A Novel Methodology for Enhancing Cross-language and Domain Adaptability in Temporal Expression Normalization

Alejandro Sánchez de Castro, Lourdes Araujo, Juan Martinez-Romo

NLP & IR, Universidad Nacional de Educación a Distancia (UNED), Spain
asanchez@lsi.uned.es, lurdes@lsi.uned.es, juaner@lsi.uned.es

Accurate temporal expression normalization, the process of assigning a numerical value to a temporal expression, is essential for tasks such as timeline creation and temporal reasoning. While rule-based normalization systems are limited in adaptability across different domains and languages, deep-learning solutions in this area have not been extensively explored. An additional challenge is the scarcity of manually annotated corpora with temporal annotations. To address the adaptability limitations of current systems, we propose a highly adaptable methodology that can be applied to multiple domains and languages. This can be achieved by leveraging a multilingual Pre-trained Language Model (PTLM) with a fill-mask architecture, using a Value Intermediate Representation (VIR) where the temporal expression value format is adjusted to the fill-mask representation. Our approach involves a two-phase training process. Initially, the model is trained with a novel masking policy on a large English biomedical corpus that is automatically annotated with normalized temporal expressions, along with a complementary hand-crafted temporal expressions corpus. This addresses the lack of manually annotated data and helps to achieve sufficient capacity for adaptation to diverse domains or languages. In the second phase, we show how the model can be tailored to different domains and languages using various techniques, showcasing the versatility of the proposed methodology. This approach significantly outperforms existing systems.

1. Introduction

A **temporal expression** (TE) refers to a linguistic construct or phrase within a sentence or discourse that conveys information about date, time, duration, or sets. Every TE is associated with a corresponding value; for instance, *April 1990* is represented by the value 1990–04. The process of predicting this value is termed temporal normalization. TimeML (Pustejovsky et al. 2010) is an ISO standard that includes the TIMEX3 tag, which defines what and how TEs should be annotated.

Action Editor: Ruihong Huang. Submission received: 7 November 2024; revised version received: 4 March 2025; accepted for publication: 19 April 2025.

https://doi.org/10.1162/COLI.a.12

This is a complicated task due to several factors. On one hand, there are basic constructions like *"today"* or *"yesterday"* which are very well processed by current systems. On the other hand, there is a great variety of expressions and many ways of expressing the same thing like *"a while"*, *"some time"*, *"period"*, *"moment"*, and so forth. Also, some TEs need to be anchored since they do not contain enough information to be normalized, for example, in *Next 26 July* it is necessary to know which is the current month to know which is the next July. There are also temporal expressions that might be interpretable, for example, *10 hours later* can be treated as a time or as a duration depending on the annotator's interpretation. Therefore, merging different corpora with different annotator's interpretations might lead to contradictions. Attention must also be paid to the context, for example, the sentence *"On the 3rd and 4th day"* is split into two expressions *"3rd"* and *"4th day"*. Therefore, the system in charge of normalizing the first expression must be aware of the context to see that *"3rd"* refers to *day*.

Normalization of TEs is very helpful in all tasks that involve temporal ordering such as timeline creation (Najafabadipour et al. 2020), summarization of texts (Barros et al. 2019), or question answering (Cole et al. 2023). It also plays a crucial role in reasoning, since providing the systems with the capability of being temporarily aware is a crucial step to achieving reasoning. It is basic to apply induction and deduction to infer the temporal order of things.

Until now, rule-based systems (Strötgen and Gertz 2010; Navas-Loro et al. 2020) have dominated the landscape of normalization, offering precise adaptation to specific domains and languages. However, these architectures prove highly susceptible to changes in domain and language, demanding labor-intensive hand-crafting of new rules for adaptation. Attempts to transition to machine and deep learning solutions (Ning et al. 2018; Ding et al. 2021; Chang and Manning 2012) have encountered obstacles that hinder their progress, such as the limited availability of hand-annotated data, together with a lack of work done for languages other than English. Additionally, the representation of TE-formatted values complicates the adaptability of deep learning architectures. Furthermore, addressing the task requires a significant linguistic capacity, as observed, and conventional deep-learning architectures have proven insufficient. Lastly, there has been limited exploration of the Pre-trained Language Models (PTLMs) approach, and the existing research has not delved into sophisticated fine-tuning techniques to enhance performance.

This work aims to solve the existing challenges on the TE normalization task: data scarcity, language and domain adaptation, and TE-formatted value representation on deep learning architectures through the exploration of sophisticated fine-tuning and data representation techniques for TE normalization, improving the performance of existing solutions.

In order to do this, we introduce a novel deep-learning methodology for normalizing TEs. We will train a *XLM-RoBERTa* model (Conneau et al. 2020), which possesses a much higher linguistic capacity than rule-based systems and non-transformers' deep learning architectures. This model provides multilingual capabilities, which through the use of the proposed methodology will provide a much more adaptable solution. The intuition behind the use of fill-mask and this type of model is that by being able to independently mask the expression text, type, and value, the model can learn all the semantic structures and relationships between the three components. Understanding the relationship between these three components can lead to better standardization. Also, it also allows the integration of multiple languages and domains to enrich the capabilities of the model. Thus, a performance superior to the current state of the art can be achieved. We have used a fill-mask approach inspired by Lange et al. (2023), for which we introduce a Value Intermediate Representation (VIR) designed for token-based architectures, like fill-mask, to represent the TE values effectively. With this approach, we minimize the TE-formatted values representation problem. The training process will be divided into two phases. In the first one, we will mitigate the data scarcity problem. We propose the use of a current state-of-the-art rule-based system called *Annotador* (Navas-Loro et al. 2020) for automatically annotating the temporal expressions of a large English biomedical domain corpus—*MIMIC-III* (Johnson, Pollard, and Mark 2016)—from which we have used only the plain text. To complement them we have hand-crafted a TE dataset of 12*k* uncommon date expressions, like calendar dates, centuries, or decades. We will train the model over these two corpora while exploring different novel masking policies, with which the performance of classical techniques can be improved.

On the second one, we will perform a domain and language adaptation of the model, to biomedical (*E3C*; Magnini et al. 2020) and news (*Timebank*; Verhagen et al. 2007) domains using the available multilingual annotated TEs in Italian, French, English, and Spanish present in these corpora, together with the freezing layers technique to minimize catastrophic forgetting.

We show how, with just a few annotations, our model can be adapted to the biomedical and news domain in multiple languages: Italian, French, English, and Spanish. With the sum of all, we improve the performance of the current best multilingual normalization model, *XLM_Bosch* (Lange et al. 2023) and other monolingual normalization systems in E3C and Timebank corpora. Our model and complete pipeline, along with training scripts and all training data, will be publicly available as long as licenses allow.[1]

2. Related Work

The most widely used schema for TEs is *TimeML ISO standard* (Pustejovsky et al. 2010), in which TEs are defined using the *TIMEX3* tag, although there are other minority schemes such as *scate* (Laparra, Xu, and Bethard 2018). This tag classifies TEs into 4 types (*DATE*, *DURATION*, *TIME*, and *SET*) along with some attributes, from which the *value* is the most relevant. The TimeML schema is under constant review by the community (Suleymanova and Trofimov 2021), as the annotation system is flexible and open to interpretation.

HeidelTime (Strötgen and Gertz 2010) is the most recognized rule-based system. The main limitation of these systems is that they are difficult to adapt, as they require analyzing the text and hand-crafting new rules for a new domain or language. This issue of domain and language adaptation has been studied, specifically for low-resource languages (Skukan, Glavaš, and Šnajder 2014; Li et al. 2014; Moriceau and Tannier 2013; Mansouri et al. 2018). After HeidelTime, other rule-based systems have arisen like *Annotador* (Navas-Loro et al. 2020), the state-of-the-art for the Spanish and English legal domain, *Parstime* (Mansouri et al. 2018) for Persian, or *CogCompTime* (Ning et al. 2018) which can be considered the current state-of-the-art on English. There are other systems that use different techniques such as *UWTime* (Lee et al. 2014), which uses a Combinatory Categorial Grammar to construct compositional meaning representations, while

1 `https://github.com/asdc-s5/Temporal-expression-normalization-with-fill-mask`.

considering contextual cues, to compute the final time values. Or *DNPTime* (Ding et al. 2021), where the authors model temporal expression normalization as a sequence of operations to construct the normalized temporal value, and they present a novel method that can automatically generate normalization rules from training data without expert interventions. Or *TimeNorm* (Escribano, Rigau, and Agerri 2023), where the authors use a synchronous context-free grammar using an extended version of the CYK+ algorithm for normalization.

Temporal expression normalization has also been treated with mixed machine learning and rule-based systems like those of Ning et al. (2018) and Ding et al. (2021). These systems bring more flexibility regarding domain and rule adaptation, which requires less human intervention. Still, generally, they obtain worse performance than rule-based systems.

Deep learning approaches (Kim and Jeong 2016; Etcheverry and Wonsever 2017; Lange et al. 2020; Sánchez-de Castro, Araujo, and Martinez-Romo 2023) have contested rule-based systems, but their data-hungry nature posed challenges due to small and scarce TE corpora.

Recently, research has slowed in TE normalization, especially in languages other than English, mainly because it is a hard problem to tackle with recent deep learning approaches and the range of solutions is not that wide. However, the disruption of PTLM has brought new opportunities as shown in Sánchez-de Castro, Araujo, and Martinez-Romo (2023) or Lange et al. (2023). In the latter, the authors propose the use of an XLM-based model with a fill-mask objective for token prediction, where each token to be predicted is part of the TE value. Their results show the potential of PTLMs for this task, particularly in low-resource languages, where they outperform HeidelTime.

Some studies, such as Gautam, Lange, and Strötgen (2024) explore the use of recent GPT models—specifically GPT-3.5—for TE normalization. In that work, the authors propose several prompting techniques for improving the model's performance. While their approach shows advantages in specific cases, it proves less effective when evaluated on well-established datasets like Timebank. In particular, their method underperforms by 6.4 points compared with the approach proposed in Lange et al. (2023) on English Timebank with a much higher energy cost.

Fill-mask or masked language modeling has been used for pre-training language models like BERT (Devlin et al. 2018) and it can be applied to a bunch of tasks, like data augmentation (Zhou et al. 2022), named entity recognition (Liu, Zhu, and Zhu 2020), text classification (Moon et al. 2021), or sentiment transfer (Wu et al. 2019). A key element in fill-mask is the masking policy, determining which and how many tokens are masked. This process involves selecting and masking certain words in the input. The model is then trained on this masked input to predict the original tokens, enabling the learning of syntactic structures and token relationships. The general rule is to mask 15% of the input tokens, but recent work casts doubt on this (Wettig et al. 2023; Yang, Zhang, and Zhao 2023). As for what to mask, there are some suggestions on how to select the most relevant tokens, such as Pointwise Mutual Information (Levine et al. 2021), Salient Span Masking (Guu et al. 2020; Cole et al. 2023), SpanBERT (Joshi et al. 2020), or Weighted Sampling (Zhang et al. 2023).

In summary, current solutions for TE normalization lack adaptability to domain and language, which hampers addressing the shortage of annotated data. This could be mitigated by utilizing contemporary language model architectures along with various training techniques, including specific masking methods tailored for temporal expressions.

3. Approach

In this section, we will describe all the technical details that have been addressed.

3.1 Temporal Expression Anchoring

TEs often lack complete information about their value. For instance, *"20th April"* specifies a day and month but omits the year, necessitating the use of an additional reference point, known as a reference date or reference date, to resolve this ambiguity.

This reference date might sometimes be found implicitly within the surrounding context; for instance, in the sentence *"the patient was admitted on 12 April...the following day..."* where *"the following day"* refers to the 13th of April. However, in many cases, such implicit temporal references are absent, leaving the document creation time (DCT) as the only available point of reference for resolving the expression. In these situations, the DCT provides the necessary context for interpreting ambiguous TEs. For instance, if the document was created on 15 April 2023, the phrase *"the following day"* would be assumed to refer to 16 April 2023. The reliance on DCT underscores the importance of knowing when a document was produced to accurately normalize temporal expressions in texts lacking clear temporal cues.

Normalization systems usually incorporate an anchoring mechanism, like *Annotador*, which uses a two-step approach to normalize TEs. First, they extract an unfinished TE value called context intermediate representation (CIR), based only on the TE textual content. For instance, for the TE *"20th of April"* the corresponding CIR would be *XXXX-04-20*, indicating an unspecified year since the TE does not reference any year. This CIR is then anchored to the reference date, whether it is the DCT or another identified TE in the text. Furthermore, the anchoring mechanism must possess the ability to perform various date-related operations, such as adding or subtracting units of time, including days, weeks, months, years, hours, or even larger segments like seasons. There are also operations where access to a calendar is required. For example, in the temporal expression *Next Wednesday* it is essential to know the day of the week of the reference date, as the upcoming Wednesday will differ depending on whether the reference date falls on a Thursday or a Monday. Ultimately, these operations enable the normalization process to accurately resolve relative and incomplete temporal expressions within a given context.

It is important to note that not all TEs require an anchoring process. Expressions classified as *DURATION* or *SET* inherently lack a relationship with any specific reference date and thus do not need anchoring. To determine the complete value of a *DURATION* or *SET*, it is sufficient to identify the time period over which the expression extends. Establishing which exact time point the expression refers to is a separate process from normalization, which involves situating the various temporal expressions on a timeline and linking them accordingly. For example, TEs such as *"for 8 hours"*, *"about two years ago"*, *"nine months"*, or *"every month"* do not need to be associated with a reference date to know their value. Consequently, when anchoring is not required, the CIR and the final normalized value will be identical. Conversely, most TEs classified as *DATES* and *TIMES* require an anchoring process to fully resolve their value. *TIME* TEs are composed of a full date and a time, for instance, the TE *"12 a.m"* with a reference date of *"2024-01-01"* would have a value of *"2024-01-01T12:00"*. However, in practice, these TEs often specify only the time component without an accompanying date, making it necessary to anchor them to a reference date. For example, expressions like *"3 p.m."*, *"this morning"*, or *"in the evening"* lack any explicit date information and must be anchored to the reference date to form a complete TE.

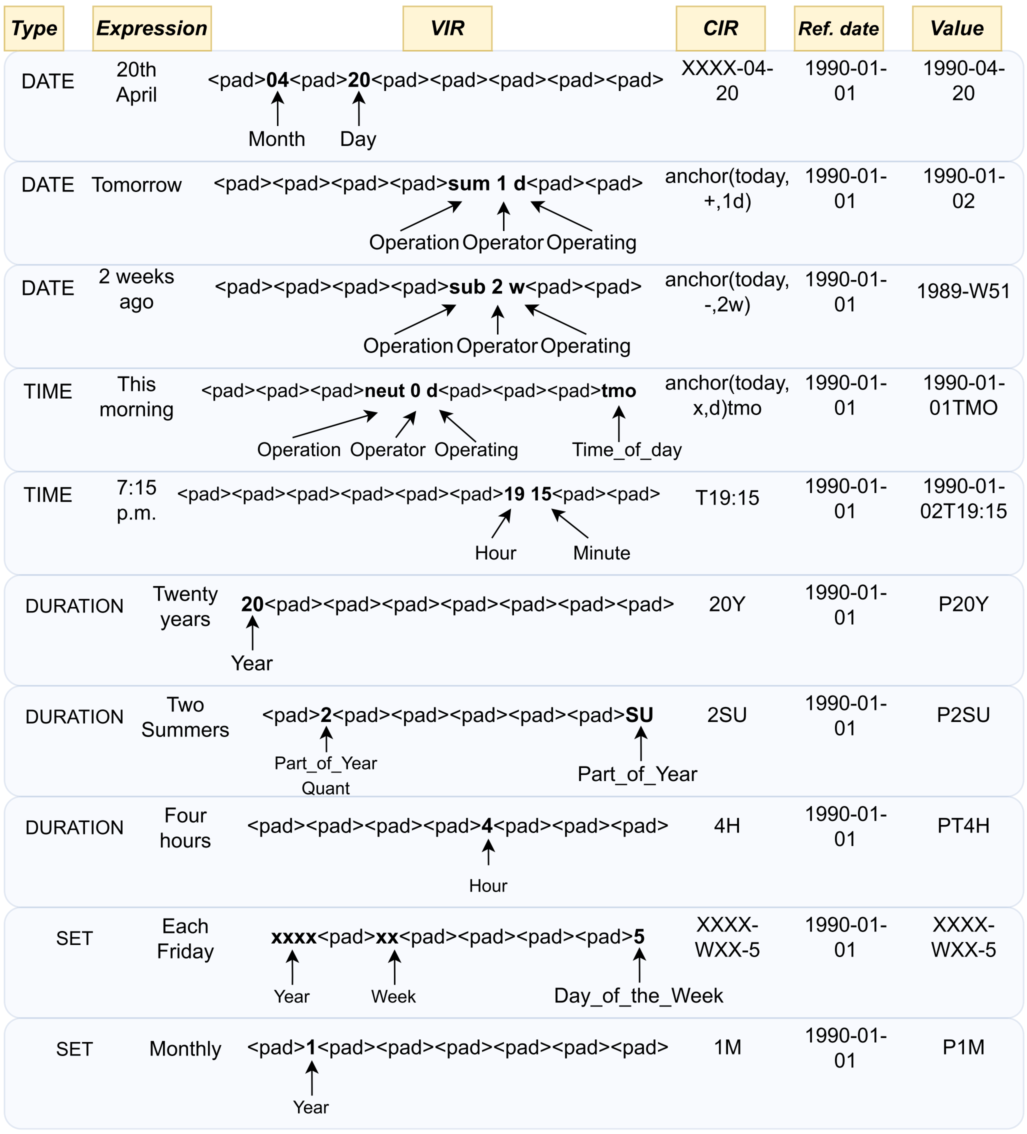
A guiding principle is that the granularity specified by the temporal expression sets the minimum level of detail, while the anchoring process must provide the higher granularity that is not explicitly stated. For example, in the TE *"3 p.m"* the specified granularity is the hour, requiring the anchoring system to supply the higher missing levels of day, month, and year, or alternatively, week and year. On the other hand, granularities below the minimum level indicated by the TE cannot be inferred from the reference date. For example, in the TE *"April"* the granularity is at the month level, making it impossible to deduce the exact day, hour, minute, or second. This leads to the fact that most *DATES* and *TIMES* need anchoring since they rarely state all levels of granularities, such as months rarely specify the year or days often omitting the month and year. This is primarily due to the economy of language, where details at broader granularities can typically be inferred from the surrounding context or textual clues, rather than being explicitly stated. Hence, anchoring plays a crucial role in filling in these gaps to achieve a complete and accurate temporal normalization.

3.2 Fill-mask

The normalization problem can be seen as a translation from a string into a *TIMEX3* value. Seq2Seq models (Britz et al. 2017) generate a string based on another string, which is great for language translation. Models like T5 (Raffel et al. 2020) have been applied with good results (Edwards et al. 2022; Xue et al. 2021; Mager et al. 2021). This kind of architecture has been explored for TE normalization in Ding et al. (2023), where they use a T5 model, obtaining similar or worse performance than other existing models, concluding how hard these kinds of models are to align to the constraints of the TimeML schema.

On its side, the fill-mask approach consists of masking certain words or tokens in a text so that the model predicts them. This method enables the model to predict a sequence of tokens with a predetermined length defined by the user, making it suitable for predicting the value of a TE. However, the main issue with fill-mask models is to know in advance the exact length of the sequence that has to be generated. This means that the length of the TE CIR must be known before normalizing it. For example, in TEs of type *DURATION* like *"1 year"* the value would be *"P1Y"* but for *"1 year and 2 months"* the value would be *"P1Y2M"*. Or in TEs of type *DATE* like *"The past year"*, *"This month"*, or *"Tomorrow"*, where the values would be respectively *1989*, *1989-12*, and *1989-12-31* with a reference date *1990-01-01*. As it can be seen, the length of the value varies and predicting this length with absolute accuracy is highly challenging. Therefore, a suitable solution is to regularize the length of every TE CIR so that all have the same length.

Inspired by Lange et al.'s (2023) intermediate representation for fill-mask, we introduce a more space efficient Value Intermediate Representation (VIR). The VIR is a structured, fixed-length format that encapsulates all potential variations of the CIR, capturing TE values. This format is organized into a series of slots, where each slot corresponds to a distinct component of the CIR. Importantly, since the VIR must accommodate any possible TE representation within a predefined structure, it is not expected that all slots will be utilized for a single TE. Any unused slots are therefore populated with padding tokens to preserve the fixed-length format. Examples illustrating the relationship between a TE's CIR and its corresponding VIR are provided in Figure 1. In the first example, a TE of type *DATE* is given as *"20th April"*, which contains no



Type	Expression	VIR	CIR	Ref. date	Value
DATE	20th April	<pad>**04**<pad>**20**<pad><pad><pad><pad><pad> Month Day	XXXX-04-20	1990-01-01	1990-04-20
DATE	Tomorrow	<pad><pad><pad><pad>**sum 1 d**<pad><pad> Operation Operator Operating	anchor(today, +,1d)	1990-01-01	1990-01-02
DATE	2 weeks ago	<pad><pad><pad><pad>**sub 2 w**<pad><pad> Operation Operator Operating	anchor(today, -,2w)	1990-01-01	1989-W51
TIME	This morning	<pad><pad><pad>**neut 0 d**<pad><pad><pad>**tmo** Operation Operator Operating Time_of_day	anchor(today, x,d)tmo	1990-01-01	1990-01-01TMO
TIME	7:15 p.m.	<pad><pad><pad><pad><pad><pad>**19 15**<pad><pad> Hour Minute	T19:15	1990-01-01	1990-01-02T19:15
DURATION	Twenty years	**20**<pad><pad><pad><pad><pad><pad><pad> Year	20Y	1990-01-01	P20Y
DURATION	Two Summers	<pad>**2**<pad><pad><pad><pad><pad>**SU** Part_of_Year Quant Part_of_Year	2SU	1990-01-01	P2SU
DURATION	Four hours	<pad><pad><pad><pad>**4**<pad><pad><pad> Hour	4H	1990-01-01	PT4H
SET	Each Friday	**xxxx**<pad>**xx**<pad><pad><pad><pad>**5** Year Week Day_of_the_Week	XXXX-WXX-5	1990-01-01	XXXX-WXX-5
SET	Monthly	<pad>**1**<pad><pad><pad><pad><pad><pad> Year	1M	1990-01-01	P1M

Figure 1
Equivalence between a *TIMEX3* value and its Value Intermediate Representation (VIR).

explicit reference to a specific year. Its VIR, reflecting only the information available in the TE, records the month and day, while unused slots are filled with padding tokens. Accordingly, its CIR is represented as *"XXXX-04-20"*. When normalized using the reference year *1990*, the complete normalized date is derived as *"1990-04-20"*.

The second example presents another *DATE* TE with text *"Tomorrow"*. This case involves an operation, requiring one day to be added to the reference date. The CIR represents this operation explicitly with an *anchor* operation, which performs simple additions or subtractions. This operation requires three arguments: the reference date, the operation and the quantity. Therefore, the VIR encodes these three parameters as *"sum 1 d"* (denoting the addition of one day). Using the Annotador anchoring system, the final normalized value is then calculated as one day after the reference date as *"1990-01-02"*.

Therefore, the pipeline to obtain the value of a TE follows these steps: First, the VIR is obtained by the model and translated to CIR through a regularization process using regular expressions. Finally, the CIR is anchored to the reference date using *Annotador's* anchoring system to obtain the final value. This structure allows for the application of a fill-mask model, where the system predicts each slot's value.

We define one VIR per type of TE since each one has its own format. This way we reduce the length, optimizing the space and the tokens that have to be unmasked:

DATE: The VIR is composed of 9 slots:

- YEAR | MONTH | WEEK | DAY: These four slots are self-explanatory. The only detail to keep in mind is that a well-formed value cannot simultaneously reference both a week and a month at the same time. For instance, the expression *"the first week of the year"* would result in a VIR containing only a value in the week slot (set to 01), while all other slots remain filled with padding tokens. This would translate to a CIR of *"XXXX-W01"*, and, using the reference date *"1990-01-01"*, the final normalized value would become *"1990-W01"*. As this example illustrates, it is not possible to encode both a month and a week within the same representation. For TEs that specify days of the week, both the week and day slots are utilized. For example, the TE *"Monday of the first week of the year"* would have 01 in the week slot and 01 in the day slot. Based on the same reference date, this would normalize to *"1990-W01-01"*. Note that days of the week are consistently encoded with numerical values ranging from 01 (Monday) to 07 (Sunday).
- OPERATION: This slot can store three different values: SUM, SUB, and NEUT, which stand for addition, subtraction, and neutral operations like in the TE *"Today"*. This TE would have a VIR of *NEUT 0 Y* for the slots OPERATION OPERATOR OPERATING. This slot also contains the *"BC"* token, which is used to represent years in the B.C. era, such as *"the year 4000 B.C."*.
- OPERATOR: This slot stores the value on which the operation is performed.
- OPERATING/FUZZY/TIME OF THE YEAR: This slot when working as *OPERATING* can refer to any time unit, e.g., year, month, days, quarters, seasons, etc. For example, in the TEs *"the next/past year/month/week/summer/quarter"* the slot would store *Y/M/W/SU/Q*, respectively. As for the *FUZZY* term it refers to *PRESENT_REF*, *FUTURE_REF*, and *PAST_REF*, which refers to TEs like *"now"*, *"soon"*, and *"recently"*, respectively. With regard to *TIME OF THE YEAR* there are cases when it is necessary to store values such as seasons or quarters without the need for any accompanying operations. For example, in the TE *"Summer"* there is no operation needed so this slot just stores the value *SU*. These three possibilities share the same slot since there cannot exist an expression

that contains all three terms. This saves two slots, optimizing space. We will apply the same practice to the rest of the VIR's TE types.

Since dates have a wide variety of possibilities, multiple types of operations can be defined. Therefore we use five types of operations defined by Annotador:

- Day-of-the-week-based operations: This kind of operation needs the current day of the week to properly anchor the final value. For example, in TEs such as *"next Friday"*, identifying the next Friday depends on the day of the week of the reference date. If the reference date is a Thursday, the next Friday is the following day. However, if the reference date is a Sunday, the next Friday falls five days later.
- Month-based operations: This includes TEs such as *"next September"* where it is necessary to identify the current month to know which is the next September.
- Season-based operations: With TEs like *"past summer"* where knowing the current season is needed.
- Date-based operations: As in *"next 20 of April"* where the current date is needed.
- Numerical operations: Including TEs like *"two days before"* or *"three months after"* which involve arithmetic operations relative to the reference date, but they don't need access to a calendar in order to be anchored.

- CENTURY: This slot stores the value for centuries. For example, in the TE *"The 20th century"* the slot would store the value *19*, which refers to the first two digits of the century representation in decimal format, that is, *19XX*.
- DECADE: This slot stores the value for decades, which is just the first digit of the decimal representation of the decade. For example, in the TE *"The decade of the 20s"* the slot would store the value 2. Thus, if we join the slot of centuries and decades, we can represent TEs such as *"The second decade of the 20th century"* with the VIR *19 2*, which would result in the value of *192*.

TIME: It is composed of 10 slots. It is similar to the *DATE* VIR, mainly because times contain dates:

- YEAR | MONTH | DAY | OPERATION | OPERATOR | OPERATING | HOUR | MINUTE | SECOND: As it can be seen in this listing, we have dispensed with the *WEEKS* slot because they imply a vague date. For example *"the second week of the year"* implies seven days, which makes it impossible to specify particular hours. The same happens with decades, centuries, and fuzzy expressions. *TIME* expression can also have operations like *"Tomorrow morning"* as can be seen in the first *TIME* example in Figure 1, but the operation will always refer to a date with a fixed time so we use the same kind of operations. It is worth mentioning that expressions like *"two hours later"* are usually marked as *DURATIONS*, since there is usually no other time hour to anchor the operation and for event and time relation anchoring are equally useful.

- TIME OF THE DAY: We have added this slot for referring to *afternoon*, *morning*, *midnight*, *evening*, and *night* with the tokens *AF*, *MO*, *MI*, *EV*, and *NI*, respectively.

DURATION: This kind of expression does not include operations but comprehends durations of any granularity. The VIR is composed of 8 slots:

- YEAR | MONTH/QUANT TIME OF THE YEAR | WEEK | DAY: A *DURATION* must store values for years, months, weeks, and days for TEs such as *"a month"*, *"a year"* or *"two weeks"*. But it also must store time of the year quantifiers such as *quarters*, *weekends*, or *seasons* in TE like *"for 2 springs"*.
- HOUR/QUANT TIME OF THE DAY | MINUTE | SECOND: Smaller granularities like hours, minutes, and seconds must also be stored for TEs like *"three hours"* or *"five minutes"* and also times of the day quantifiers such as *morning* or *night* in TE like *"nineteen mornings"* or *"five hundred nights"*.
- TIME OF THE DAY/YEAR: This slot stores which time of the day or year the quantifier slots refer to. For the TE *"nineteen mornings"* the *QUANT TIME OF THE DAY* slot would store 19 and this slot would store *MO* (from morning). It should be noted that a time of the year and a time of the day cannot coexist in the same TE, since they have incompatible scales in the same way that a month and a week cannot coexist. In the case of an expression in which both options are mentioned consecutively, they will be annotated as different TEs.

SET: This VIR closely resembles that of *DURATION*, sharing a similar format with minor differences and the same length of 8 slots: [*YEAR | MONTH/QUANT TIME OF THE YEAR | WEEK | DAY | HOUR/QUANT TIME OF THE DAY | MINUTE | SECOND | TIME OF THE DAY/YEAR/DAY OF THE WEEK*]. In the *SET* type can appear TEs such as *"Each Friday"*, generally referring to specific days of the week. To accommodate this, we have added the option in the final slot to specify the day of the week as can be seen in the final example of Figure 1. The rest of the VIR is the same as the *DURATION's*.

In summary, we propose the use of the VIR, a fixed-length equivalent of the CIR, which represents the unanchored value of a TE. This design enables the application of a fill-mask PTLM architecture in conjunction with an existing and tested anchoring system. The VIR approach offers a key advantage since it relieves the model from anchoring TEs, avoiding the need for complex date calculations or an internal calendar representation, thereby simplifying the task significantly.

The proposed representation differs significantly from the one introduced by Lange et al. (2023). To understand these differences, it is necessary to first examine how their representation is structured:

- **First slot**: This slot stores the BC token, serving as an auxiliary component for operations alongside the last three slots.
- **Second and third slots**: These slots are used to store the year, split into two parts (e.g., the year *1940* is divided into *"19"* and *"40"*). These slots also encode undetermined dates, temporal references such as past, present, and future, and certain *DURATION* values.
- **Fourth and fifth slots**: These are designated for storing month and day information.
- **Sixth to eighth slots**: These slots store hours, minutes, and seconds. The hour slot additionally encodes part-of-day tokens (e.g., evening, night) and certain duration values related to hours, minutes, and seconds.
- **Ninth to eleventh slots**: The final three slots capture information related to operations as defined in HeidelTime.

Once we introduce their representation, several key differences emerge. Even though both representations share certain fundamental features, such as slots for months, days, and years, Lange's representation lacks dedicated slots for certain temporal elements such as specific parts of the year (such as summer or quarters), weeks, centuries, and decades. Also, they do not explicitly define the typology of used operations, although they use 4 slots while we use 3. In addition, their approach relies on a single representation for all types of TEs. As a result, their method squeezes multiple types of temporal elements into a small number of slots. As a result, these slots become overcrowded and the information is heavily condensed. This approach forces important details to be overgeneralized, compromising the accuracy and efficiency of the representation. In contrast, our representation is more space-efficient and tailored to the specific needs of each type of TE. For instance, when applied to the Timebank and E3C corpora, Lange's representation requires 51,392 tokens to unmask, while our representation only needs 40,662 tokens—a reduction of approximately 21%. Lange's representation, in theory, could be optimized to achieve similar efficiencies. For example, DATES could be encoded with only 8 slots by omitting the hour, minute, and second slots. However, based on the information provided in the paper, it remains unclear whether removing these slots would still allow for the full encoding of all possible DATE information. These differences are understandable, as Lange's representation was designed to reflect the way HeidelTime handles TE values, while our proposed representation is based on Annotador. Since HeidelTime and Annotador are different systems, their representations naturally diverge, like in the definition of partial values or operations, which are defined significantly different in each system. Consequently, although both representations originate from the same conceptual foundation, they are implemented within distinct systems, employ different approaches, and yield notably different results.

3.3 Masking Policy

Within a fill-mask approach, the masking policy comprises the masking strategy, which determines which tokens to mask, and the masking function, which specifies the number of tokens to mask.

3.3.1 Masking Strategy. Regarding the masking strategy, we distinguish between masking TE and masking regular tokens. For TEs, we introduce a tag in the text following the format *"<TYPE EXPRESSION VIR>"* where, for example, the expression *"tomorrow"* is tagged as *"<DATE tomorrow <pad>...sum 1 d...<pad> >"*. The tag components are deeply related: The type defines the VIR's structure and potential values, and the string provides values and positions. Just knowing two of them is enough to predict the remaining one, therefore masking one component at a time enriches the model's knowledge. Experimental results suggest that masking 40% of the type, 40% of the string, and 20% of the VIR improves the model's ability to predict the VIR, fostering attention to its structure and values, although it may not be optimal.

Additionally, regular tokens are masked with a 10% probability, for two main reasons. First, as TEs constitute a small fraction of total tokens, masking only TEs would result in a negligible overall percentage of masked tokens. Second, to prevent catastrophic forgetting, a diverse set of masked tokens is crucial; hence, a 10% probability is deemed sufficient. Contrary to the 80-10-10 corruption rule proposed by Devlin et al. (2018), recent research (Wettig et al. 2023) suggests that simple masking without the proposed rule may be more beneficial for downstream tasks. Furthermore, whole-word masking is applied to regular tokens.

3.3.2 Masking Function. Regarding the masking function, a constant 10% is applied to regular tokens, while time-variant decreasing masking is used for TEs. Observations by (Yang, Zhang, and Zhao 2023) suggest that training a BERT model with a 30% fixed masking ratio versus 15% results in the former showing an early performance spike, and the latter exhibiting better performance towards the end of training. This phenomenon is akin to learning rate decay, where the model explores the entire search space in the initial stages and converges to a final minimum in a shrinking space as training progresses. Larger training corpora accentuate this effect, while it may not be as noticeable in smaller datasets.

When applied to the masking function, the more tokens that are masked, the more search space the model has available, and vice versa. To fulfill the above-mentioned requirements, those authors propose to use a masking ratio decay. Using a linear and a cosine function to decrease the masking ratio, they improved the performance of a *BERT* model on a series of downstream tasks, with the cosine function proving superior due to its ability to maintain a larger initial masking ratio.

When trying to apply this function to the context of TEs, we found that both linear and cosine functions brought several problems. Figure 2 shows a comparison between the constant function and the linear and cosine functions proposed in the above-mentioned work, as well as our proposed *linear+cosine* and *linear+logistic* functions. We plot a masking decay from 100% to 20%, dividing the training into initial exploration and final optimization phases with a mean masking ratio of 60%. It can be seen that the cosine function exhibits a notably high masking ratio in the initial 15% training steps, potentially causing prolonged random guessing. Conversely, the linear function may decay too quickly initially and too slowly later, compromising the advantages of masking decay. Combining *linear+cosine* or *linear+logistic* functions addresses these

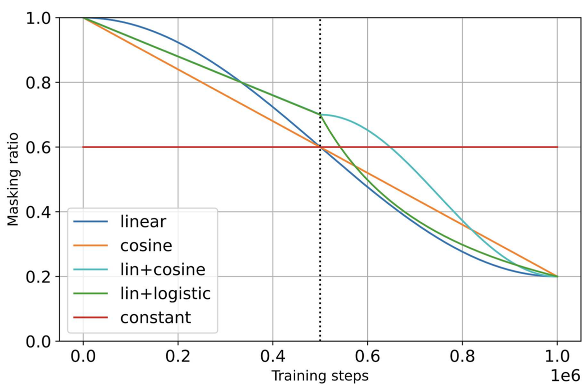


Figure 2
Evolution of the masking ratio for different functions over 1 million steps.

issues. For the first half of training, a linear function with reduced gradient avoids rapid decreases and overly high initial masking ratios. In the second half, the logistic function ensures a rapid yet decelerating decrease, while the cosine function provides a slower but accelerating decrease. This combination maintains the same mean masking ratio as linear, cosine, and constant functions but with more suitable initial and final masking rates.

Accordingly, we have defined the *linear+logistic* and *linear+cosine* functions as piecewise functions.

$$M_{lin\_cos}(t) = \begin{cases} m \cdot t + c & 0 \leq t < \frac{T}{p} \\ (1 + \cos(\frac{\pi \cdot r}{s}))(a + b) & \frac{T}{p} \leq t < T \end{cases}$$

$$M_{lin\_log}(t) = \begin{cases} m \cdot t + c & 0 \leq t < \frac{T}{p} \\ \frac{1}{(d-l)+(l \cdot e^{r/T})} & \frac{T}{p} \leq t < T \end{cases}$$

For the first T/p steps of the $M(t)$ range we define the linear function, where p is the percentage of steps. In the case of Figure 2 where $p = 0.5$, t is the actual step, c corresponds to the maximum value of the linear function for T steps, and m is the slope, defined as

$$m = \frac{minimum_{lin} - 1}{T \cdot p}$$

For the second part of the $M_{lin\_cos}(t)$ range, we define the cosine function. As the function starts at step T/p, it must be taken as if this were the starting point, therefore we normalize the starting point through $r = t - (T \cdot p + 1)$ and $s = T - (T \cdot p + 1)$. On the other hand, b represents the minimum value of $M_{lin\_cos}(t)$ for T steps and a is defined as

$$a = \frac{maximum_{cos} - b}{2}$$

The first part of the $M_{lin\_log}(t)$ range is the same as $M_{lin\_cos}(t)$. As for the second part of the range we define the logistic function, where $d = 1/minimum_{linear}$, r is defined the same as in $M_{lin\_cos}(t)$, and l defines the curvature of the function as

$$l = \frac{\frac{1}{n} - d}{e^p - 1}$$

Where n refers to the minimum value of $M_{lin\_log}(t)$.

In conclusion, we introduce a comprehensive masking policy that defines what tokens will be masked and in what percentage, introducing a novel masking strategy and masking function.

3.4 Freezing Layers Technique

TE expressions vary based on writing style, domain, and language, as evident in prior research like Bethard et al. (2017), which showed a performance drop of around 20 points in TE detection and classification due to a slight change in the domain. Annotator interpretations also contribute to significant differences, given the flexibility of the TimeML schema, allowing a single TE to be annotated in various ways.

A second fine-tuning process enhances the model's performance on E3C and Timebank domains and languages, but it may result in catastrophic forgetting (Ke et al. 2021; Kar et al. 2022), where previously acquired knowledge is lost. To mitigate this, techniques like *layer freezing* are used. This involves freezing certain model layers during training to reduce forgetting, improve domain and language adaptation, and decrease training time. The optimal number of layers to freeze lacks consensus, but for a 12-layer model like the one used, freezing less than 6 has minimal impact, and freezing more than 9 updates too little information.

3.5 Corpora

For training our model we have used four different corpora:

Medical Information Mart for Intensive Care III (*MIMIC III*): It is an English-exclusive biomedical domain corpus, which includes de-identified hospital patient records. We have used over *300K* records from which we have automatically detected, classified, and normalized around 1,080*K* TEs with *Annotador*. From which 200*K* are *DATE* TE, 480*K* are *TIME* TE, 240*K* are *DURATION* TE, and 150*K* are *SET* TE. During this process, we realized that *Annotador* had problems detecting and normalizing some TE. Since it is a rule-based system, we have corrected and added some rules so it could handle more TEs:

- **Detection and normalization rules**
  - We have added two lemmas to the rule *"Rule$PARTDAY"* for the detection of *"midnight"* and *"on midnight"* in the text.
  - We have changed the *NI* in *"PARTDAY_MAP"* for *TNI*. This meant that when normalizing expressions that included references to *"night"*, such as *"tonight"*, the value was not assigned correctly. The correct value to assign when there are references to night is TNI according to the annotation standard.

- We have changed the rule *"Rule$TIMEdelfullDATE"* because it added an extra *T* on CIRs like *anchor(today,+,1D)**TT**15:00*

- **Anchoring process**
  - We have added a rule so that Annotador could anchor CIRs like *anchor(TODAY, +, 1WE)*.
  - We have modified Annotador so that it does not erase the month on CIRs like *Sanchor(+, XXXX-XX-12)*.
  - We have modified Annotador so it does process only decades like *XX8*.
  - We have added the option of neutral Danchor for expressions like *"on the 5th"*. This is to accommodate the behavior of the model because it mixed the neutral anchor operation with Danchor. This is not an incorrect behavior so we opted to modify the anchoring process rather than the training data.

Hand-crafted TEs Dataset: In addition to the presented corpora, we have developed a new hand-crafted TEs dataset. While automatically annotating MIMIC-III, we observed significant gaps in the variety of basic date-related TEs. Approximately 89% of the TEs were classified as *past*, *present*, or *future*, demonstrating a limited range of expressions. To address this, we created a 12K TE dataset with fundamental structures to complement and enrich the automatically annotated dataset, enabling the model to acquire stronger foundational capabilities. Each TE in this dataset includes both its textual form and corresponding VIR, with values derivable through an automatic de-regularization process, if needed. The TEs were generated using templates based on semantic structures, with components randomly varied using a predefined pool of options. For instance, one template might follow the structure *"number + part-of-year + present/future/past"*, where *"number"* is a digit, *"part-of-the-year"* can be *years, months, weeks or days* and *future/past* can be temporal modifiers like *previous, past, former, present, same, subsequent, next, coming,* and so forth. Note that when the verbal tense is present the number part is omitted as well as with certain temporal modifiers like *next* or *previous* in TEs like *"the next year"* or *"the previous month"*. This approach automates the generation process, allowing for scalability by adjusting the variety within the pool of options while reducing the production time from months if done manually to just a few days. We did not provide a surrounding context for these artificial TEs, as the primary goal was to address the limitations of the automatically annotated corpus by increasing the diversity of TEs. Adding contextual information would introduce domain-specific biases, whereas our intention was to keep the dataset as domain-neutral as possible. These templates were designed following the TE typology defined by Annotador, a framework we were already familiar with, having utilized it as a base system for normalization. The primary typologies are as follows:

1. ***Calendar dates***: We generate 8,000 calendar dates with the form *"2001"*, *"21st of August"*, *"first week of the year"*, *"late December"*, etc.

2. ***Centuries and decades***: We generate 200 TEs like *"XX century"*, *"second century"*,*"the twenties"*, etc.
3. ***Regular operations*** (*anchor*): These are operations with years, months, weeks and days, for example, *"past week"*, *"next year"*, *"this month"*, *"two days ago"*, etc. We have added 2,000 expressions.

    (a) There is a specific type of regular operation that involves adding or subtracting days of the week, like in *"previous Monday"*, *"on Saturday"* or *"coming Thursday"*. We have added 600 expressions.

4. ***Seasons operations*** (*Sanchor*): Operations like *"next spring"*, *"past autumn"*, *"this summer"*, etc. We add 600 of these expressions.
5. ***Date operations*** (*Danchor*): These are operations that involve adding or subtracting whole dates like in *"on the first of coming June"* or *"the past day 10"*. We have added 600 date operations.
6. ***Specific expressions***: We add some specific expressions which are regular operations but we treat them differently since they are very common. Some examples are *"the day before yesterday"*, *"the day after tomorrow"*, *"the day after the day after tomorrow"*, etc. We have added 300 specific expressions.

The amount of TEs per type is mainly based on the variety of expressions we were able to generate. The greater the variety the greater the number of TEs. For example, it is relatively easy to create a great variety of calendar dates just varying days, weeks, months, and years. But it is harder to create such a great variety of season operations when there are just 4 seasons. We have also taken into account the quantity of each kind of TEs in the automatically annotated dataset. As can be seen in Table 1, there are no *Sanchor*, *Danchor*, and *specific expressions*, and there are very few calendar dates. It is important to remember that these expressions only represent 10% of the total number of *DATES*, since the rest are expressions such as present, past, future, and so on. Regarding *anchor* operations, while the quantity is higher the variety is still scarce, as 87% of these values consists in just adding one day. Given these imbalances, we have focused exclusively on DATES as other TE types in the dataset demonstrated a more balanced distribution and did not require similar augmentation. While creating this dataset, we carefully considered potential data leakage risks concerning the Timebank and E3C test sets. To prevent any contamination, we ensured that none of the expressions in the

Table 1
TE distribution in automatically annotated MIMIC-III of *DATES* based in the typology defined in Annotador and used for the hand-crafted dateset. The point represents thousands not decimals.

Calendar	Cent & Dec	Anchor	Sanchor	Danchor	Specific	Total
1.256	2.282	18.118	0	0	0	21.656

Table 2
TE distribution in E3C and Timebank corpora based on each language.

	Language	Training	Test	Total
Timebank	English	1,053	156	1,209
	French	205	81	286
	Italian	522	126	648
	Spanish	1,093	198	1,291
E3C	English	153	174	327
	French	118	157	275
	Italian	110	177	287
	Spanish	146	200	346

dataset overlap with those in the test sets. Although the primary goal of this dataset is to complement the automatically annotated MIMIC-III corpus, it is versatile enough to serve as a *DATES* TEs foundational corpus. Its broad yet basic coverage of *DATES* makes it a valuable auxiliary dataset for training other systems.

European Clinical Case Corpus (*E3C*): This is a manually annotated corpus of clinical cases. It includes multiple language annotations such as Spanish, Italian, French, and English, as shown in Table 2.

Timebank: This is a multilingual news manually annotated corpora. The corpus contains multiple language annotations such as Spanish, Italian, French, and English, as shown in Table 2 in which it can be seen that English and Spanish account for a large part of the representation. It can also be observed that the E3C test sets are generally larger than the training and Timebank test sets.

The frequency of TE types varies by domain, being inversely related when considering individual datasets, as shown in Figure 3. This inverted relation between types in

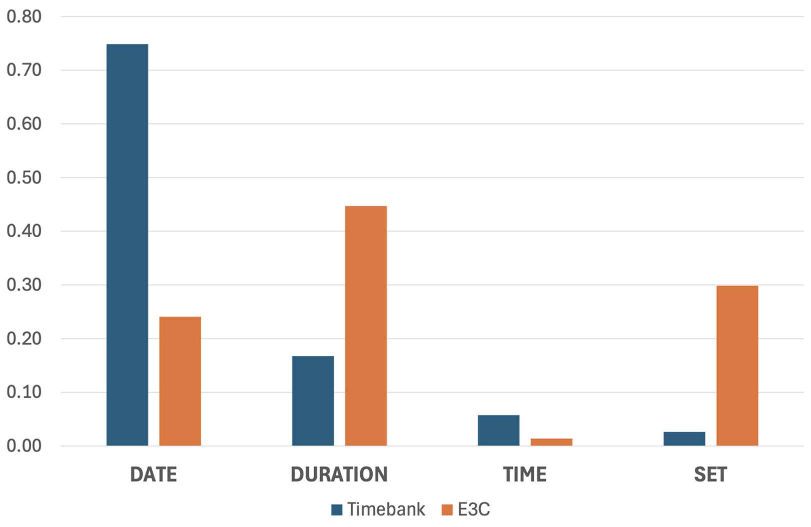


Figure 3
Distribution of TE type of manually annotated data by corpus.

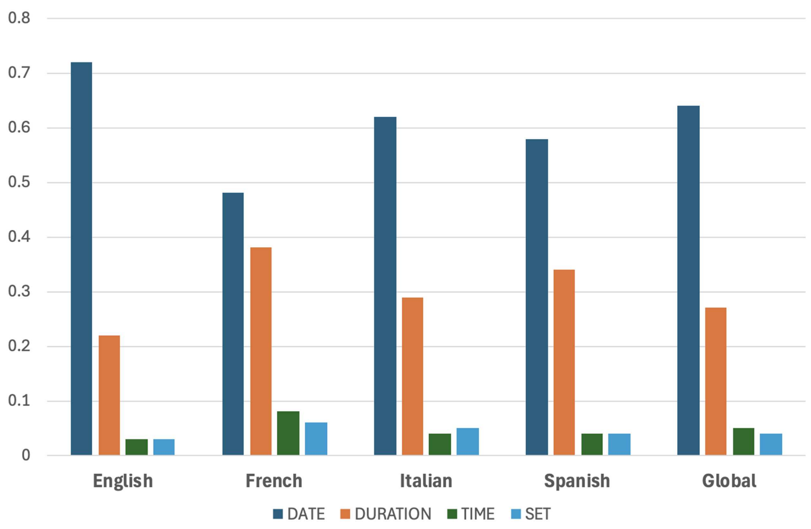


Figure 4
Distribution of TE type per language of manually annotated data (Timebank plus E3C).

both domains may be explained due to the nature of them. In the clinical cases domain of E3C, it may be more frequent to talk about the durations of treatments or recoveries instead of particular dates. Also, it may be more frequent to talk about sets, for example, when prescribing medication with TEs like *"every 8 hours"*. On the other hand, in the news domain of Timebank, it makes sense to use dates and times to refer to the described events more frequently. But in the general picture, the *DATE* type consistently appears as the most frequent across all languages, as shown in Figure 4. This trend is largely due to the higher density of temporal expressions in the Timebank corpus compared to the E3C corpus. The *DURATION* type follows in frequency, whereas the *TIME* and *SET* types occur less frequently, representing a minority in the datasets.

At this stage, we have two sets of data: MIMIC-III, which contains automatically annotated data processed by *Annotador*—including TEs, their corresponding CIR, and their final anchored values—together with the hand-crafted TEs dataset and the manually annotated corpora TimeBank and E3C, which include TEs and their final anchored values but lack CIRs. As discussed in Sections 3.1 and 3.2, using the fill-mask architecture requires a representation like the VIR. Therefore, these corpora must be adapted to this format. Given that CIR and VIR are equivalent, as outlined in Section 3.2, we developed a process called **regularization** allowing automatic conversion from CIR to VIR, and a **de-regularization** process to revert from VIR to CIR. Both processes have been automated through a series of rule sets and regular expressions. For example, *DURATIONS* are built like PxYxMxWxDTxHxMxS, representing years, months, weeks, days, hours, minutes, and seconds, respectively. We converted this structure into our VIR through regular expressions to detect each component. Figure 5 illustrates the pipeline for converting a TE into its equivalent VIR for training on the top part and how the predictions are generated on the bottom part.

Furthermore, since TimeBank and E3C do not contain CIRs for TEs, it was necessary to manually review certain expressions to derive the CIR. As noted in Section 3.1, the CIR and final value are identical if anchoring is unnecessary. In the cases of *DURATIONS* and *SETS*, anchoring is not required, as these represent time spans that do not

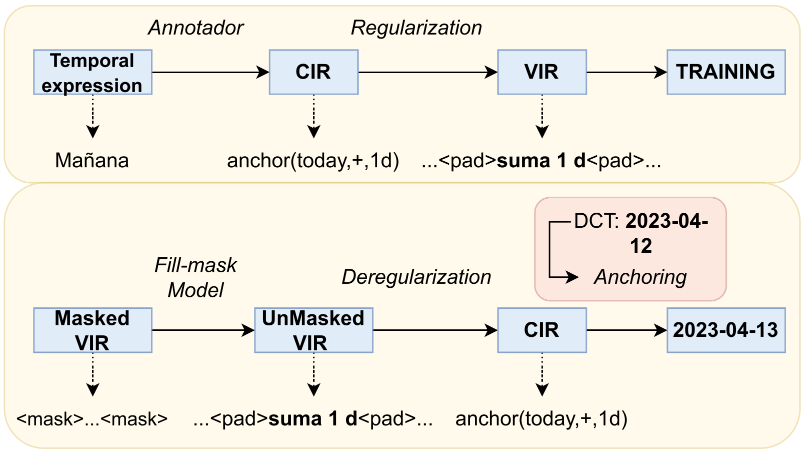


Figure 5
Preprocessing of TE values into VIR for training and post-processing of VIR into final TE value.

depend on a reference date. However, for *DATES* and *TIMES*, anchoring is generally necessary, meaning that the CIR and final value often differ, requiring manual CIR annotation. For instance, a human annotator can easily observe that if the TE is *"20th of April,"* the CIR would be *XXXX-04-20*, since no year is mentioned in the text. It is worth reiterating, that in such cases, the final anchored value would include the year based on the reference date; however, since the model does not handle anchoring, it needs only the CIR rather than the final anchored value. Therefore we have manually annotated CIRs for all necessary TE in TimeBank and E3C. Although it is manual work, it is a light workload, which only needs to be done once per language and in any case the workload is much less than in the adaptation of rule-based systems. Along the way, some misstatements on the E3C and Timebank corpora annotations have been corrected following the TimeML annotation guide (Saurí, Saquete, and Pustejovsky 2010).

3.6 Experiments

Figure 6 illustrates the training process, divided into two phases: initial training with the proposed masking policy and subsequent domain and language adaptation using the freezing layers technique. On this basis, four experiments have been carried out.

The first experiment compares our proposed VIR with the representation introduced by Lange et al. (2023) using the XLM_Bosch model, referred to hereafter as Lange's representation. This experiment aims to evaluate whether the proposed representation, in addition to being more spatially efficient, can achieve comparable or superior performance. Both representations are inherently tied to the rule-based systems upon which they are built. Lange's representation is integrated with HeidelTime, whereas the proposed VIR is designed around Annotador. This dependency arises because each representation closely aligns with the internal data structures and anchoring mechanisms of its respective framework. Notably, HeidelTime and Annotador differ in their definitions of typologies, operations, and the handling of partial *DATE* and *TIME* values, which directly impact the resulting representations. Since Lange's representation cannot be decoupled from HeidelTime, the same 300K MIMIC-III documents

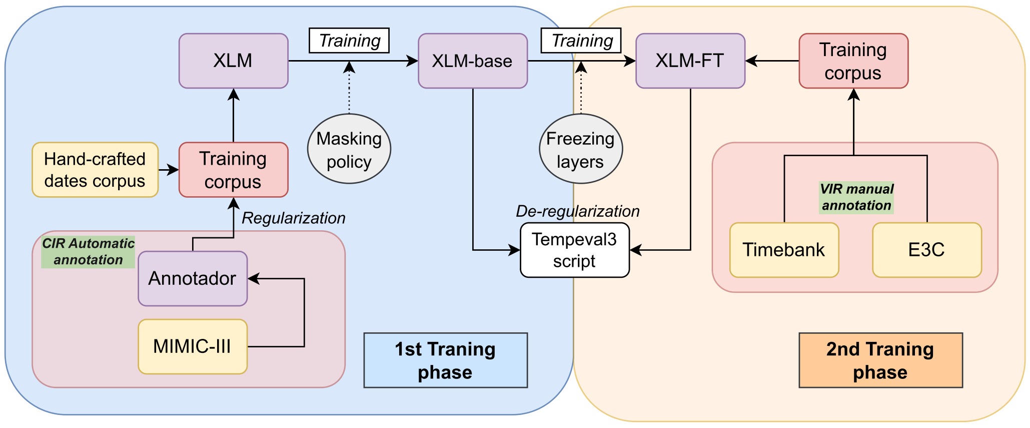


Figure 6
Global pipeline for training. The whole process is divided into two training phases with their corresponding corpora.

(described in Section 3.5) were annotated using HeidelTime. By utilizing the tools provided in its repository, TEs and their corresponding representations were extracted. This process yielded a distribution nearly identical to that obtained via Annotador: 1080K TEs, comprising 202K *DATE* TEs, 478K *TIME* TEs, 243K *DURATION* TEs, and 147K *SET* TEs. This similarity was expected, as both HeidelTime and Annotador demonstrate comparable performance, as shown in Table 7 and in prior studies (Sánchez-de Castro, Araujo, and Martinez-Romo 2023; Navas-Loro et al. 2020). Although using two different training sets could theoretically introduce noise, the high degree of similarity between the datasets suggests that any resulting differences would be minimal. It is important to note, however, that since HeidelTime uses its own typology of expressions, it is not possible to perform a detailed distributional analysis (such as that in Table 1) to verify alignment with Annotador's outputs. To evaluate the performance of these representations, a XLM-RoBERTa large model was trained on the MIMIC-III corpus using both VIR and Lange's representation. The model's performance was then tested on the Timebank and E3C datasets in English, with careful consideration of potential noise when interpreting the results.

The second experiment compares the impact of the hand-crafted TEs corpus by evaluating the performance of a *large* model trained on MIMIC-III against a model trained on these MIMIC-III plus the hand-crafted TEs corpus, comprising up to 112K TEs.

Thirdly, we will compare the performance of three different masking functions: constant, *linear+cosine*, and *linear+logistic*, for training the model over MIMIC-III and the hand-crafted TEs corpus. We have used a mean masking ratio value of 60% for all three functions and a 10% masking ratio for regular tokens. Therefore, an average masking of around 15% is achieved in the training and dev set. Additionally, we will train two sizes of *XLM-RoBERTa*, *base* and *large*, so that we can compare the performance-size trade-off.

Finally, in the fourth experiment, we will test whether it is better to freeze 6, 9, or no layers for the domain and language adaptation phase, where we will train the model on all E3C and Timebank languages training sets (English, Italian, French, and Spanish).

In all experiments we have randomly split the training into 90% training and 10% development, keeping the distribution of temporal expression types between the two sets.

Finally, our model will be compared with other recognized systems such as *GPT-4o mini*, *HeidelTime*,[2] *UWTime*,[3] *CogCompN*,[4] *DNPTime*,[5] *TimeNorm*,[6] *SUTime*,[7] *ARTime*,[8] and *XLM_Bosch*.[9] We have named the last model on our own since the authors did not provide it.

Regarding GPT-4o mini we have developed two experiments. In the first one, we just generate the predictions without any prompt technique or special hyper-parameters. We just introduce the TimeML schema and prompt the TE asking for the value, parse the responses and evaluate them, as follows:

- Base prompt: *Temporal expressions are those constructions referring to points or intervals on the timeline. They can express: (a) day times, e.g., "noon", "3:00 pm", "this morning"; (b) dates, e.g., "April 28th, 2008", "yesterday", "next week", "three months later", "last year"; (c) durations, e.g., "two months", "five hours", "the coming 9 years"; and (d) sets, e.g., "once a month, "every Tuesday". Your work is to obtain the value of the following temporal expressions according to the TimeML schema. Use the document creation time, the type and the text of the temporal expression to obtain the value. Finally, display your response in a JSON format.*
- Consecutive prompts: *Obtain the value of the following temporal expression with DCT "20-04-1990", type "DATE" and text "Yesterday".*

Secondly, we introduce a few-shot prompt to boost the performance of the model. We use the same prompt as in the previous experiment but introduce static normalized examples and a bulleted style prompt for improved clarity:

- Few-shot prompt: *Temporal expressions are those constructions referring to points or intervals on the timeline. They can express:*
  - *day times, e.g.*

    "noon" value:"XXXX-XX-XXTMI"

    "3:00 pm" value: "XXXX-XX-XXT15:00"

    "this morning" value:"XXXX-XX-XXTMO"

2 https://github.com/HeidelTime/heideltime.
3 https://bitbucket.org/kentonl/uwtime-standalone/src/master/.
4 https://github.com/qiangning/CogCompTime.
5 https://github.com/nju-websoft/ARTime.
6 https://github.com/NGEscribano/XTN-timexes.
7 https://stanfordnlp.github.io/CoreNLP/sutime.html.
8 https://github.com/nju-websoft/ARTime.
9 https://github.com/boschresearch/temporal-tagging-eacl.

  - *dates, e.g.*

    "April 28th, 2008" value:"2008-04-28"

    "yesterday" value:"one day before the DCT". If DCT:"1990-04-20" value:"1990-04-19"

    "next week" value:"one week after the DCT". If DCT:"1990-04-20" value:"1990-W17"

    "three months later" value:"three months after the DCT". If DCT:"1990-04-20" value:"1990-07"

    "last year" value:"one year before the DCT". If DCT:"1990-04-20" value:"1989"

  - *durations, e.g.*

    "two months" value:"P2M"

    "five hours" value:"PT5H"

    "the coming 9 years" value:"P9Y"

  - *sets, e.g.*

    "once a month" value:"P1M"

    "every Tuesday" value:"XXXX-WXX-4"

  Your work is to obtain the value of the following temporal expressions according to the TimeML schema. Use the document creation time, the type and the text of the temporal expression to obtain the value. Finally, display your response in a JSON format.

- *Consecutive prompts: Obtain the value of the following temporal expression with DCT "1990-04-20", type "DATE" and text "Yesterday".*

These prompts have been designed with a fundamental structure that includes defining the task, identifying its components, and articulating the objective using straightforward language. The purpose of these experiments is to obtain an initial insight into the performance of the most well-known LLM in a basic scenario. Rather than focusing on optimization, we rely on existing studies, such as Gautam, Lange, and Strötgen (2024), that have already addressed this aspect in the English Timebank corpus.

As for the pre-processing of the training corpora, the text has been split into sentences and all sentences without at least one TE have been removed, to reduce the size. Each possible value of the VIR slots must be represented by one model token. Therefore, we have added 368 new tokens to the model tokenizer, 214 of which already were included. We evaluate all systems on both E3C and Timebank predefined test sets with the TempEval evaluation script (UzZaman et al. 2013). For all experiments, we report the weighted-mean accuracy of TE normalization. All the mean values are weighted based on the number of expressions of each corpus shown in Table 2.

For all experiments, we have used a *XLM-RoBERTa* model, with $batch\_size = 10$, $lr = 8e-5$, $wd = 0.01$, $seed = 42$ and *3 epochs*, selecting the training checkpoint with lower dev loss. For the masking function experiments we have used $p = 0.5$, $minimum_{lin} = 0.7$, $c = 1$, $maximum_{cos} = 0.7$, $b = 0.2$, $minimum_{log} = 0.2$. We will repeat each experiment three times and will take the mean value to mitigate randomness. Replicating the entire two-phase training takes around 6 hours of computing time. The presented experiments have been run on a configuration of two RTX 3090 GPUs.

For training, we mark each TE with a label of the form *"<TYPE EXPRESSION VIR>"*, for example, *"She was discharged home <duration **2 weeks postoperatively** <pad><pad>*2*<pad><pad><pad><pad><pad>>"*. This input is passed during the fine-tuning process to the model after substituting the VIR slots, type or expression with MASK tokens based on our proposed masking policy. For example: *"<date **five days later**<mask><mask><mask><mask><mask><mask><mask><mask><mask>>"*, *"<<*mask*>**five days later**<pad><pad><pad><sum>* <5>*<d><pad><pad><pad>>"* or *"<date **<mask>** <pad><pad><pad><sum>*<5>*<d><pad><pad><pad>>"*. Once the VIR is unmasked, we perform an automatic de-regularization process followed by an anchoring process, based on the reference date to obtain the final value, as summarized in Figure 5.

4. Results

In this section, we will discuss the results of the experiments proposed in Section 3.6. It should be noted that all results shown are weighted accuracy as explained in Section 3.6.

4.1 Representation Comparison

As can be seen in Table 3, our representation offers a 2.64 point advantage over Lange's. By optimizing the representation space we reduce in more than 18% the number of masked tokens in the training set and in 22% the amount of tokens that have to be unmasked in the test sets. Considering that the representations are composed mostly of *pad* tokens, reducing the length greatly reduces the noise of these *pad* tokens, improving the performance. Furthermore, utilizing a single representation for all types requires condensing substantial information into a limited number of slots. This can lead the model to mix information and worsen its performance. In contrast, utilizing a dedicated representation for each type provides more specific slots tailored to the type

Table 3
Performance of a large XLM-RoBERTa model trained over automatic annotated MIMIC-III with Lange et al. (2023) representation and our proposed VIR. The best results are in **bold**.

Representation	E3C	Timebank	Mean
Our VIR	**49.96**	**69.62**	**58.65**
(Lange et al. 2023)	47.71	66.49	56.01

of expression. This design allows the model to establish stronger associations between the expression type and its VIR, effectively segmenting the information and thereby enhancing overall performance.

As already mentioned in Section 3.6, representations cannot be decoupled from the system of rules on which they are designed. In our case it is Annotador and in the case of Lange's representation, it is HeidelTime. So to perform this experiment we had to annotate MIMIC-III on the one hand with Annotador and our representation and on the other hand with HeidelTime and Lange's representation. This may lead to some interference in the results of the experiment, as the results are not fully comparable, but as we cannot decouple HeidelTime and Annotador from their respective representations there is no way to eliminate this interference completely. Still, HeidelTime and Annotador are very similar systems in terms of English TE normalization performance. This can be seen in Table 8 and in Sánchez-de Castro, Araujo, and Martinez-Romo (2023) and Navas-Loro et al. (2020), where the difference in performance is only a few tenths of a percent. The performance difference between the two systems does not fully justify such a large discrepancy between our representation and Lange's. While the performance gap is undoubtedly influenced by the differences between HeidelTime and Annotador, the size of this gap suggests that with all other factors being equal, our representation would outperform Lange's in this scenario.

4.2 Hand-crafted TE Corpus Impact

Table 4 shows the comparison in performance on English E3C and Timebank for training a *large XLM-RoBERTa* model with and without the hand-crafted TE corpus. It can be observed that the hand-crafted TEs boost the performance by over 5.5 points. The increase is due to the lack of diversity of TEs in MIMIC-III, especially full dates, days of the week, and operations like *"next month"*. It is important to remark that there is no data leakage from the hand-crafted TE corpus.

4.3 Masking Function Results

We have proposed three different masking functions for training the model. The results can be seen in Table 5. The best performance is offered by the combination of *linear+logistic* masking function. This function offers the best combination of early high masking rate and lately low masking rate. Behind in performance is the *linear+cosine* masking function. The main reason why this function is worse than the logistic function might be the accelerating decrease exposed in Section 3.3. Lastly, the worst performance is the *constant* masking function. Regarding the base and large versions of *XLM-RoBERTa*, it is clear that the large version offers better performance than the base, with around 4 points more on E3C and Timebank.

Table 4
Performance of a large XLM-RoBERTa model on English E3C and Timebank trained with (W) and without (W/O) the hand-crafted TE corpus. The best results are in **bold**.

Hand-crafted TEs	E3C	Timebank	Mean
W	**54.25**	**76.89**	**64.26**
W/O	49.96	69.62	58.65

Table 5
Results comparing the training base and large XLM-RoBERTa model with three different masking functions in MIMIC-III over English E3C and Timebank corpora. Showing the mean values for each model and each function. The best results are in **bold**.

Model Size	Constant	Linear+Cosine	Linear+Logistic	Mean
		E3C		
Large	54.25	54.62	**55**	**54.62**
Base	49.73	51.61	**52.54**	51.29
Mean	51.99	53.12	**53.77**	
		Timebank		
Large	76.89	76.89	**78.76**	**77.51**
Base	70.92	72.21	**73.77**	72.30
Mean	73.90	74.55	**76.26**	

From now on we will refer to the large model trained with *linear+logistic* function as our *base model*.

4.4 Freezing Layers Result

In this experiment, our base model is subjected to a fine-tuning process on E3C and Timebank multilingual training sets using the freezing layers technique planned in Section 3.4. We evaluate based on the mean value of all languages across both corpora. Table 6 shows a mean 1.22 advantage on training with 9 against 6 frozen layers on both corpora and a 2.45 advantage on freezing 9 layers against not freezing. Therefore, it can be concluded that freezing layers improves the adaptability of the model to new domains and languages. In addition, for this particular case freezing 9 layers is consistently better than freezing 6, but this may not be transferable to other cases.

4.5 Final Results

Table 7 shows the final results. Our base model, which excels on the E3C dataset, achieves a comparable global mean score to XLM_Bosch, with only a 0.28 point difference, despite XLM_Bosch performing best on the Timebank dataset. This is an understandable behavior because XLM_Bosch is trained with Timebank data and our base

Table 6
Results on multilingual Timebank and E3C comparing fine-tuning the best model found in Section 4.3 on multilingual Timebank and E3C train sets with 0, 6, and 9 frozen layers. The best results are in **bold**.

Layers	E3C	Timebank	Mean
0	68.31	73.99	71.48
6	68.78	75.82	72.71
9	**69.71**	**77.27**	**73.93**

Table 7
Comparison between our base model without the hand-crafted corpus, our base model, our base model without the hand-crafted corpus plus second fine-tune, our base model plus second fine-tune, GPT-4o mini, GPT-4o mini with few-shot and current state-of-the-art system XLM_Bosch results on multilingual E3C and Timebank corpora—English, French, Italian, and Spanish—by order. The weighted mean for each corpus and the global weighted mean between both corpora are also presented. The best results per language, corpus, and global are in **bold**.

	English	French	Italian	Spanish	Mean	
			Timebank			**Global Mean**
Ours base W/O	74.82	54.52	77.50	73.76	72.12	65.98
Ours base	78.76	57.69	81.58	77.64	75.96	69.52
Our base W/O + ft	79.16	58.33	82.93	78.93	76.92	70.91
Ours base + ft	**83.33**	**61.73**	**87.30**	**83.08**	**81.01**	**74.71**
XLM_Bosch	79.49	**61.73**	85.74	81.52	79.05	69.80
GPT-4o mini	58.64	43.44	61.43	58.46	57.01	58.11
GPT-4o mini Few-Shot	61.19	45.26	63.76	61.15	59.45	60.57
			E3C			
Ours base W/O	52.25	62.29	59.52	69.35	61.12	
Ours base	55.00	65.92	62.65	73.00	64.42	
Our base W/O + ft	56.54	67.42	64.41	75.05	66.15	
Ours base + ft	**59.52**	**71.34**	**67.80**	**79.00**	**69.71**	
XLM_Bosch	54.17	61.15	58.76	74.00	62.47	
GPT-4o mini	50.36	60.36	57.36	66.84	58.98	
GPT-4o mini Few-Shot	52.52	62.71	59.72	69.78	61.46	

model is only trained with biomedical domain data. Considering that our baseline model has not seen data from either Timebank or E3C or any language other than English, it highlights the value of the training techniques used and the zero-shot capabilities of the model. This scenario is improved on the fine-tuned version, which outperforms XLM_Bosch with a 7.94 point advantage on E3C, 1.96 points on Timebank, and 4.91 points in general, in part thanks to the layer freezing technique. Also, our model outperforms in all languages, except in French Timebank where both models score the same.

To highlight the value of the hand-crafted corpus, we trained a model following the same procedure used for the final model but excluding this corpus from the training process. The results are summarized in Table 7, where the first row presents the performance of the base model trained without the hand-crafted corpus. As the table indicates, incorporating the corpus leads to a 3.54 point improvement in the global mean performance. Conversely, fine-tuning the model without the hand-crafted corpus results in a 3.8 point decrease in performance. In addition, including both the hand-crafted dataset and the fine-tuning process with E3C and Timebank shows the best results, indicating the complementarity of both parts and empirically ruling out unnecessary overlapping. These findings underscore the significant positive influence of the hand-crafted corpus on the model's performance. Although its impact is not as pronounced as that of fine-tuning, the hand-crafted corpus could play an equally critical role in zero-shot scenarios across other domains and languages, as it was designed to serve as a foundational dataset. This also highlights the current lack of manually annotated corpora to achieve sufficiently varied training sets, emphasizing the need to build larger, well-designed foundational corpora that address global requirements rather than domain- or language-specific needs, as our hand-crafted corpus. Also, it should be recalled that none of the expressions included in the hand-crafted corpus are in the test sets of either Timebank or E3C.

Regarding GPT-4o mini, it is the worst performing model of the comparison by language and in general means. The performance gap is greater in Timebank than in E3C, where the GPT-4o mini is just 3.49 points behind *XLM_Bosch* without any intended training. But it is still far from the rest of the options in the global mean, with 16.6 points of difference from our fine-tuned model and even 11.41 points from our base model, which in this scenario is working on zero-shot in terms of languages and domains.

Few-shot prompting enhances GPT-4o Mini's global mean performance by 2.46 points, a notable improvement that brings its performance close to XLM_Bosch in E3C, with only a 1.01 point difference in favor of XLM_Bosch. Nevertheless, it remains 14.14 points below our proposed approach in terms of global mean performance. In the English Timebank dataset results shown in Table 8, GPT-4o Mini falls 9.11 points behind the weakest baseline system, SuTime. Additionally, prior work (Gautam, Lange, and Strötgen 2024) that uses complex prompting strategies for TE normalization reports a 10.73 point lower performance on the English Timebank compared with our approach. Also, those authors conclude that XLM_Bosch's performance is superior to their proposed GPT approach in English Timebank. Considering these results, we do not expect further exploration of more complex prompting techniques to provide substantial additional insights for improving the baselines.

Despite these limitations, we will analyze the errors of GPT models to gain a deeper understanding of their performance. Given the model's considerable size and its exceptional capabilities across diverse tasks, one would expect a strong performance in this domain. In our baseline experiment, we observed that the model struggles with correctly handling *DURATIONS* and *SETS*, often misinterpreting them and mistakenly mixing them with date formats. This issue is partially mitigated by few-shot prompting, which also improves the handling of date-related TEs involving weeks—another area where the model initially exhibited formatting errors. However, in most other cases, few-shot prompting did not produce any significant improvements. Given the broad

Table 8
Comparison between our best model and other well-known TE normalization solutions in English Timebank. The best results are in **bold** and second-best results are underlined.

TE normalization systems	English Timebank
HeidelTime	81.6
Annotador	81.54
SuTime	70.3
UWTime	82.6
CogCompN	**83.4**
ARTime	75.4
DNPTime-Large	80.4
TimeNorm	79
XLM_Bosch	79.49
Ours	<u>83.33</u>

and varied nature of the errors, relying solely on prompting techniques may not yield significant improvements, as evidenced by Gautam, Lange, and Strötgen (2024). Other potential avenues for enhancement accuracy and robustness include aligning the model through reinforcement learning or developing intelligent agents capable of integrating external tools, such as calendars, code-for-date operations, or format verification systems.

It should be noted that it is possible that GPT-4o mini may have been trained with some or all of Timebank and E3C, as they are public corpora and there is no guarantee that this has not occurred.

Finally, Table 8 presents a comparison of our proposed model against other TE normalization solutions within English Timebank. Most alternatives focus exclusively on English and are specifically tailored for Timebank, whereas our method supports multiple languages while achieving state-of-the-art performance. As shown, our model nearly matches the leading solution, *CogCompN*, with only a 0.07 point difference, while outperforming the others. Additionally, our approach is adaptable across various languages and domains and can work in zero-shot scenarios for unseeing languages, unlike rule-based systems, making it a much more effective option for TE normalization.

5. Conclusions

In this work, we achieve the best results in multilingual TE normalization among all public normalization systems for two different domain corpora, E3C and Timebank. To accomplish this, we have built a new value intermediate representation (VIR) for adapting the normalization task to the fill-mask architecture. We have applied this new representation to automatically annotate a biomedical corpus—MIMIC-III—with a state-of-the-art normalization system called Annotador. Also, we have hand-crafted a TE corpus to complement the automatic annotations. This corpus has been of vital importance in improving the results of the final model. On the training side, we have created a custom masking policy that is fully adapted to normalization tasks. This policy greatly improves the performance of the model against training with a general policy. We believe that these results show the great potential of choosing a well-fitted masking strategy and show an appropriate way to do so, which can be applied to many other problems beyond normalization. In addition, we show that the freezing layers technique can be of great benefit for domain and language adaptation. By freezing 9 model layers we achieve the best performance on E3C and Timebank, outperforming the results of the best current multilingual normalization and monolingual systems. Also, we show how our intermediate representation is more optimal than the one proposed in Lange et al. (2023). Therefore, we show how our methodology contributes to improving cross-language and domain adaptability by enabling and optimizing the use of multilingual fill-masked PTLMs, demonstrating how they are a great alternative to conventional systems to achieve multilingual capabilities.

6. Limitations

This work presents a series of limitations regarding the methodology and the final model capabilities.

First, we would like to acknowledge that our work was inspired by Lange et al. (2023) in the use of fill-mask models alongside an intermediate representation. However, while we adopted the concept of an intermediate representation, our implementation significantly diverges from theirs, both in design and demonstrated performance.

We have shown that our representation is more space-efficient and achieves superior results. Beyond this shared inspiration, the rest of our work is entirely novel. We introduce a completely new corpus, which has proven highly effective and can serve as a valuable resource for other TE normalization approaches. Additionally, we propose an entirely new masking policy that is broadly applicable to any fill-mask model, further distinguishing our methodology from previous efforts.

As the proposed methodology and techniques are universal, we plan to apply them to other languages and domains in the future. Also, we plan to adapt auto-retrogressive Large Language Models (LLM) like *Mistral*, *Gemma*, or *LLaMA-3* to the normalization task, since it has a promising avenue for study. Also, we plan to test the application of agentic approaches where LLMs can use external tools like calendars, code for date operations, or format verifiers. In addition, we have only tested with *XLM-RoBERTa*. Other fill-mask models might achieve a better balance between performance and size. We also plan to use Low Ranking Adaptation given the good results it has been producing in recent LLM. Also, we have not tested other possible VIR formats with different lengths or configurations, which might result in a better understanding of the representation by the model.

The use of TE detection models has not been studied in this work, as it has been considered that it could interfere with the study of the performance of the proposed normalization methodology. A normalization system can only process TEs that provide sufficient information for normalization. If the detection system partially fails, it is likely that the normalization process will also fail. Moreover, if the detection system fails entirely and does not detect the expression, there is nothing for the normalization system to process. Thus, evaluating a normalization system's performance together with a detection system would be inefficient for studying the normalization process.

In conducting our GPT-4o Mini experiments, we recognize that our approach does not aim to maximize the model's potential performance. Although more advanced prompting techniques could have been incorporated, they fall beyond the scope of this study. Existing research (Gautam, Lange, and Strötgen 2024) has already investigated such methods, reporting lower performance than our approach on English Timebank, 10.73 points precisely. Consequently, we see little value in further exploring prompting strategies for GPT-4o Mini in this work. Instead, our inclusion of GPT-4o serves to illustrate the performance of a widely recognized model in a basic scenario, rather than to highlight its full capabilities.

Regarding the final model, we are conscious that it has certain limitations when it comes to TE normalization. Most of these are one-off errors. However, we have detected a recurrent error when trying to normalize long-context TEs. For example, when some part of the expression is implicit like in *"On 4 and 5 April, the patient went to the doctor for consultation...On 4 underwent various tests"*. The model does not relate 4 with a day, it just takes a random guess. This is an acceptable error since most of the TEs are self-contained and relating both parts of the text is a very hard task in deep learning. We have not tried to solve these problems because they rarely appear in real scenarios. Also, building artificial expressions with this characteristic would imply filling in the space between both TEs, hindering the task. Finally, we regularize and de-regularize every value into its corresponding VIR and vice versa with a Python script, using rules, regular expressions, and pattern recognition. This script has been done meticulously, taking care and debugging errors. Nevertheless, the potential range of values for the TEs is very extensive and there may be some misbehavior in the regularization and de-regularization processes.

As for the final results in Table 8, the scores of HeidelTime and Annotador have been extracted from Navas-Loro et al. (2020); UWTime, CogCompn, and DNPTime-Large have been extracted from Ding et al. (2023); and the scores of TimeNorm from Escribano, Rigau, and Agerri (2023), as all of them use the same test sets as ours.

This work has been developed taking into account the possible impacts, both positive and negative, of our proposal and methodology. This is why we have not repeated the experiments more than three times, to reduce the environmental impact. On the other hand, some of the used corpora include personal information that is fully anonymized by the corresponding authors. Finally, we have not found any possible biases that could be detrimental or have a negative impact on any community.

Acknowledgments
This work has been funded by the following projects: OBSER-MENH (MCIN/AEI/10.13039/501100011033 and NextGenerationEU"/PRTR with identification TED2021-130398B-C21), SICAMESP (2023-VICE-0029), and by the project EDHER-MED (PID2022-136522OB-C21)".

References

Barros, Cristina, Elena Lloret, Estela Saquete, and Borja Navarro-Colorado. 2019. NATSUM: Narrative abstractive summarization through cross-document timeline generation. *Information Processing & Management*, 56(5):1775–1793. https://doi.org/10.1016/j.ipm.2019.02.010

Bethard, Steven, Guergana Savova, Martha Palmer, and James Pustejovsky. 2017. SemEval-2017 Task 12: Clinical TempEval. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 565–572. https://doi.org/10.18653/v1/S17-2093

Britz, Denny, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451. https://doi.org/10.18653/v1/D17-1151

Chang, Angel X. and Christopher Manning. 2012. SUTime: A library for recognizing and normalizing time expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3735–3740.

Cole, Jeremy R., Aditi Chaudhary, Bhuwan Dhingra, and Partha Talukdar. 2023. Salient span masking for temporal understanding. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3052–3060. https://doi.org/10.18653/v1/2023.eacl-main.222

Conneau, Alexis, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451. https://doi.org/10.18653/v1/2020.acl-main.747

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ding, Wentao, Jianhao Chen, Longfei E., Jinmao Li, and Yuzhong Qu. 2023. Time expression as update operations: Normalizing time expressions via a distantly supervised neural semantic parser. *Knowledge-Based Systems*, 278:110870. https://doi.org/10.1016/j.knosys.2023.110870

Ding, Wentao, Jianhao Chen, Jinmao Li, and Yuzhong Qu. 2021. Automatic rule generation for time expression normalization. In *Findings of the Association for Computational Linguistics, Findings of ACL: EMNLP 2021*, pages 3135–3144. https://doi.org/10.18653/v1/2021.findings-emnlp.269

Edwards, Carl, Tuan Lai, Kevin Ros, Garrett Honke, Kyunghyun Cho, and Heng Ji. 2022. Translation between molecules and natural language. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 375–413. https://doi.org/10.18653/v1/2022.emnlp-main.26

Escribano, Nayla, German Rigau, and Rodrigo Agerri. 2023. A modular approach for multilingual timex detection and normalization using deep learning and grammar-based methods. *Knowledge-Based Systems*, 273:110612. https://doi.org/10.1016/j.knosys.2023.110612

Etcheverry, Mathias and Dina Wonsever. 2017. Time expressions recognition with word vectors and neural networks. In *24th International Symposium on Temporal Representation and Reasoning (TIME 2017)*, volume 90 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:20.

Gautam, Akash, Lukas Lange, and Jannik Strötgen. 2024. Discourse-aware in-context learning for temporal expression normalization. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 306–315. https://doi.org/10.18653/v1/2024.naacl-short.27

Guu, Kelvin, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938.

Johnson, Alistair, Tom Pollard, and Roger Mark. 2016. MIMIC III clinical database (Version 1.4). *PhysioNet*, 10(C2XW26):2.

Joshi, Mandar, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77. https://doi.org/10.1162/tacl_a_00300

Kar, Sudipta, Giuseppe Castellucci, Simone Filice, Shervin Malmasi, and Oleg Rokhlenko. 2022. Preventing catastrophic forgetting in continual learning of new natural language tasks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, pages 3137–3145. https://doi.org/10.1145/3534678.3539169

Ke, Zixuan, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021. Achieving forgetting prevention and knowledge transfer in continual learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 22443–22456. Curran Associates, Inc.

Kim, Zae Myung and Young-Seob Jeong. 2016. TIMEX3 and event extraction using recurrent neural networks. In *2016 International Conference on Big Data and Smart Computing (BigComp)*, pages 450–453. https://doi.org/10.1109/BIGCOMP.2016.7425968

Lange, Lukas, Anastasiia Iurshina, Heike Adel, and Jannik 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. https://doi.org/10.18653/v1/2020.repl4nlp-1.14

Lange, Lukas, Jannik Strötgen, Heike Adel, and Dietrich Klakow. 2023. Multilingual normalization of temporal expressions with masked language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1174–1186. https://doi.org/10.18653/v1/2023.eacl-main.84

Laparra, Egoitz, Dongfang Xu, and Steven Bethard. 2018. From characters to time intervals: New paradigms for evaluation and neural parsing of time normalizations. *Transactions of the Association for Computational Linguistics*, 6:343–356. https://doi.org/10.1162/tacl_a_00025, PubMed: 32432133

Lee, Kenton, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. 2014. Context-dependent semantic parsing for time expressions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447. https://doi.org/10.3115/v1/P14-1135

Levine, Yoav, Barak Lenz, Opher Lieber, Omri Abend, Kevin Leyton-Brown, Moshe Tennenholtz, and Yoav Shoham. 2021. {PMI}-masking: Principled masking of correlated spans. In *International Conference on Learning Representations*.

Li, Hui, Jannik Strötgen, Julian Zell, and Michael Gertz. 2014. Chinese temporal tagging with HeidelTime. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 133–137. https://doi.org/10.3115/v1/E14-4026

Liu, Chao, Cui Zhu, and Wenjun Zhu. 2020. Chinese named entity recognition based on BERT with whole word masking. In *Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence*, ICCAI '20, pages 311–316. https://doi.org/10.1145/3404555.3404563

Mager, Manuel, Arturo Oncevay, Abteen Ebrahimi, John Ortega, Annette Rios, Angela Fan, Ximena Gutierrez-Vasques, Luis Chiruzzo, Gustavo Giménez-Lugo, Ricardo Ramos, Ivan Vladimir Meza Ruiz, Rolando Coto-Solano, Alexis Palmer, Elisabeth Mager-Hois, Vishrav Chaudhary, Graham Neubig, Ngoc Thang Vu, and Katharina Kann. 2021. Findings of the AmericasNLP 2021 shared task on open machine translation for indigenous languages of the Americas. In *Proceedings of the First Workshop on Natural Language Processing for Indigenous Languages of the Americas*, pages 202–217. https://doi.org/10.18653/v1/2021.americasnlp-1.23

Magnini, Bernardo, Begoña Altuna, Alberto Lavelli, Manuela Speranza, and Roberto Zanoli. 2020. The E3C project: Collection and annotation of a multilingual corpus of clinical cases. In *CLiC-it*. https://doi.org/10.4000/books.aaccademia.8663

Mansouri, Behrooz, Mohammad Sadegh Zahedi, Ricardo Campos, Mojgan Farhoodi, and Maseud Rahgozar. 2018. ParsTime: Rule-based extraction and normalization of Persian temporal expressions. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10772 LNCS:715–721. https://doi.org/10.1007/978-3-319-76941-7_67

Moon, Seung Jun, Sangwoo Mo, Kimin Lee, Jaeho Lee, and Jinwoo Shin. 2021. MASKER: Masked keyword regularization for reliable text classification. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(15):13578–13586. https://doi.org/10.1609/aaai.v35i15.17601

Moriceau, Véronique and Xavier Tannier. 2013. French resources for extraction and normalization of temporal expressions with HeidelTime. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC13)*.

Najafabadipour, Marjan, Massimiliano Zanin, Alejandro Rodríguez-González, Maria Torrente, Beatriz Nuñez García, Juan Luis Cruz Bermudez, Mariano Provencio, and Ernestina Menasalvas. 2020. Reconstructing the patient's natural history from electronic health records. *Artificial Intelligence in Medicine*, 105:101860. https://doi.org/10.1016/j.artmed.2020.101860, PubMed: 32505419

Navas-Loro, María, Víctor Rodríguez-Doncel, David Pinto, Vivek Singh, and Fernando Perez. 2020. Annotador: A temporal tagger for Spanish. *Journal of Intelligent & Fuzzy Systems*, 39(2):1979–1991. https://doi.org/10.3233/JIFS-179865

Ning, Qiang, Ben Zhou, Zhili Feng, Haoruo Peng, and Dan Roth. 2018. CogCompTime: A tool for understanding time in natural language. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 72–77. https://doi.org/10.18653/v1/D18-2013

Pustejovsky, James, Kiyong Lee, Harry Bunt, and Laurent Romary. 2010. ISO-TimeML: An international standard for semantic annotation. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(1).

Sánchez-de Castro, Alejandro, Lourdes Araujo, and Juan Martinez-Romo. 2023. Robertime: A novel model for the detection of temporal expressions in spanish. *Procesamiento del Lenguaje Natural*, 70:39–51.

Sánchez-de Castro, Alejandro, Lourdes Araujo, and Juan Martinez-Romo. 2024. Generative LLMs for multilingual temporal expression normalization. In *European Conference on Artificial Intelligence*, pages 3789–3796. https://doi.org/10.3233/FAIA240940

Saurí, Roser, Estela Saquete, and James Pustejovsky. 2010. Annotating time expressions in Spanish. TimeML annotation guidelines (Version TempEval-2010), Barcelona Media Technical Report 2010-02.

Skukan, Luka, Goran Glavaš, and Jan Šnajder. 2014. HeidelTime. Hr: Extracting and normalizing temporal expressions in Croatian. In *Proceedings of the 9th Slovenian Language Technologies Conferences (IS-LT 2014)*, pages 99–103.

Strötgen, Jannik and Michael Gertz. 2010. HeidelTime: High quality rule-based extraction and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 321–324.

Suleymanova, E. A. and I. V. Trofimov. 2021. "A year ago": On normalization of one type of temporal expression. *Scientific and Technical Information Processing*, 48:376–387. https://doi.org/10.3103/S0147688221050105

UzZaman, Naushad, 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.

Verhagen, Marc, 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. https://doi.org/10.3115/1621474.1621488

Wettig, Alexander, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2023. Should you mask 15% in masked language modeling? In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2985–3000. https://doi.org/10.18653/v1/2023.eacl-main.217

Wu, Xing, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Mask and infill: Applying masked language model for sentiment transfer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5271–5277. https://doi.org/10.24963/ijcai.2019/732

Xue, Linting, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498. https://doi.org/10.18653/v1/2021.naacl-main.41

Yang, Dongjie, Zhuosheng Zhang, and Hai Zhao. 2023. Learning better masking for better language model pre-training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7255–7267. https://doi.org/10.18653/v1/2023.acl-long.400

Zhang, Linhan, Qian Chen, Wen Wang, Chong Deng, Xin Cao, Kongzhang Hao, Yuxin Jiang, and Wei Wang. 2023. Weighted sampling for masked language modeling. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. https://doi.org/10.1109/ICASSP49357.2023.10096946

Zhou, Ran, Xin Li, Ruidan He, Lidong Bing, Erik Cambria, Luo Si, and Chunyan Miao. 2022. MELM: Data augmentation with masked entity language modeling for low-resource NER. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2251–2262. https://doi.org/10.18653/v1/2022.acl-long.160