–84.
- Marc Verhagen, Roser Sauri, Tommaso Caselli, and James Pustejovsky. 2010. Semeval-2010 task 13: Tempeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 57–62.
- Xiaoshi Zhong, Aixin Sun, and Erik Cambria. 2017. Time expression analysis and recognition using syntactic token types and general heuristic rules. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 420–429.

A Experiment setup

HengamTransformer trained with a learning rate of $2e - 5$, a batch size of 16, and the maximum sequence length of 512 tokens for the entire training set. Additionally, during the training the weights belonging to the first 8 layers are frozen. Furthermore, for the adversarial training part, we used K -PGD, with $K = 3$.

B Uniform data selection over temporal profiles

HengamTagger has identified 3016 and 31,272 profiles from time and date patterns respectively. In the creation of *HengamCorpus*, to maximize the diversity of patterns for training and evaluation, we uniformly draw samples from sets of sentences of unique “temporal pattern profile”, presence/absence vector of different temporal patterns within the sentence. Figure 3 illustrates how these profiles are skewed in the raw collections. Each row in this diagram indicates the presence of particular pattern IDs.

C HengamGold Parameters Description

We provide the conditions in creation of *HengamGold* in Table 3. These conditions are chosen to maximize the coverage of diverse Persian temporal patterns in this evaluation dataset (e.g., formal and informal styles).

D Hints on extension of HengamTagger for other languages

HengamTagger can be easily extended in supporting languages other than Persian. In this section we, provide an example to extend the framework for another language, in particular for English. Suppose we want to extract English temporal expressions such as “August 12”, “June 21”, etc. For detection of this pattern, firstly we need to define two pattern units: (i) the **MNTH** pattern unit, which includes the Gregorian months, and (ii) the **N31** pattern unit to support numbers from 1 to 31. We then only use the primitives **MNTH** and **N31** to define the pattern “**MNTH N31**”. Subsequently, the “**MNTH N31**” pattern generates the following regular expression to support the mentioned temporal expression.

$[January|February|...|December]\backslash s[1 - 31]$

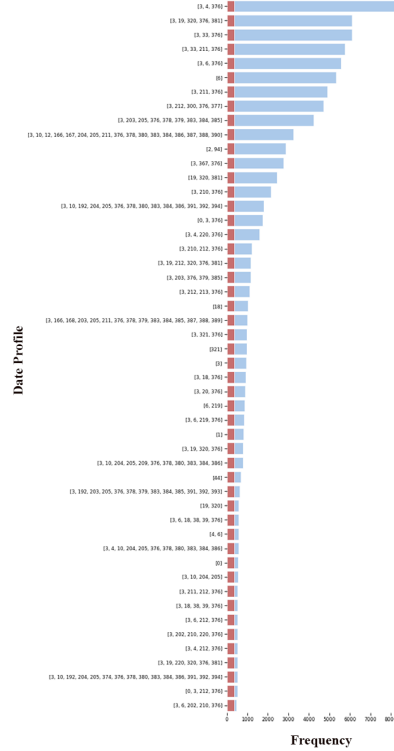


Figure 3: **Skewness of date/time profile distributions.** This figure illustrates the frequency distribution of date profiles calculated over *PersianTwitter*. *HengamTagger* has identified 3016 and 31, 272 profiles from time and date patterns. In the figure the skewness of temporal profile distributions is demonstrated for the most frequent profiles. In the next step, we uniformly sample from the identified profiles (the red parts of the bar for each pattern profile) to have maximum diversity of patterns in the training.

Condition	Matching Cases
Is there any temporal expression in the sentence?	187
Is there any date expression in the sentence?	134
Is there any time expression in the sentence?	79
Is there a place name that contains temporal tokens?	7
Is there a person's name that contains temporal tokens?	14
Does any other named entity contain temporal tokens besides place and person?	15
Is the temporal expression explicit?	150
Does the sentence contain any symbols?	16
Can temporal expression be expressed as a set?	15
Can temporal expression be expressed as a duration?	9
Does the sentence have a formal tone?	130
Is there a digit in the sentence?	112
Does the sentence refer to a solar calendar?	33
Does the sentence refer to a Gregorian calendar?	24
Does the sentence refer to a lunar calendar?	8
Is there a month name in the sentence?	36
Is there any temporal token that indicates the day part in this sentence?	33
Is there any temporal token that indicates the relative time?	28
Is there any season name in the sentence?	7
Is there any weekday name in the sentence?	17

Table 3: **Parameters used in the creation of HengamGold:** we provide a list of conditions considered in the design of the *HengamGold* evaluation dataset along with the number of sentences that satisfying each condition.