This is a previous version of the article published in Journal of Biomedical Informatics. 2021, 116: 103716. https://doi.org/10.1016/j.jbi.2021.103716

Automatic Extension of Corpora from the Intelligent Ensembling of eHealth Knowledge Discovery Systems Outputs

Juan Pablo Consuegra-Ayala[a,*], Yoan Gutiérrez[b,c], Alejandro Piad-Morffis[a], Yudivian Almeida-Cruz[a], Manuel Palomar[b,c]

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

Abstract

Corpora are one of the most valuable resources at present for building machine learning systems. However, building new corpora is an expensive task, which makes the automatic extension of corpora a highly attractive task to develop. Hence, finding new strategies that reduce the cost and effort involved in this task, while at the same time guaranteeing quality, remains an open and important challenge for the research community. In this paper, we present a set of ensembling strategies oriented toward entity and relation extraction tasks. The main goal is to combine several automatically annotated versions of corpora to produce a single version with improved quality. An ensembler is built by exploring a configuration space in search of the combination that maximizes the fitness of the ensembled collection according to a reference collection. The eHealth-KD 2019 challenge was chosen for the case study. The submitted systems' outputs were ensembled, resulting in the construction of

*Corresponding author.
Email addresses: jpconsuegra@matcom.uh.cu (Juan Pablo Consuegra-Ayala), ygutierrez@dlsi.ua.es (Yoan Gutiérrez), apiad@matcom.uh.cu (Alejandro Piad-Morffis), yudy@matcom.uh.cu (Yudivian Almeida-Cruz), mpalomar@dlsi.ua.es (Manuel Palomar)

an automatically annotated collection of 8 000 sentences. We show that using this collection as additional training input for a baseline algorithm has a positive impact on its performance. Additionally, the ensembling pipeline was used as a participant system in the 2020 edition of the challenge. The ensembled run achieved a slightly better performance than the individual runs.

Keywords:

Ensemble Methods, Annotated Corpora, Information Extraction, Entity Recognition, Relation Extraction, Natural Language Processing

---

1. Introduction

In recent years, there has been an increasing interest in applying machine learning and Natural Language Processing (NLP) techniques to solve several types of complex tasks. Named entity recognition (NER), relation extraction, machine translation, and sentiment analysis are just a few well-known examples of these tasks [1, 2, 3, 4]. These and other NLP tasks have a considerable impact on society nowadays and machine learning techniques have been able to achieve good performance [5, 6]. Although systems based on handcrafted rules might be able to perform fairly well in some of these tasks [7], the use of machine learning models eliminates the need for expert knowledge when designing new rules, and this reduces the cost of solving new problems.

One of the major issues involved in applying machine learning technologies is that they require large collections of corpora for building learning models. In some cases, the corpora are a natural component of the problem and therefore can be easily obtained by collecting known examples. That is the case with more classical approaches, such as stock market predictions based on features computed from automatically collected data [8]. However, in other cases, the corpora must be

handcrafted to establish a training base for solving certain subtasks. This frequently occurs with information extraction problems, where specific annotations schemes are defined and then extracted from plain text [9].

These corpora are considered highly valuable resources nowadays, not only because they are highly demanded, but they are also difficult and expensive to build. Hence, techniques have been developed for reducing the complexity of building corpora. Some technologies assist the corpus annotation directly, making it easier for humans to manually build the corpus. Such systems have applied active learning techniques to train a tagging model [10]. The human annotator acts as an oracle that corrects the model predictions and supplies sensitive information when necessary. This approach helps to reduce the corpus construction time since annotators can focus on simpler pieces of annotation.

Even with computer assistance, the amount of human effort, the required expertise, and the time spent to build corpora, make this task an expensive one. Automatic corpus extension through bootstrapping techniques is an effective way to increment the corpus creation efficiency [11, 12]. Corpus bootstrapping denotes the process of (1) manually annotating a small subset of a corpus, (2) training a model to predict the annotations, and (3) using that model to automatically annotate the remaining collection. This process can be repeated incrementally as long as the required quality is ensured by some validation estimator. Even with low-performance systems, this strategy is valuable if error correction is faster than manual annotation.

As with any automatic processing technique, there is a trade-off between quality and efficiency that needs to be balanced. On the one hand, manual expert annotation of the corpus should achieve the highest quality at the expense of efficiency. On the other hand, automatic annotation allows a quicker annotation, but is sensitive to overfit the initial collection and not generalize well to unseen situations. The use of

fine-tuned machine learning models can help to address this issue, but there is no out-of-the-box solution.

1.1. Ensemble Methods

Ensemble methods are designed to address the low bias / high variance problem exhibited by most machine learning models, making them suitable for producing more robust classifications [13]. An ensemble model is made of several low bias models whose predictions are combined to produce a final prediction. The main assumption is that the combination of the low-level predictions will produce an output with a lower variance while keeping a low bias. Having a diverse set of low-level models is a key feature to achieve this [14].

The multi-hypothesis nature of ensemble models ensures that, if fine-tuned, it will perform better than any of the individual models in the general case. This also allows them to estimate the degree of confidence or quality of the predictions they output. Classic ensembling techniques include *voting* and *weighted-voting* [15], *boosting* [16], and *bagging* [17].

In a classification problem, *voting* produces as output the label that achieved the majority of votes, treating each low-level model prediction as a vote. *Weighted-voting* works just like voting, but each low-level model is assigned a weight that indicates the importance of its votes. The label with the highest cumulative score is returned as output. *Boosting* runs an iterative process where models are trained sequentially, each one trying to improve its performance in the training examples that the previous models performed the worst. During this process, each sub-model is also assigned a score that weights the importance of its prediction. *Bagging* trains each sub-model in a different selection (with replacement) of the original training examples. This way, a model with a high variance should produce trained models with high diversity.

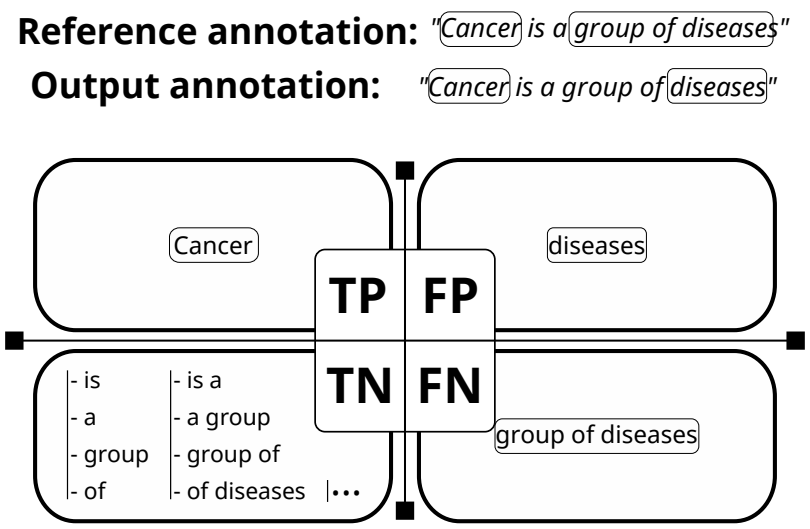

Figure 1: Classification of the annotations generated by a NER model according to a reference annotated version. **TP**, **TN**, **FP**, and **FN** stand for True Positive, True Negative, False Positive, and False Negative, respectively.

Alternatively, *feature bagging* works similarly by selecting a subset of the features instead of the training examples, causing correlated features to be analyzed separately in some sub-models.

While this works well for standard classification problems [18, 19], more complex corpora require additional factors to be taken into account. For example, information extraction tasks such as NER require a wider definition of what is being "voted" by each sub-model and what should be reported as output. The NER task consists of locating entities within the text by extracting spans of text and then assigning a label to them. Overlapping entities with different labels produced by two sub-model might imply that either the two entities are returned, or only one of them, or even neither of them. Figure 1 shows how the annotations generated by a NER model are classified into four categories according to a reference annotated version.

Some models might have better accuracy at predicting a certain type of entity

than others, and therefore, the weighted-voting should not be done uniformly across all the different types. Advances in this area suggest that models should be weighted independently for each entity class in order to obtain a better performance [20].

1.2. Scenario for Ensembling

Many information extraction competitions (e.g., eHealth-KD[1], IberLegal[2], and CAPITEL[3], at IberLEF 2020[4]) are hosted every year to encourage the research community to produce new state-of-the-art results in the area. These competitions allow the coordination of different systems to solve the same problem by defining an evaluation corpus and a set of input and output specifications shared by all systems. This turns each competition into a source of different annotated versions of their corpora. By ensembling these results, the original corpus of the competition can be extended and used as additional training resources in future versions of the competition, which may have a positive impact on the performance of the participating systems.

The eHealth-KD 2019 (or eHealth-KD v2 [21]) corpus, is a noteworthy example of information extraction related corpora available in the literature [22]. It is made of three types of annotations: entities, relations, and attributes, all of them manually annotated in a collection of Spanish sentences from the medical domain. The corpus was used as the main evaluation scenario for the eHealth Knowledge Discovery Challenge at IberLEF 2019 (eHealth-KD 2019 for short) [23]. In the challenge, participants were asked to automatically annotate a set of plain text sentences with

[1] https://knowledge-learning.github.io/ehealthkd-2020/
[2] https://temu.bsc.es/iberlegal/
[3] https://sites.google.com/view/capitel2020
[4] https://sites.google.com/view/iberlef2020/home

the corresponding annotation scheme (attributes were excluded from the scope of the challenge). Different systems were submitted and ranked according to their performance in a gold standard collection. The participants were provided with a set of 8 700 additional untagged sentences that were merged with the actual test collection. As a result, a collection with several different versions of the same automatically annotated sentences was obtained. A critical ensembling of these versions can be used to automatically extend the previous corpus with a higher performance than simply using the winning model.

This paper's goal is to design and validate an ensembling strategy to automatically extend Knowledge Discovery corpora, particularly oriented toward NER and relation extraction tasks, from the systems' outputs. This type of corpora extension resembles an improved way of doing corpus extension through bootstrapping: instead of using a single model to annotate the unlabeled sentences of a corpus, the predictions (outputs) of several models are ensembled to produce a single, more robust, annotated version. In this line, the eHealth-KD 2019 challenge was taken as a case of study, resulting in the automatic annotation of a collection of 8 000 sentences [24]. The results are compared against both classical voting systems as well as automatic learning models according to their performance in the original evaluation scenarios. An upper bound of the performance that can be achieved by only merging the submissions of the challenge is presented. Additionally, the ensembling pipeline presented in this paper was used as a system in the 2020 edition of the eHealth-KD challenge [25] by one of the participating teams [26]. The team designed several deep-learning models to solve the challenge and ensembled them to produce a final submission.

The specific contributions of this research are as follows:

- The introduction of ensemble methods for corpora extension as a better al-

ternative to classic corpus bootstrapping. The main advantages are reducing the bias towards a particular architecture or set of rules and providing richer information on the quality of the final corpus.

- The definition of an ensembling pipeline that works directly on the system outputs rather than on the systems themself. The main advantage is allowing the integration of any group of systems, regardless of their architecture, and non-programmatic annotation sources, i.e., human annotations.

- The automatic extension of the *eHealth-KD v2* corpus [21], a manually annotated corpus of Spanish language sentences in the health domain. The newly annotated sentences are publicly available as the *eHealth-KD 2019 ensembled corpus* [24]

- Comparison of the performance of several ensembling techniques in a particular corpus and the study of the additional quality metrics obtained from extending the corpus using ensembling methods rather than classical bootstrapping methods.

The remainder of the paper is organized as follows. Section 2 provides relevant details on the eHealth-KD challenge and its corpus. Section 3 presents the ensembling pipeline proposed in this paper, along with the different ensembling techniques it includes. Section 4 shows the results of evaluating the ensembler, along with the most promising strategies found. Additionally, it provides some statistics about the collection of sentences that resulted from ensembling the eHealth-KD challenge's submissions. Section 5 discusses the most relevant aspects of the whole research. Finally, Section 6 presents the conclusions, along with future lines of work.

2. Related Work

This section provides further information on the eHealth-KD challenge and its corpus. The methods presented in this paper were directly applied and evaluated in the resources provided by the challenge.

2.1. eHealth-KD Challenge and Corpus

The eHealth-KD challenge hosted at IberLEF proposes a set of resources and evaluation scenarios 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. The F_1 score is used as the main evaluation metric. Both partial matching of annotations and label misclassification are considered.

Figure 2 shows an example annotation of three sentences. Each entity annotation is identified by its span of text, i.e., the position that the entity holds in text. Each relation annotation is identified by the pair of entities it relates. Four types of entities are defined: Concept, Action, Predicate, and Reference. The annotation scheme has also 13 types of relations. Entity and relation representations match the specification described in Section 3.2.

The challenge had the participation of 10 different contestant systems. Three evaluation scenarios were developed, each one oriented toward entity detection, relation extraction, or both tasks. The *main scenario* (Scenario 1) asked participants to output both the entities and relations present in a collection of plain text sentences. The *entity detection task* (Scenario 2) asked participants to find and label only the relevant entities present in a collection of sentences, and, therefore, relation annotations were ignored. For the *relation extraction task* (Scenario 3), both the sentences and entities were given to the participants for them to output the relations occurring between

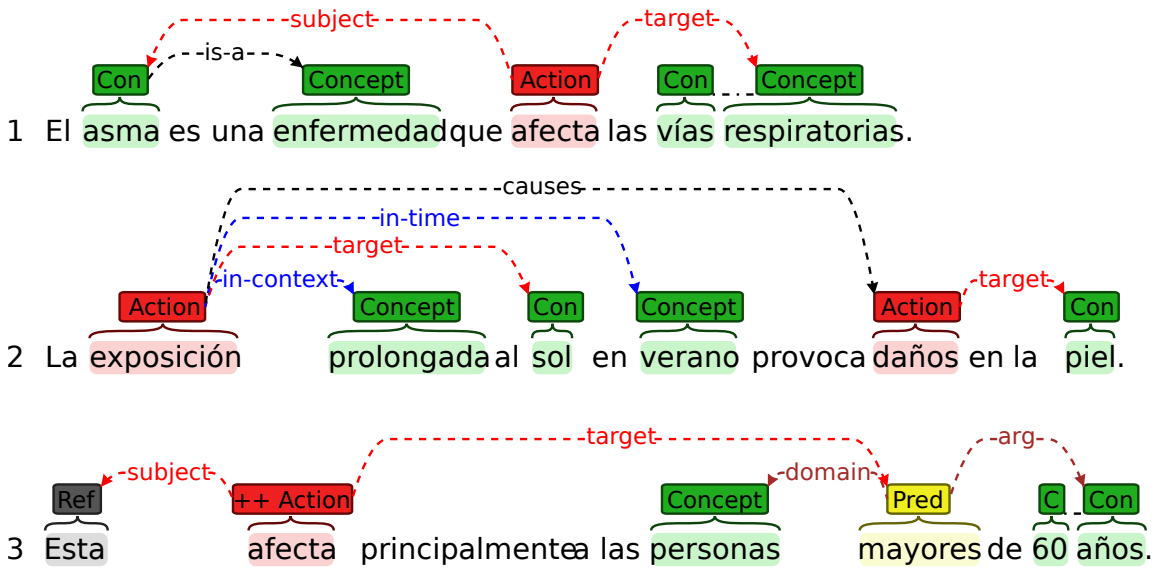

Figure 2: Example annotation of three Spanish sentences. The annotation shows the most relevant entities and relations defined.

them. A total of 32 successful submissions were made to Scenario 1, 48 to Scenario 2, and 48 to Scenario 3.

Each scenario is made of 100 sentences, for a total of 300 sentences with a human-annotated equivalent. A set of 8 700 additional sentences was added to the evaluation sentences of Scenario 1. Though these sentences were not used to measure the performance of the participant (since no reference annotation was available for them), this enforced participants to provide an automatically annotated version of the sentences while also discouraging them to manually annotate the test collection to cheat the evaluation system.

The ensembling pipeline presented in this paper can be applied to make use of the different versions[5] of the 8 700 non-gold-annotated sentences to automatically extend

[5] https://github.com/knowledge-learning/ehealthkd-2019/tree/master/data/

the corpus. These versions consist of the submissions made by the contestants to the additional sentences included in the test collection.

3. Method

This section presents our ensembling pipeline (also referred to as "the ensembler"), its input, the set of algorithms and parameters it has available to select from, the optimization algorithm used to find the best configuration, and an upper bound of ensemble-based method's performance. Section 3.1 provides a general overview of the ensembling pipeline. Section 3.2 describes the input expected by the ensembler, along with the requirements imposed on it. Section 3.3 presents the pool of algorithms that the ensembler has available to select from, along with their parameters. Section 3.4 describes how to build a fake annotation system, called the "Oracle Ensembler", to obtain an upper bound of the performance obtained by ensembler systems based only on other systems' outputs. Finally, Section 3.5 defines an agreement metric to enhance the information about the collections that result from the ensembling pipeline.

3.1. Overview

The ensembling pipeline takes as input a collection of different annotated versions of a document and produces a single annotated version of it (d^e). A key feature of the ensemble is its ability to work directly with different versions of a document rather than with the program or individual who generated it. The particularities of the input expected by the ensembler are described in Section 3.2.

`submissions/all`

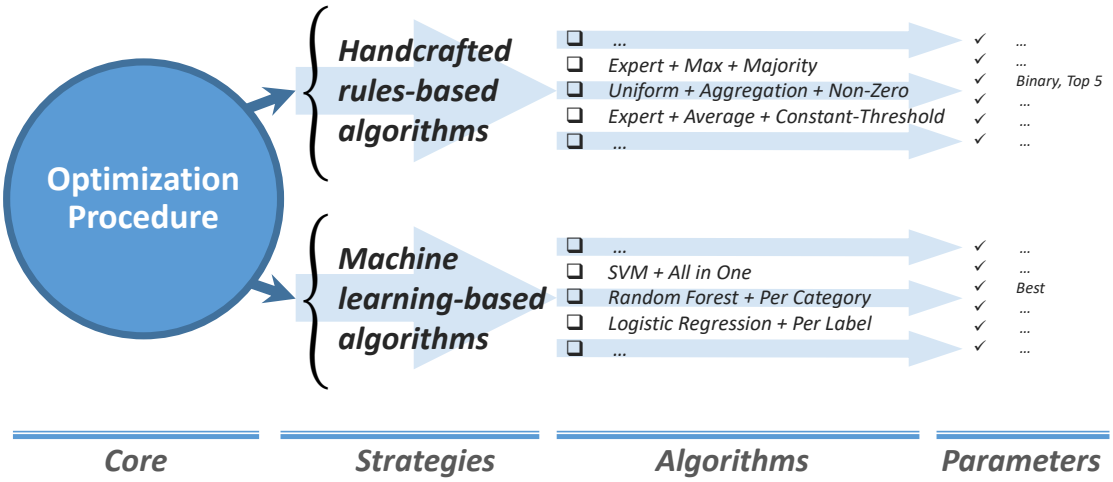

Figure 3: Overview of the ensembling pipeline. The ensembler's core is an optimization procedure that searches for the best configuration according to a reference collection.

The ensembler consists of a pool of algorithms and parameters from which the "best configuration" is selected. The best configuration is defined as the combination of algorithms and parameters that generates the ensembled document with the highest score according to a predefined evaluation metric F and a reference annotated document (d^*). Figure 3 shows a high-level definition of the ensembler.

The F_1 metric is the standard fitness function to be used due to the information extraction nature of the NER and relation extraction tasks. However, any metric that scores the quality of a document d^e with respect to the reference d^* can be used. For example, additional considerations, such as partial match between entities, can be taken into account depending on the use case. A macro-F_1 metric can be applied to give each annotation type the same weight (even when some of them are

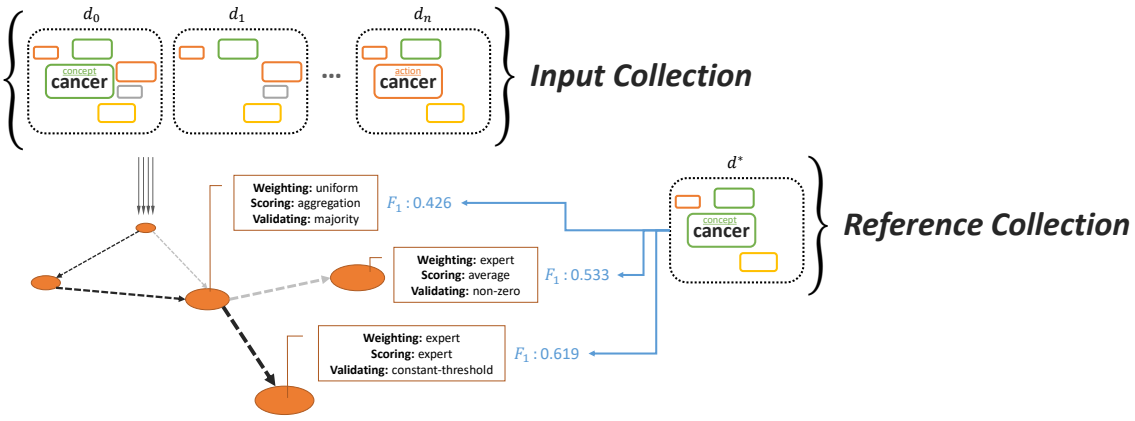

Figure 4: Optimization search performed on top of the pipeline configuration space. Each d_i, with $i \in [0, n]$, stands for an annotated version of the documents; d^* stands for the reference version of the document.

less represented in the whole collection). To apply these metrics, annotations are categorized into four classes, as shown in Figure 1. Precision is computed according to the number of True Positives and False Positives, which are the number of correct and spurious annotations, respectively. Recall is computed according to the number of True Positives and False Negatives, which are the number of correct and missing annotations, respectively.

To find the best configuration, a probabilistic evolutionary search is performed on top of the space of possible configurations (algorithms and parameters). The search starts with a random sampling strategy, but as it evaluates more pipelines, it modifies a probabilistic sampling model so that pipelines similar to the best ones found are more commonly sampled. Figure 4 illustrates the optimization procedure. The fitness function is set to directly maximize the F score of the resulting ensemble according to the reference collection.

The ensembler's pool of algorithms is divided into two categories: (1) the hand-crafted rules-based algorithms and (2) the machine learning-based algorithms. The

first group contains the algorithms that apply a sequence of handcrafted rules to merge the different versions of a document into a single ensembled version. The other group contains the algorithms that train one or more machine learning models to predict, based on the annotations made in the input collection of versions, which annotations to include in the ensembled version. Section 3.3 provides the full specification of the algorithms that are considered by the ensembler.

3.2. Input

Given a set of different annotated versions of a document $C = \{d_0, \cdots, d_n\}$, $n > 1$, the goal of our ensembling system is to produce a single annotated version d^e of the document. Each d_i, with $i \in [0, n]$, is an annotated version of the document, also referred to in this paper as a system, in reference to a system that generated the document as output. An annotated document, in the scope of this paper, consists of a set of annotated sentences $\{s_0, \cdots, s_p\}$. The ensembler explores a space of parameters in search of the configuration that produces the best ensembled document according to a reference collection d^*. That is, the reference annotated version of some sentences is provided, and their automatically ensembled versions are evaluated with respect to them using a fitness function F.

Only entity and relation annotations are considered by our ensembler. An entity is represented by a pair $\langle spans, type \rangle$, where *spans* is a sequence $\{\langle b_0, e_0 \rangle, \cdots, \langle b_n, e_n \rangle\}$ of spans of text in the sentence, with each $\langle b_i, e_i \rangle$ representing the beginning and end of the *i-th* span, and *type*, a label that indicates the type of entity it was assigned. Each entity belongs to a single sentence, i.e., all their spans of text are contained in the same sentence. There might be overlapping between entities spans, but it is assumed that there is no full overlap between entities of the same type, i.e., there is no pair of entities with the same type that share the exact same span of text. Figure 5

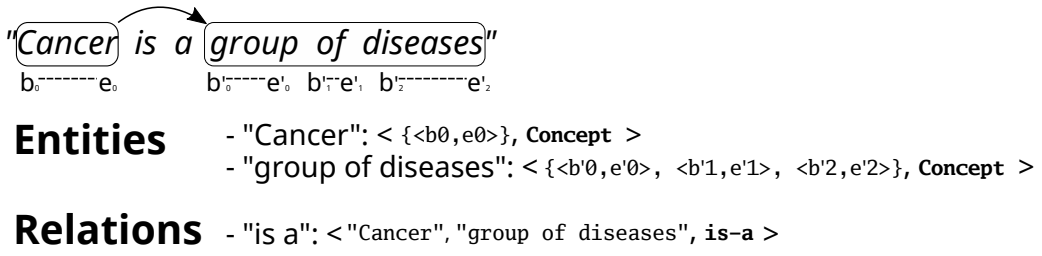

Figure 5: Summary of the representation of entities and relations expected by the ensembler.

shows a sample sentence with annotated entities and their representation.

A relation is represented by a triplet $\langle E_s, E_h, type \rangle$, where E_s and E_h are the source and head entities of the relation respectively, and *type*, a label that indicates the type of relation annotated. Relations occur only between entities that belong to the same sentence, i.e., there are no edges between entities in different sentences. Figure 5 shows a sample sentence with an annotated relation and its representation.

3.3. Algorithms and Parameters

Two different types of algorithms are explored by the ensembler's optimization procedure as presented in Section 3.1:

- Those that pipeline a set of handcrafted rules to combine the collection of documents C (as described in Section 3.3.1).
- Those that train machine learning models to predict which annotations from the collection of documents C to include (as described in Section 3.3.2).

3.3.1. Handcrafted Rules-based Strategy

The ensembling pipeline has available a set of handcrafted rules-based algorithms to ensemble the collection of documents C. These algorithms sequentially apply a combination of manually designed rules to merge the output of several information

extraction systems. First, the ensemble determines, for each annotation in the union of all input documents, which systems include it and which ones do not; we say that systems "vote" for the annotations included in their output. Secondly, in the case of conflicting annotations, the prevalent one(s) must be selected. Afterwards, the ensembler must decide which proposed annotations are good enough to be included. All of these decisions have an important role in producing the final output.

According to this, handcrafted rules-based algorithms, grouped into the next Strategy Development section, are organized as follows:

1. **Vote Aggregation:** Given the collection of annotated documents C, the union set of all annotations is built while keeping track of which system voted for each annotation. It is said that system i voted for an annotation r if r is included in document d_i. More details below in Section 3.3.1.

2. **Vote Weighting:** The votes given by each system to each annotation are weighted. More details below in Section 3.3.1.

3. **Annotation Scoring:** An aggregated score of the quality of each annotation is computed according to its votes. More details below in Section 3.3.1.

4. **Validation and Selection:** A decision is made as to whether the annotation is kept and, in the case of conflicting annotations, a decision is reached on which one to keep. More details below in Section 3.3.1.

Figure 6 summarizes the previous architecture. Several ensembling strategies can be generated by assigning algorithms to all previous steps. For example, a classical voting ensembler assigns a uniform weight to each voting system. Only the annotations that exceed a predefined count are reported and in the case of conflict, only those that achieve the majority of votes. Another example is an expert ensembler,

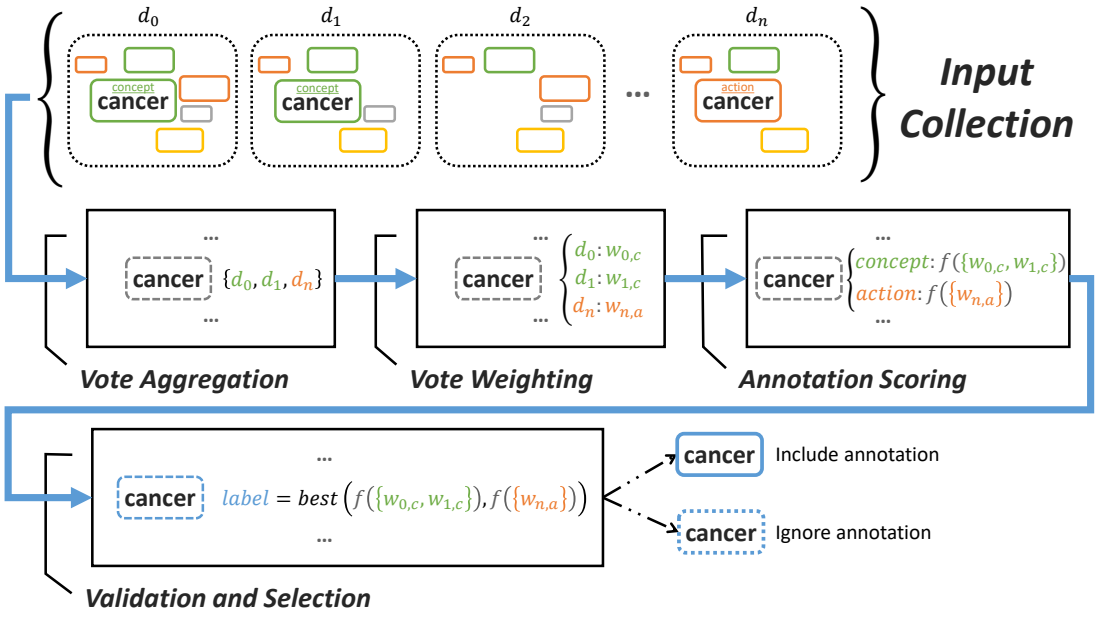

Figure 6: Summary of the handcrafted rules-based strategy. All annotations are aggregated and, afterward, a pipeline comprising three tasks decides which annotations to include in the ensembled collection. Each d_i, with $i \in [0, n]$, stands for an annotated version of the documents; each $w_{i,j}$ stands for the weight assigned to system i at predicting annotations of type j; f stands for a function that scores the quality of annotation type j given all the active weights for type j; *best* stands for a function that selects the annotation type that was scored the highest (arbitrarily chosen in the case of a tie) and decides whether it is high enough to be included in the output collection.

which can be implemented by precomputing the importance per annotation type of each system and using it to weigh the votes. The votes emitted by the corresponding best system are weighted accordingly, and others valued as zero. Only the annotations with a non-zero vote are reported and in the case of conflict, only those that achieved the best score.

Strategy Development

The next sections provide details on each stage that takes part in the pipeline of handcrafted rules-based algorithms.

Vote Aggregation. The ensembler has two ways to aggregate the annotations:

- *binary decisions*, in which the ensembler considers each annotation with a fixed label as a possible candidate.

- *non-binary decisions*, in which the ensembler has to select only one label from all the proposed labels for the same annotation.

The vote aggregation phase has a critical impact on the final ensemble collection. Depending on whether binary decisions are made, the output will have full overlapping annotations or not. Binary ensembling can build collections more similar to the reference collection than non-binary ensembling since the output of the latter is a subset of the former. On the other hand, non-binary ensembling should be easier to tune since the final output is relaxed across all versions, i.e., at most one annotation will be reported per overlapping annotation, instead of multiple ones.

Vote Weighting. Votes made by each system are weighted in two ways:

- The *uniform voting* approach is the most basic strategy, in which every vote is considered to have the same weight. This strategy, combined with others later in the pipeline, can produce some classic ensembling algorithms. For example, by adding up the weighted votes per label of each annotation, assigning to the annotation the label that most systems voted for, and keeping it only if the score was greater than 50% of the total number of systems, a classic voting ensembler can be built.

- An *expert-based weighting* approach exploits the fact that some systems have better performance than others (according to the reference collection). The quality of each system's votes is estimated using the fitness function F. Since systems might perform differently at each type of annotation, the weighting function depends not only on the system identifier but also on the type of annotation that is being voted. Each weight $w_{i,j}$ is computed by measuring the performance of system i according to function F while considering only the annotations of type j in both the reference and submitted collections.

Annotation Scoring. Once the collection of weighted votes per label has been built for each annotation, all candidate labels are assigned a score. This score is meant to capture the degree of confidence given by the voting systems, indicating whether the label should be included. Several strategies are implemented:

- *aggregation scoring*, in which all weighted votes are added up.
- *average scoring*, in which weighted votes are averaged.
- *max scoring*, in which the maximum weighted vote is returned.
- *expert scoring*, in which the weight of the highest weighted system is returned if voted by the same, or zero otherwise.
- *union scoring*, in which the maximum score (1) is returned if at least one system voted.

The scoring can be done relative to the voting systems only or to the whole collection of systems. In the former case, the fact that a system did not vote does not influence the resulting score, while that is not for the latter case. This is valid,

for example, for *average scoring*, in which added weighted votes can be divided either by the number of voting systems or the total number of systems. Another possible modification to the previous strategies is to consider only the top k best-weighted votes, for any value k between 1 and the total number of systems.

Validation and Selection. Depending on whether the pipeline decided to use binary or non-binary decisions for classification, each annotation will have a single candidate label or a list of them, respectively. In the first case, annotations with different labels are being treated as different annotations and therefore validated independently. In the second case, overlapping annotations have been grouped, and each candidate label has been scored. The selected label will be the one that was scored the highest (arbitrarily chosen in the case of a tie), resulting in a single annotation per group of overlapping annotations. In either case, the final candidate label has to be validated to decide whether it is good enough to be included in the ensembled collection.

Different strategies are used:

- *non-zero validation*, in which an annotation will be reported if the label selected for it has a positive score.
- *majority validation*, in which an annotation will be reported if the label selected for it has a score greater than half the total number of systems.
- *per-label-threshold validation*, in which the score for a label has to exceed a predefined (per label) threshold for an annotation with that label to be reported.
- *constant-threshold validation*, which works similarly to the *per-label-threshold validation* but uses the same threshold for every type of label.

All the previous strategies are particular cases of the *per-label-threshold validation*, in which the thresholds are set to zero for *non-zero validation*, to half the number of systems for *majority validation*, or to parametric constant value for *constant-threshold validation*. Defining these particular cases is useful to improve the performance while searching the space of possible configurations, as described in Section 3.1. Some of these validations are oriented toward certain strategies previously applied. For example, the *majority validation* intuitively should have a better performance when combined with *uniform weighting* and *aggregation scoring*, since that represents a classic voting ensembler.

Optimization Parameters

The task of deciding which strategies and parameters combine the best is given to the optimization search algorithm as presented in Section 3.1.

Table 1 summarizes the space of handcrafted-rules based algorithms to be explored. Strategies are chosen from categorical values. Their parameters are requested to the space sampler, which is then updated according to the performance achieved by the pipeline.

In some case, the input systems are highly correlated or might belong to the same case study. For example, in the case of the eHealth-KD challenge, each participant might submit multiple systems, which in some cases share the exact same architecture but with small fine-tuned parameter changes. Having multiple systems biased towards the same type of annotations might (or might not) have a bad influence on the final ensemble. To deal with this, the *best* parameter was added to the optimization space. This parameter controls whether to consider only the best performing system of each category (e.g., the best performing submission of each participant) or all of them. The inclusion of the *best* parameter in the building phase allows the optimization algorithm

Phase	Strategy	Parameters
Building	voting	binary^{b}
	submissions	best^{b}
		top^{k}
		selection^{l}
Weighting	uniform	
	expert	
Scoring	aggregation	top^{k}
	average	top^{k}, strict^{b}
	max	
	expert	discrete^{b}
	union	
Validating	non-zero	
	majority	
	constant-threshold	threshold^{t}
	per-label-threshold	thresholds $^{t^*}$

Table 1: Summary of the handcrafted rules-based algorithms and their parameters to be explored. Each tag b, k, l, t and t^* stands for Boolean, integer, list, continuous, and vector type, respectively.

to turn it off if proved convenient. Additionally, the *top* and *selection* parameters were added to ensemble the top best-performing systems only or a selection of them.

3.3.2. Machine Learning-based Strategy

Aside from the handcrafted rules, the ensembling pipeline has also available a set of machine learning-based algorithms to ensemble the collection of documents C. Machine learning models are trained to predict the probability of a labeled annotation to be correct, i.e., the probability of it belonging to a gold annotated collection, according to the votes given to it by each system. Two key aspects are configured here: first, how each annotation is transformed into a feature vector and which model handles it, and second, which concrete model architecture is used. Figure 7 summarizes the general structure of this strategy.

Strategy Development

Annotation Representation The main features for representing any labeled annotation comes from which systems voted for it. That means that part of the feature vector will hold the information of whether a system voted or not for the labeled annotation to be included. This is captured with a vector $v = \langle v_0 \cdots v_n \rangle$, n being the total number of systems, in which $v_i = 1$ if system i voted to include the annotation with that label, and $v_i = 0$ otherwise. Instead of using binary weighting, the importance of the corresponding voting system can also be used, this is, $v_i = w_{i,j}$ if system i voted to include the annotation with label j, with $w_{i,j}$ being the system weight. Additionally, a one-hot encoded vector l is built to encode the selected label, this is, l is a vector whose size equals the number of possible labels, and in which all component equals 0 but the one that corresponds to the selected label. Using this information, the feature vectors and models can be handled in one of the following ways:

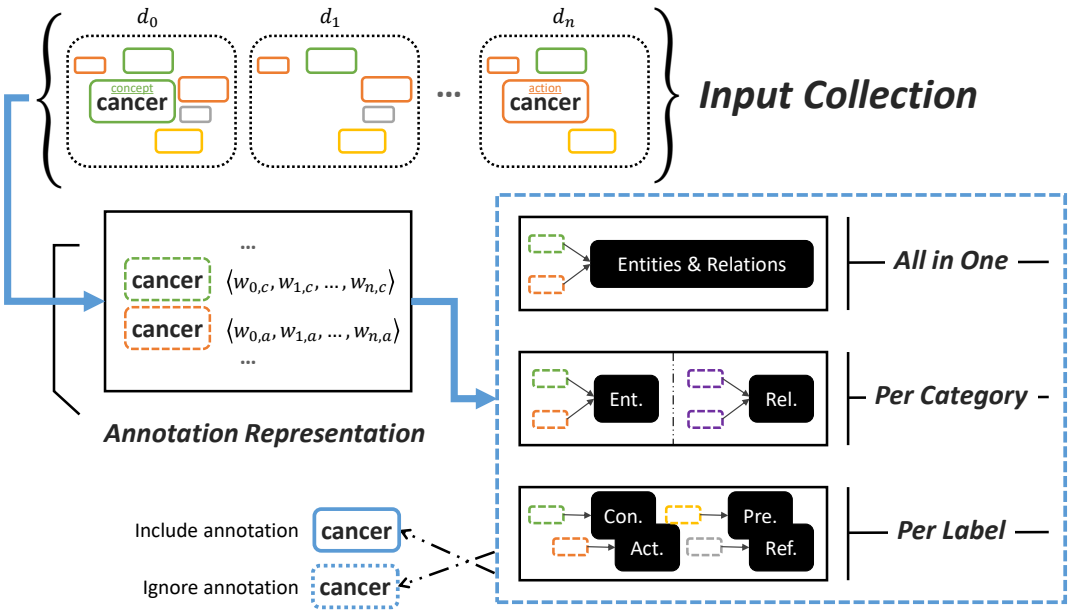

Figure 7: Summary of the machine learning-based architecture. Each annotation is transformed into a feature vector and, afterward, a machine learning model computes the probability of it belonging to a gold annotated collection. Each d_i, with $i \in [0, n]$, stands for an annotated version of the documents; each $w_{i,j}$ stands for the weight assigned to system i at predicting annotations of type j.

- *all in one model*, in which a single model is trained to handle every labeled annotation. Each feature vector is built by concatenating vector v and vector l.

- *per category model*, in which two models are trained, one to handle entities and one to handle relations. Each feature vector is built by concatenation vector v with vector l (using the entity or relation labels accordingly).

- *per label model*, in which a model is trained for each type of label. Each feature vector is fully composed of vector v.

Model Architecture In the context of this paper, three classic *scikit-learn*[6] classifiers were considered: Random Forest, SVM, and Logistic Regression. The default parameters of the respective models in the `0.22.2.post1` version of the library are used. The three architectures were evaluated as reported in Section 4. In general, any machine learning classifier can be registered to solve the prediction task. The task consists in a binary classification problem in which the feature vector of each labeled annotation is used to predict whether to output the annotation or not.

Optimization Parameters

Table 2 summarizes the space of machine learning-based algorithms to be explored by the optimization procedure described in Section 3.1. Aside from the weighting and training modes and the model architecture, an additional parameter "best" was included. This parameter controls whether to use all systems or only the best one per category. Binary voting is used in this situation for vote representation to fit the learning systems requirements.

[6]https://scikit-learn.org/

Phase	Strategy	Parameters
Building	submissions	best^b
Weighting	uniform	
	expert	
Model	SVM	
	Random Forest	
	Logistic Regression	
Training	all in one	
	per category	
	per label	

Table 2: Summary of the space of machine learning-based algorithms to be explored. Tag b stands for boolean.

3.4. Upper Bound

Though ensemble methods are a great way to improve the performance of individual systems, the diversity of those systems plays a key role in the quality of the final results. There is a limit to what an ensemble method can achieve based on its subsystem. To obtain an upper bound of the performance that the best ensembling method can achieve, an oracle ensembler can be built.

The oracle ensembler consists in an algorithm that always makes the correct choice on which voter to trust. That means that for a given annotation, even if only one system voted for the inclusion of the annotation, or only one system voted for the exclusion of the annotation, the oracle ensembler will trust it if doing so is the right decision according to the reference collection. As a consequence, every annotation present in the reference collection that was identified by at least one system will be included. On the other hand, annotations that were not detected by any subsystem will not be included. Only if all subsystems vote for the inclusion of an annotation not present in the reference collection, a spurious annotation will be reported.

It only makes sense to build such an ensemble for comparison purposes, since it requires a gold annotated sentence to make a prediction on that same sentence, which means that it will solve a problem already solved. Section 4 shows the obtained upper bound of the performance of any ensembling system based only on the previous submissions of the eHealth-KD challenge can obtain.

3.5. Quality Measures

The use of ensembling methods to produce a single merged version from different versions of a document allows computing an agreement metric to further enhance the information available on the quality of the final result. Though this condition is not determinant in the quality of the resulting ensemble (since part of the performance

of ensemble methods comes from the diversity of their subsystems), the agreement between annotations estimates the confidence in the final document. Sentences that have a higher agreement are probably easier to annotate. That means that a corpus made of those sentences is more likely to be correctly annotated since subsystems may have achieved better performance on their own.

To measure the agreement in a sentence, the following equation is proposed:

$$agreement(anns) = \frac{\sum_{i \in anns} | \, votes(i) \, |}{N \cdot | \, anns \, |}$$

where *anns* is the set of annotations in the sentence, *votes* returns the systems than voted for the annotation, and N is the total number of systems. This metric value ranges from 0 to 1. The highest value is achieved when every annotation received the maximum number of votes (N).

4. Results

The methods described in this paper were directly applied and evaluated in the resources provided by the eHealth-KD challenge. The eHealth-KD challenge provides a collection of 9 000 sentences automatically annotated by each participant system of the challenge's 2019 edition.[7] The gold-annotated version (manually annotated by experts) of 600 of those sentences is also available. Those manually annotated sentences are divided into two packs of 300 sentences, used to evaluate the submissions

[7] https://github.com/knowledge-learning/ehealthkd-2019/tree/master/data/submissions/all

of the challenge's 2019 [8] and 2020[9] editions, respectively.

In the context of this paper, the previous collections of sentences are used in the following way:

- **Reference Collection [**300**]** The sentences used to evaluate the submissions of the challenge's 2020 edition. Both the gold-annotated version and the submissions of each participant (to the challenge's 2019 edition) are available. Conveniently, these sentences were selected from different agreement ranges (as described in Section 3.5), which means that some of them (100) had high agreement, others (100) low agreement, and the remaining (100) an in-between agreement. [27]

- **Validation Collection [**100**]** The sentences used to evaluate the submissions to the main scenario (Scenario 1) of the challenge's 2019 edition. Both the gold-annotated version and the submissions of each participant are available.

- **Blank Collection [**8 400**]** The remaining sentences, for which the gold-annotated version is not provided. Only the submissions of each participant to the challenge's 2019 edition are available.

Three different aspects of the ensembler described in this paper were evaluated.

[8] Available in the testing collection (https://github.com/knowledge-learning/ehealthkd-2019/tree/master/data/testing) of the eHealth-KD 2019 challenge (scenarios 1, 2, and 3).

[9] Available in the testing collection (https://github.com/knowledge-learning/ehealthkd-2020/tree/master/data/testing) of the eHealth-KD 2020 challenge (scenarios 1, 2, and 3).

- **Quality of the Ensembled Corpus** First, the ensembler was used to produce the *eHealth-KD 2019 ensembled corpus* [24], a collection of 8 000 automatically annotated sentences, from the Blank Collection's sentences. Of these, 3 000 were published in the context of the eHealth-KD 2020 challenge as an additional source of development data[10]. The Reference Collection was used as a reference collection for tuning the ensembling pipeline. Section 4.1 presents some statistics about the final corpus, including an analysis of quality metrics.

- **Quality of the Ensembling Algorithms** Secondly, the performance of the most interesting ensembling strategies is evaluated in the Reference and Validation Collections. Section 4.2 presents these results in a greater detail.

- **Quality of the Ensembler Optimization** Finally, the performance of the ensembler's optimization procedure is evaluated. With this goal in mind, the ensembler was used as a system to participate in the eHealth-KD 2020 challenge [25]. Team *UH-MatCom* ensembled several deep-learning architectures to produce an additional submission to the challenge [26]. The ensemble run obtained the best results out of the three runs of the team, except for an incorrectly submitted scenario. This shows that the ensembling pipeline produces outputs that perform better than the systems given as input. Section 4.3 elaborates on these results.

4.1. Quality of the Ensembled Corpus

In this section, the experiments applied to evaluate the ensembler's quality in extending corpora are presented. The ensembler was used to produce the *eHealth-KD 2019 ensembled corpus* [24], a collection of 8 000 automatically annotated sentences [24],

[10] https://github.com/knowledge-learning/ehealthkd-2020/tree/master/data/ensemble

from the Blank Collection of sentences. The selection criteria was to keep the top 8 000 sentences with the highest agreement (as described in Section 3.5). Figures 8, 9, 10, 11, and 12 summarize the steps applied to produce the final collection of automatically annotated sentences. The Reference Collection was used as a reference collection for tuning the ensembling pipeline. The impact of using the ensembled corpus as an extra source of training data for solving the challenge was tested, as shown in Section 4.1.3. Additionally, the performance of already tuned ensembler was evaluated in the three original evaluation scenarios of the eHealth-KD 2019 challenge, as shown in Section 4.1.4. Finally, the quality of the resulting corpus was measured with the agreement metric presented in Section 3.5, as shown in Section 4.1.5.

4.1.1. Ensembler Configuration

The ensembling pipeline was tuned in the 300 sentences of the Reference Collection. This means the configuration was selected from the best configurations found in the Reference Collection. Figure 8 illustrates the procedure.

Instead of the F_1 metric proposed in the challenge, a macro-F_1 variant of this metric was applied in the ensembling pipeline. The macro-F_1 metric averages the F_1 obtained across each annotation type. This was done to ensure that all annotations types were represented in the corpus. The F_1 metric used in the challenge benefits systems that decide to ignore certain types of annotations to achieve a better score. For example, the ensembler shown in Table 8 that resulted from optimizing the F_1 metrics decides not to report *causes*, *entails* and *has-property* relations. This is not a problem of the ensembler but of the evaluation metric selected. The ensembler was able to find a configuration that maximizes that metric. The correct metric to be used is subject to the problem that is being solved. For example, in the context of a system participating in the eHealth-KD challenge, maximizing the F_1 score is

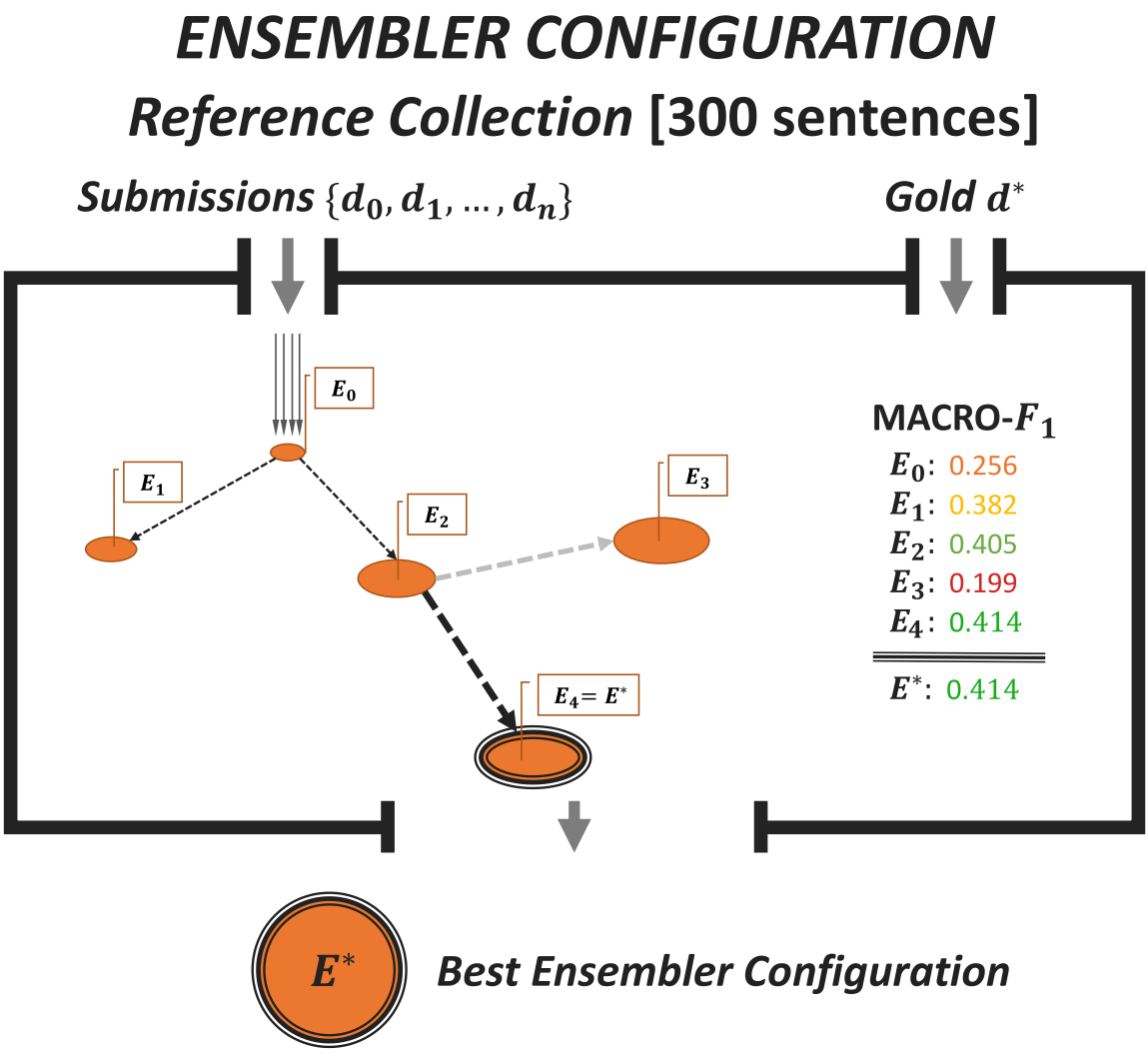

Figure 8: Ensembler configuration. The optimization search is performed using the Reference Collection. The submissions $\{d_0, d_1, \cdots, d_n\}$ are ensembled using several configurations $(E_0 \cdots E_4)$ looking for the ensembler configuration E^* for which the macro-F_1 in the gold document is maximized. The found E^* configuration was used to build the *eHealth-KD 2019 ensembled corpus*. Section 4.1.1 provides further information on E^*.

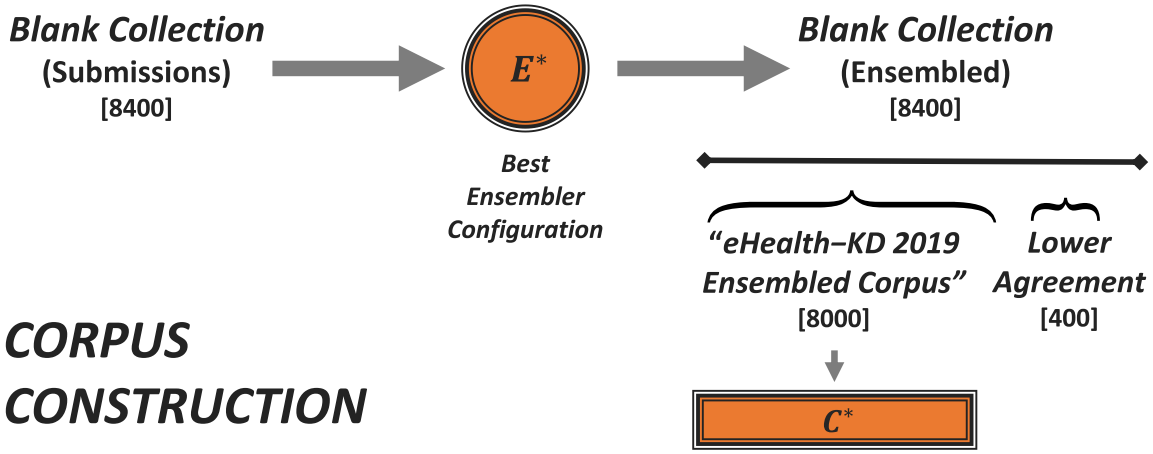

Figure 9: Corpus construction. The best ensembler configuration (E^*) found in the Reference Collection is used to build the *eHealth-KD 2019 ensembled corpus*. All sentences in the Black Collection are ensembled. From those 8 400 sentences, the 8 000 ones with higher agreement are included in the corpus (C^*). Section 4.1.2 provides statistics of the corpus (C^*).

the correct decision. However, for automatically annotated corpora, it is relevant to ensure that all annotation types are represented.

Table 3 summarizes the configuration that was used to ensemble the 8 000 sentences of the published *eHealth-KD 2019 ensembled corpus*. The macro-F_1 metric was used in the optimization search to measure the performance in the Reference Collection. The selected configuration achieved a macro-F_1 of 0.414 and an F_1 of 0.616 in the Reference Collection. Similarly, the selected configuration achieved a macro-F_1 of 0.455 and an F_1 of 0.647 in the Validation Collection. Figure 9 illustrates the corpus construction procedure.

4.1.2. Corpus Statistics

Table 4 shows the number of entities and relations annotated in the *eHealth-KD 2019 ensembled corpus*. A total of 86 112 elements were annotated, 50 249 of them as entities and the remaining 35 863 as relations. It is noteworthy that although

Phase	Strategy	Parameters
Building	voting	binary `True`
	submissions	best `False` top ∞ selection 15
Weighting	expert	
Scoring	expert	discrete `True`
Validating	constant-threshold	threshold `0.211`

Table 3: Best ensembler configuration found according to the macro-F_1 metric. The ensembler configuration was used to produce the *eHealth-KD 2019 ensembled corpus*, a collection of 8 000 automatically annotated sentences from the Black Collection. The Reference Collection was used as a reference collection for tuning the ensembling pipeline. Strategies and parameters are taken from Table 1. The collection of 15 selected eHealth-KD 2019 submissions is made of: `lsi2_uned/574805`, `lsi2_uned/575373`, `lsi_uned/575160`, `abravo/577194`, `iakesg/576597`, `iakesg/576606`, `vsp/576664`, `vsp/576666`, `vsp/576670`, `vsp/576671`, `jlcuad/577097`, `jlcuad/577119`, `talp/576647`, `hulat-taskAB/576454`, `hulat-taskAB/576998`.

attributes are part of the original annotation scheme of the *eHealth-KD v2* corpus, they were not considered in the context of the challenge and, therefore, are not available for the automatic construction of this ensembled corpus. Just as the original corpus, the most represented entity classes are *Concept* and *Action* [22]. The most represented relations are *target*, *subject*, and *in-context*.

Metric	eHealth-KD v2	Ensembled
Sentences	1 045	8 000
Entities	6 612	50 249
`Concept`	4 092	31 126
`Action`	1 742	13 522
`Predicate`	563	4 204
`Reference`	215	1 397
Relations	6 049	35 863
`target`	1 729	13 289
`subject`	894	5 144
`in-context`	677	4 975
`is-a`	566	3 284
`in-place`	400	1 580
`causes`	367	1 323
`domain`	364	1 217
`argument`	343	1 165
`entails`	167	423
`in-time`	165	565
`has-property`	159	2 242
`same-as`	124	467
`part-of`	94	189

Table 4: Summary statistics for the *eHealth-KD 2019 ensembled corpus*. The statistics of the *eHealth-KD v2* corpus are taken from (Piad et al. [22]).

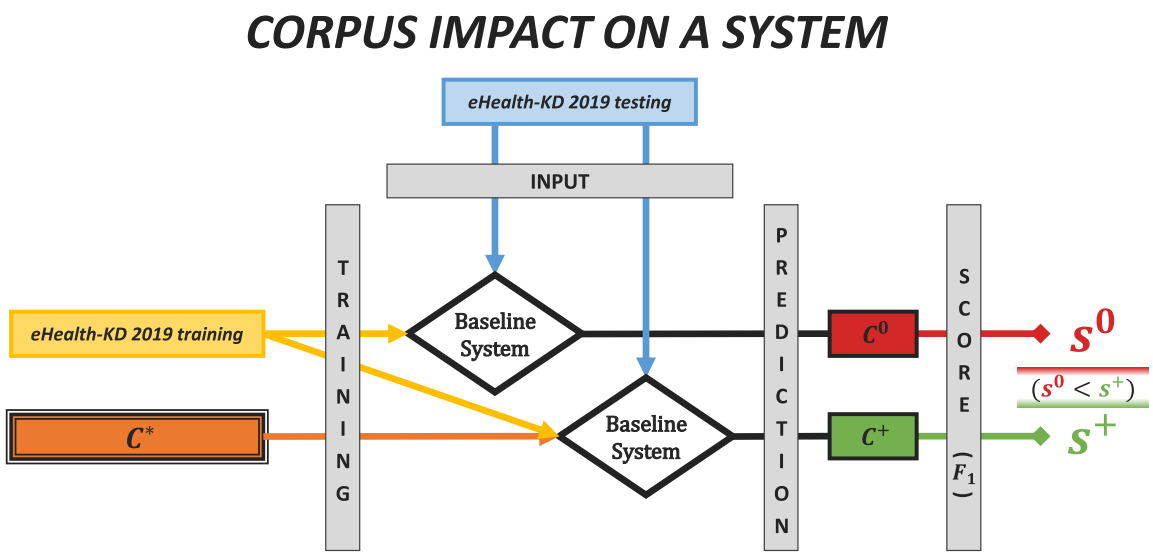

Figure 10: Corpus impact on a system. Two instances of the same system are trained: one using only the eHealth-KD 2019's training collection, and the other one using also the *eHealth-KD 2019 ensembled corpus* (C^*). Both systems are then used to predict the eHealth-KD 2019's testing collections, for which the F_1 score is computed as proposed in the challenge. The system that uses both training collection performs better than the other one, as shown in Section 4.1.3.

4.1.3. Corpus Impact on a System

To evaluate the usefulness of the 8 000 automatically annotated sentences as additional corpora, a system that used such collection as additional training data is evaluated. Specifically, the system was first trained using only the original training collection of the eHealth-KD 2019 challenge and another instance of the same system was then trained in a collection containing both the original sentences and the *eHealth-KD 2019 ensembled corpus*'s sentences. The original evaluation scenarios of the eHealth-KD 2019 challenge were used to evaluate both instances of the system. The F_1 metric proposed in the challenge was used in the three scenarios. Figure 10 illustrates this procedure.

The selected system is a new baseline algorithm that was implemented and trained

to measure the impact of using the ensembled collection for future annotations.[11] The new baseline consists of two machine learning models, each one responsible for solving one of the tasks of the challenge.

The new baseline solves the entity extraction task as a sequence labeling problem. Sentences are tokenized and each token is assigned a label in the BILOUV [28] entity tagging scheme[12]. With the exception of the ***O****ther* tag, all BILOUV tags are suffixed with the four entity types, for a total of 21 labels. Several linguistic features (e.g., text, lemma, POS-tag, dependency tree label, etc.) obtained from *spaCy*[13] are used to represent the tokens. A CRF [29] estimator is used to predict the sequence of output labels from the input sequence of tokens.

The new baseline solves the relation extraction task by using a Logistic Regression model to predict which type of relation (if any) exists between each pair of entities in the same sentence. The concatenation of the features of both tokens is used as input to the model.

Table 5 summarizes performance achieved by the new baseline, when trained in the original training collection only and when trained in both collections (the *eHealth-KD 2019*'s training collection plus the *eHealth-KD 2019 ensembler corpus*), for the three evaluation scenarios. In every scenario, there is an improvement in performance while using the information in the ensemble collection. This indicates that there are new patterns not found in the original collection that can now be learned from the ensembled collection. More complex systems might make a better use of this information. A subset of the ensembled collection (3 000 sentences) is provided to the participants of the eHealth-KD challenge at IberLEF 2020. Participants are free to

[11] https://github.com/jpconsuegra/ensemble-baseline

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

[13] https://spacy.io/

Baseline	1-main	2-taskA	3-taskB
training	0.533	0.778	0.356
training + ensemble	**0.556**	**0.800**	**0.363**
human	0.727	0.861	0.735

Table 5: Comparison of performance of a baseline system when trained only on the original training collection provided in the eHealth-KD 2019 challenge, and that trained with the addition of the *eHealth-KD 2019 ensembled corpus*. The system is evaluated in the three evaluation scenarios of the eHealth-KD 2019 challenge. A human baseline is also provided for comparison purposes.

use this collection or not. The impact of using this additional information will be later measured according to how the performance of their system was influenced by the inclusion of this collection.

For reference purposes, the human baseline presented by Piad et al. [22] is included in Table 5. The performance of the human baseline approximates the best performance that can be achieved by any automatic annotation system. Relatively to this score, the performance of the baseline system when trained only on the original training collection provided in the eHealth-KD 2019 challenge, and that trained with the addition of the ensemble collection, change to 0.733 and 0.765 in Scenario 1, respectively. In Scenario 2, the scores change to 0.904 and 0.929, respectively. Finally, in Scenario 3, the scores change to 0.484 and 0.494, respectively.

4.1.4. Estimated Quality

To estimate the quality of the selected ensembler configuration (shown in Table 3) in annotating sentences and, therefore, the quality of the *eHealth-KD ensembled corpus*' sentences, that same ensembler configuration was used to annotate a manually annotated collection of sentences. The three evaluation collections of the eHealth-KD

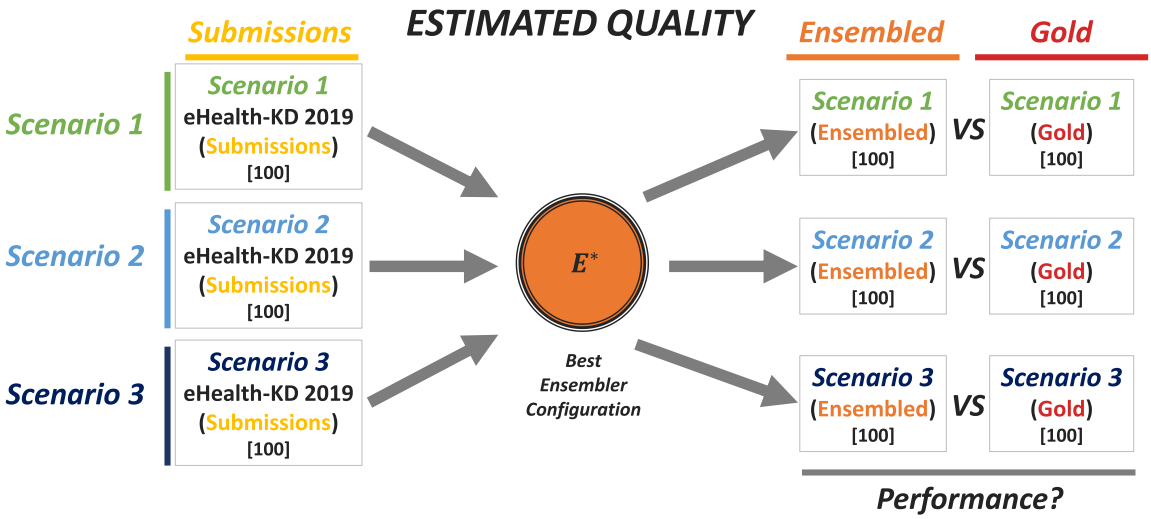

Figure 11: Estimated quality. The quality of the ensembler configuration E^* used to generate the *eHealth-KD 2019 ensembled corpus* is estimated by ensembling collections for which there are human-annotated versions. The three evaluation scenarios of the eHealth-KD 2019 challenge are used. Those collections were not used to tune the ensembler configuration E^*. Section 4.1.4 provides further information in this regard.

2019 challenge were used for this purpose. The sentences of those collections are not part of the Reference Collection used to tune the ensembler. Those sentences were manually annotated by the organizers of the challenge. Figure 11 illustrates the quality estimation procedure.

Table 6 compares in the three evaluation scenarios the performance achieved by the selected ensembler against the baseline and the best performing submission to the challenge (TALP-UPC). Additionally, the human baseline presented by Piad et al. [22] is included. The ensembler was able to outperform TALP-UPC's performance in all scenarios but the third one. This scenario deals only with the extraction of relevant relations. It has been shown that this task is considerably harder than the entity recognition one [23]. The results achieved by the ensembler suggest that there

System	F_1			macro-F_1		
	1-main	**2-taskA**	**3-taskB**	**1-main**	**2-taskA**	**3-taskB**
baseline	0.431	0.546	0.123	0.182	0.461	0.095
TALP-UPC	0.639	0.820	**0.627**	0.451	**0.792**	**0.520**
ensemble (Opt.)	**0.647**	**0.823**	0.610	**0.455**	**0.792**	0.491
human baseline	0.727	0.861	0.735	0.592	0.830	0.666

Table 6: Summary of the performance, in the three evaluation scenarios, of the best ensembler configuration (according to the macro-F_1 metric) against the baseline, the best performing submission, and a human baseline.

is no fixed rule to follow for choosing which sub-system to trust. No model decided to completely trust TALP-UPC's annotations, which suggests that for certain situations (present in the reference collection) trusting other systems might be a better option. In the long run, this rule can not be extended to the whole collection of sentences.

The selected ensembler configuration can produce better annotations for Scenario 1[14] than all of the individual systems. As shown in Table 6, this is true even for sentences that were not present in the reference collection. This fact supports the idea that corpora extension using our ensembling pipeline produces better results than corpora extension through bootstrapping (since the latter would use only the output of the best performing system, TALP-UPC in this case).

[14]Scenario 1 is considered the main evaluation scenario of the challenge and includes both tasks (the entity and relation extraction tasks).

4.1.5. Sentence Agreement

As explained in Section 3.5, the use of ensembling methods to produce a single merged version from different versions of a document allows computing an agreement metric to further enhance the information available on the quality of the final result. In this section, the agreement on each *eHealth-KD ensembled corpus*' sentence is computed. Then, it is shown that there is a relation between the agreement of the sentences in a corpus and the quality of its annotations. Figure 12 illustrates this procedure.

Figure 13 shows the number of sentences in the collection per agreement range. Only one system per participant was considered to do this measure. This is done to avoid biasing the results toward a specific participant if they submit a large number of similar systems. Most sentences had an agreement around 0.2 and 0.3, which means that in most sentences only $20 - 30\%$ of the sub-systems agreed. This reveals how difficult it is to get an agreement among many knowledge discovery systems.

Figure 14 shows the performance, of the ensembler configuration used to build the *eHealth-KD ensembled corpus*, in different subsets of the Reference and Validation Collections as the number of sentences increases, sorted by ascending (and descending) agreement. As expected, as the number of sentences increases, the collections with higher agreement tend to perform better than those with a lower agreement. This shows that the agreement metric unlocked by the ensembler can be used to estimate and control the quality of a collection of sentences for which there is no available human-annotated version. The sentences published in the *eHealth-KD 2019 ensembled corpus* [24] are sorted by agreement so that the top agreement sentences can be easily filtered. The explicit agreement of each sentence is also provided.

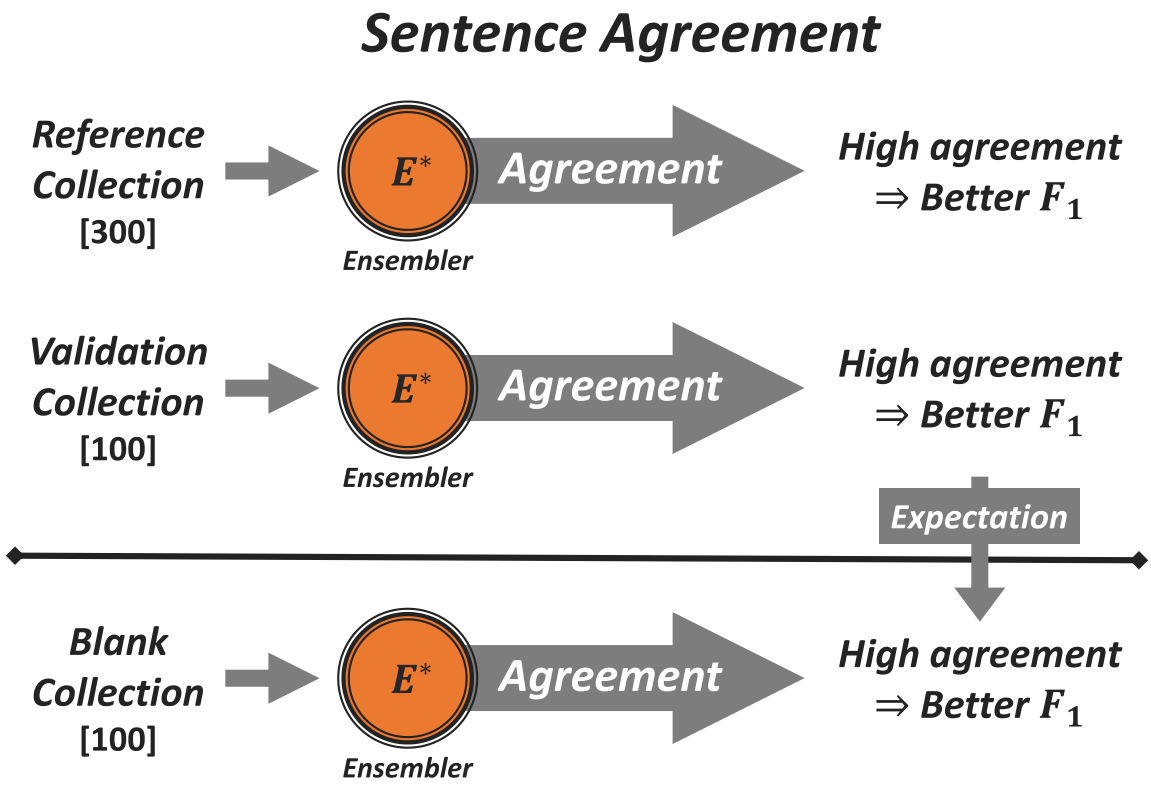

Figure 12: Sentence agreement. The agreement of all sentences in the Reference Collection and Validation Collection is computed. Such analysis shows that subsets of sentences with a higher agreement tend to have a higher F_1 score. This fact allows using the agreement of the sentences in the *eHealth-KD 2019 ensembled corpus* to control its quality. Section 4.1.5 provides further information in this regard.

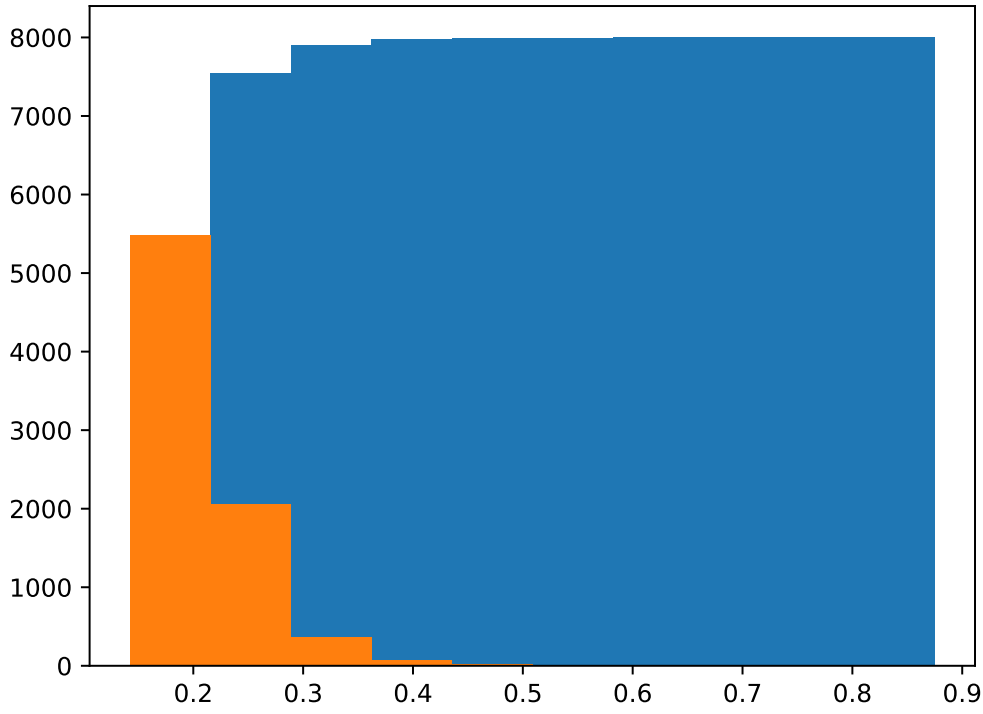

Figure 13: Number of sentences (in the *eHealth-KD 2019 ensembled corpus*) per agreement range between submissions. The histogram is shown in orange. The cumulative distribution is shown in blue.

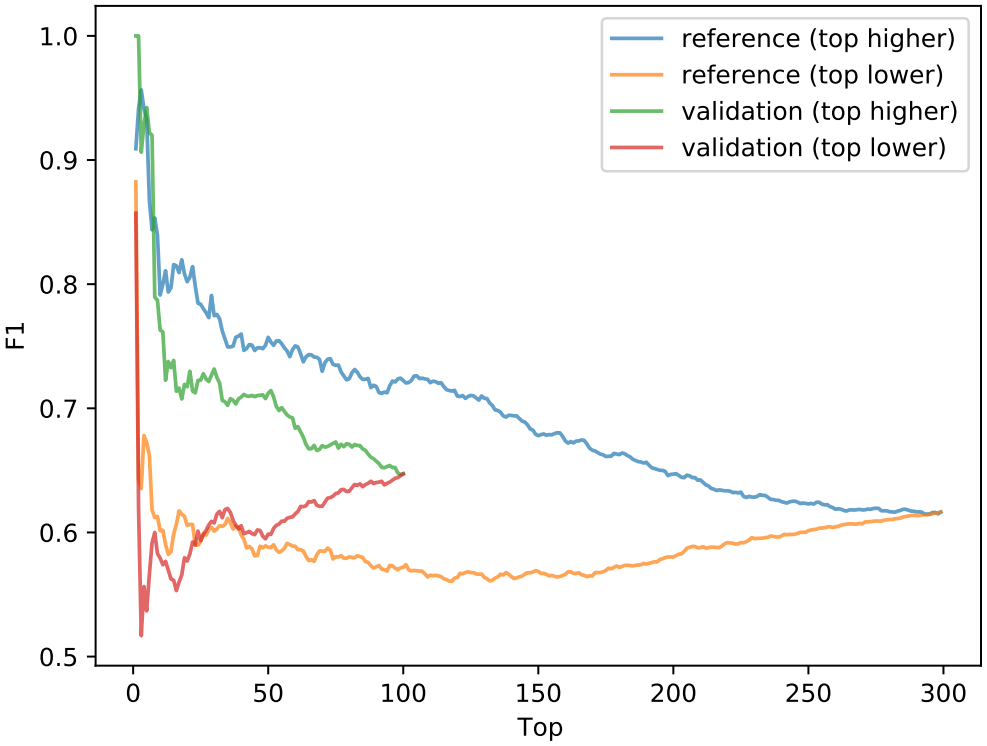

Figure 14: F_1 achieved by using only the top k sentences with higher or lower agreement respectively. The `reference (top higher)` and `reference (top lower)` lines use the sentences in the Reference Collection. Similarly, the `validation (top higher)` and `validation (top lower)` lines use the sentences in the Validation Collection. The `(top higher)` lines show the F_1 achieved by selecting only the top x sentences with higher agreement. Similarly, the `(top lower)` lines show the F_1 achieved by selecting only the top x sentences with lower agreement. As the number of sentences decreases, the collections with a higher agreement have their F_1 increased while the collections with a lower agreement have their F_1 decreased.

4.2. Quality of the Ensembling Algorithms

The ensembling pipeline presented in this paper was directly applied to the Blank Collection of sentences.[15] The ensembler was tuned in the Reference Collection and validated in the Validation Collection.

Table 7 summarizes the performance obtained by several ensembler configurations in the Reference Collection and the Validation Collection. Additionally, both the baseline annotation algorithm and the submission that achieved the best score in the challenge [23] (TALP-UPC [30]) are included at the start of the table. Also, the oracle ensemblers are added at the end of the table for comparative purposes (e.i., they provide an upper bound of the performance that can achieve a perfect ensembler). Both the binary and non-binary version of the oracle ensemble achieve the same approximate result.

The system that resulted from optimizing the combination of handcrafted strategies achieves a slightly better performance than by only using the highest scored submission (0.650 and 0.639, respectively). Table 8 summarizes the architecture of the best combination found. The use of expert information through the inclusion of expert weighting over uniform weighting resulted in an important improvement. The optimization algorithm ran for about 500 generations of 10 individuals before finishing due to no-improvement early stopping.

Most automatic learning systems achieved a superior performance in the Reference Collection than in the Validation Collection. This was expected since weights were directly optimized to maximize the performance in the training collection, i.e., the Reference Collection. Additional considerations were made to measure their generalization capability, like using a "leave one out" cross-validation process. The validation

[15] https://github.com/jpconsuegra/ensemble-kd2019

System	Strategy		Ref.	Val.
Source	baseline		0.355	0.431
	TALP-UPC		0.605	0.639
Handcrafted	majority		0.426	0.486
	union		0.357	0.377
	optimization		**0.619**	**0.650**
Learning	all in one	RF	0.672	0.622
	per category	RF	0.675	0.626
	per label	SVM	0.635	0.623
		LR	0.607	0.640
Oracle	best		0.810	0.843
	all		0.834	0.873

Table 7: Summary of the performance of some relevant ensembler configurations. The baseline algorithm and the best performing submission to the challenge are included at the beginning of the table. Column **Ref** shows the F_1 score achieved in the Reference Collection. Column **Val** shows the F_1 score achieved in the Validation Collection after configuration selection. The strategy labeled as *optimization* corresponds to the ensembler configuration that resulted from exploring the space of handcrafted rules to find the optimal configuration according to the F_1 metric (shown in Table 8). RF, SVM, and LR stand for Random Forest, Support Vector Machine, and Logistic Regression, respectively.

revealed that indeed their performance was overestimated. The *best* and *weighting* variations did not have a significant impact on the best performing learning systems, though the inclusion of expert weighting produced a slightly positive improvement.

The performance achieved by the oracle ensemblers provides an upper bound of

Phase	Strategy	Parameters
Building	voting	binary `True`
	submissions	best `False`, top `6`
Weighting	expert	
Scoring	expert	discrete `True`
Validating	constant-threshold	threshold `0.211`

Table 8: Best ensembler configuration found in the space of handcrafted rules according to the F_1 metric. The Reference Collection was used for tuning the ensembling pipeline. Strategies and parameters are taken from Table 1.

the performance that can be achieved by any ensembling method that relies only on the information contained across all the submissions to the eHealth-KD 2019 challenge. The oracle ensembler that considers all submissions achieves a performance of 0.873 in the Validation Collection. This means that it is impossible to correctly annotate 13% of the Validation Collection based on the voting models. Relatively to this score, the ensembler that resulted from the optimization process reaches approximately 74% of the total capacity. This shows that there might not be any set of consistent rules to decide which model votes to trust in each case.

4.3. Quality of the Ensembler Optimization

Team UH-MatCom designed and trained four deep-learning models in the 2020 edition of the eHealth-KD challenge [26]. Two of the four trained models were submitted to the challenge. Additionally, the ensemble method described in this paper was applied to produce a third run.

The best found ensembling pipeline for Scenario 2 and Scenario 3 was to use the

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 9: Summary of the performance (F_1) achieved by the system of team *UH-MatCom* across the three runs of the eHealth-KD 2020 challenge. 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*".

automatic learning rules, with a *per label model* configuration using a Support Vector Machine (SVM) binary classifier for each different type of entity and relation. Only two models were ensembled for Scenario 1 and 4 due to not having the corresponding submissions of those models in these scenarios. The best found ensembling pipeline was to use the automatic learning rules, with an *all in one model* configuration using a single Logistic Regression classifier for all the entity types and relations.

Table 9 shows the results obtained by the three runs in the challenge. There was 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. This shows that the ensembling pipeline produces outputs that perform better than the systems given as input.

5. Discussion

The ensembling pipeline presented in this paper is meant to take advantage of information extraction competitions, as a source of different annotated versions of the same sentences, to produce an extended version of the original corpus. This method responds to corpus bootstrapping, the technique in which a corpus is automatically extended using programmatic instructions. The use of ensembling techniques instead of using a single version has several advantages, such as: reducing the bias towards a particular architecture or set of rules and providing richer information about the quality of the resulting corpora. The effectiveness of the proposed method in achieving these goals was evaluated in Section 4.

Since the ensembler presented in this paper works directly on the annotations instead of working with the algorithms, it allows the integration of any group of systems independently of their architecture. As long as they share the same annotation scheme (which specifies the possible labels of entities and relations and the lexicographic rules of the tokens), there is no need for them to share the same intermediate steps, architecture, or resources. That turns scientific competitions into an opportunity for automatic corpus extension since all the submitted systems share the same annotation scheme.

Another advantage that derives from working directly on the outputs, is that the ensembler can be used to combine non-programmatic sources of annotations, i.e. human annotations. This advantage is especially relevant when there is no initial collection of labeled data that can be used to train learning models. Unlike algorithms, humans can begin annotating documents from the specification of an annotation scheme. By evaluating their performance on a small trial collection, the ensembling pipeline can measure the quality of their annotation as it would with computer systems.

Future annotations made by the same person can be intelligently combined with other annotated versions of the same document to produce an improved variant. This way, people from all around the world can collaborate with the construction of corpora by incrementally providing their own manually annotated versions of arbitrary subsets of the corpus under construction. This reduces the need for expertise or supervision of the annotation task, since the ensembler can be used as a way to regularize the annotation quality. Applying such a mechanism constitutes an important step into the crowdsourcing corpora annotation, since more people can contribute with a little supervision.

Compared to active learning techniques, the ensembling approach does not require expert knowledge, but instead it takes advantage of having numerous sources of information to build the corpus, although they may not be completely accurate. The precision is regulated by the quality of the systems involved in the annotation and the coverage that the creator of the corpus wishes to give to the corpus. Most of the current research in active learning field involves finding the best method to choose the samples to be asked to a human oracle. The ensembling pipeline applied for crowdsourcing allows the annotation of arbitrary amounts of data given the softer restriction on quality and the higher availability of pseudo-oracles.

6. Conclusions and Future Work

This paper discusses several strategies for building an ensembling pipeline oriented toward NER and relation extraction tasks. Using an optimization search algorithm, the space of possible ensemblers that can be built with the given strategies is explored to find the optimal configuration for a concrete corpus. The *eHealth-KD v2* corpus [22] was taken as a case of study, resulting in the automatic annotation of the relevant key phrases and relations present across 8 000 sentences. The submissions presented

in the eHealth Knowledge Discovery Challenge at IberLEF 2019 were used as the only source of votes (no additional voting models were trained). The experiments show that the collection of ensembled sentences contains information that can be used by learning algorithms to improve their performance in the original eHealth-KD tasks. This fact encouraged the distribution of the collection as an auxiliary training corpus in the 2020 edition of the eHealth-KD challenge. Since error correction in the eHealth-KD corpus is faster than manual annotation, further progress can be made now in extending it. The agreement metric computed for the sentences was shown to hold information about the quality of the annotation. Additionally, the results show that the ensembling pipeline produces more accurate outputs than the systems it takes as input. This kind of ensembling technology can be used to produce highly competitive information extraction systems.

Since the ensembling techniques discussed in this paper rely only on the resulting annotations and not on the systems themselves, these methods can be used to assist crowd annotation of NER and relation extraction related corpora. Large collections of corpora can be annotated by multiple individuals with overlapping annotations. A smaller collection of gold annotated sentences can be built to estimate the quality of each annotation. The ensemble system will then be asked to find the best way to combine the different votes to obtain a more accurate corpus. As major future work we plan to study the application of machine learning techniques to deal with corpora bias during the ensembling procedure to obtain a impartial version of it.

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). Moreover, it has been backed by the work of both COST Actions: CA19134 - "Distributed Knowledge Graphs" and CA19142 - "Leading Platform for European Citizens, Industries, Academia and Policymakers in Media Accessibility".

References

[1] J. P. Chiu, E. Nichols, Named entity recognition with bidirectional lstm-cnns, Transactions of the Association for Computational Linguistics 4 (2016) 357–370.

[2] L. Gligic, A. Kormilitzin, P. Goldberg, A. Nevado-Holgado, Named entity recognition in electronic health records using transfer learning bootstrapped neural networks, Neural Networks 121 (2020) 132–139.

[3] B. Agarwal, N. Mittal, Machine learning approach for sentiment analysis, in: Prominent feature extraction for sentiment analysis, Springer, 2016, pp. 21–45.

[4] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, arXiv preprint arXiv:1409.0473 (2014).

[5] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer, Neural architectures for named entity recognition, arXiv preprint arXiv:1603.01360 (2016).

[6] N. Limsopatham, N. Collier, Bidirectional lstm for named entity recognition in twitter messages (2016).

[7] P. J. Gorinski, H. Wu, C. Grover, R. Tobin, C. Talbot, H. Whalley, C. Sudlow, W. Whiteley, B. Alex, Named entity recognition for electronic health records:

a comparison of rule-based and machine learning approaches, arXiv preprint arXiv:1903.03985 (2019).

[8] O. Hegazy, O. S. Soliman, M. A. Salam, A machine learning model for stock market prediction, arXiv preprint arXiv:1402.7351 (2014).

[9] A. Piad-Morffis, Y. Gutiérrez, R. Muñoz, A corpus to support ehealth knowledge discovery technologies, Journal of biomedical informatics 94 (2019) 103172.

[10] E. K. Ringger, M. Carmen, R. Haertel, K. D. Seppi, D. Lonsdale, P. McClanahan, J. L. Carroll, N. Ellison, Assessing the costs of machine-assisted corpus annotation through a user study., in: LREC, Vol. 8, 2008, pp. 3318–3324.

[11] J. Zavrel, W. Daelemans, Bootstrapping a tagged corpus through combination of existing heterogeneous taggers, arXiv preprint cs/0007018 (2000).

[12] O. Ikechukwu E, O. Ebele G, A. Godwin E, et al., Bootstrapping method for developing part-of-speech tagged corpus in low resource languages tagset-a focus on an african igbo, arXiv preprint arXiv:1903.05225 (2019).

[13] R. Polikar, Ensemble based systems in decision making, IEEE Circuits and systems magazine 6 (3) (2006) 21–45.

[14] P. Yang, Y. Hwa Yang, B. B Zhou, A. Y Zomaya, A review of ensemble methods in bioinformatics, Current Bioinformatics 5 (4) (2010) 296–308.

[15] T. G. Dietterich, Ensemble methods in machine learning, in: International workshop on multiple classifier systems, Springer, 2000, pp. 1–15.

[16] R. E. Schapire, The strength of weak learnability, Machine learning 5 (2) (1990) 197–227.

[17] L. Breiman, Bagging predictors, Machine learning 24 (2) (1996) 123–140.

[18] L. Rokach, Ensemble-based classifiers, Artificial Intelligence Review 33 (1-2) (2010) 1–39.

[19] D. Opitz, R. Maclin, Popular ensemble methods: An empirical study, Journal of artificial intelligence research 11 (1999) 169–198.

[20] R. Speck, A.-C. N. Ngomo, Ensemble learning for named entity recognition, in: International semantic web conference, Springer, 2014, pp. 519–534.

[21] A. Piad-Morffis, Y. Gutiérrez, Y. Almeida-Cruz, R. Muñoz, [dataset] eHealth-KD v2 (Mar. 2020). doi:10.5281/zenodo.3696792.
URL https://doi.org/10.5281/zenodo.3696792

[22] A. Piad-Morffis, Y. Gutiérrez, Y. Almeida-Cruz, R. Muñoz, A computational ecosystem to support ehealth knowledge discovery technologies in spanish, Journal of Biomedical Informatics (2020) 103517doi:https://doi.org/10.1016/j.jbi.2020.103517.
URL http://www.sciencedirect.com/science/article/pii/S1532046420301453

[23] A. Piad-Morffis, Y. Gutiérrez, J. P. Consuegra-Ayala, S. Estevez-Velarde, Y. Almeida-Cruz, R. Munoz, A. Montoyo, Overview of the ehealth knowledge discovery challenge at iberlef 2019, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019). CEUR Workshop Proceedings, CEUR-WS. org, 2019.

[24] J. P. Consuegra, jpconsuegra/ehealthkd-2019-ensembled-corpus: Version 20.07

(Jul. 2020). doi:10.5281/zenodo.3926746.
URL https://doi.org/10.5281/zenodo.3926746

[25] A. Piad-Morffis, Y. Gutiérrez, 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 (IberLEF 2020), 2020.

[26] J. P. Consuegra-Ayala, M. Palomar, UH-MatCom at eHealth-KD Challenge 2020, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020), 2020.

[27] A. Piad-Morffis, J. P. Consuegra-Ayala, Y. Gutiérrez, S. Estevez-Velarde, ehealth-kd 2020 (Apr. 2020). doi:10.5281/zenodo.3870509.
URL https://doi.org/10.5281/zenodo.3870509

[28] 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.

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

[30] S. Medina Herrera, J. Turmo Borras, Talp-upc at ehealth-kd challenge 2019: A joint model with contextual embeddings for clinical information extraction, in: Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2019): co-located with 35th Conference of the Spanish Society for Natural Language Processing (SEPLN 2019): Bilbao, Spain, September 24th, 2019, CEUR-WS. org, 2019, pp. 78–84.