UH-MatCom at eHealth-KD Challenge 2020: Deep-Learning and Ensemble Models for Knowledge Discovery in Spanish Documents

Juan Pablo Consuegra-Ayala[a], Manuel Palomar[b,c]

[a]*Faculty of Math and Computer Science, University of Habana, 10200 La Habana, Cuba*
[b]*University Institute for Computing Research (IUII), University of Alicante, 03690 Alicante, Spain*
[c]*Department of Language and Computing Systems, University of Alicante, 03690 Alicante, Spain*

Abstract
The eHealth-KD challenge hosted at IberLEF 2020 proposes a set of resources and evaluation scenarios to encourage the development of systems for the automatic extraction of knowledge from unstructured text. This paper describes the system presented by team UH-MatCom in the challenge. Several deep-learning models are trained and ensembled to automatically extract relevant entities and relations from plain text documents. State of the art techniques such as BERT, Bi-LSTM, and CRF are applied. The use of external knowledge sources such as ConceptNet is explored. The system achieved average results in the challenge, ranking fifth across all different evaluation scenarios. The ensemble method produced a slight improvement in performance. Additional work needs to be done for the relation extraction task to successfully benefit from external knowledge sources.

Keywords
eHealth, Knowledge Discovery, Natural Language Processing, Machine Learning, Entity Recognition, Relation Extraction

1. Introduction

This paper describes the system presented by team UH-MatCom in the eHealth-KD challenge at IberLEF 2020 [1]. The challenge proposes a set of evaluation scenarios and corpora to encourage the development of systems for the automatic extraction of knowledge from unstructured text. Both the relevant entities and the relations between them that occur in plain text documents are asked to be identified. Though health-related documents are the main source of text, the knowledge extracted is general-purpose, and non-health domain documents were also available as additional evaluation collections.

Our system makes use of several models with different architectures, which were evaluated across three different runs. Two of the runs submitted consist of deep learning-based architectures, using state of the art NLP techniques for named entity recognition (NER) and relation extraction. BERT [2], CRF [3], and Bi-LSTM [4] layers are the main components of these models. The first architecture uses general domain knowledge extracted from ConceptNet [5]

Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020)
EMAIL: jpconsuegra@matcom.uh.cu (J.P. Consuegra-Ayala); mpalomar@dlsi.ua.es (M. Palomar)
ORCID: 0000-0003-2009-393X (J.P. Consuegra-Ayala)

to enrich the information about entity pairs in the relation extraction task. The second one uses additional syntactic features, such as dependency tree information instead of ConceptNet, to encode the relation between entity pairs. The third run ensembles these two models and two additional ones to produce an aggregated result. The ensembler parameters are automatically tuned according to the training and development collection.

The remainder of the paper is organized as follows. Section 2 and 3 describe the different architectures used by the system and the ensemble method applied. In Section 4, the official results achieved in the challenge are presented. In Section 5, some insights found about the quality of each strategy are shared. Finally, Section 6 presents the conclusions of the paper along with some future work recommendations.

2. System Description

Our system makes use of two main architectures, from which several components are turned on or off, to produce different variants. Both architectures are deep learning-based, and they will be referenced onwards in this paper as *BILUOV tagger* and *pairwise classifier*. The distinction between the two main architectures raises from the type of problem they solve. Each one is used independently to solve the corresponding task of the challenge.

The *BILUOV tagger* solves a sequence tagging problem, in which each token from an input sequence is assigned a label from the BILUOV [6] entity tagging scheme[1]. This model is used to solve Task A of the challenge, oriented toward extracting the relevant entities mentioned in a plain text sentence. The four types of entities proposed in the challenge (i.e., Concept, Action, Predicate, and Reference) are handled independently. One *BILUOV tagger* instance is trained for each entity type. These instances can optionally share intermediate layers during training, a parameter that was explored and optimized during the challenge development phase.

The *pairwise classifier* solves a multi-class classification problem, in which a sequence of tokens is processed and then assigned a single label. This model is used to solve Task B of the challenge, oriented toward identifying the relevant relations (if any) between the previously extracted entities. A single instance of this model is trained to handle all types of relations. This model can optionally reuse some of the previously trained layers of the *BILUOV tagger*, depending on whether the entity extraction models shared those layers or not.

2.1. Input Handling

Both models work in the most general case on sequences of tokens. The *BILUOV tagger* receives the sentences tokenized as it is, and produces an output for each token in the input sequence. The *pairwise tagger* receives the path of tokens in the dependency tree of the sentence between each pair of entities occurring in the sentence (or between their Lowest Common Ancestor (LCA) in the case of multi-word entities). Additionally, the two sequences of tokens that make the source and head entities, respectively, are also provided to the pairwise tagger with their corresponding previously assigned entity type. Figures 1 and 2 show the way inputs are transformed by each architecture.

[1]https://devopedia.org/named-entity-recognition (Accessed 2020-05-01).

2.1.1. Common to Both Tasks

Each plain text sentence is tokenized at the word level. The following features are extracted for each token. Figure 1 illustrates how these are directly used in the *BILUOV tagger*.

- **Character Representation**, that consists of assigning an integer value to each character of the token according to its index in a predefined vocabulary. The selected vocabulary was one of all ASCII letters, digits, and punctuation symbols. Padding is added at the end to ensure that all tokens have the same number of characters.
- **Word Embedding**, that consists of a vector representation of the token in a continuous semantic space. Classical pretrained word embeddings or contextual ones, such as BERT, were considered.
- **POS Tag Representation**, that consists of a one-hot encoding of the Part-of-speech tag of the token.

2.1.2. Task B specifics

On top of the previous features, some additional features are included to each token for the *pairwise classifier*. Figure 2 illustrates how these are directly used in the *pairwise-classifier*.

- **Dependency Tree Representation**, that consists of a one-hot encoding of the dependency label of the token on the sentence's tree.
- **Entity Type Representation**, that consists of a one-hot encoding of the entity type of the token according to the label it was assigned in the previously finished entity extraction task. Some tokens do not belong to any entity so they are assigned an extra *Not An Entity* (NaE) label. No special consideration was taken into account for tokens belonging to multiple entities, but instead, an arbitrary label was selected from the candidates.

The dependency tree representation is only used for the tokens that form the path between the source and head entities of the relation. Similarly, the entity type representation is not provided for the sequence of source and head tokens, but instead, it is given as an additional feature for the whole sequence.

2.1.3. ConceptNet

Finally, external knowledge from ConceptNet about the source and head entities of the relation is optionally provided to the *pairwise classifier*. The information is captured in 36 ConceptNet relations [5] divided across three different matching ranges.

The **direct match** range stands for relations found between the exact entities (possibly multi-word) in ConceptNet or their lemmatized versions.

The **related match** range stands for relations found between the exact entities, their lemmas, or words belonging to the same equivalence class that them according to ConceptNet's *SimilarTo*, *Synonym*, *RelatedTo*, and *EtymologicallyDerivedFrom* relations. The use of this kind of match allows to check inter-language relations, which is crucial to make full use of the knowledge contained in ConceptNet since some relations only exist between their English equivalents.

The **partial match** range stands for direct and related matches found between the individual words that make up the entities.

This information is encoded in a k-hot vector whose active components (i.e., set to 1) are the ones corresponding to the relations for which a match was found. Only the subset of relations occurring between Spanish and/or English terms was preprocessed due to computational resource constraints. Nonetheless, this information is enough in the context of the challenge since all documents are written in Spanish, and the English subset allows to capture the most relevant relations between Spanish terms through equivalence relations.

2.2. Entity Recognition Model

To solve the entity recognition task, the *BILUOV tagger* architecture is used. Figure 1 summarizes the architecture shared by these models.

At first instance, a character encoding layer is used to transform the character representation of each token into a single vector that captures morphological dependencies on the token. This technique has been proven effective in dealing with words out of vocabulary that share some similarities with previously seen words. The character encoding layer consists of an embedding layer *(A)* (a lookup table) followed by a Bi-LSTM layer *(B)*. The Bi-LSTM consumes the sequence of embedded characters and produces the concatenation of the internal LSTM's outputs.

The previously computed character level representation, the word embedding, and the POS-tag representation are concatenated for each token in the input sequence. These vectors are processed by a list of stacked Bi-LSTM layers *(C)* to produce a sequence of vectors that encode the tokens in the input sentence. The number of stacked layers is a parameter to be optimized, but during the challenge, all evaluations were made with a single layer.

Finally, a linear dense layer and CRF *(D)* are applied to transform each token vector into the corresponding most probable tag in the BILUOV tagging scheme.

2.2.1. Task A

Four independent instances with this architecture are trained, each one corresponding to an entity type. Doing so has two major benefits. First, this simplifies the entity encoding and decoding in the sentence since there is no need to differentiate the BILUOV tags between entities types in the same sentences. Second, it allows to model overlapping between entities with different types without additional considerations. On the other hand, splitting the task into four subproblems has the disadvantage of reducing the relevant training data for each model (i.e., the number of blank annotations per sentence increase), and the errors while predicting with each model are accumulated and propagated.

To mitigate the previous drawbacks, the character encoding layer is optionally shared across models and jointly trained. This allows benefiting from additional training examples for certain components of the models while continuing to model different hypothesis spaces in each model.

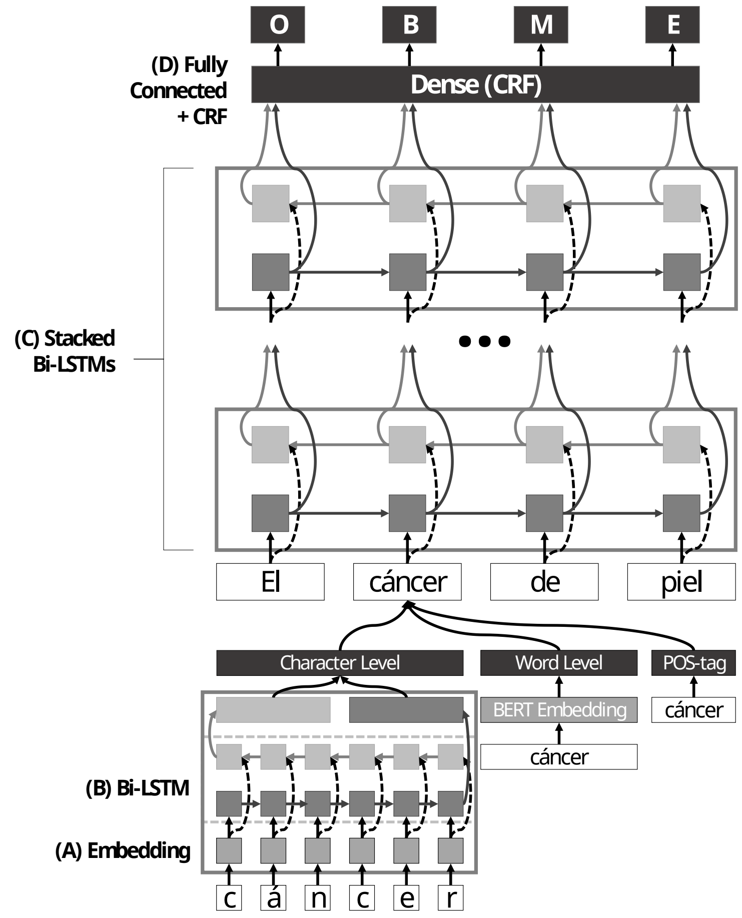


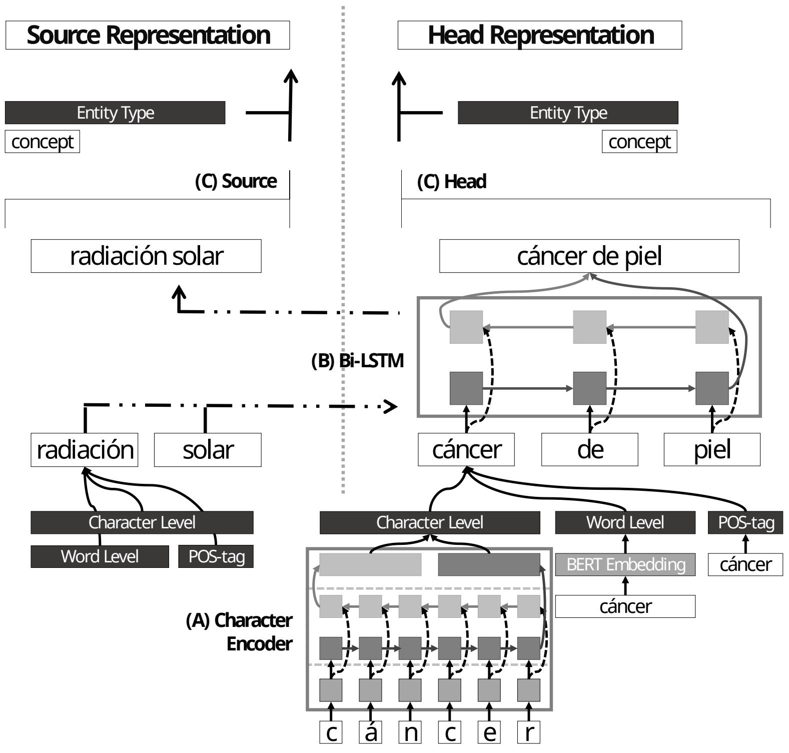
Figure 1: Summary of the *BILUOV tagger* architecture. An example sentence (*"El cáncer de piel"*) is processed by the *BILUOV tagger* that was created for the Concept entity type. The model outputs a sequence of tags (*O-B-M-E*), which is then decoded to obtain the mentioned Concepts (only "cáncer de piel" in this example).

The option to share layers is a parameter of the system, but in the context of the challenge, all tested models were jointly trained due to an empirically observed improvement on performance. Dropout layers are optionally added during training between each set pair of contiguous layers.

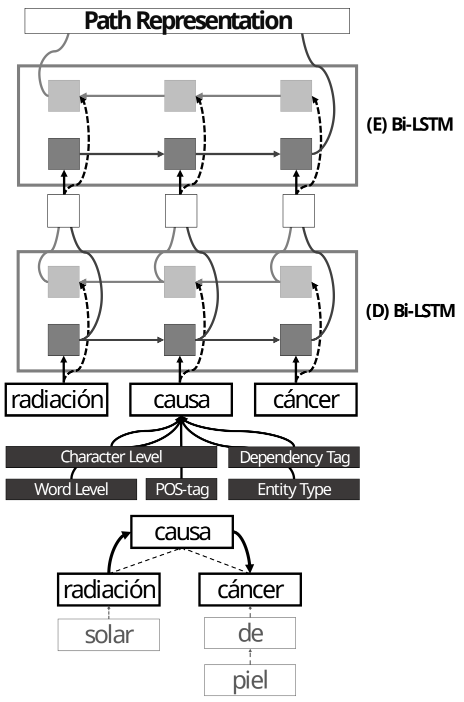
2.3. Relation Extraction Model

To solve the relation extraction task, the *pairwise classifier* architecture is used. Figure 2 summarizes the architecture used by this model. The architecture is divided in three stages: encoding of the source and head entities (Figure 2a), encoding of the path between the entities (Figure 2b), and representation combination and prediction (Figure 2c).

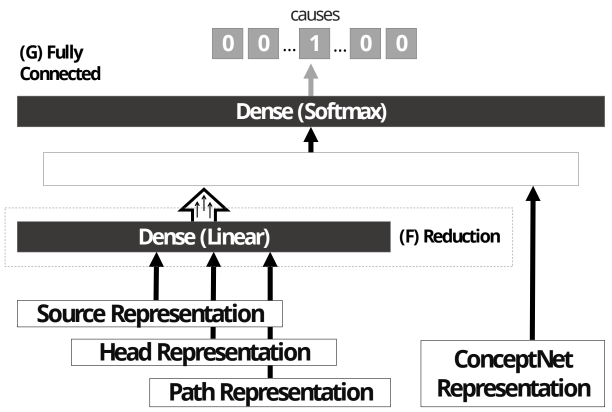
Similar to the *BILUOV tagger*, the *pairwise classifier* uses a character encoder layer *(A)* to transform the character representation of a token into a single vector. In case the four entity recognition models were trained sharing this layer, the previously computed weights are frozen



(a) Fragment of the *pairwise classifier* responsible of computing the source and head entities representations.



(b) Fragment of the *pairwise classifier* responsible of encoding the path between the source and head entities.



(c) Fragment of the *pairwise classifier* responsible of combining all representations to finally predict the most probable relation mentioned about the source and head entities.

Figure 2: Summary of the *pairwise classifier* architecture.

and reused here. In other case, a new character encoder layer is created and trained.

This architecture makes use of three Bi-LSTM layers to encode the tokens and produce intermediate representations that capture dependencies between pairs of entities. The first Bi-LSTM *(B)* consumes the sequence of tokens that made up the source entity (and head entity) to produce a compact representation for the entity. The entity type representation of the source and head entities are concatenated to the previously obtained vectors *(C)*. The second Bi-LSTM *(D)* processes the tokens in the path between the source and head tokens (i.e., the LCA of their respective entity's tokens) over the dependency tree, and transform them into a hidden representation. Afterward, the third Bi-LSTM *(E)* consumes the hidden representations to produce a vector that encodes the whole path.

The encoded representation of the head and source entities and the path between them are concatenated. An optional linear dense layer *(F)* is included afterward, to make the concatenated vector match three times the size of the ConceptNet representation (if provided). This dimension reduction was applied in some instances looking for the ConceptNet representation to be considered as much as other features. The resulting vector is processed by a final linear dense layer *(G)*, that produces the logits for the most probable type of relation (if any) between the pair in question.

2.3.1. Task B

A single model with this architecture is trained to solve the task. All pairs of entities occurring in the same sentence are tested. The computation of the dependency path can optionally be skipped to consider only the information of the source and head entities. Since computing the path is an expensive task to perform, the run in the challenge that considers this information was unable to complete in time for the first and fourth evaluation scenario, and the baseline solution was used in that case.

The lack of relations between a pair of entities is modeled with an additional relation type. This label is the most representative in the whole training collection since just a few entities are related in comparison to all the possible pairs. To mitigate the unbalance of the obtained dataset, we optionally employed a class-oriented weighting scheme during the training phase. This way, the model gets to "pay more attention" to samples from an under-represented class.

2.4. Output Handling

The *BILUOV* tagger outputs a sequence of BILUOV tags. These tags stand for: ***B**egin*, to mark the start of an entity; ***I**nner*, to mark its continuation; ***L**ast*, to mark its end; ***U**nit*, to represent single token entities; and ***O**ther*, to represent tokens that do not belong to any entity. Additionally, the *o**V**erlapping* tag is used to deal with tokens that belong to multiples entities. For example, in the sentence *"El cáncer de estómago y de esófago causan problemas"*, the model that detects Concepts must output: *O-V-I-L-O-I-L-O-U*, resulting in the detection of *"cáncer de estómago"*, *"cáncer de esófago"*, and *"problemas"*.

The decoding process is handled in two phases, based on the methodology described by team UH-MAJA-KD in the previous edition of the challenge [7]. First, discontinuous entities

Table 1
Summary of low-level parameters used to train the concrete models' variations that were used in the challenge. The four variations use the same set of parameters.

Common params		**BILUOV tagger**		**Pairwise classifier**	
char_vocab_size	96	*num_labels*	6	*num_labels*	14
char_embedding_dim	100	*dropout*	0.0	*dep_repr_dim*	29
padding_idx	0	*stacked_layers*	1	*entity_repr_dim*	5
char_repr_dim	200			*subtree_repr_dim*	300
word_repr_dim	768			*token_repr_dim*	300
postag_repr_dim	19				
token_repr_dim	300				

Table 2
Summary of the high-level parameters of the four models that were trained and used in the challenge.

Model	**jointly**	**weight**	**cnet**	**path**	**reduce**
cnet	x	x	x		
deptree	x	x		x	
cnet-deptree	x	x	x	x	
only-bert	x	x	x		x

matching either one of following regular expression are extracted.

$$(VO^*)^+((I|O)^*L)^+ \qquad \Longleftarrow \| \Longrightarrow \qquad (B(I|O)^*)^+(O^*V)^+$$

Afterward, all remaining entities are considered to be continuous sequences of tokens. The encoding and decoding process was tested to measure the error caused by simply changing the representation. This resulted in an F_1 of 0.997, a recall of 0.998, and a precision of 0.995, in the training collection.

To solve Task A, the four *BILUOV taggers* are asked to produce the BILUOV sequence for each sentence. Each output sequence is decoded independently to obtain a set of entities. Afterward, all entities are merged into a single collection and reported as output.

To solve Task B, all pairs of entities that belong to the same sentence are built. Afterward, the *pairwise classifier* is asked to predict the most probable relation between each pair. All non-empty relations are collected and reported as output.

2.5. System Training

Four concrete variations of the previously described architectures were trained and pipelined. All of them share the parameters shown in Table 1. The difference between them is given by the selection of optional decisions. Table 2 summarizes the high-level configuration of each one of them.

The *cnet* model ignores the path between the source and head entities in the *pairwise classifier* and disables the reduction layer. The *deptree* model does not use the ConceptNet information and disable the reduction layer. The *cnet-deptree* model combines both strategies.

The *only-bert* model uses a simplified version of the *pairwise classifier* in which only the ConceptNet information and the BERT embedding vector of the source and head tokens are used. The path between them, their character embedding, POS-tag representation, and entity type representation are ignored. The already discussed reduction layer of the pairwise classifier is applied in this case.

Only the training collection provided in the challenge was used to train the models. The development collection was used during training as validation collection to avoid overfitting in the training collection. The collection of additional sentences was not used at all. For scenario 4, no particular considerations were taken into account (beyond what was done in the other scenarios). The use of ConceptNet as an external source of knowledge was motivated by the idea of achieving a better performance in unseen terms during training, and therefore, in non-health domains.

Sentences were tokenized and parsed using the python library *spaCy*[2] (v2.2.4). Both the dependency tree, lemmatization, and POS-tag information were extracted from there, using the `es_core_news_md` pretrained model. A pretrained BERT multilingual model[3] was used to precompute the embedding vectors for each token in the sentence. No fine-tuning process was applied. A single spacy token translates to multiple BERT tokens (BERT uses WordPiece tokenization [8]). The embedding vectors obtained from BERT were mapped to spacy's token by choosing the first BERT token that derives from the token of spacy. Averaging the vectors, instead of choosing the first one, produced a slightly worst performance.

All models were trained for 100 epochs at most, but in practice, they converged to their final accuracy in around 10 or 20 epochs. Models were implemented using the python library *PyTorch*[4] (v1.4.0). Experiments were run in a machine with the following stats: 8 core Intel Core i9-9900K (-MT-MCP-) CPU, speed/max: 3651/5000 MHz, cache: 16384 KB, and RAM: 64 GB.

3. Ensemble

An ensemble system is made of several low bias models whose predictions are combined to produce a more robust final prediction. Multiple considerations have to be applied when ensembling information extraction models. For example, the system must decide which annotations are being voted by a common set of models and which ones are not. In case of conflicting annotations, the prevalent one(s) must be selected. Afterwards, the system must also decide which voted annotations are good enough to be included. All of these decisions have an important role in producing the final output.

According to this, we defined a set of handcrafted rules that were pipelined to built an ensembling system as follows:

1. Given the collection of annotations per model, build the union set of all annotations while keeping track of which models voted for each annotation, i.e., the model included the annotation.

[2]https://spacy.io/
[3]https://github.com/google-research/bert (`bert-base-multilingual-cased`)
[4]https://pytorch.org/

2. Weight the votes given by each model to each annotation.
3. Compute an aggregated score of the quality of each annotation according to its votes.
4. Decide whether to keep the annotation or not and, in case of conflicting annotation, decide which ones to keep.

A classical voting system might assign a uniform weight to each voting model. Only the annotations that exceed a predefined count are reported and in case of conflict, only those that achieve the majority of votes. An expert system can be implemented by precomputing the importance per annotation type of each model and using it to weigh the votes. The votes emitted by the corresponding best model are weighted accordingly, and others valued as zero. Only the annotations with a non-zero vote are reported and in case of conflict, only those that achieved the best score.

On top of the handcrafted rules, a set of machine learning algorithms can be trained to ensemble all systems votes. The machine learning model predicts the probability of a labeled annotation to be correct, i.e., the probability that it belongs to a gold annotated collection, according to the votes each system gave to it. Two key aspects can be configured here: first, how each annotation is transformed into a feature vector and which model will handle it, and second, which concrete model architecture will be used.

In the context of the challenge, a probabilistic evolutionary search was performed to explore the ensemble configuration space. The search starts with a random sampling strategy, but as it evaluates more pipelines, it modifies an probabilistic sampling model so that pipelines similar to the best ones found are more commonly sampled. The fitness function is set to directly maximize the F_1 score of the resulting ensemble according to the reference collection. The reference collection was set to scenario 1 of training collection.

The configurations explored will not be covered in this paper. The four models shown in Table 2 were ensembled. The best found ensembling pipeline for scenario 2 and scenario 3 was to use a different Support Vector Machine (SVM) binary classifier for each different type of entity and relation. Only two models (*cnet* and *only-bert*) were ensembled for scenarios 1 and 4 due to not having the corresponding submissions of the remaining models (*deptree* and *cnet-deptree*) in these scenarios (due to time limitations). The best found ensembling pipeline was to use a single Logistic Regression classifier for all the entity types and relations.

4. Results

Two of the four trained models were submitted to the challenge: the *cnet* and *deptree* models. Additionally, the ensemble method described in Section 3 was applied to produce a third run. Table 3 shows the results obtained by the three runs in the challenge. Due to time issues, some scenarios were not submitted for some runs and instead the baseline implementation was used (highlighted with a "*b*" in the table). There was also an issue with the ensemble method in scenario 3 (Task B), which resulted in an ill-formed submission. The ensemble run obtained the best results out of the three runs, except for the incorrectly submitted scenario.

Table 4 summarizes the overall results obtained by the team's system in the challenge. It ranked fifth in all evaluation scenarios, with a tie for fourth place in the main evaluation scenario. The most promising results of the system were achieved in scenario 2 (Task A).

Table 3
Summary of the performance (F_1) achieved by the system across the three runs. The best score achieved in each scenario of the challenge is included for comparison purposes. Also, the baseline implementation provided by the challenge organizers is included. Runs that use the baseline solution are marked with "b".

System	1-main	2-taskA	3-taskB	4-transfer
top performance	**0.666**	**0.825**	**0.633**	**0.584**
run 1: cnet	0.552	0.792	0.306	0.367
run 2: deptree	0.395^b	**0.795**	**0.545**	0.138^b
run 3: ensemble	**0.557**	**0.795**	0.005	**0.373**
baseline	0.395^b	0.542^b	0.131^b	0.138^b

Table 4
Summary of the performance (F_1) achieved by each participant in the challenge. Teams are sorted according to their performance in scenario 1.

Team	1-main	2-taskA	3-taskB	4-transfer
Vicomtech	**0.666**	0.821	0.583	0.563
Talp-UPC	0.627	0.816	0.575	**0.584**
UH-MAJA-KD	0.625	0.814	0.599	0.548
IXA-NER-RE	0.557	0.692	**0.633**	0.479
UH-MatCom	0.557	0.795	0.545	0.373
SINAI	0.421	**0.825**	0.462	0.281
HAPLAP	0.395	0.542	0.316	0.138
baseline	0.395	0.542	0.131	0.138
ExSim	0.246	0.314	0.131	0.122

5. Discussion

Several models were trained and tested in the development collection though they were not run in the test collection and therefore not used in the ensemble run. Using BERT to embed the words caused the biggest improvement in performance in Task A. In fact, by simply using the BERT representation and a linear dense layer, competitive results are achieved in the development collection. This is not enough for Task B, in which only using the concatenated BERT representation of the source and head tokens and a linear dense layer results in very low performance.

Another big improvement in the performance of the system was observed when using class weighting to train the *pairwise classifier*. Penalizing the most represented classes (the lack of relation in this context) helped to avoid ill-trained models that maximize the training accuracy but without relating to the F_1 metric. Alternative training procedures need to be explored, such as solving a multi-label classification problem with threshold selection to avoid the inclusion of the lack of relation class. Directly optimizing the F_1 metric during the training process is another alternative to be explored.

The addition of ConceptNet information did not have the positive impact we expected. This might have been caused by limitations in representation (zero-one vector) and the way it was introduced in the model. However, we think that the use of this kind of information can be really valuable in tasks like this one, in which few reference examples are available. Future work needs to be done in this respect.

6. Conclusions

This paper describes the system presented by team UH-MatCom in the eHealth-KD challenge at IberLEF 2020. Several deep-learning models were trained and ensembled to automatically extract relevant entities and relations from plain text documents. The system achieved average results in the challenge, ranking fifth across all different evaluation scenarios. Due to resource issues, some promising strategies were not fully tested. Future work will be done to solve those limitations.

Both syntactic and semantic information was used. The use of external knowledge bases to boots the relation extraction task and transfer learning scenario did not performed as expected. More work needs to be done in this sense, to improve the performance in future editions of the challenge.

Acknowledgments

Funding: This research has been partially funded by the University of Alicante and the University of Havana, the Generalitat Valenciana (*Conselleria d'Educació, Investigació, Cultura i Esport*) and the Spanish Government through the projects LIVING-LANG (RTI2018-094653-B-C22) and SIIA (PROMETEO/2018/089, PROMETEU/2018/089).

References

[1] A. Piad-Morffis, Y. Gutiérrez, H. Cañizares-Diaz, S. Estevez-Velarde, Y. Almeida-Cruz, R. Muñoz, A. Montoyo, Overview of the eHealth Knowledge Discovery Challenge at IberLEF 2020, in: Proceedings of the Iberian Languages Evaluation Forum co-located with 36th Conference of the Spanish Society for Natural Language Processing, IberLEF@SEPLN 2020, 2020.

[2] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[3] J. Lafferty, A. McCallum, F. C. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001).

[4] A. Graves, J. Schmidhuber, Framewise phoneme classification with bidirectional lstm and other neural network architectures, Neural networks 18 (2005) 602–610.

[5] R. Speer, J. Chin, C. Havasi, Conceptnet 5.5: An open multilingual graph of general knowledge, 2017. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14972.

[6] L. Ratinov, D. Roth, Design challenges and misconceptions in named entity recognition, in: Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009), 2009, pp. 147–155.

[7] J. M. Alvarado, E. Q. Caballero, A. Rodrıguez, Uh-maja-kd at ehealth-kd challenge 2019 (2019).

[8] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., Google's neural machine translation system: Bridging the gap between human and machine translation, arXiv preprint arXiv:1609.08144 (2016).