Data augmentation process to improve deep learning-based NER task in the automotive industry field

Abdenacer KERAGHEL
Lizeo IT
42 Quai Rambaud, 69002 Lyon
Lyon, France
abdenacer.keraghel@lizeo-group.com

Khalid BENABDESLEM
Lyon 1 university
43, Bd du 11 Novembre 1918
Villeurbanne, Cedex 69622, France
khalid.benabdeslem@univ-lyon1.fr

Bruno CANITIA
Lizeo IT
42 Quai Rambaud, 69002 Lyon
Lyon, France
bruno.canitia@lizeo-group.com

***Abstract*—Searching and extracting information in a textual sequence drawn from scientific articles, queries on a search engine and posts in a discussion forum necessitate a process called Named Entity Recognition (NER). Nevertheless, the data available to achieve this process diverge depending on their nature and field of study. In this article, we look at the performance of Named Entity Recognition systems, and their complexity and ability to process data from different backgrounds. A comparative study between several state-of-the-art approaches, applied to different types of data (search engine queries and discussion forum posts) related to the automotive industry field, is proposed in order to select the approach that will best suit our instance. To do this, we shall rely on the results of metrics evaluating machine learning models such as precision, recall and F-score.**

I. Introduction

The «Named Entity» (NE) term is a referential linguistic unit, it is widely used in Natural Language Processing (NLP), which is coined for the Message Understanding Conference (MUC) [7]. At that time, MUC was focusing on Information Extraction (IE) tasks, where structured information of company activities and defense related activities is extracted from unstructured text[1]. NE is essential to recognize information units like names, including person (PER) as «Barack Obama» and organization (ORG) as «Google» and location (LOC) as «London». Identifying references to these entities in text was recognized as one of important sub-tasks of Information Extraction and was called «Named Entity Recognition» [15]. In this paper, we look at automotive industry field, whose entities are selected referring to the type and brand of vehicle, the size and season of tyre, etc.

The principle of our Named Entity Recognition system is the same as the traditional one (extraction of units called named entities from textual sequences). The difference lies in the set of named entities, instead of LOC, PERS, ORG (cf. Tab. II) we have Vehicle-Maker, Vehicle-Segment, Vehicle-Cartype, Tyre-Dimension, etc (cf. Tab. III). And also in the nature of the text to be processed (search engine queries in our case).

[1]Unstructured text is often user-generated information such as email or instant messages, social media postings or search engine queries

An unambiguously named entity recognition system is designed to analyze input data (via tokenization[2] and confidence score calculation) to detect their associated classes. In this case, a token[3] of a sequence may belong to several classes, hence the need for a stage of disambiguation. The latter aims to determine the right class of this token. After a review of prevailing literature, the main approaches to existing named entity recognition are the following (see Fig. 1).

Two families caught our attention:

- The **Probabilistic** family alike the hidden Markov chains (HMMs) [20], [21] and the CRFs (Conditional Random Fields) [13]. This family consists of representing the named entity recognition problem under the form of an unoriented graph, where nodes correspond to the named entity classes and edges to transition probabilities. The goal is to predict the most likely class sequence using the set of transition probabilities. This corresponds to the observed token sequence (a query on a search engine in our case);
- The **Deep Learning** family [1], [8], [9]. It is a family of neural networks of different types: recurrent (LSTM, GRU) or hybrid (hybridization of convolutional (CNN) and recurrent neural networks). It consists of learning long-term dependencies between the different classes of named entities possible in the learning stage. The objective of this family is to predict the subsequent class of entity named in a sequence, from the previous classes.

The **Machine Learning** family was discarded because it seemed overall to be less efficient than the previous two (Deep Learning and probabilistic). The time constraints and those imposed by the application domain (multi-support, heterogeneity of the universes of named entities) made any additional experimentation impossible.

In this article, we will first present the families of approaches we have chosen in order to draw up a comparative study. We will then present our test datasets as well as

[2]The operation consists of cutting a text into tokens, most often words.
[3]A lexical unit.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

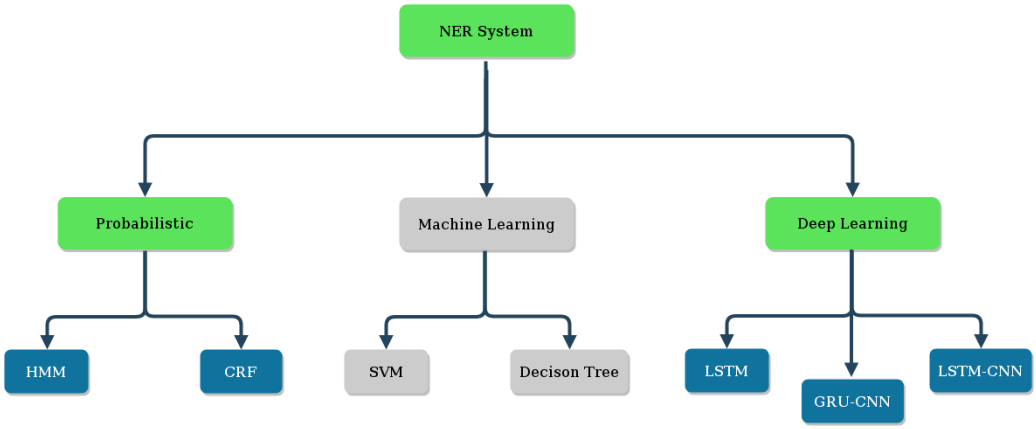


Fig. 1: Main approaches of named entity recognition.

our experimental protocol before making a restitution of the results.

II. Learning methods for named entity recognition

Based on the complexity and the capacity of the models to process data from different origins, our choice fell on the following five approaches (without exhaustiveness):

A. Conditional Random Filed (CRF)

CRFs[4] are non-directed statistical graphical models, of which a particular case is a linear chain which corresponds to a finite state automaton (HMM or hidden Markov chain), conditionally trained on a dataset previously annotated by experts. These types of models are well suited for sequential analysis, and CRFs in particular have proven useful for partial speech tagging (Lafferty and al. 2001) and named entity recognition for press wire data [13]. They have also been applied to the task of recognizing mentions of genes and proteins (McDonald and Pereira. 2004), with interesting preliminary results [14] (86.4% as precision and 78.7% as recall and 82.4% as F-score).

B. Long Short-Term Memory (LSTM)

This model[3] is a special case of RNN (Recurrent Neural Network), the only difference is that it contains memory cells allowing it to keep information for a theoretically indefinite period. It makes it easy to learn long term dependencies for sequential labeling tasks such as voice recognition or named entity recognition [8]. The goal of an LSTM is to create a memory cell for each of the words in the sequence, the latter containing the context of the associated word, which allows easy recognition of its named entity class.

[4]Our implementation cannot be released due to intellectual property reasons.

C. GRU-CNN

It is a hybridization[3] of two neural networks. The first is an RNN using the GRU (Gated Recurrent Units) [10] Architecture, which aims at solving the vanishing gradient problem of an RNN. GRU uses its oblivion doors to decide what information should be passed to the exit. It is able to store information from the distant past without losing it over time, which is relevant for predicting named feature classes.

The second (CNN) [12] is a Convolutional Neural Network whose purpose is to extract a matrix of characteristics of fixed size for each input and to filter the noise using a product of convolution. The objective of this hybridization is to be able to keep both the history and the characteristics of a given word, which in turn will allow us to easily identify the equivalence classes (words which belong to the same class).

D. LSTM-CNN

It is a hybridization[5] of the two neural networks CNN and LSTM previously defined. The objective is the same as that of the GRU-CNN, consisting in preserving without loss of characteristics of the sequence as well as the context of each word (its history from the past to the future). This makes it easy to identify and predict the corresponding named entity class. This type of models is well suited for the detection of anomalies in the time series, and also for the recognition of named entity [1] For press wire data as well as informal sequences[6].

E. BLSTM-CNN

This model is based on the LSTM-CNN with bidirectional functionality[7] [1]. The purpose of this functionality is to be able to keep the history of a given word in both directions, from past to future and from future to past. This will strengthen

[5]Code adapted from: https: //github.com/kamalkraj/Named-Entity-Recognition-with-Bidirectional-LSTM-CNNs.

[6]Text sequences containing abbreviations such as tweets, blogs, discussion forum posts, etc.

[7]Code adapted from: https: //github.com/kamalkraj/Named-Entity-Recognition-with-Bidirectional-LSTM-CNNs.

the decision of the model in terms of predicting the equivalence classes of named entities.

III. Data sources

In order to carry out our comparative study, we used the following two datasets:

A. CONLL-2003

It is a public dataset[8], comprising news articles, cut into sentences in the form of posts. It is used to calibrate an automatic learning model on the named entity recognition task. It consists of eight files covering the two languages: German and English. In our case, we only used English. For each language, we have 3 main files: learning, testing, and validation. The following two tables illustrate the description and the volume [16] of each of them.

TABLE I: Description of the different files in the CONLL'03 dataset (english).

Files	Nb of news articles	Nb of sentences	Nb of tokens
Learning	946	14 987	203 621
Test	231	3 684	46 435
Validation	216	3466	51 362

TABLE II: Number of named entities per file.

Files / NEs	LOC	MISC	ORG	PER
Leaning	7140	3438	6321	6600
Test	1668	702	1661	1617
Validation	1837	922	1341	1842

B. Requests from the automotive industry field

The automotive industry is a particular field, with specific named entities. These are classified by domain; the Table III illustrates these different domains as well as their associated NEs classes.

TABLE III: The set of domains and named entities classes in the automotive industry domain.

Domain	Named Entity Class	Example
Vehicle	Vehicle-Maker	Volkswagen
	Vehicle-Segment	Golf
	Vehicle-Cartype	Golf 4
	Vehicle-Motor	Hdi 1.6
Pattern	Pattern-Brand	Pirelli
	Pattern-Product	Sp 22
	Pattern-Name	AMH2
	Pattern-Season	Winter
Dimension	Dimension-Complete	255/62 R15 96h
	Dimension-GeoBox	255/62 R15
	Dimension-Diameter	R15
Dealer	Dealer-Name	Norauto
Localization	Localisation-City	Lyon

One of the major problems we encountered when carrying out our comparative study was the insufficient quantity of labeled data available from the automotive industry. We internally had a restricted dataset containing a few hundred queries (491) in French, issued by users of an online tyre price comparator, annotated by experts in the field. This volume of data remains insufficient for a task of learning and validating state-of-the-art models. As a palliative, we decided to set up a data generation process based on machine learning on existing data. To do this, several approaches have been considered such as:

- **The joint low**: it is a probability law with several variables. The goal is to estimate the parameters of the transitions between each EN and the others. For example, the probability of having a Vehicle-Maker after a Location-City is 0.02. This approach is suitable, but seems very sensitive to the insufficiency of the learning dataset, which is our case;
- **Recurrent neural networks** [17]: these are character-based models, used to predict the next character in a sequence. These models are not fitting to our case, because we are going to have too much noise at the level of the generated sequences (incorrect sequences) which implies a corrupt annotation;
- **Bayesian networks** [19]: these are probabilistic graphical models, based on a set of conditional probabilities (parameters) estimated from a learning step on a dataset. They are very similar to the attached law, except that the latter can inject expert knowledge during the learning stage. These models are very appropriate to our case, in particular due to the fact that we have some business knowledge as well as relations between the different classes of named entities via a dictionary containing the sphere of the automotive industry available internally. For this we will study these in detail.

1) Description of a Bayesian network: A Bayesian network is an acyclic graphical model (of the generative type) of a distribution of probabilities common to a set of variables. The Bayesian network is composed of two components: a graphical structure and a set of conditional probabilities which represent the parameters of the model. The graph structure is a set of nodes, each of which represents a discrete or continuous variable in the training dataset. These are linked together via oriented arches which represent the outbuildings. If an arc between two nodes is present, a link of type father node and node child is established as well as an association of a distribution of conditional probabilities for each node.

The blue rectangles in Fig. 2, represent the nodes of the graph (variables), each node has its own table of parameters (distribution of conditional probabilities) calculated from knowing its parents probabilities. Ex: the parameters of the **Breakdown** node are calculated from its probability distribution knowing the **Fuel, Battery** nodes, which gives the following formula: $P(Br = 1|Ba = 1, Fu = 1)$ = 0.8, etc. Using this formula, we can regenerate the parameter table of the Breakdown node by replacing the torque (Fuel, Battery) with the associated values.

In our case, the variables are the named entity classes such

[8]https: //www.clips.uantwerpen.be/conll2003/ner.tgz.

as Vehicle-Maker, Pattern-Brand, etc. Parameter tables are the probabilities of existence or not (1 = exists, 0 = does not exist) of a named entity knowing the others (those in direct relation with it) in a text sequence (request or post).

2) Data generation with a Bayesian network: As a reminder, Bayesian networks consist of a set of random variables $V = \{V_1, V_2, ..., V_n\}$, and a set of parameters $\Theta = \{\theta_1, \theta_2, ..., \theta_m\}$ calculated in the learning stage. The number of parameters m = $\sum_{i=1}^{|V|} 2^{nb_i}$, where nb_i is the number of parents of nodes v_i. These two sets define a probability distribution [4] P on V which factorizes as follows:

$$P(V) = \prod_{v_i \in V} P(v_i | parents(v_i)). \quad (1)$$

The generation of data by a Bayesian network is therefore done by multiple imputation [19], i.e. for each observation (sample), we impute each variable using the others (inference process), starting with the one with the fewest parents. Ex: for the example illustrated in Fig. 2, we have 3 variables (Breakdown, Fuel, Battery). To generate an observation (line in a dataset), we start by imputing the Fuel and Battery variables (fewer parents in the graph). By taking for example the variable Fuel, we have 90% for 0 and 10% for 1, any process of simulation of random variables according to a probability distribution can do it. Assuming we want to get 1, to calculate the value of Battery, we need to infer the probability $P(Ba|Fu = 1)$ using the joint law given by Eq. (1), and so on for Breakdown.

By repeating this process several times, each time with different values from a simulation process, we fill a dataset whose rows contain 0 and 1. At the end of this process, we process each row of this dataset, replacing each variable containing 1 (exists) with a real value drawn randomly from a dictionary containing the automotive industry universe (all makes of cars, tyre sizes, etc.). Finally, an addition of noise and a well-studied permutation as well as a concatenation of the columns is applied on each line. The Figure 3 illustrates the process of associating the true values on a given example. One of the possible sequences after concatenating the columns and adding noise is as follows: "**I'm looking for a Michelin tyre for my Renault clio**".

This generation process requires a pre-trained model (structure + parameters) on a dataset. To do this we used the **PyAGrum**[9] framework that contains a module called Bayesian Network (BN). This module provides a flexible and efficient implementation of Bayesian Networks. Those can be learnt from data using the Learning module or generated randomly from several generators. The Leaning module can be used for structure and parameters BNs learning, where structure learning algorithms are a combination of handler for reading database, a score among (BD, BDeu, AIC, BIC/MDL), with possibly, some additional a *priori* (smoothing or Drichlet), component for scheduling local structure changes and a set of constraints that the user wishes to be satisfied. The latter icnludes also a structural constraints like requiring/forbidding arcs, limiting the indegrees and imposing a partial ordering on the BN nodes. And as far as learning parameters is concerned, the framework allows users to define BNs using traditional Conditional Probability Tables (CPT), but also using Logit models, aggregators (and, or, max, etc). CPTs are exploited in various inference algorithms as Lazy Propagation and Gibbs sampling for data generation. The BN parameters can also be learnt either by maximum likelihood or maximum aposteriori [6]. PyAgrum can be used with jupyter notebooks, in docker container or directly on web, it requires a **Python** version higher than 3.4, it can be installed like any other package using "pip" packages installer.

In our case, we've trained several models (on the 491 requests), using algorithms (K2, GHC, 3off2) in order to select the best based on evaluation metrics like: precision, recall and F-score.

There are no exploitable methods to evaluate a dataset generated by a Bayesian network, so we calculated for each algorithm, distances (Hellinger, etc) and divergences (KL divergence) between the structure and the probability distribution (BN parameters) of the model trained on the real data and that of the model trained on the generated data. Finally, we took the model for which the structure remains unchanged. The Table IV shows the algorithms used in the comparison as well as the scores obtained, with the best in bold.

Based on comparison results such as; Hellinger, recall, precision, F-score, between the models trained on real data and data generated by each algorithm illustrated by Tab. IV, the best algorithm is K2 (based on the maximization of score). The latter is very robust and efficient for the task of data generation. We will therefore use it in the rest of our study to generate query-type sequences.

3) Analysis of data generated by K2: In this part we present a comparison of the frequency of appearance (illustrated by the bar diagrams on the right of Fig. 4) of each named entity class, as well as the size of the sequences (illustrated by the 2 curves to the left of Fig. 4) between the actual data and that generated by K2.

We note that the frequency of appearance of the NE Vehicle-Segment is different in the generated dataset compared to the real one, which is due to the injection of expert knowledge (the NE Vehicle-Cartype and Vehicle-Segment must not appear together in a query) before the learning step of the parameters of the Bayesian network. This injection is achieved by prohibiting the dependencies (arcs) between the nodes which represent these 2 NEs . On the other hand, the frequency of appearance of the other NEs and the distribution of the size of the sequences are perfectly preserved, which motivates the choice of the K2 algorithm for the task of data generation.

[9]A Python wrapper, which provides a high-level interface to the part of aGrUM, allowing to create, model, learn, use, calculate with and integrate Bayesian networks and other graphical models.

[10]Three algorithms for learning the structure of a Bayesian network available in PyAgrum.

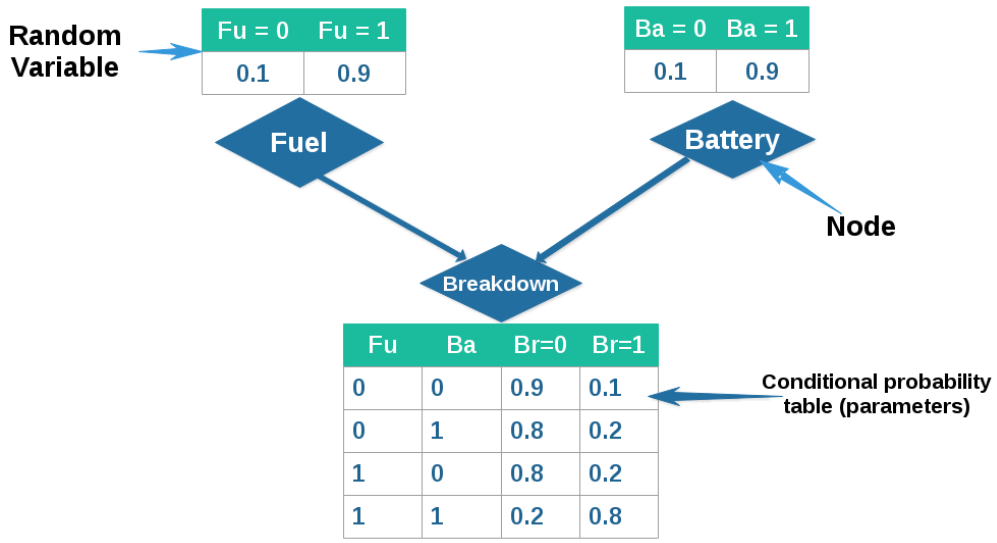


Fig. 2: Example of a Bayesian network with parameter tables.

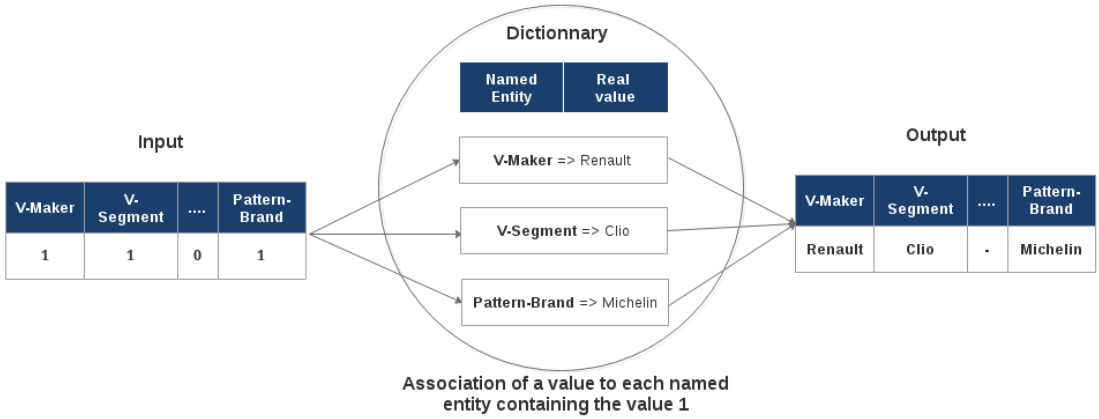


Fig. 3: Process of associating real values using a dictionary containing the automotive industry universe.

TABLE IV: Comparaison scores of the 3 algorithms "*K2, GHC, 3off2*"[10].

	klPQ	errorPQ	klQP	errorQP	hellinger	bhattacharya	jensen-shannon	recall	precision	f-score	dist2opt
K2	**0.001486**	0.000000	**0.001263**	0.000000	**0.021086**	**0.000222**	**0.000309**	**0.986301**	**1.000000**	**0.993103**	**0.013699**
GHC	0.004978	0.000000	0.005196	0.000000	0.040392	0.000816	0.001123	0.853333	0.984615	0.914286	0.147471
3off2	0.036641	0.000000	0.042804	0.000000	0.103878	0.005410	0.007088	0.870130	0.943662	0.905405	0.141563

kl(PQ and QP): called *Kullback-Leibler divergence*. It is used to measure the difference between two probability distributions (P, Q) over the same variable x. The KL divergence of P(x) from Q(x), is the information lost when the probability distribution P is used to estimate Q.

hellinger and bhattacharya: measures whiches used to quantify the similarity between two probability distributions [3], [18].

jensen-shannon: a symmetrized and smoothed version of divergence measure between two probability distributions (P, Q). It is defined by:

$$JSD(P||Q) = \frac{1}{2}(D(P||M) + D(Q||M)) \quad (2)$$

with M = $\frac{1}{2}$(P + Q) and $D(P||M)$ is the Kl divergence [5], [11].

dist2opt: represents the euclidian distance to the ideal point (precision=1, recall=1). It is defined by:

$$dist2opt = \sqrt{(1 - precision)^2 + (1 - recall)^2} \quad (3)$$

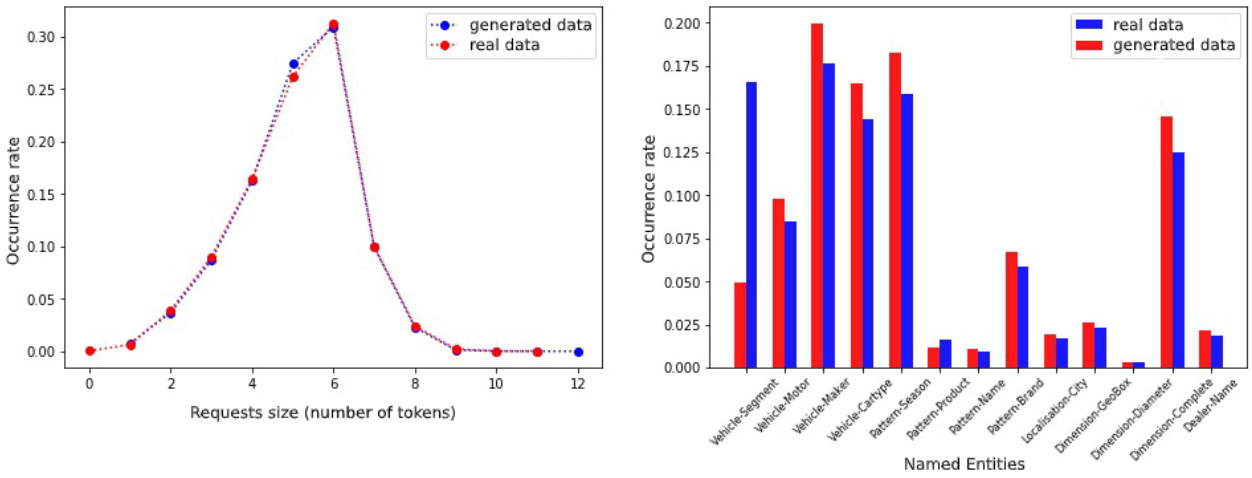


Fig. 4: Comparison of occurrence frequency of NEs and sequences length distribution, between generated (1 000 000 requests) and real (491 requests) data.

IV. Comparative study

A. Data used in the evaluation of the models

In this part, we present the volume as well as the description of the datasets, used in the evaluation of the models (see Tab. V).

Finally, to carry out statistical tests, we created eight other datasets by cutting the CONLL'03 in half and the queries in six. We will thus obtain the datasets: requests A, requests B, requests C, requests D, requests E, requests F, CONLL'03 A and CONLL'03 B.

B. Methodology

The five models collected from the state-of-the-art have first performed their task of recognizing and disambiguating a named entity on the various datasets. They were then assessed by evaluation metrics such as precision, recall and F-score. Precision measures the number of well-ranked NEs compared to the total number of NEs . The recall will measure the number of relevant NEs found compared to the total number of relevant NEs . The F-score is the harmonic mean of these two metrics. The confidence interval is given by the Wilson score interval, with 95% of confidence.

C. Statistical tests

For the statistical study we used the following tests [2]:

- **Friedman test**: the aim is to contest the following hypothesis: "All algorithms have equivalent performance". A confidence level of 0.95 will be used;
- **Nemenyi test**: once the Friedman test is validated, we can use it to show that one algorithm or a group of algorithms is significantly more efficient than another algorithm. The confidence level used will be 0.1 because of the small number of datasets to conduct this study.

D. Results

In this part, we present the results of our comparative and statistical study. To be able to compare machine learning and deep learning approaches, the variety of assessment datasets is very important. For this, we used the 10 aforementioned datasets (see Sec. IV-A for more details) to carry out our experiment.

The Table VI illustrates the results achieved by the five models, with the best in bold. We see from these results, that the performance in terms of named entity recognition of the Deep Learning family is better than that of the probabilistic family. Thus, we note that the bidirectional functionality of the LSTM model has a noteworthy impact on all types of data (requests or posts CONLL'03).

Once the Friedman test is validated for the different models, we proceeded to the Nemenyi test ensuing in the Fig. 5. The results of the Nemenyi tests show that the CNN-BLSTM model is significantly more efficient than all the others models, which confirms the results of Tab. VI.

V. Conclusion & perspectives

In this article, we have established a state of the art on named entity recognition (**NER**), after examining the automotive industry data available internally. We then presented five models belonging to two families of **Probabilistic** and **Deep Learning** approaches. This research allowed us to recover and even implement the architectures of selected models and to adapt them to our case study. Fronting an insufficient volume of data, we set up a data generation process in order to carry out a comparative study between these models. All of the developments carried out were tested at each stage to ensure that the quality of the results obtained was sufficient to proceed. Our comparative study (cf. Tab. VI and Fig. 5) on real data (CONLL'03) and generated data (queries generated by a Bayesian network), showed that the Deep Learning vision is significantly better than the probabilistic. In perspective, we

TABLE V: Description and volumetry of the two main datasets used in models evaluation.

Source	**Data from the automotive industry domain**		**CoNLL03**	
Type / Sequences size	**Generated requests / [1 to 23] tokens**		**Posts/ [1 to 124] tokens**	
Nb of sequences / Language	24 000 / french		6703 / english	
NEs and occurrence frequency	Dealer-Name	1.9%	LOC	16.89%
	Dimension-Complete	13%		
	Dimension-GeoBox	2.4%		
	Dimension-Diameter	0.3%	MISC	7.51%
	Localisation-City	1.7%		
	Pattern-Brand	6.1%		
	Pattern-Product	1%	ORG	16.4%
	Pattern-Name	1.9%		
	Pattern-Season	16.4%		
	Vehicle-Maker	18.3%	PER	13.68
	Vehicle-Segment	4.6%		
	Vehicle-Cartype	15%		
	Vehicle-Motor	8.7%	O	45.52%
	O	9.7%		

TABLE VI: Results of different approaches studied across the ten datasets.

Datasets	**Metrics**	**CRF**	**LSTM**	**CNN-GRU**	**CNN-LSTM**	**CNN-BLSTM**
Requests	**Precision**	0.8072 +/- 0.005	0.7200 +/- 0.005	0.8193 +/- 0.005	0.8371 +/- 0.004	**0.9651 +/- 0.002**
	Recall	0.8059 +/- 0.005	0.6421 +/- 0.006	0.7716 +/- 0.005	0.7903 +/- 0.005	**0.9622 +/- 0.002**
	F-score	0.8065 +/- 0.005	0.6788 +/- 0.006	0.7947 +/- 0.005	0.8132 +/- 0.005	**0.9636 +/- 0.002**
Requests A	**Precision**	0.8310 +/- 0.012	0.7183 +/- 0.014	0.8202 +/- 0.012	0.8378 +/- 0.011	**0.9649 +/- 0.006**
	Recall	0.8307 +/- 0.012	0.6412 +/- 0.015	0.7726 +/- 0.013	0.7912 +/- 0.012	**0.9620 +/- 0.006**
	F-score	0.8309 +/- 0.012	0.6776 +/- 0.014	0.7957 +/- 0.012	0.8138 +/- 0.012	**0.9634 +/- 0.006**
Requests B	**Precision**	0.8033 +/- 0.012	0.7189 +/- 0.014	0.8186 +/- 0.012	0.8359 +/- 0.011	**0.9647 +/- 0.006**
	Recall	0.8017 +/- 0.012	0.6407 +/- 0.015	0.7702 +/- 0.013	0.7888 +/- 0.013	**0.9622 +/- 0.006**
	F-score	0.8025 +/- 0.012	0.6776 +/- 0.014	0.7937 +/- 0.013	0.8117 +/- 0.012	**0.9634 +/- 0.006**
Requests C	**Precision**	0.8027 +/- 0.012	0.7210 +/- 0.014	0.8208 +/- 0.012	0.8384 +/- 0.011	**0.9658 +/- 0.006**
	Recall	0.8010 +/- 0.012	0.6427 +/- 0.015	0.7732 +/- 0.013	0.7915 +/- 0.013	**0.9627 +/- 0.006**
	F-score	0.8018 +/- 0.012	0.6796 +/- 0.014	0.7963 +/- 0.012	0.8143 +/- 0.012	**0.9643 +/- 0.006**
Requests D	**Precision**	0.8010 +/- 0.012	0.7209 +/- 0.014	0.8191 +/- 0.012	0.8379 +/- 0.011	**0.9652 +/- 0.006**
	Recall	0.7995 +/- 0.012	0.6424 +/- 0.015	0.7705 +/- 0.013	0.7901 +/- 0.013	**0.9622 +/- 0.006**
	F-score	0.8003 +/- 0.012	0.6794 +/- 0.014	0.7941 +/- 0.013	0.8133 +/- 0.012	**0.9637 +/- 0.006**
Requests E	**Precision**	0.8030 +/- 0.012	0.7207 +/- 0.014	0.8183 +/- 0.012	0.8375 +/- 0.011	**0.9658 +/- 0.006**
	Recall	0.8011 +/- 0.012	0.6425 +/- 0.015	0.7704 +/- 0.013	0.7894 +/- 0.013	**0.9627 +/- 0.006**
	F-score	0.8021 +/- 0.012	0.6794 +/- 0.014	0.7937 +/- 0.013	0.8128 +/- 0.012	**0.9643 +/- 0.006**
Requests F	**Precision**	0.8034 +/- 0.012	0.7204 +/- 0.014	0.8187 +/- 0.012	0.8370 +/- 0.011	**0.9651 +/- 0.006**
	Recall	0.8016 +/- 0.012	0.6430 +/- 0.015	0.7727 +/- 0.013	0.7908 +/- 0.013	**0.9622 +/- 0.006**
	F-score	0.8025 +/- 0.012	0.6795 +/- 0.014	0.7950 +/- 0.013	0.8133 +/- 0.012	**0.9637 +/- 0.006**
CONLL'03	**Precision**	0.8465 +/- 0.009	0.7929 +/- 0.010	0.7528 +/- 0.010	0.7759 +/- 0.010	**0.8626 +/- 0.008**
	Recall	0.8065 +/- 0.009	0.7517 +/- 0.010	0.7351 +/- 0.011	0.7666 +/- 0.010	**0.8627 +/- 0.008**
	F-score	0.8261 +/- 0.009	0.7718 +/- 0.010	0.7438 +/- 0.010	0.7712 +/- 0.010	**0.8627 +/- 0.008**
CONLL'03 A	**Precision**	0.8457 +/- 0.012	0.7932 +/- 0.014	0.7521 +/- 0.015	0.7806 +/- 0.014	**0.8648 +/- 0.012**
	Recall	0.8067 +/- 0.013	0.7548 +/- 0.015	0.7367 +/- 0.015	0.7762 +/- 0.014	**0.8704 +/- 0.011**
	F-score	0.8258 +/- 0.013	0.7735 +/- 0.014	0.7443 +/- 0.015	0.7784 +/- 0.014	**0.8676 +/- 0.011**
CONLL'03 B	**Precision**	0.8473 +/- 0.012	0.7926 +/- 0.014	0.7534 +/- 0.015	0.7712 +/- 0.014	**0.8604 +/- 0.012**
	Recall	0.8064 +/- 0.013	0.7486 +/- 0.015	0.7336 +/- 0.015	0.7572 +/- 0.015	**0.8552 +/- 0.012**
	F-score	0.8263 +/- 0.013	0.7700 +/- 0.014	0.7434 +/- 0.015	0.7641 +/- 0.014	**0.8578 +/- 0.012**

assume that the good performance of our reimplementation of CNN-BLSTM could be even better by adding a mechanism of evolution over time such as self-learning.

REFERENCES

[1] Jason P. C. Chiu and Eric Nichols. Named Entity Recognition with Bidirectional LSTM-CNNs. *arXiv:1511.08308 [cs]*, November 2015.

[2] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[3] Konstantinos G Derpanis. The bhattacharyya measure. *Mendeley Computer*, 1(4):1990–1992, 2008.

[4] F. Forbes. Modelling structured data with Probabilistic Graphical Models. *EAS Publications Series*, 77:195–219, 2016.

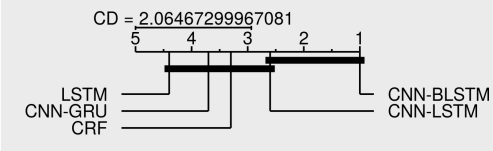
[5] Bent Fuglede and Flemming Topsoe. Jensen-shannon divergence and hilbert space embedding. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, page 31. IEEE, 2004.

[6] Christophe Gonzales, Lionel Torti, and Pierre-Henri Wuillemin. agrum: a graphical universal model framework. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 171–177. Springer, 2017.

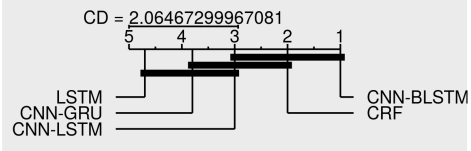
[7] Ralph Grishman and Beth M Sundheim. Message understanding conference-6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996.

[8] James Hammerton. Named entity recognition with long short-term memory. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 172–175, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

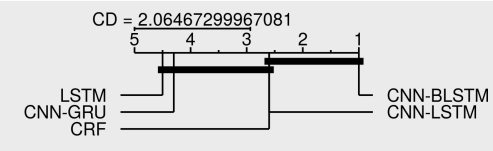
[9] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997.



(a) Nemenyi critical distance - Precision



(b) Nemenyi critical distance - Recall



(c) Nemenyi critical distance - F-score

Fig. 5: Nemenyi test results.

[10] Zhenyu Jiao, Shuqi Sun, and Ke Sun. Chinese lexical analysis with deep bi-gru-crf network, 2018.

[11] Shivakumar Jolad, Ahmed Roman, Mahesh C Shastry, Mihir Gadgil, and Ayanendranath Basu. A new family of bounded divergence measures and application to signal detection. *arXiv preprint arXiv:1201.0418*, 2012.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105. Curran Associates Inc., 2012.

[13] Andrew McCallum and Wei Li. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-enhanced Lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, 2003.

[14] Ryan McDonald and Fernando Pereira. Identifying gene and protein mentions in text using conditional random fields. *BMC bioinformatics*, 6(S1):S6, 2005.

[15] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[16] Erik F Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*, 2003.

[17] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

[18] Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

[19] Jim Young, Patrick Graham, and Richard Penny. Using bayesian networks to create synthetic data. *Journal of Official Statistics*, 25(4):549, 2009.

[20] Shaojun Zhao. Named Entity Recognition in Biomedical Texts Using an HMM Model. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and Its Applications*, JNLPBA '04, pages 84–87. Association for Computational Linguistics, 2004.

[21] GuoDong Zhou and Jian Su. Named Entity Recognition Using an HMM-based Chunk Tagger. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 473–480. Association for Computational Linguistics, 2002.