Understanding language - one evaluation at a time

Marco Concetto Rudilosso

***Abstract*—In this survey I will firstly introduce the reader to the basic concepts that power the modern NLP research, describing concepts like neural networks, optimizations techniques and word embeddings. Then, I will describe the status of the art of evaluation in the NLP community, what problems exist and solutions. I will conclude exploring by future directions in evaluating NLP systems.**

INTRODUCTION

A brief history of NLP

Since the very beginning of the computing revolution in the 50s, the understanding of natural language has been a challenge for artificial intelligence. In his famous article "Computing Machinery and Intelligence"[1] Alan Turing proposes the Turing test as a standard by which we can judge intelligence. He states that a machine that is indistinguishable from a human in a conversation, would be seen as intelligent.

The 50s were a period of optimism, especially after the success of the Georgetown experiment, where researchers where able to translate more than 60 Russian phrases into English. It was stated that Machine Translation (MT) would have been a solved problem in 3-5 years[2]. In 1966, after the ALPAC Report, which concluded that MT was nowhere near being solved, the optimism faded[3].

Most of the systems developed in the 70s and 80s were based on hand-written rule and their success was mostly based on the ability of their users to adapt to the restricted use cases that they understood. In the 90s a new trend appeared: the application of statistical methods to natural language. This new approach had incredible results, leading to strong improvements in the performance of different NLP systems[4].

Such methods paved the way for what we now refer as Machine Learning approaches, leading to a revolution in NLP.

A vision for the future

Due to recent advances in the Machine Learning field, especially Deep Learning, the belief, that we could reach the potential of creating a system that can, by itself, learn and solve any problem, has surfaced.

It has been shown that using black-box optimizations makes possible to tune hyper-parameters automatically[5]. Moreover, the idea of substituting optimization methodologies such as SDG (Stochastic gradient descent), or the more recent Adam, with a neural optimizer has shown good results[6].

Given these results, hopes are high that we will be able to automate most of the time consuming tasks in Machine Learning, such as a correct choice of hyper-parameters.

The evaluation problem

In order to actually achieve the vision outlined in the previous section, we will need a rigorous and repeatable method of evaluating new techniques and their improvements.

In this literature review, we will mainly focus on NLP focused Machine Learning, this decisions entails the list of obstacles which arise when it comes to finding a good evaluation methodology:

- Intrinsic vs Extrinsic evaluation
- What tasks we should use
- What datasets we should test on
- What baselines we should test against
- Fair hyper-parameter tuning

Such method is needed for 2 main reasons: firstly, when new papers are published, stating an improvement over the state of the art, it would be beneficial if the evaluation part of the paper was standardized and repeatable. If different research groups were to test their techniques over different datasets and tasks, a comparison between their discoveries would lead to incoherent results.

Secondly, when we let programs optimize our algorithms, we must have a good evaluation strategy, vice versa it would lead the optimizer to bad results.

BACKGROUND KNOWLEDGE

Modern NLP makes large use of neural networks, thus an elementary understanding of how this models are created, how they are optimized and what kind of input they are fed is crucial before diving into mode advanced topics. In this section I will briefly explain some of these concepts.

Neural networks

I will briefly explain the basics of a feed-forward neural network. I must inform the reader that other kind of neural network exist[7].

Neural networks, as it is easy to imply by their names, are inspired by the understanding of how our brain works. In the attempt of emulating human behavior, this models have been designed to contain neurons, that are computational units. Every unit applies a set of weights to its input, group them using some function, usually summing them, and then applies a non-linear function, passing the result to its output. Multiple neuron can be used together to form Neural Networks.

A neural network can be visualized as in figure 1. Each hidden layer node is a neuron with its non-linear function, the arrows are the weights (which are not shown).

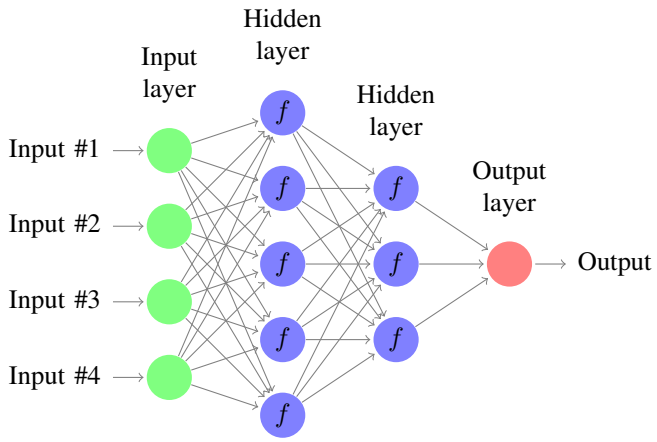

Fig. 1. Neural network with 2 hidden layers

Mathematical notation: As much as one can like the brain metaphor, understanding the mathematical notation of Neural Networks (NNs) is crucial in order to code them and to read papers. Due to this, I will now talk about NNs using matrices and vectors.

The simplest NN is the Perceptron:

$$\begin{gathered} Perceptron : \mathbb{R}^{in} -> \mathbb{R}^{out} \\ Perceptron(x) = xW + b \\ x \in \mathbb{R}^{in},\ W \in \mathbb{R}^{in \times out},\ b \in \mathbb{R}^{out} \end{gathered} \tag{1}$$

The Perceptron takes the input vector, multiplies it by a matrix and adds some bias. This model is quite constrained, due to the fact that it can only reproduce linear function, because of its nature.

By adding an hidden layer and a non-linear function, which is often called activation function, we have a much more powerful machinery. It can be described by the following equation:

$$NN_1(x) = g(xW_1 + b_1)W_2 + b_2 \tag{2}$$

Where g is a non-linear function. Adding a second layer is relatively straightforward:

$$NN_2(x) = g_2(g_1(xW_1 + b_1)W_2 + b_2)W_3 + b_3 \tag{3}$$

Where g_1 and g_2 are 2 non-linear functions. Examples of non-linear functions are the sigmoid, the hyperbolic tangent and the Rectifier (ReLU).

Training methods

In order to make these models actually produce a meaningful result, we need to train them. This means that we need to find a set of parameters, that given a loss function, which judges how good the outputs of our NN are, it minimize the loss.

Most of the methods used work by computing the loss function over a set of inputs, then the gradient with respect to the error is calculated, and the parameters are updated by moving them in the opposite direction of the gradient. This is why they are called gradient-descent methods.

SGD: SGD, stochastic gradient descent, is one of the simplest of such methods and it works as follow:

Algorithm 1 SGD

```
procedure SGD(f, xs, ys, L, learningRate)
    params ← initialize()
    while stopping criteria not met do
        (x, y) ← pick(xs, ys)
        loss ← L(f(x, parameters), y)
        gradient ← grad(loss, parameters)
        params ← params − learningRate ∗ gradient
    return params
```

SGD takes a function f, which it tries to optimize with regards to a set of parameters, a set of possible inputs xs paired with a set of desired output ys. It then takes a loss function, which describes how well the function f is behaving, and a learning rate, which scales the speed of optimization.

First of all it initialize the parameters to some value, it could be random or set by the programmer. While we have not reached the stopping criteria, it samples a pair of x and y, calculates the loss and then updates the parameters in the opposite direction of the gradient of the loss function w.r.t the parameters.

Pure SGD approximate the loss function on the entire dataset using a single local one, this may lead to inaccurate gradient calculation. In order to mitigate this, sampling the gradient on subsets of the entire dataset can be a valid solution. This subsets are often referred as mini-batches.

Other training methods exist[8], which have advantages over SGD, such as AdaGrad, AdaDelta and Adam.

Word embeddings

When applying Deep Learning techniques to a problem, we feed into a neural net inputs encoded in a mathematical form. For example images could be represented as matrices of RGB values. When it comes to NLP, words, or sentences, do not have an immediate mathematical representation.

Previous to the work carried on by different research groups in 2013-2014[9][10][11], words were represented using sparse vectors, where each dimension was a singular feature of the word. For example one feature could have been the previous word, another could have been the presence of a specific prefix or suffix and so on.

Thanks to the aforementioned research groups, we now have the ability to represent words using dense vectors, which have a low dimensionality (compared to sparse vectors) and where each dimension does not just represent a single feature. The advantages of dense vectors are: low dimensionality, thus decreasing the compute power needed; a greater generalization power, dense vector are better at transfer information between similar words[7].

word2vec: word2vec[9] is one of the techniques that is now being used to create dense vector representation of words. It works on any unlabeled corpus. word2vec creates vector representations using the skip-gram model, where we try to

maximize the probability of predicting the right context, the words around a target, given a specific word.

It works by using negative sampling, an approach whose goal is to approximate the softmax function. Given 2 sets of words D, that contains true word-contexts pairing, and $\neg D$, that contains randomly sampled words and contexts, the model tries to maximize the probability that the a pair comes from the true pairings and not the random ones. By doing so, it produces for each word, that it is contained in the corpora fed into it, a dense vector.

THE EVALUATION PROBLEM

After the early days of Deep Learning applied to NLP, where significant improvements were made, we are now in a different period. Improvements are not as significant and a vast amount of work is done simultaneously, making it incredibly difficult to properly evaluate the novelty and the significance of a new discovery. The need for a more methodological approach to evaluation is increasing, in order to drive the progress to more powerful NLP systems.

Intrinsic vs Extrinsic

When it comes to assessing the quality of word embeddings, two approaches can be taken: intrinsic or extrinsic evaluation.

Intrinsic Evaluation: Usually, it evaluates the quality of given word embeddings by their ability to predict the similarity of pairs of humanly annotated words. Each pair of words has a similarity score given by humans and it is compared with the vector similarity of the word embeddings representing the 2 words.

Extrinsic Evaluation: It consists in using the word embeddings as inputs in models that solve downstream tasks, such as part of speech tagging (POS), chunking or name entity recognition (NER). Good word embeddings are the ones which perform well in these tasks according to metrics related to each task.

Intrinsic evaluation has the advantage that is faster than extrinsic evaluation and it is assumed that they can usually predict the performance of word embeddings in downstream tasks. Unfortunately, this is not true for most of the intrinsic evaluation datasets, as shown in table I, which summarize the work done in [12]

TABLE I
CORRELATION BETWEEN INTRINSIC AND EXTRINSIC EVALUATIONS.

	Chunking	NER	POS
WordSim-353	-0.90	-0.75	-0.88
MC-30	-0.87	-0.77	-0.90
MEN-TR-3K	-0.98	-0.83	-0.97
MTurk-287	-0.57	-0.29	-0.50
MTurk-771	0.28	0.37	0.27
Rare Word	-0.57	-0.29	-0.50
YP130	-0.82	-0.93	-0.50
SimLex-999	1.00	0.85	0.98

Replication is the key

Replication in scientific papers is an incredibly important feature of an experiment, as it adds a layer of certainty that the results of original authors were accurate. This is because when a study is replicated, the person that is carrying it follows as exactly as possible the methodologies described in the paper, whereas the original authors could have unconsciously (or consciously) made choices that changed the results of the experiment after it was started.

Going back to NLP, given the complexity of the systems we are developing and the amount of parameters and hyper-parameters, replication is an ongoing issue. As shown in [13], after the release of doc2vec[14], a model to create embeddings for sentences, group of sentences and documents, the replication of the results of the paper[14] was an ongoing controversy in the NLP community.

In light of these kind of events, the RepEVAL workshops were organized. The principal goal of the workshop is to foster the discussion about the evaluation techniques that we use to asses the quality of general-purpose representations of words, sentences and documents.

A suite of evaluations tasks: In the RepEVAL 2016 workshop, one of the accepted papers[15] proposed an interesting solution to the evaluation problem for word embeddings: create a suite of moderate size tasks that tests different properties of the embeddings. They test syntactic properties with tasks like POS and chunking. They use NER and IOB tagging to test the semantic properties at the word level. Semantic properties at the sentence level are evaluated through sentiment classification and question classification.

They released the evaluation suite as a script and also made it available on-line, thus making it easy to have a fair and replicable evaluation.

Where is the lie?

As stated in the previous sub-section, replication is an important role in the evaluation of embeddings and models. Nonetheless, replication is a necessary but not sufficient to show that a new proposed technique actually improves on the status of the art. An experiment could be replicable but faulty at the same time.

The baseline problem: When new research introduces architectural and algorithmic improvements over vanilla NLP systems, we should be mainly interested in understanding if these new proposed techniques would actually have an improvement over the state of the art systems.

This appears to be a difficult problem to tackle, as the choice of a baseline, the so called vanilla implementation, can change the results of a study. If a very poor one is chosen, the results of a study could be meaningless in a real world scenario, where much more complex systems are deployed.

Neural machine translations has a variety of datasets which are widely used and a score system, called BLEU, that quantitatively evaluates the accuracy of a model. Nonetheless, understanding the real world improvements of new techniques remains a daunting task. As shown in [16] a stronger baseline can show how recently published methods can actually lead

to no improvements over existent systems. This is graphically demonstrated by the table II, where different recently proposed techniques do not show any improvement over the hardened baseline in 3 out of 4 cases.

TABLE II
BLEU SCORES EN-FR

	BLEU Score
Baseline	37.3
Baseline + Dropout	37.2
Baseline + Lexicon Bias	37.1
Baseline + Pre-Translation	36.6
Baseline + Bootstrapping	37.4

The hyper-parameters dilemma: Even if we were to choose the right baseline, or if we were to prove that our model really improves on the state of the art, would we be always be right? Unfortunately the answer is no.

The models that we use are sensible to hyper-parameters and this could affect their performance. In [17] 3 different recurrent neural networks architectures are compared: LSTM[18], RHN[19] and NAS[20]. The last two are recent advancements of the recurrent neural networks architecture which promises improvements over the state of the art. LSTM are relatively old RNN models.

After fine-tuning the hyper-parameters of these 3 models using Google Vizier[5], a black box optimizer, LSTM shows to have better results over the other models, thus contradicting published claims.

FUTURE HORIZONS AND CONCLUSION

Evaluation is a key requirement for progress in NLP, problems do exist, but I am hopeful that they will be overcome. Firstly, Google Vizier[5] has shown interesting results in automating hyper-parameters, by improving internal systems at Google notably, but also allowing to understand the differences and performances of a variety of RNN architectures over a large amount of data[21]. Thanks to the highly scalable nature of this system, running such experiments is now possible, giving us new insights over the advantages and disadvantages of different models.

Secondly, as shown in [16], the choice of optimizer in the training phase of a model can make a difference in its final performance. Could we automate this? Maybe. It has been shown that it is possible to create a neural optimizer[20] that can create new update rules for the model that is being trained. Such optimizer uses reinforcement learning in order to maximize the performance of a model after a few epochs. This system has been able to discover new update rules that perform well, opening up a new ways of designing update rules.

Thirdly, it would be interesting to see if Transformers could be used as new baselines for different NLP tasks[22]. They are a recently proposed architecture that uses simple feed-forward neural networks paired with attention, which shows improvements in machine translation over the state of the art RNN models. I propose such models as baselines because of their strong performance, but also because they are faster to train, thanks to their reduced complexity compared to RNNs.

Finally, it would be beneficial if, in line with the work done in [15], more evaluations suites could be made. One such example is the CommAI initiative[23], whose goal is to create a set of tasks to evaluate intelligence with a language focused approach.

REFERENCES

[1] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[2] J. Hutchins, "Example-based machine translation: a review and commentary," *Machine Translation*, vol. 19, no. 3, pp. 197–211, 2005.

[3] K. S. Jones, "Natural language processing: a historical review," in *Current issues in computational linguistics: in honour of Don Walker*. Springer, 1994, pp. 3–16.

[4] M. Marcus, "New trends in natural language processing: statistical natural language processing," *Proceedings of the National Academy of Sciences*, vol. 92, no. 22, pp. 10 052–10 059, 1995.

[5] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1487–1495.

[6] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," *arXiv preprint arXiv:1709.07417*, 2017.

[7] Y. Goldberg, "A primer on neural network models for natural language processing," *J. Artif. Intell. Res.(JAIR)*, vol. 57, pp. 345–420, 2016.

[8] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[10] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Advances in neural information processing systems*, 2013, pp. 2265–2273.

[11] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[12] B. Chiu, A. Korhonen, and S. Pyysalo, "Intrinsic evaluation of word vectors fails to predict extrinsic performance," in *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP*, 2016, pp. 1–6.

[13] J. H. Lau and T. Baldwin, "An empirical evaluation of doc2vec with practical insights into document embedding generation," *arXiv preprint arXiv:1607.05368*, 2016.

[14] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 1188–1196.

[15] N. Nayak, G. Angeli, and C. D. Manning, "Evaluating word embeddings using a representative suite of practical tasks," *ACL 2016*, p. 19, 2016.

[16] M. Denkowski and G. Neubig, "Stronger baselines for trustable results in neural machine translation," *arXiv preprint arXiv:1706.09733*, 2017.

[17] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," *arXiv preprint arXiv:1707.05589*, 2017.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] J. G. Zilly, R. K. Srivastava, J. Koutník, and J. Schmidhuber, "Recurrent highway networks," *arXiv preprint arXiv:1607.03474*, 2016.

[20] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[21] J. Collins, J. Sohl-Dickstein, and D. Sussillo, "Capacity and trainability in recurrent neural networks," *stat*, vol. 1050, p. 28, 2017.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[23] M. Baroni, A. Joulin, A. Jabri, G. Kruszewski, A. Lazaridou, K. Simonic, and T. Mikolov, "Commai: Evaluating the first steps towards a useful general ai," *arXiv preprint arXiv:1701.08954*, 2017.