Time Expressions Recognition with Word Vectors and Neural Networks

Mathias Etcheverry[1] **and Dina Wonsever**[2]

1 **Universidad de la República, Facultad de Ingeniería, Instituto de Computación, Montevideo, Uruguay**
mathiase@fing.edu.uy
2 **Universidad de la República, Facultad de Ingeniería, Instituto de Computación, Montevideo, Uruguay**
wonsever@fing.edu.uy

Abstract

This work re-examines the widely addressed problem of the recognition and interpretation of time expressions, and suggests an approach based on distributed representations and artificial neural networks. Artificial neural networks allow us to build highly generic models, but the large variety of hyperparameters makes it difficult to determine the best configuration. In this work we study the behavior of different models by varying the number of layers, sizes and normalization techniques. We also analyze the behavior of distributed representations in the temporal domain, where we find interesting properties regarding order and granularity. The experiments were conducted mainly for Spanish, although this does not affect the approach, given its generic nature. This work aims to be a starting point towards processing temporality in texts via word vectors and neural networks, without the need of any kind of feature engineering.

1998 ACM Subject Classification I.2.7 Natural Language Processing

Keywords and phrases Natural Language Processing, Time Expressions, Word Embeddings, Neural Networks

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.12

1 Introduction

Detecting and interpreting the linguistic expressions that we use to refer to the physical time we live in (e.g. “21 September”, “2001” or “yesterday”) poses an interesting problem of natural language processing. Time expressions or timexes are a sub-language made up of specific lexicon and with an interpretation linked to the calendar system that we use, numbers, conditions and relations with entities that are external to the expression. Additionally, as many other NLP tasks, the analysis of timexes presents ambiguous situations where contextual information is needed.

Several approaches have been proposed to detect and interpret time expressions. Rule-based systems like HeidelTime [35] and SUTime [9] yield very good results. Also systems based on machine learning techniques like Support Vector Machines [3] or Conditional Random Fields [1] produce good results without the cost of defining the rules, though it is necessary to define specific characteristics and to use external resources.

Furthermore, vector representations of words, obtained from large text collections, have produced interesting results regarding association and analogies between words [25]. Word vectors can be seen as a set of artificial micro features that may be part of the input of a machine-learning algorithm.

Artificial neural networks organized in layers can be interpreted as successive sequent transformations of a representation to obtain an expected result. Using word vectors as input enables us to resolve problems without having to specify rules or machine-learning features with information about the problem. The information about the problem used to define the models considered in this work is limited to the output layer and the way the model is used.

In this work we study the behavior of neural models that use as input distributed representations of words to detect and classify the time expressions occurring in a text. Word representations make associations between similar words or semantically related words, which gives the model the capacity to consider cases with lexical entries that do not exist in the training data. We consider feedforward and LSTM models, and analyze the effect of considering different dimensions for word representations, for hidden layers and normalization techniques such as noise in representations, dropout, L1 and L2. We conducted experiments using a small corpus for Spanish obtaining interesting results. We present results relative to word vectors that suggest the possibility of interpreting time expressions without defining rules or adding knowledge about the temporal domain. Finally, we include experiments in English with encouraging results, and in the different, yet related, task of events detection for Spanish and English.

2 Related Work

2.1 In Time Expressions

Extensive work has been done in the detection and interpretation of time expressions. Furthermore, neural networks and vector representations of words have made great progress recently. However, to our knowledge, no studies using neural models to resolve problems with time expressions have been presented.

Regarding existing rule-based systems, [24] presents a system that resolves recognition and interpretation through manually developed and machine-learned rules. [28] tackles the problem by processing the input text, where information about expressions is cumulatively added in each stage through heuristics and rules. [16] resolves recognition and interpretation with a formal set of rules on the morphosyntactic information of the input. [13] builds a system based on the *TRIOS* system [38], adding pre and post processing stages to improve their results.

The *HeidelTime* system [34] uses regular expressions and resources from a temporal lexicon to recognize and interpret the expressions, reaching the best results at *TempEval-2*. These results were later improved by the *SUTime* system [9]. At *TempEval-3*, the system that obtained the best results was a new version of *HeidelTime* [35]. [8] outperforms this result with a system based on a manually developed context-free grammar.

Furthermore, traditional machine learning methods are essentially classifiers based on conveniently defined features. It is from annotated examples that mechanisms are set to determine the desired information in arbitrary entries. This type of method is suitable to identify and classify time expressions, but its application to interpretation is not direct.

These systems are usually based on features such as: words part-of-speech tags in a context window, belonging to word classes that are manually specified, and restrictions on a dependency or constituents analysis, among many others. The variety of possible features is unlimited, and the results depend mainly on features engineering. It is also important to note that some features may require considerable computing time, thus making their use questionable.

We could mention many existing machine learning based systems. [1] detects time expressions with *Conditional Random Fields (CRF)*, as does [2]. Later on, [3] uses *Support*

Vector Machines (SVM) as classifiers and simplifies the rules for interpretation through a classifiers cascade.

Semi-supervised approaches may include *bootstrapping* techniques to improve recognition [27]. [21] expands positive cases using *WordNet*. The *ManTIME* system [14] runs *CRF* for detection, considering attributes derived from *WordNet*, but does not reach significant improvements.

The*ClearTK-TimeML* system [7] trains multiple supervised classifiers to identify and classify time expressions, events and relations. Different methods are made to compete in the system (*CRF*, *SVM* and logistic regression), with a specific tune of hyperparameters.

Hybrids that include machine learning techniques and a formal set of rules have produced good recognition results. As for interpretation, rules are used relatively naturally given their compositional properties. Some approaches combine the advantages of formal sets of rules and annotated examples to interpret expressions. [4] inferred a probabilistic context-free grammar on the expressions. This system can be easily applied to different languages [5].

[22] uses combinatory categorial grammar system to detect and interpret time expressions. The work considers 287 manually designed entries, as well as automatically generated entries (such as numbers and formats of dates), obtaining 83.1% F-score for detection, 85.4% for classification and 82.4% for interpretation in the evaluation data of *TempEval-3*; this is the current state-of-the-art.

2.2 In Neural Networks

Artificial neural networks currently play a major role in the artificial intelligence community and natural language processing isn't an exception[10, 11]. This has increased with the progress made in the construction of vector representations of words from the contexts where they occur [25, 26].

As for the use of word representations with neural network models to solve NLP tasks, many studies with significants results can be mentioned. Among them, paraphrase detection [32], parsing [31] and sentiment analysis [33]. [18] uses recurrent models and word embeddings to resolve the issue of opinion mining. This work shares ideas with [17] that uses a feedforward network and word2vec embeddings for time expression recognition in English for clinical domain.

3 Time Related Words and Distributed Representations

To study the quality of information provided by words vector representations in the temporal domain we reduced the dimension of the representations to be able to represent their relative positions graphically. In turn, the ordering of the cosine distances respect to a specific word were considered. We used word representations inferred from Wikipedia in Spanish through GloVe [26], presented by [12].

3.1 Clustering

It is relevant to study the structure of the space of vector representations for the words that refer to temporal information. Figure 1 shows the representations of a selection of time related words after reduction to 2 dimensions with *t-sne* [40] for its graphical representation. The presented 2-d representation shows how semantically related words tend to form clusters. The following clusters are formed: days of the week, months, years, adverbs and low numbers

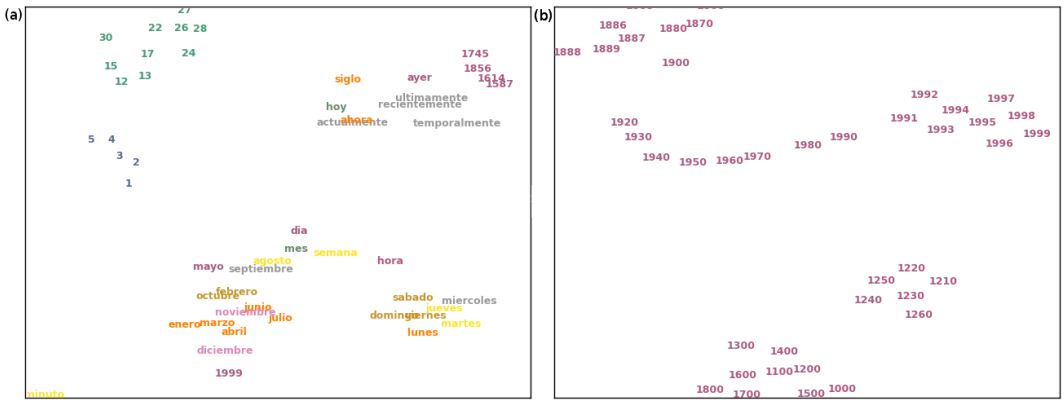

Figure 1 (a) Representation in 2 dimensions using *t-sne* of 200 dimension representations of a time related words selection. (b) Representation in 2 dimensions using *t-sne* of 200 dimension representations of numbers.

Table 1 Table with the closest terms (to the heading) ordered by distance to the time expressions representations.[1]

Amanecer	Neolítico	Comienzo	Antes	Repentinamente	Apresuradamente
atardecer	paleolítico	inicio	después	súbitamente	marchar
mañana	mesolítico	dio	tras	muere	replegarse
noche	calcolítico	antes	ya	falleció	precipitadamente
día	neolítico	final	días	murió	desecaba
medianoche	datan	dando	luego	prematuramente	mudarse
anochece	pleistoceno	llegada	ese	trágicamente	periódicamente
mediodía	precerámico	finales	meses	tempranamente	dirigiera
ocaso	epipaleolítico	principio	tiempo	...	...
madrugada	bronce	momento	comenzar		
...	...	...	...		

used in days of the month. In tasks related to time expressions, this enables the generalization to cases that are not included in the training corpus.

Besides the days of the week and the months of the year, it is interesting to observe, for example, the behavior of the words denoting times of day (e.g. sunrise), prehistoric times (e.g. Neolithic) or time adverbs. Table 1 shows the closest words to a given term ordered by distance.

3.2 Ordering and Granularity

In the examples presented, besides the formation of clusters we notice that the words that follow a sequential order (common in time domain), such as the days of the week, months, etc., tend to hold the order in terms of the representations. For instance, the representation of the word *miércoles* (Wednesday) occurs closer to that of *martes* (Tuesday) and of *jueves* (Thursday) than to other week day names. This shows that representations tend to preserve the chronological order of the terms.

[1] There is an English translated version in the Appendix B, Table 22.

Table 2 Table with the closest terms (to the heading) ordered by distance to the representations of ordinal and numerical terms.[2]

Primero	Segundo	Vigésimo	1853	1850	1700	1999
luego	tercer	trigésimo	1855	1840	1600	1998
segundo	primer	décimo	1854	1849	1800	1995
mismo	cuarto	cuadragésimo	1856	1870	1500	1997
último	último	noveno	1852	1860	1400	1996
primer	quinto	quincuagésimo	1851	1880	1200	2002
posteriormente	primero	octavo	1865	1851	1100	2003
después	tercero	quinto	1849	1830	1300	1994
...	...	...	...	...	...	...

Besides this tendency to preserve the order, ordinal terms and numbers present a granularity issue. When considering terms such as *primero* (first), *segundo* (second) and *tercero* (third), the term primero is close to terms like *segundo* and also to *último* (last). Then, when considering the terms close to *segundo*, terms like *tercero* (third), *cuarto* (fourth) and *quinto* (fifth) prevail. Additionally, next to the term *vigésimo* (twentieth) we find terms like *désimo* (tenth) and *trigésimo* (thirtieth) (see Table 2).

Similarly, granularity is considered in the vector representation of numerical terms. For instance, if the number is in hundreds granularity, for example 1700, other numbers from the hundreds like 1600 or 1800 are found close. Note as well that these terms are sequentially the previous and next ones to 1700 in the hundreds granularity. Something similar happens with numbers from the tens such as 1850, with close numbers like 1840, 1860 and 1870; and if 1853 is considered, its representation is close to that of 1855 and 1854.

Figure 1(b) shows the representations of a selection of numerical terms after reduction to 2 dimensions with t-sne. We can see clusters that reflect the granularity and order properties. It is also interesting to note how the sequence 1920, 1930, ..., 1970, 1980, 1990 connects the cluster close to 1900 (1888, 1889, ...) with the one close to 1990 (1991, 1992, ...). These order and granularity phenomena of the terms are potentially useful to interpret time expressions. Note that these properties are inferred only from the local contexts where words occur.

3.3 Regression on Years vectors

Experiments were conducted on neural network models trained to infer their respective numerical value from the vector representation of a numerical term. A sampling of 300 random numbers was considered in the range from 1000 to 2000 with their respective vector representations. The set was used to train a regression feedforward network with one 100-size hidden layer, and the lowest absolute error was used as target function for training. Though deeper studies are needed, the results obtained are encouraging. The experiments include results such as 1985.21; 1986.84; 1986.72; 1988.58 and 1989.02 for the $1985 - 1989$ sequence, where only 1989 was in the training data. However, not all the results were as accurated, although, they were close to the expected value.

[2] There is an English translated version in the Appendix B, Table 23.

Table 3 Labeling scheme used to classify time expressions.

Type	**Label**
date (da)	L_{da}, U_{da}
time (t)	L_t, U_t
duration (du)	L_{du}, U_{du}
set (s)	L_s, U_s
other	B, I, O

4 Timex Detection and Classification

Detecting time expressions involves identifying the expression and indicating its extension. The classification of time expressions means to determine the type of the expressions identified using a predefined set of types.

With a focus on the TimeML annotation system of time expressions and events [29], the *date* type was considered for dates (e.g. "three years before"), *time* for the times of day (e.g. "at 3 pm"), *duration* for durations (e.g. "5 minutes") and *set* for the expressions that represent frequencies and sets (e.g. "every Monday").

Detection and classification are resolved simultaneously by formulating the problem as the labeling of the words in the text. Each word is assigned a label indicating if a time expression applies and which type of expression it is. The labels from the *BILOU* scheme were used, as it has shown better results than the *BIO* scheme for the extraction of named entities [30]. The original scheme determines for each word if it is the beginning (B), the inside (I) or the last (L) token of an expression; if it is a single word expression (U) or a word that does not belong to any expression (O).

To consider classification we distinguish the last word in the expression adding an $\mathbf{L}_i$ and $\mathbf{U}_i$ label for each type, the other labels remain unchanged (see Table 3).

4.1 Model

To resolve the detection and classification of time expressions we consider models organized in layers to label words according to the *BILOU* variant presented. The model is applied sequentially to each word giving the word label on each activation.

The input layer receives word vectors, the information is transformed along the hidden layers to the output layer, where the *softmax* function is used. We consider *feedforward* and recurrent models with different numbers and size of hidden layers and regularization techniques.

We include local contextual information concatenating vector representation of fixed size window to both sides from the word to label (window context).

For example, if a size 2 left window context and 3 size right context is considered, the input received by the neural network would be $x = [w_{i-2}w_{i-1}w_iw_{i+1}w_{i+2}w_{i+3}]$; the concatenation vector of the word representations in the input text, where w_i is the representation of the word to classify. The disadvantage of this type of context is that it has a rigid size regardless of the case presented to the model. This might be inappropriate for selective context considerations or to capture long-range dependencies.

Table 4 Information about time expressions in the used corpus.

		Expresiones Temporales				
Name	**Words**	**Date**	**Duration**	**Time**	**Set**	**Total**
TEval13_es_train	46.687	585	215	49	29	878
TEval13_es_test	12.197	164	36	8	8	216

4.2 Corpus and Training

Training and evaluations were conducted with the data for Spanish included in the TempEval-3 task [39]. As the evaluation data is not available, the training data was divided into a training set and an evaluation set. The training set has **878** time expressions, and the evaluation set has **216** of them. Table 4 shows information from the corpora used.

The limited size of the corpus is a setback for supervised learning approaches in contrast to those that include rule-based knowledge. However, it is a good scene to test the unsupervised word vector representations as a generalization tool when there is limited data. A further disadvantage is that the reduced size of the evaluation set affects the quality of the evaluation and reduces the impact of small variations on the experiment results.

As for training, all the models considered were trained with *RMSprop* [37], a variant of backpropagation. In all cases we used a learning rate of 1×10^{-4} and a moment value of 0.9. The stopping criterion is that the improvement of the target function does not exceed 1×10^{-5} per 30 epochs. The experiments were conducted with *Theanets* package [19] implemented over *Theano* [36].

The evaluation is made through precision and recall at the expression level of the output of each model against the evaluation data. The global evaluation measure is taken through the F-score with the same balance for both.

4.3 Experiments

There follow the experiments conducted and the results obtained. The experiments focus on the dimensions of words and internal representations, number of layers, regularization techniques and variations on local context.

4.3.1 Dimension

[25] and [26] show that by increasing the dimension, vector quality improves. This tendency seems not to be unlimited, although there are no known results. Furthermore, compact vectors have desirable characteristics and benefits regarding computational cost.

To observe the impact of dimension empirically on the detection task we considered several dimensions representations under similar conditions. Models were trained avoiding to alter the rest of the environment.

The models used were three-layer feedforward models with three words of symmetric context. To preserve the proportion between layers, we adjusted the size of the hidden layer to three times the word dimension. We observed a steady increase in the recall, but the precision began to decrease. The results are shown in Table 5.

We observed the behavior that takes place when halving the size of the models considered for dimensions 150 and 200. In both cases we observed a slight improvement in the results, mainly contributed by the increase in precision, thus preserving global results. The results are shown in Table 6.

Table 5 Comparison of detection results in similar environments varying the dimension or the word representations using feedforward models with one hidden layer and three words (to each side) of symmetric context were used.

Dim	Train Acc	Prec	Rec	F
25	1x10-6	72.97	50.00	59.34
50	1x10-6	76.14	62.04	68.37
100	396x10-4	**82.35**	64.81	72.54
150	396x10-4	80.79	66.20	72.77
200	396x10-4	78.61	**68.06**	**72.95**

Table 6 Detection and classification results when halving the hidden layer dimensions in the most significant models in Table 5. We present between brackets the classification results.

Dim	Hid	Prec	Rec	F
150	450	80.79 (77.46)	66.20 (62.04)	72.77 (68.89)
	225	**82.28** (75.00)	66.67 (62.50)	**73.66** (68.18)
200	600	78.61 (77.97)	68.06 (**63.89**)	72.95 (**70.22**)
	300	79.14 (**78.61**)	**68.52** (62.96)	73.45 (69.92)

4.3.2 Detection with/without Classification

As the models that resolve the classification of expressions also conduct detection tasks, it would be interesting to know how classification affects detection. Table 7 shows the comparison of the detection results between models with output layers for detection and the same model but changing the output layer for classification. In general, detection precision was improved in the models that also classify the expression. This might be due to the fact that the classification information is used in detection.

As for lexical generalization to cases that are not included in the training data, we observed at least one positive case with the word *semestre* (semester). This word does not occur in the training data, but it does appear twice in the test data. A few of the models can detect at least one of the two occurrences. This might be the case because the model was able to generalize from words like *trimestre* (trimester) and *cuatrimestre* (four-month period) that do occur in the training data.

4.3.3 Hidden Layer Size

Hidden layers are a fundamental part of the model that build intermediate representations to resolve the task. The sizes of hidden layers correspond to the dimensions of the spaces of intermediate representations. Hidden layers that are too small might hinder the right resolution of the task, while a very large size is more inefficient and might lead the model to overfit the training data.

We previously observed that the size of hidden layers (although up to now only models with one hidden layer have been considered) has an impact on results. To observe the effect of considering different sizes in the hidden layer, we trained a sequence of one-layer models to classify expressions varying its size (Table 8).

As for the number of hidden layers, we conducted a initial experiment where an additional 100-size layer before output was considered for a 300-size layered model. The results mainly improved the recall, with 79.67 precision and 67.13 recall, which entails an F-score of 72.86.

Table 7 Comparison of results in detection in models trained for classification with the same model but with detection output layer (between brackets).

Dim	Hid	Prec	Rec	F	
150	450	82.66 (80.79)	66.20 (66.20)	73.52 (72.77)	+0.75
	225	80.00 (82.28)	66.67 (66.67)	72.73 (73.66)	-0.93
200	600	83.05 (78.61)	**68.06 (68.06)**	**74.81 (72.95)**	+1.86
	300	**83.81 (79.14)**	67.13 (68.52)	74.55 (73.45)	+1.10

Table 8 Results in the classification of expressions with feedforward models on 200-dimension words, with three words of symmetric context, varying the size of the unique hidden layer.

Hid	P	R	F
100	74.58	61.11	67.18
200	75.28	62.04	68.02
300	78.61	62.96	69.92
400	77.27	62.96	69.39
500	76.95	63.42	69.54
600	79.21	**65.27**	**71.57**
700	**79.31**	63.89	70.77
1000	76.40	62.96	69.04
2000	79.19	63.43	70.44

This looks promising in the deeper consideration of models. We will come back to this point below.

4.3.4 Window Context Size

The local context is extremely important for time expressions. We tested various context window configurations. Models with isolated left and right contexts were considered, and also models with symmetric contexts. The model that served as base had a hidden layer whose size is defined according to context length. The inclusion of any context produced better results in all cases than the context-less version, however, large contexts can negatively affect the results. Table 9 shows the results.

As expected, considering the symmetric context yielded far better results than considering only the left or right context. Although the best global result was achieved with two context words (two left and two right), the best recall result was reached with three words whose F-score is also next to the maximum result. The results also show that the right context seems to provide more information than the left one for this task.

4.3.5 Regularizations

Regularization techniques often can improve the way the neural network model is trained giving as result a best generalizatión of unseen cases. We consider input and hidden noise, dropout, l1 and l2 regularizations. We try different values for each regularization technique considered and here we present the conclusions obtained. The details of the results obtained are in Appendix A.

We try input and hidden noise with positive results mainly in the first. Input noise improves substantially the recall (in comparison with the same model without any noise)

Table 9 Comparison of classification results when increasing the left context without considering the right (left value), left (center) and symmtric context (right value). The base model is feedforward of 200 word dimension, and according calculated hidden size.

Context	Hid	Prec	Rec	F
0	100	60.64	26.39	36.77
1	100	67.31 / **64.97** / 72.78	32.41 / 47.22 / 56.94	43.75 / 54.69 / 63.90
2	200	**69.83** / 63.64 / **80.84**	**37.50** / 48.61 / 62.50	**48.79** / 55.12 / **70.50**
3	300	67.00 / 62.42 / 76.92	31.02 / 45.37 / **64.81**	42.40 / 52.55 / 70.35
4	400	69.81 / 62.42 / 76.86	34.26 / 43.06 / 43.06	45.96 / 50.96 / 55.19

Table 10 Results of the classification of expressions using BLSTM models based on 200-dimension words with no window context an one single hidden layer.

Hid	Steps	P	R	F
150	100	54.20	32.87	40.92
200	100	60.75	30.09	40.25
600	100	63.16	**33.33**	**43.64**
300	15	64.36	30.09	41.25
600	15	64.44	27.31	38.43
1000	15	**71.44**	30.56	42.85
2000	15	71.25	26.39	38.51
600	3	64.00	14.09	24.06

and minor improvement in precision. The grade of input noise that gives the best results was 0.2. Respect to hidden noise, a much lesser improvement was detected with its best configuration in 0.05.

Dropout also improved precision and recall if a very low value was considered, the value that gives the best results was 0.01. Respect l1 (sparsity) and l2 (weight decay) a minor improvement was noticed in both cases. In the case of l2, the results fluctuate and for that reason is hard to determine which configuration is better. For l1, the best results were obtained with 0.01 affecting positively to the recall of the model.

4.3.6 Recurrent Networks

A non-exclusive alternative to the window context is the context considered by recurrent models. Previously established activations are considered through feedback in the hidden layer, allowing the sequential application of the network to consider the context that corresponds to the inputs previously applied.

Context considerations of the recurrent models are more flexible than the explicit information provided by the window context. However, its interpretation is more complex, and the experiments conducted did not produce good results without the additional consideration of the window context.

As the results improve significantly when considering both left and right contexts, bidirectional models are considered, especially *Bidirectional Long-Short Term Memory* (BLSTM) [15].

In the experiments conducted with BLSTMs, the results obtained were lower than those of the feedforward models with a window context. Table 10 shows the results. Different sizes for the recurrent layer and depths of recurrence were considered.

Table 11 Results of the classification of expressions using feedforward models on 200-dimension words with 3 words of symmetric window context varying hidden layers number and sizes.

h1	h2	h3	h4	P	R	F
300	–	–	–	78.61	62.96	69.92
300	100	–	–	**79.67**	**67.13**	**72.86**
300	200	100	–	74.07	64.81	69.13
600	400	200	100	64.25	57.41	60.64

Table 12 Results of the classification of expressions using BLSTM models on 200-dimension words with no window context, varying hidden (recurrent) layers number and sizes.

Hid1	Hid2	Hid3	Prec	Rec	F
200	–	–	60.75	30.09	40.25
300	150	–	**68.42**	**54.17**	**60.46**
300	200	100	61.15	45.83	52.52

4.3.7 Network Depth

The number of hidden layers is a key aspect when defining layered neural models. It is known that the training of networks with several hidden layers presents difficulties. We experiment with models up to four hidden layers empirically.

Different depths were considered in homogeneous models, that is, with all layers of the same type. The first experiment consisted in adding an extra hidden layer between the existing hidden layer and the output layer to the model previously used as a base. The inclusion of the additional layer substantially improved the results, especially regarding recall (see Table 11).

Upon observing the favorable effect of considering a model with two hidden layers (versus the single hidden layer model), we trained and assessed models with three and four hidden layers. In this case, results show that considering more than two hidden layers the model was not adequately trained.

Regarding recurrent models, even though their results were far lower than those of the feedforward models with a context window, we studied the effect of considering more hidden layers, also formed by BLSTMs. As in the previous case, considering one additional layer resulted in a considerable improvement and the F measure decreased when more than two hidden layers were considered (Table 12). These models were trained considering 100 recurrence steps.

4.3.8 Combining Variations

The results yielded by different variations of neural models, including feedforward and recurrent BLSTM models, were shown above. We tested different structural configurations and regularization techniques.

A general observation regarding the experiments conducted is that the random initialization of the weights of the model can cause different instances to produce different results. To reduce the impact of this situation, we repeated some of the experiments in questionable situations and others randomly, and we found no high differences between the different instances of the same experiment in most cases.

Regarding the structural considerations of the model, considering two hidden layers, instead of one, had a positive effect. This behavior was not sustained for greater depths.

Table 13 Results of the classification of best models presented before (top), combinations of which show positive effects independently (middle) and good combinations with an improved word vectors set.

h1	h2	$noise_I$	dropout	L1	L2	P	R	F
300	–	–	–	–	–	78.61	62.96	69.92
600	–	–	–	–	–	79.21	65.27	71.57
300	100	–	–	–	–	79.67	67.13	72.86
300	**–**	**0.20**	**–**	**–**	**–**	**80.66**	**67.59**	**73.55**
300	–	–	0.01	–	–	80.57	65.28	72.12
300	–	–	–	0.01	–	78.89	65.74	71.72
300	–	–	–	–	0.001	80.11	65.28	71.94
300	–	0.20	–	–	–	80.66	67.59	73.55
700	–	0.2	–	–	–	81.03	65.28	72.31
700	400	0.2	–	–	–	**81.36**	66.67	73.28
400	150	0.2	0.1	–	–	80.35	64.35	71.46
350	120	0.1	0.1	–	–	80.32	**68.06**	**73.68**
600	–	0.1	–	–	–	81.21	**68.06**	**74.05**
450	200	0.1	–	–	–	**81.92**	67.13	73.79

The size of the hidden layers fluctuated for sizes over 300, gradually decreasing for lower values. Input and hidden layer noise, dropout, L1 and L2 were considered as regularization techniques. The most significant improvement was observed with the consideration of noise in the input. Table 13 shows the best results obtained for each family of experiments.

In order to see how different techniques which yielded good results perform when combined, we considered experiments with two hidden layers, input noise and dropout simultaneously. Middle of Table 13 shows the results and bottom of the table includes results of models trained using 300-dimension vectors provided by [6].

4.4 Comparison with SVM

We compare the obtained results with Support Vector Machines (SVM) in order to empirically validate the advantage of considering neural models versus other alternatives. We consider SVM classifier with a word dimension of 200 and 3 words with a local context in both directions yelding 60.8 of F measure in time expressions recognition and classification. This result indicates, at least initially, that neural models make a better use of word embeddings and local context to resolve this task.

4.5 Comparison with Other Works

It is not possible to compare this to other works because we have not been able to access the same evaluation data. Nevertheless, we think it is useful to at least include some comparative values. As we explained before, the training set was split to have an evaluation set. In essence, the model was trained with a part of the training set and evaluated with an evaluation set of similar characteristics, but which was different.

The comparison is made based on the model that produced the best results in detecting time expressions. The model considered has a single 300-size hidden layer, with 3 words with symmetric contexts, a 200-dimension word and a variance value of 0.2 for the input noise.

Table 14 Results of the classification for Spanish in the tempeval-3 task. ANNTime refers to the best results obtained in this work. ANNTime* refers to the same model applying heuristics to rectify inconsistent labels. ANNTime-nabu and ANNTime-nabu* correspond to the best models using the vectors of [6].

	P(r)	R(r)	F_1(r)	F_1(s)
HeidelTime	**96.0**	**84.9**	**90.1**	**85.3**
TIPSemB-F	93.7	81.9	87.4	82.6
FSS-TimEx	86.6	52.3	65.2	49.5
ANNTime	92.8	77.8	84.6	78.6
ANNTime*	91.2	81.5	86.1	79.2
ANNTime-nabu	91.7	76.8	83.6	79.6
ANNTime-nabu*	91.4	83.3	87.2	79.9

Table 15 Information of the TempEval 2013 corpus for English.

Name	Words	Date	Duration	Time	Set	Total
TEval13_en_silver	718.746	11.133	1.346	192	68	12.739
TEval13_en_platinum	7.003	96	34	4	4	138

Table 14 shows the comparison of the results. We also included the results obtained with the model that produced the best results using the vectors of [6].

For each model we include the result of applying the model and, subsequently, a basic heuristics to correct inconsistent labels (this is marked with an asterisk in Table 14). Heuristics consists of correcting an incorrect label in three situations. First, if an O is followed by an I, the I is replaced by B or U_{da} accordingly[3]. Second, if an O is preceded by a B or I and followed by an I or L, said label is replaced by I. Third and last, if an I is followed by an O, said I is replaced by L_{da} o U_{da} accordingly. The application of heuristics improved results by almost 4% regarding the recall of the model, and reduced precision by 1.6%, thus resulting in a global improvement (F) of 1.5%, in the case of overlapping for basic vectors and a 3.6% improvement for 300-dimension vectors.

4.6 Time expressions in English

The results using same neural models, but applied to resolve the task in English, are shown, taking into account the lessons learned throughout the experiments conducted for Spanish. We use the data conveyed for English in TempEval 2013 (see Table 15). The representations presented by [26] of 200 dimensions, built with a six billion word corpus, are used.

The approach used is applied directly because it does not include specific knowledge of the language or domain. We trained a feedforward model with three words with symmetric contexts, based on the 200-dimension representations, with a single, 300-size hidden layer and a noise level of 0.2 in the input, which was the model that produced the best results in detection for Spanish. We also consider two hidden layers models detecting faster convergence. Table 16 shows the results obtained.

Regarding the classification of expressions, in all cases models showed an F-score for classification lower than the one in the work of [22], which is to be expected because this

[3] The date type was used because it is the most frequent.

Table 16 Results of the detection and classification of expressions for English.

h1	h2	$noise_I$	P_{det}	R_{det}	F_{det}	Acc_{class}	F_{class}
300	–	0.20	93.88	66.67	77.97	93.47	72.88
300	100	–	95.83	66.67	78.63	**95.65**	75.21
900	200	–	**96.80**	65.94	78.45	95.60	75.00
600	100	0.20	95.88	**67.39**	**79.15**	93.54	74.04
[22]			86.1	80.4	83.1	93.4	**85.4**

Table 17 Top: Results of events detection model in Spanish with and without time expressions detection. Middle: Results of events detection model in English with time expressions detection. Bottom: Time expressions detection in English including events detection.

	P	R	F
Events (without timexes) (600x200-0.2) (SP)	81.16	79.40	80.27
Events (with timexes) (600x200-0.2) (SP)	84.31	79.08	81.61
CRF+Morph+SRL+WNet [23]	83.43	79.54	81.40
Events (with timexes) (600x200-0.2) (EN)	79.31	79.62	79.46
ATT1(MaxEnt+Syn+Sem) [20]	81.44	80.67	81.05
Timexes (with Events) (600x200-0.2) (EN)	98.99	71.01	82.70
Semantic Parsing [22]	86.1	80.4	83.1

value is influenced by detection. Regarding accuracy, we obtained better results in all cases[4], but it should be noted that this value is positively affected by the lower result in detection.

5 Events Detection

The model presented only uses supervised training data and word representations built through self-supervised methods. This means that the approach is rapidly adaptable to other languages, as it was observed for English. Similarly, it can be considered to deal with tasks that have a compatible formulation, for instance, events detection. The main difference is that, as opposed to events, time expressions often include words such as the prepositions “in” and “during”, days of the week, names of months, etc., which can act as indicators that aid their detection.

It is interesting to notice that in events detection part-of-speech information becomes extremely useful because of the abundance of events denoted as verbs. In this approach any part-of-speech is included, just word embeddings. This could be interpreted as analogous to the human common understanding of a language, where the knowledge of what is a verb or a noun it is not particularly needed to understand. Another interesting observation is that the jointly detection of time expressions and events performs better that each isolated task (Table 17).

[4] This measure was calculated with $AttrAccuracy = AttrF1/EntityExtractionF1$ [39].

6 Conclusion

This work tackles the problem of detecting and classifying time expressions with approaches based purely on distributed representations of words and artificial neural networks, and it presents interesting results for Spanish with a relatively small dataset.

Initial results that support the possibility of interpreting time expressions purely from data were presented. We found that word representations tend to contain information that refers to the sequential nature of units such as the names of months, days and numbers, among others. This trait is potentially useful for interpreting expressions. Regarding numerical and ordinal entities, we found that they take into account granularity information in the vector representations.

We showed the behavior of neural network models in detecting and classifying expressions, as well as the effect of varying the definition of the model. Results show the relevance of the local linguistic context. Recurrent models did not produce significant improvements in the experiments conducted, when compared to simple feedforward models, and the latter have the advantage that their training is much less costly. We also showed the effect of applying regularization techniques, especially emphasizing the benefit of considering noise in the input data and dropout.

Based on the previous results, we trained models for the problem in English and we also added the events detection problem. The results for English are encouraging, and they provide an example of the approach behavior when more data is available, both for supervised data to train models and for unsupervised data used to build the word embeddings. Regarding events detection, it is of special interest because it is not strongly based on specific lexicon, as is the case of time expressions. The results obtained were encouraging and the inclusion of said task, together with the detection and classification of time expressions brings an improvement in the results as compared to the isolated treatment presented before.

References

1 Sisay Fissaha Adafre and Maarten de Rijke. Feature engineering and post-processing for temporal expression recognition using conditional random fields. In *Proceedings of the ACL Workshop on Feature Engineering for Machine Learning in Natural Language Processing*, FeatureEng'05, pages 9–16, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL: `http://www.aclweb.org/anthology/W05-0402`.

2 David Ahn, Sisay Fissaha Adafre, and Maarten de Rijke. Towards task-based temporal extraction and recognition. In Graham Katz, James Pustejovsky, and Frank Schilder, editors, *Annotating, Extracting and Reasoning about Time and Events*, number 05151 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

3 David Ahn, Joris Rantwijk, and Maarten Rijke. A cascaded machine learning approach to interpreting temporal expressions. In *in Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2007*. Citeseer, 2007.

4 Gabor Angeli, Christopher D. Manning, and Daniel Jurafsky. Parsing time: Learning to interpret time expressions. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 3-8, 2012, Montréal, Canada*, pages 446–455, 2012. URL: `http://www.aclweb.org/anthology/N12-1049`.

5 Gabor Angeli and Jakob Uszkoreit. Language-independent discriminative parsing of temporal expressions. In *Proceedings of the 51st Annual Meeting of the Association for Compu-*

tational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers, pages 83–92, 2013. URL: http://www.aclweb.org/anthology/P/P13/P13-1009.pdf.

6 Agustín Azzinnari and Alejandro Martínez. Representación de Palabras en Espacios de Vectores. Proyecto de grado, Universidad de la República, Uruguay, 2016.

7 Steven Bethard. Cleartk-timeml: A minimalist approach to tempeval 2013. *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, 2013.

8 Steven Bethard. A synchronous context free grammar for time normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 821–826, 2013. URL: http://www.aclweb.org/anthology/D/D13/D13-1078.pdf.

9 Angel X. Chang and Christopher Manning. Sutime: A library for recognizing and normalizing time expressions. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

10 R. Collobert and J. Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. *In Proceedings of ICML, pages 160–167*, 2008.

11 R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *JMLR, 12:2493–2537*, 2011.

12 Mathias Etcheverry and Dina Wonsever. Spanish word vectors from wikipedia. *Language Resource Conference (LREC 2016)*, 2016.

13 Michele Filannino. Temporal expression normalisation in natural language texts. *CoRR*, abs/1206.2010, 2012. URL: http://arxiv.org/abs/1206.2010.

14 Michele Filannino, Gavin Brown, and Goran Nenadic. Mantime: Temporal expression identification and normalization in the tempeval-3 challenge. *CoRR*, abs/1304.7942, 2013. URL: http://arxiv.org/abs/1304.7942.

15 A. Graves and J. Schmidhuber. Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures. *Neural Networks*, 18(5–6):602–610, 2005.

16 Claire Grover, Richard Tobin, Beatrice Alex, and Kate Byrne. Edinburgh-LTG: TempEval-2 System Description. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval'10, pages 333–336, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL: http://dl.acm.org/citation.cfm?id=1859664.1859738.

17 Naman Gupta, Aditya Joshi, and Pushpak Bhattacharyya. A temporal expression recognition system for medical documents by taking help of news domain corpora. *12th International Conference on Natural Language Processing (ICON)*, 2015.

18 Ozan Irsoy and Claire Cardie. Opinion mining with deep recurrent neural networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

19 Leif Johnson, Majid alDosari, Filip Juricek, John, Kyle Kastner, Yoav Goldberg, talbaumel, Yu Yang, mhr, Eben Olson, and Sergey Romanov. theanets: v0.6.1, July 2015. doi:10.5281/zenodo.19930.

20 Hyuckchul Jung and Amanda Stent. ATT1: temporal annotation using big windows and rich syntactic and semantic features. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pages 20–24, 2013.

21 Oleksandr Kolomiyets and Marie-Francine Moens. Kul: Recognition and normalization of temporal expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval'10, pages 325–328, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL: `http://www.aclweb.org/anthology/S10-1072`.

22 Kenton Lee, Yoav Artzi, Jesse Dodge, and Luke Zettlemoyer. *Context-dependent semantic parsing for time expressions*, volume 1, pages 1437–1447. Association for Computational Linguistics (ACL), 2014.

23 H. Llorens, E. Saquete, and B. Navarro-Colorado. Timeml events recognition and classification: Learning crf models with semantic roles. In *In COLING 2010, 23rd International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2010, Beijing, China*, page 725–733, 2010.

24 Inderjeet Mani and D. George Wilson. Robust temporal processing of news. In *38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China, October 1-8, 2000.*, 2000. URL: `http://www.aclweb.org/anthology/P00-1010`.

25 Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *In Proceedings of Workshop at ICLR*, 2013.

26 Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

27 Jordi Poveda, Mihai Surdeanu, and Jordi Turmo. An analysis of bootstrapping for the recognition of temporal expressions. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-Supervised Learning for Natural Language Processing*, SemiSupLearn'09, pages 49–57, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. URL: `http://dl.acm.org/citation.cfm?id=1621829.1621836`.

28 Georgiana Puscasu. A framework for temporal resolution. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*, 2004. URL: `http://www.lrec-conf.org/proceedings/lrec2004/pdf/664.pdf`.

29 James Pustejovsky, José M. Castaño, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. Timeml: Robust specification of event and temporal expressions in text. In *New Directions in Question Answering, Papers from 2003 AAAI Spring Symposium, Stanford University, Stanford, CA, USA*, pages 28–34, 2003.

30 Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. *Proceedings of the Thirteenth Conference on Computational Natural Language Learning , CoNLL '09, pages 147–155, Stroudsburg, PA, USA.*, 2009.

31 Richard Socher, John Bauer, Christopher D, Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. *Association for Computational Linguistics 2013 Conference (ACL 2013)*, 2013.

32 Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in Neural Information Processing Systems (NIPS 2011)*, 2011.

33 Richard Socher, Alex Perelygin, Jason Chuang Jean Wu, Chris Manning, Andrew Ng, and Chris Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, 2013.

34 Jannik Strötgen and Michael Gertz. HeidelTime: High Quality Rule-based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval'10, pages 321–324, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL: `http://dl.acm.org/citation.cfm?id=1859664.1859735`.

35 Jannik Strötgen and Michael Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298, 2013. doi:10.1007/s10579-012-9179-y.

36 Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL: http://arxiv.org/abs/1605.02688.

37 Tijmen Tieleman and Geoffrey E. Hinton. Lecture 6.5 – rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 2012.

38 Naushad UzZaman and James F. Allen. TRIPS and TRIOS System for TempEval-2: Extracting Temporal Information from Text. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval'10, pages 276–283, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL: http://www.aclweb.org/anthology/S10-1062.

39 Naushad UzZaman, Hector Llorens, James F. Allen, Leon Derczynski, Marc Verhagen, and James Pustejovsky. Tempeval-3: Evaluating events, time expressions, and temporal relations. *CoRR*, abs/1206.5333, 2012. URL: http://arxiv.org/abs/1206.5333.

40 L. J. P. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research 9(Nov):2579-2605*, 2008.

A Result Tables

A.1 Noise

We tested the effects of applying noise to the input. We slightly modified the input, according to Gaussian distribution centered at zero and using several variance values, in the input data and intermediate representations. The variance regulates the noise level injected.

Including noise substantially improved the recall, and it slightly improved the precision for noise values from 0.1 (Table 18). The results for noise values under 0.1 were slightly lower. The best results were obtained when considering 0.2 injected noise variance, with significant degradation when considering a 0.3 variance. The hidden layer noise has a less substantial impact and is more sensitive to the amount of noise injected.

We trained a model with the best noise levels in the input and in the hidden layer to observe the interaction of both cases. It seems that the effects add up degrading the precision.

As for expression detection, as well as in classification, we obtained the best result when considering a 0.2 noise value at the input and with no noise in the hidden layer. This was the same case that produced the best results in classification. The best results achieved

Table 18 Results in the classification of expressions in feedforward models on 200-dimension words, with three words of symmetric context and a 300-size hidden layer, with varying noise levels at the input and hidden layer. The results between brackets correspond to cases with noise in the hidden layer.

Noise	Prec	Rec	F
0.00	78.61	62.96	69.92
0.01	77.20 (**79.41**)	61.11 (62.50)	68.22 (69.95)
0.05	77.40 (77.60)	63.43 (**65.74**)	69.72 (**71.18**)
0.10	79.33 (77.40)	65.74 (63.43)	71.90 (69.72)
0.20	**80.66** (77.01)	**67.59** (62.04)	**73.55** (68.72)
0.30	78.65 (–)	64.81 (–)	71.06 (–)

Table 19 Results in the classification of expressions from feedforward models on 200-dimension words, with three words of symmetric context and a 300-size hidden layer, with varying levels of dropout in the hidden layer.

Dropout	Prec	Rec	F
0.00	78.61	62.96	69.92
0.01	**80.57**	**65.28**	**72.12**
0.05	78.29	63.43	70.08
0.10	75.82	63.89	69.35
0.20	77.14	62.50	69.05

Table 20 Results in the classification of expressions from feedforward models on 200-dimension words, with three words of symmetric context and a 300-size hidden layer, with varying levels of L2 regularization.

L2	P	R	F
0.00	78.61	62.96	69.92
0.0001	78.21	64.81	70.89
0.001	80.11	65.28	**71.94**
0.01	77.58	62.50	69.23
0.05	79.21	**65.43**	71.57
0.10	**81.21**	62.04	70.34
0.20	78.82	62.04	69.43

for detection were 78.59 F-score for the strict case and 81.42 for the relaxed case, where displacements of a word were allowed in the extension of the expression detected.

A.2 Dropout

This section shows the behavior of dropout (or multiplicative mask) in the hidden layer. This technique randomly turns to zero some entries in intermediate layers. It can be seen as a partial network where some components are completely eliminated. The dropout value is the portion of units set to zero.

We conducted an experiment with varying dropout levels for the hidden layer of a feedforward (Table 19). Regarding detection, a dropout value of 0.01 also yielded the best result in the hidden layer, with a 75.70 F-score in the strict case and of 81.84 for the relaxed case.

A.3 L1 and L2 regularizations

Overfitting can be reflected on high values in the weights learned, so training the model avoiding high values in the learned weights may help to prevent it. This technique is called *weight decay* (or L2 regularization). The technique consist on adding the term $\lambda_{L2}\|\theta\|_2$ to the target function, where $\|.\|_2$ is the L2 norm, and θ is the weights vector to fit. This aims to a reduction in the weights magnitude besides the function to optimize.

Many instances of a base model was trained with different values for L2 regularization each (Table 20). Results improve when considering $\lambda_{L2} = 0.001$. Anyways, the results fluctuate and for that reason is hard to determine which was the best configuration.

Furthermore, besides the tendency towards low values in learned parameters, there might be a positive effect on sparse representations. A way to tend to sparse parameters is to

Table 21 Results in the classification of expressions from feedforward models on 200-dimension words, with 3 words of symmetric context and a 300-size hidden layer, with varying levels of L1.

L1	P	R	F
0.00	78.61	62.96	69.92
0.001	**80.00**	62.96	70.47
0.01	78.89	**65.74**	**71.72**
0.05	74.85	59.26	66.15
0.10	74.30	61.57	67.34
0.20	77.21	56.50	65.24

include the term $\lambda_{L1}\|\theta\|_1$, where $\|.\|_1$ is the L1 norm. Including this term reduces some components if the result is not affected.

Table 21 shows the results obtained in the detection and classification of expressions. We can see that values higher than 0.05 for L1 regularization significantly degrade the results. The best result was obtained with 0.01, which showed an improvement of 2 points in the F-score.

B English Translated Tables

Table 22 Table with the closest terms (to the heading) ordered by distance to the time expressions representations.

Dawn	Neolithic	Start	Before	Impromptu	Hurriedly
sunset	paleolithic	initiation	after	suddenly	march
morning	mesolithic	gave	behind	die	retract
night	chalcolitic	before	already	perish	hastily
day	neolthic	final	days	died	dried out
midnight	date	giving	then	prematurely	move
nightfall	pleistocene	arrival	that	tragically	periodically
midday	preceramic	endings	months	early	direct
twilight	epipaleolytic	beginning	time	...	...
early morning	bronze	moment	begin		
...	...	...	...		

Table 23 Table with the closest terms (to the heading) ordered by distance to the representations of ordinal and numerical terms.

First	Second	Twentieth	1853	1850	1700	1999
then	third	thirtieth	1855	1840	1600	1998
second	first	tenth	1854	1849	1800	1995
same	fourth	fortieth	1856	1870	1500	1997
last	last	nineth	1852	1860	1400	1996
first	fifth	fiftieth	1851	1880	1200	2002
later	first	eighth	1865	1851	1100	2003
after	third	fifth	1849	1830	1300	1994
...	...	...	...	...	...	...