Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

A Weak Supervision Data Annotation Method For Large Scale Natural Language Understanding Tasks

Massimo Ruffolo, Ermelinda Oro, Francesco Visalli, Mariella Pupo, Francesco Luppino, Fausto Pupo

RT-ICAR-CS-19-02 **Giugno 2019**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 8-9C, 87036 Rende, Italy, URL: *www.icar.cnr.it*
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: *www.icar.cnr.it*
– Sezione di Palermo, Via Ugo La Malfa, 153, 90146 Palermo, URL: *www.icar.cnr.it*

A WEAK SUPERVISION DATA ANNOTATION METHOD FOR LARGE SCALE NATURAL LANGUAGE UNDERSTANDING TASKS *

A PREPRINT

Massimo Ruffolo
High Performance Computing and Networking Institute - National Research Counsil
Via Pietro Bucci 8/9C Rende (CS), 87036, Italy
massimo.ruffolo@icar.cnr.it

Ermelind Oro
High Performance Computing and Networking Institute - National Research Counsil
Via Pietro Bucci 8/9C Rende (CS), 87036, Italy
linda.oro@icar.cnr.it

Francesco Visalli
High Performance Computing and Networking Institute - National Research Counsil
Via Pietro Bucci 8/9C Rende (CS), 87036, Italy
francesco.visalli@icar.cnr.it

Mariella Pupo
Altilia srl
TechNest Unical Piazza Vermicelli Rende (CS), 87036, Italy
mariella.pupo@altiliagroup.com

Francesco Luppino
Altilia srl
TechNest Unical Piazza Vermicelli Rende (CS), 87036, Italy
francesco.luppino@altiliagroup.com

Fausto Pupo
Altilia srl
TechNest Unical Piazza Vermicelli Rende (CS), 87036, Italy
fausto.pupo@altiliagroup.com

June 29, 2019

ABSTRACT

Deep learning is gaining growing attention but it still poses many challenges to researches and practitioners. To learn accurate and useful deep learning models requires large labeled datasets precisely describing examples of the task to address. To facilitate the wide adoption of deep learning methods to solve real world use cases, new ways to simplify data labeling are strongly required. In this paper we propose a method and a system that adopts weak supervision and crowd-sourcing techniques to make domain experts capable to autonomously label large dataset with a small effort.

***K*eywords** Human-in-the-loop AI · Machine Learning · Deep Learning · Weak Supervision

*This work has been supported by POR-CALABRIA

1 Introduction

In the last years, deep learning is gaining growing attention from researchers and practitioners belonging to many application fields because it automates the feature engineering process. Despite such a success developing enterprise-grade deep learning-based applications still pose many challenges. In particular, it still needs large labeled datasets obtained by manual annotation of training data. This is one of the most costly bottlenecks to a wide and pervasive adoption of deep learning methods in real world use cases particularly when subject matter experts have to annotate textual data. For this reason, in industry and academy are emerging new approaches, often referred to as *weak supervision*, aiming to simplify the annotation process in order to make it more automatic and scalable, even though less accurate and noisier. Recently, [1] proposed data programming as a paradigm for semi-automatic datasets labeling, and Snorkel [2] a system that implement it. Bach et al. [3] at Google extended Snorkel in order to achieve better scalability and enterprise knowledge base re-usability.

In this work we describe a new weak supervision method for textual data named MANTRA (Model, ANnotate, TRain, and Assess) along with the system that implement it. The method simplifies how human annotate large documents and contents corpora available within organizations by reusable labeling functions, and by mechanisms for quickly training machine learning models to the highest possible quality. One of the most important contribution of our method consists in the fact that it enable to annotate large textual datasets for multiple NLP tasks such as: text classification, entity recognition, question answering, and sentiment analysis. The MANTRA system allows to easily create labeled textual datasets by adopting a human-in-the-loop mechanisms where *subject matter experts (SMEs)* are able to specify weak supervision in rapid, iterative, and interactive way without taking into account the details of labeling functions execution and model training over industrial scale datasets. A usual annotation pipeline in MANTRA has two steps: (i) dataset labeling by labeling functions; (ii) usage of a generative modeling approach that estimates the accuracy of different labeling functions based on their observed agreements and disagreements as in [1], but our also method exploits the manually curated ground truth (when it is available) to better estimate the accuracy; (iv) creation of a probabilistic (confidence-weighted) labeled training set.

In MANTRA datasets annotation can be performed in three different ways. First SMEs writes full text queries that enable to navigate, by a rich query language that combines keywords with sophisticated logical and distance operators, huge amount of textual contents and documents. Second SMEs can assign query results to specific labels (in this case full text queries acts as labeling functions). Third SMEs can write labeling functions by a textual data annotation language (TAL) that exploits various form of linguistic and domain knowledge representations such as: regular expressions, POS-Tagging, knowledge graphs (semantic networks), ontologies. Finally, users can perform point-and-click annotation actions aiming at manually creating annotations or updating those generated by the queries or TAL programs for defining a ground truth. It is noteworthy that full text queries and TAL programs can be combined. In this case results of full text queries represent passages where to apply TAL programs for generating labels for question answering tasks.

To test the effectiveness of the MANTRA method and system we describe result of an application to unlabeled textual data coming from a real world use case to prove that the method effectively helps in quickly creating training sets and speeding up machine learning adoption in vertical domains.

The rest of this paper is organized as follows...

2 Related Work

Since 2012, when AlexNet [4] won the ImageNet Large Scale Visual Recognition Competition (ILSVRC) beating others no neural network methods, the enormous strand of learning algorithms working on networks built of many layers (named deep learning) has begun. In the last several years, these deep architectures have contaminated also the Natural Language Processing (NLP) world, becoming the state of the art for many tasks of such field. Deep learning models are often hungry for data, as they are rich of parameters. Leveraging supervised learning methods to train these models requires a large amount of annotated data. Such datasets are enormously expensive to create, especially when domain expertise is required. Moreover, the specifications of a system often change, requiring the re-labeling of the datasets. Therefore, it is not always possible to rely on subject expert matters for labeling the data.
For the reasons described above, the interest in weak supervision systems is increasingly greater. One of the most interesting paradigms of weak supervision is the data programming [1]. Data programming is a weak supervision model that allows to annotate datasets through label functions, which are user-defined programs that provide labels for data. Labeling functions can express different weak supervision approaches such as pattern-based, distant supervision, weak classifiers or labeling function generators. The problem of considering different sources in order to label data is that often these data overlap. Moreover, the labeling functions output noisy labels. Data programming models the functions as a generative process, which automatically denoise the resulting dataset by learning the accuracies of the

labeling functions along with their correlation structure.
Data programming has lead to the creation of Snorkel [2], the first end-to-end system that, leveraging on data programming model, combine many sources of weak supervision to build training set rapidly. In order to get labeled data from different sources, Snorkel built an interface layer around the abstract concept of labeling function. It lays the foundation for how data programming should be used. Once the functions have been written, Snorkel automatically learns a generative model over these, which allows it to estimate their accuracies and correlations, with no ground-truth data needed. The output of this process is a set of probabilistic labels that can be used as input for discriminative models (e.g. SVM classifiers or neural networks) that should use a noise-aware loss function.
Snorkel framework has been extended by Snorkel DryBell [3], a production scale weak supervision management system. Snorkel DryBell underlines the importance of weak supervision in industrial and other organizational deployments, where the savings, intended both in terms of costs and time to realize these datasets, is fundamental.
However, frameworks like Snorkel can still take away non-technical users, such as subject domain experts, from weak supervision. Other tools like BRAT [5] guide the labeling process with an annotation interface. BRAT uses the NLP technology to support the human annotation for most NLP tasks. For example, the BRAT interface allows the user to mark a span of text for annotation by selecting it with the mouse or by double-clicking on a word. In this way the annotation productivity is improved and the human and financial cost of annotation is reduced.
Another technique for alleviating the cost of human experts is the active learning. It is the case of tools, like Prodigy [6] or Figure Eight [7], that perform the so called human in the loop process. In active learning the algorithm inspect the data and requests to humans which data should be labeled.
Semi-supervised learning is another approach to let the model require less annotated data. In semi-supervised learning both unsupervised and supervised learning are exploited. A use case is the Bidirectional Encoder Representations from Transformers (BERT) [8]. BERT is mainly a model to create strong contextual word embeddings but it is also a generic architecture for many NLP tasks. In BERT, word embeddings are learned in an unsupervised way by training the model on large corpus. The final model is then fine-tuned in order to learn various specific NLP tasks. Fine tuning requires much less data than learn the task from scratch. The idea behind fine tuning is that most of the features have already been learned and just need to be specialized for the specific task.

3 The Method

This section describes the proposed method named MANTRA (Model, ANnotate, TRain, and Assess) and the system that implement it. This method is tailored to provide highly flexible, scalable, and general ways to annotate texts for creating training sets for numerous NLP tasks such as: text classification, entity extraction, question answering, sentiment analysis. The main idea behind MANTRA is the combination of: (i) a declarative Text Annotation Language (TAL) that enables both SMEs and machine learning experts to easily create rich, expressive, reusable, and scalable labeling rules without taking into account execution details, (ii) a modeling approach to estimate the accuracies of the different labeling functions based on their observed agreements and disagreements. In this case users write labeling functions that can be very simple or sophisticated programs also exploiting already modeled domain knowledge available within enterprises. The complexity level of labeling functions depends on the machine learning task and on the kind of information to recognize. For example, a sentiment analysis problem for a financial or e-commerce use case can require more sophisticated labeling functions than a simple text classification problem. (iii) The creation and assessment of the machine learning model derived from the labeled training set. If accuracy is not satisfying these three steps are repeated in order to define a new model o the base of new and richer dataset. Steps described above makes MANTRA an iterative and interactive method that enables both SMEs, with only domain expertise, and machine learning experts to work in cooperation to create training sets for machine learning algorithms in the field of NLP.

3.1 Background: The Text Annotation Language

This section describes the Text Annotation Language (TAL) adopted in the MANTRA weak supervision method. TAL is a declarative language that enables to create labeling functions having the form of a first order logic rules. Each TAL rule is a composition of predicates that express context free grammars extended by constructs that enable to query linguistic and semantic knowledge contained in dictionaries, taxonomies, and knowledge graphs. This way users can define a wide range of labeling functions from simple compositions of regular expressions, to sophisticated programs that make use of dictionaries, knowledge graphs and NLP information like POS-tags, lemmas, and parse-tree structure of sentences. TAL allows to write reusable labeling functions that produce rich and accurate annotations that facilitate the creation of large labeled dataset. In the MANTRA weak supervision method, multiple domain experts can take labeling functions from a repository or directly write them to generate labeled examples. Results of multiple annotation

functions for the same dataset are then evaluated by the agreement and disagreement model to chose the best training set possible. More in detail a labeling function in TAL has the following syntactical form:

`head arrow body`

where head is a predicate having the form:

$$predicate(Variable\_name_1, \ldots, Variable\_name_n)$$

that computes the piece of the input text corresponding to the pattern expressed in the body. The head of the labeling functions may have, also, a set of variables (which names start with a capital letter) representing attributes used in labeling texts such as the position of the labeled piece of text within the input string, and other linguistic or structural text properties that can be used as labels metadata. The body is a pattern having the form:

$$predicate_1(), \ldots, predicate_n() \ AND \ \#builtIn() \ \{variables\_assignements\}.$$

that describes how to recognize pieces of the input text to labels by using a combination of other predicates, logical operators, and built-in functions. A predicate in the body may consist in the head of another labeling functions, this way it is possible to define labels as the result of a complex text annotation process expressed as a composition of labeling functions. This feature of the TAL augments its modularity and re-usability allowing to create libraries of labeling functions specialized in specific annotation tasks that can be reused and combined to create new labels.

The arrow is the *sequence operator* that expresses how the pattern described in the body must be evaluated or, in other words, how text pieces, in the input text, can be arranged to form the pattern to assign to the head. Example of arrows are: (i) `"<-"` (i.e. strict sequence) indicating that piece of text recognized by predicates in the body must be strictly sequential in reading order: (ii) `"«-"` (i.e weak sequence) indicating that text portions recognized by predicates can be scattered between other pieces of text in reading order.

Built-in functions allow to perform linguistic operations and other complex tasks with the aim to evaluate specific text properties that can be used to identify syntactical and semantic features in the input text. Some examples of built-in functions are the following:

- `#chunk`: identifies set of tokens identified by shallow parsing operations and consisting in nominal/noun and verb phrases.
- `#regex`: searches for basic syntactical patterns by java-style regular expressions. This built-in is useful to search specific textual patterns in the input text.
- `#dictionary`: finds terms listed in a text file called dictionary.
- `#lemma`: recognizes the lemma of a word.
- `#posTag`: returns the POS-Tag of a work in the input text.
- `#synonym`: returns the list of synonym of a word given a specific sense.
- `#sentimentClassifier`: return the sentiment polarity of a sentence or text chunk.

4 Use Cases

In general, the main problem to face in defining accurate and effective machine learning models in specific domains is the construction of labeled datasets to use for training, testing and validating them. In this section we present a some use cases coming from real world application scenarios that describe how the MANTRA weak supervision method helps in simplifying dataset labeling speeding up the construction of accurate machine learning models for different NLP tasks.

In the next section we focus our attention on the aspect based sentiment analysis problem showing how MANTRA helps in creating labeled dataset for vertical use case involving this NLP task. Other NLP tasks will be addressed in new versions of this paper.

4.1 Constructing a Labeled Datasets for Aspect-Based Sentiment Analysis

Let consider a producer of laptop computer that would like to understand how different laptops are perceived by online buyers. To accomplish this task the producer needs a machine learning tool falling in the area of Aspect-Based Sentiment Analysis (ABSA). This way s/he will be able to understand buyers evaluation criteria and which laptop features have more important impact on the evaluation. In literature there are many machine learning models already trained for the ABSA problem, but suppose that only scientific prototypes are available for laptop computers. So, how to define an industrial-grade sentiment analyzer for laptop computers quickly? In general, the problem is: how to realize a working machine learning model quickly and accurately?

Basically the user has first of all to collect buyers reviews from consumer electronics on-line stores, then s/he has to define a set of features (i.e. aspects) of laptop computers. To create the set of features s/he can use descriptions of laptop computer s/he already has or collected them from on-line stores by simple web scraping tasks.

MANTRA allows the user to index the review dataset collected from the web and write declarative labeling functions that reuse already existing knowledge about domains and/or linguistic tasks to automatically label the dataset collected from the web to create training, test, and validation sets.

```
Tool::NLTK
Language::English
#import::"english_sentiment_analysis-laptop_aspects"
?aspect_sentiment
@Sentence
aspect(AspectLemma)<- Aspect:#memberLemma(AspectLemma,"english_sentiment_analysis-laptop_aspects").
aspect_sentiment(Sentence,Sentiment,Polarity,AspectLemma) <- X:@string
[CR(PSP:#sentimentClassifierSPACY(Sentiment,Polarity),X) AND CR(AS:aspect(AspectLemma),PSP)]
{Sentence:=PSP;Classification:=Polarity;Aspect:=AS;}.
```

In the labeling function above instructions and predicates have the following.

- `Tool::NLTK`, expresses that the system uses the NLTK NLP framework to process text.
- `Language::English`, expresses that the label function expects english text in input.
- `#import::"english_sentiment_analysis-laptop_aspects"`, imports the dictionary of laptop computer features to use to define aspects of the specific ABSA problem.
- `?aspect_sentiment`, identifies the predicate that produce final labels.
- `@Sentence`, expresses that the system works on single sentences.
- `aspect(AspectLemma)`, computes lemmatization of all aspects in the aspect dictionary.
- `aspect_sentiment(Sentence,Sentiment,Polarity,AspectLemma)`, represents the predicate expressing the label. It produce, in the entire `Sentence`, the labeling of the `Sentiment` expression along with its `Polarity`, and of the lemma of the aspect related to sentiment `AspectLemma`. Roughly speaking it labels aspects and related sentiments when text chunks (i.e. noun or verbal phrase) contain at the same time an aspect of a laptop computer and a sentiment expression. More in detail: (i) the built-in predicate `#sentimentClassifierSPACY` computes a text chunk containing a sentiment expression along with its polarity by using SPACY [2]; (ii) the first operator `CR` expresses that the text chunk must be contained in the input sentence `X`; (iii) the predicate `aspect` computes all aspects in the dictionary that appear in the input sentence, and (iv) the second operator `CR` express that the aspect in the dictionary must be contained into the same text chunk of the sentence `X` where is located the sentiment expression.

The above example, shows how MANTRA leverages a declarative language that enables to write reusable text labeling functions. In particular, it is worthwhile noting that implementation details about how to use SPACY to annotate aspects are completely hidded to the user, and, more importantly user can write many different labeling functions having the same form just changing the built-in predicate `#sentimentClassifierSPACY`. For example, in the above labeling function users can exploit `#sentimentClassifierNLTK`, `#sentimentClassifierSTANFORD`, and `#sentimentClassifierALLEN` to automatically annotate the same dataset with already existing sentiment classification algorithms.

Furthermore multiple domain and/or sentiment analysis experts can write different labeling functions that generate labeled examples regarding very specific aspects. Such an approach apply weak supervision techniques that speed up machine learning adoptions within business processes in real world use cases. This way users will have extensive dataset labeling by using different already available datasets to compute a subset of labeled examples as those that agreed [1] by different adopted algorithms.

5 Experiments

See the paper M. Ruffolo, E. Oro, et al. "Review Reading Compression and Aspect Based Sentiment Analysis in E-Commerce Reviews ".

[2] https://spacy.io

6 Conclusions

In this work we presented MANTRA a scalable weak supervision method that enables to address the problem of quickly creating labeled textual dataset to use in training machine leaning algorithms for different NLP tasks. The proposed method is based on a text annotation language that allows to express labeling functions in declarative way. Such functions are highly modular and reusable and hides implementation details to end users. We have show the effectiveness of TAL by an example in the area of aspect based sentiment analysis. Experiments reported in the paper M. Ruffolo, E. Oro, et al. "Review Reading Compression and Aspect Based Sentiment Analysis in E-Commerce Reviews " show how TAL concretely speed up and scale up the labeling process enabling to learn machine learning models having state-of-the-art performances.

References

[1] Alexander J. Ratner, Christopher De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. Data programming: Creating large training sets, quickly. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3567–3575, 2016.

[2] Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *PVLDB*, 11(3):269–282, 2017.

[3] Stephen H. Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alexander Ratner, Braden Hancock, Houman Alborzi, Rahul Kuchhal, Christopher Ré, and Rob Malkin. Snorkel drybell: A case study in deploying weak supervision at industrial scale. In Peter A. Boncz, Stefan Manegold, Anastasia Ailamaki, Amol Deshpande, and Tim Kraska, editors, *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019.*, pages 362–375. ACM, 2019.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012.

[5] Pontus Stenetorp, Sampo Pyysalo, Goran Topic, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. brat: a web-based tool for nlp-assisted text annotation. In Walter Daelemans, Mirella Lapata, and Lluís Màrquez, editors, *EACL 2012, 13th Conference of the European Chapter of the Association for Computational Linguistics, Avignon, France, April 23-27, 2012*, pages 102–107. The Association for Computer Linguistics, 2012.

[6] Prodigy. `https://prodi.gy`.

[7] Figure Eight. `https://www.figure-eight.com`.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.