Read time: 19 minutes

Deep Learning for NLP: Advancements & Trends

Javier Tue, Dec 12, 2017 in **#MACHINE LEARNING**

DEEP LEARNING NLP RECAP

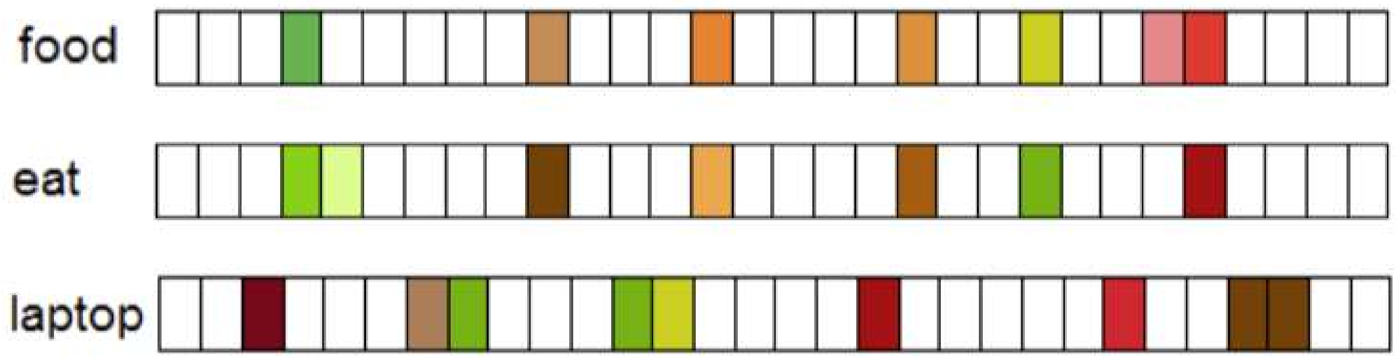
Over the past few years, **Deep Learning** (DL) architectures and algorithms have made impressive advances in fields such as **image recognition** and **speech processing**.

Their application to **Natural Language Processing** (NLP) was less impressive at first, but has now proven to make significant contributions, yielding state-of-the-art results for some common NLP tasks. **Named entity recognition** (NER), **part of speech (POS) tagging** or **sentiment analysis** are some of the problems where **neural network models** have outperformed traditional approaches. The progress in **machine translation** is perhaps the most remarkable among all.

In this article I will go through some recent advancements for NLP that rely on DL techniques. I do not pretend to be exhaustive: it would simply be impossible given the vast amount of scientific papers, frameworks and tools available. I just want to share with you some of the works that I liked the most. I think the last months have been great for our field. The use of DL in NLP keeps widening, yielding amazing results in some cases, and all signs point to the fact that this trend will not stop.

From training word2vec to using pre-trained models

Arguably, **word embeddings** is the most known technique related to DL for NLP. They follow the **distributional hypothesis**, by Harris (1954), according to which words with similar meaning usually occur in comparable contexts. For a detailed explanation of word embeddings, I suggest you to read **this great post by Gabriel Mordecki**.



Example of distributional vectors of words.

Image source

Algorithms such as **word2vec** (Mikolov et al., 2013) and **GloVe** (Pennington et al., 2014) have been pioneers in the field, and although they cannot be considered as DL (neural network in word2vec is shallow and GloVe implements a count-based method), the models trained with them are used as input data in a lot of DL for NLP approaches. This is so true that using word embeddings in our field is now generally considered a good practice.

At the beginning, for a given NLP problem that required word embeddings, we tended to train our own model from a big corpus that was domain related. This is not, of course, the best way to democratize the use of word embeddings, so pre-trained models slowly began to arrive. Trained on Wikipedia, Twitter, Google News, web crawls, and more, these models allow you to easily integrate word embeddings to your DL algorithms.

The latest developments confirmed that pre-trained word embedding models is still a key issue in NLP. For example **fastText**, from the **Facebook AI Research (FAIR) lab**, released **pre-trained vectors in 294 languages**, which represents a great work and contribution to our community. Besides the wide number of languages, this is very useful because fastText uses character n-grams as features. This allows fastText to avoid the OOV (out of vocabulary) problem, since even a very rare word (e.g. specific domain terminology) will probably share some character n-grams with more common words. In this sense, fastText behaves better than word2vec and GloVe, and outperforms them for small datasets.

However, although we can see some progress, there is still a lot to do in this area. The great NLP framework **spaCy**, for example, integrates word embeddings and DL models for tasks such as NER and Dependency Parsing in a native way, allowing the users to update the models or use their own models.

I think this is the way to go. In the future, it would be great to have pre-trained models for specific domains (e.g. biology, literature, economy, etc.) that are easy to use in a NLP framework. The icing on the cake would be the capability of fine tuning them, in the simplest way possible, for our use case. In the meantime, methods for adapting word embeddings are beginning to appear.

Adapting generic embeddings to specific use cases

Maybe the major downside of using pre-trained word embeddings is the word distributional gap existing between the training data and the actual data of our problem. Let's say you have a corpus of biology papers, food recipes or research papers in economics. It is more than likely that generic word embeddings are going to help you improve your results, since you probably don't have a corpus big enough to train good embeddings. But what if you could adapt generic embeddings to your specific use case?

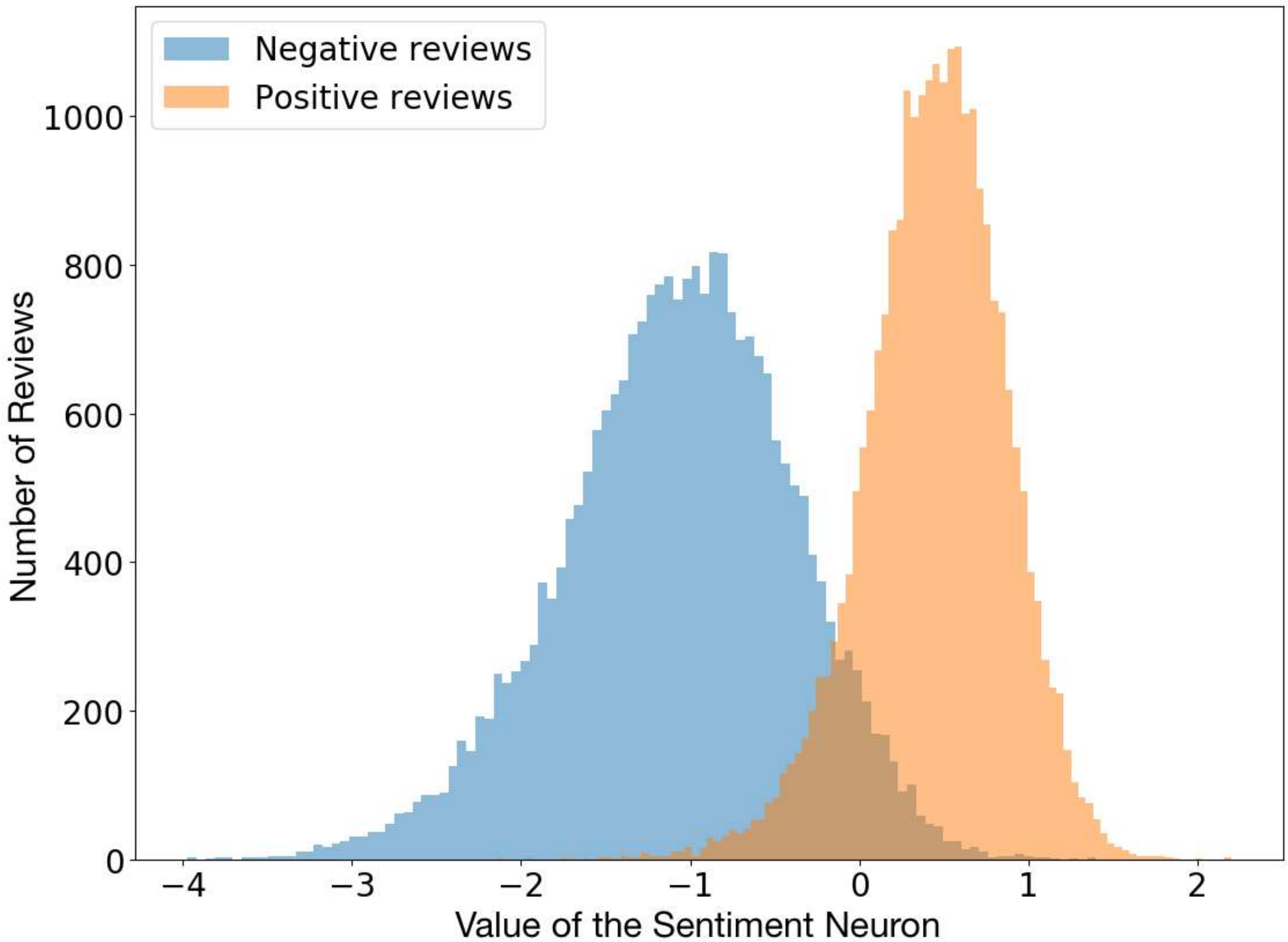
These kinds of adaptations are usually called cross-domain or **domain adaptation** techniques in NLP, and are very close to **transfer learning**. A very interesting work in this vein was proposed by **Yang et al.**. They have presented a regularized skip-gram model for learning embeddings for a target domain, given the embeddings of a source domain.

The key idea is simple yet effective. Imagine that we know the word embedding ws *for the word _w* in the source domain. To compute the embedding for w_t (target domain), the authors add to w_s a certain amount of transfer between both domains. Basically, if the word is frequent in both domains, that means that its semantics is not domain dependent. In this case, the amount of transfer is high, so the resulting embeddings tend to be similar in both domains. But since domain-specific words are more frequent in one domain than the other, the amount of transfer is small.

This is a research topic for word embeddings has not been widely explored, and I think that it is going to get more attention in the near future.

Sentiment analysis as an incredible side effect

The penicillin, the X-ray or even the post-it were unexpected discoveries. **Radford et al.** were exploring the properties of byte-level recurrent language models, with the goal of predicting the next character in the text of Amazon reviews, when they found that one single neuron in the trained model was highly predictive of the sentiment value. Yes, this single “sentiment neuron” was capable of classifying the reviews as positive or negative, in a pretty accurate way.



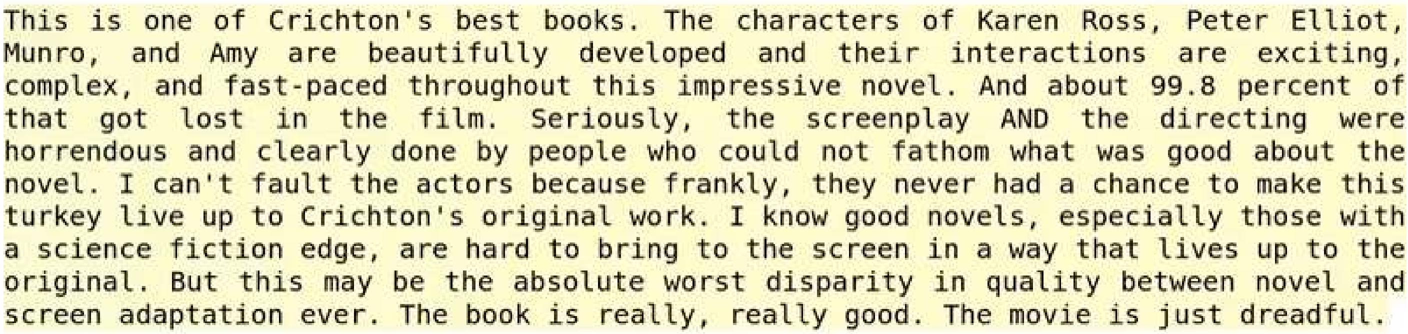
Review polarity vs Value of the neuron.

Image source

Having noted that behavior, the authors have decided to test the model on the **Stanford Sentiment Treebank**, and found that its accuracy was of 91.8%, versus the previous best of 90.2%. This means that using significantly less examples, their model, **trained in an unsupervised manner**, achieves state-of-the-art sentiment analysis, at least on one specific but extensively-studied dataset.

The sentiment neuron at work

Since the model works at the character level, the neuron changes its state for each character in a text, and it is pretty striking to see how it behaves.


This is one of Crichton's best books. The characters of Karen Ross, Peter Elliot, Munro, and Amy are beautifully developed and their interactions are exciting, complex, and fast-paced throughout this impressive novel. And about 99.8 percent of that got lost in the film. Seriously, the screenplay AND the directing were horrendous and clearly done by people who could not fathom what was good about the novel. I can't fault the actors because frankly, they never had a chance to make this turkey live up to Crichton's original work. I know good novels, especially those with a science fiction edge, are hard to bring to the screen in a way that lives up to the original. But this may be the absolute worst disparity in quality between novel and screen adaptation ever. The book is really, really good. The movie is just dreadful.

Behavior of the sentiment neuron.

Image source

After the word *best*, for example, the value of the neuron becomes strongly positive. This effect, however, disappears with the word *horrendous*, which makes sense.

Generating polarity biased text

Of course, the trained model is still a valid generative model, so it can be used to generate texts similar to the Amazon reviews. But what I find great about it, is that you can choose the polarity of the generated text by simply overwriting the value of the sentiment neuron.

Sentiment fixed to positive	Sentiment fixed to negative
Best hammock ever! Stays in place and holds its shape. Comfy (I love the deep neon pictures on it), and looks so cute.	They didn't fit either. Straight high sticks at the end. On par with other buds I have. Lesson learned to avoid.
Just what I was looking for. Nice fitted pants, exactly matched seam to color contrast with other pants I own. Highly recommended and also very happy!	The package received was blank and has no barcode. A waste of time and money.

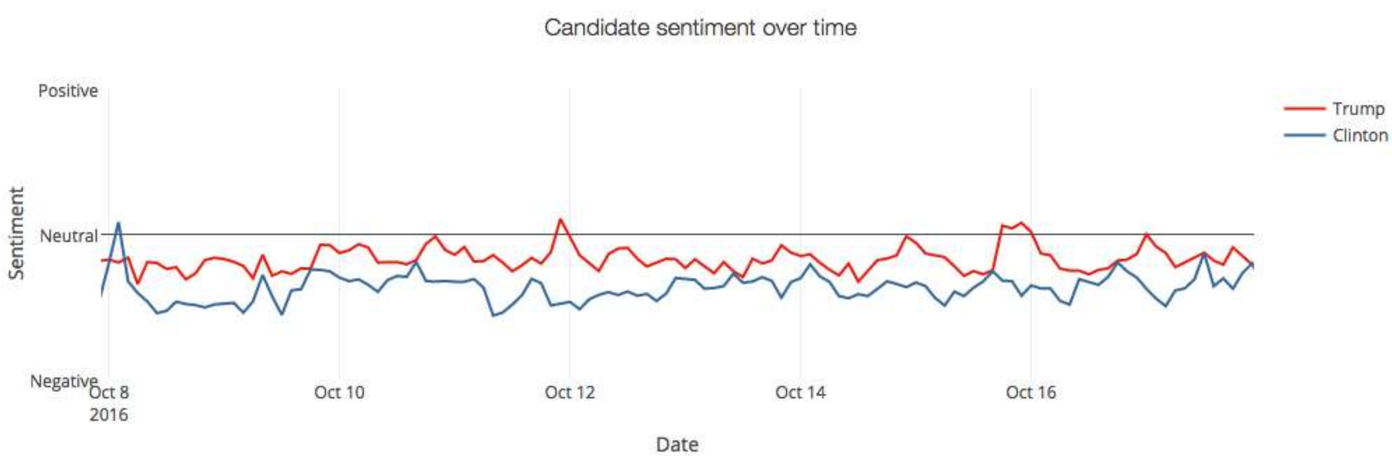
Examples of generated texts (**source**).

The NN model chosen by the authors is a **multiplicative LSTM**, presented by **Krause et al.** (2016), mainly because they observed that it converged faster than normal LSTMs for the hyperparameter settings they were exploring. It has 4,096 units and was trained on a corpus of 82 million Amazon reviews.

Why the trained model captures in such a precise way the notion of sentiment is still an open and fascinating question. Meanwhile, you can try to train you own model and experiment with it. If you have time and GPUs available, of course: the training of this particular model took one month to the authors of the work, across four NVIDIA Pascal GPUs.

Sentiment Analysis in Twitter

Whether to know what people say about your business's brand, to analyze the impact of marketing campaigns or to gauge the global feeling about Hillary Clinton and Donald Trump during the last campaign, sentiment analysis in Twitter is a very powerful tool.



Donald Trump vs Hillary Clinton: sentiment analysis on Twitter.

Image source

SemEval 2017

Sentiment analysis in Twitter has drawn a lot of attention from researchers in NLP, but also in political and social sciences. That is why since 2013, **SemEval** proposes a specific task.

In 2017, a total of 48 teams participated in the evaluation, which shows how much interest it generates. To give you an idea of **what exactly SemEval evaluates in the case of Twitter**, let's take a look at the five subtasks proposed this year.

1. **Subtask A**: given a tweet, decide whether it expresses POSITIVE, NEGATIVE or NEUTRAL sentiment.
2. **Subtask B**: given a tweet and a topic, classify the sentiment conveyed towards that topic on a two-point scale: POSITIVE vs. NEGATIVE.

3. **Subtask C**: given a tweet and a topic, classify the sentiment conveyed in the tweet towards that topic on a five-point scale: STRONGLYPOSITIVE, WEAKLYPOSITIVE, NEUTRAL, WEAKLYNEGATIVE, and STRONGLYNEGATIVE.
4. **Subtask D**: given a set of tweets about a topic, estimate the distribution of tweets across the POSITIVE and NEGATIVE classes.
5. **Subtask E**: given a set of tweets about a topic, estimate the distribution of tweets across the five classes: STRONGLYPOSITIVE, WEAKLYPOSITIVE, NEUTRAL, WEAKLYNEGATIVE, and STRONGLYNEGATIVE.

As you can see, the subtask A is the most common task, and 38 teams participated in it, but the others are more challenging. The organizers note that the use of DL methods stands out and keeps increasing, with 20 teams this year using models such as **Convolutional Neural Networks** (CNN) and **Long Short Term Memory** (LSTM). Moreover, although **SVM** models are still very popular, several participants combined them with neural network methods or used word embedding features.

The BB_twtr system

What I find remarkable is that one pure DL system, the _**BB_twtr_ system** (Cliche, 2017), was ranked first in the 5 subtasks for English. The author combines an ensemble of 10 CNNs and 10 biLSTMs trained with different hyper-parameters and different pre-training strategies. You can see the details of the architectures of the networks in the paper.

To train the models, the author uses the human labeled tweets (to give you an order of magnitude, there are 49,693 for the subtask A), and builds an unlabeled dataset of 100 million tweets, from which he extracts a distant dataset by simply labeling a tweet as positive in presence of positive emoticons such as :-) and vice versa for negative tweets. The tweets are lowercased, tokenized, urls and emoticons are replaced by specific tokens (, , etc.) and character repetitions are unified, so, for example, “niiice” and “niiiiiiiice” become both “niice”.

To pre-train the word embeddings used as input for the CNNs and the biLSTMs, the author uses word2vec, GloVe and fastText, all with the default settings, over the unlabeled dataset. Then he refines the embeddings using the distant dataset to add polarity information, and he refines them again using the human labeled dataset.

The experimentations using the previous SemEval datasets, show that using GloVe gives lower performances and that there is not one unique best model for all the gold standard datasets. The author then combines all the models with a soft voting strategy. The resulting model outperforms the previous best historical scores for 2014 and 2016 and is very close for the other years. Finally, it is ranked first in the 5 SemEval 2017 subtasks for English.

Even if the combination is not performed in an organic way but with a simple soft voting strategy, this work shows the potential of combining DL models, and also the fact that an almost end-to-end method (the input must be pre-processed) can outperform supervised methods in sentiment analysis in Twitter.

An exciting abstractive summarization system

Automatic summarization was, with **automatic translation**, one of the first NLP tasks. There are two main families of approaches: *extraction-based*, were the summary is built by extracting the most important segments from the source text, and *abstraction-based*, were the summary is built by generating the text. Historically, extraction-based approaches have been the most frequent because of their simplicity over the abstraction-based approaches.

In the last years, RNN-based models have achieved amazing results in text generation. They perform really well for short inputs and output texts, but tend to be incoherent and repetitive for long texts. In their work, **Paulus et al**. propose a new neural network model to overcome this limitation. The results are exciting, as you can see in the image below.

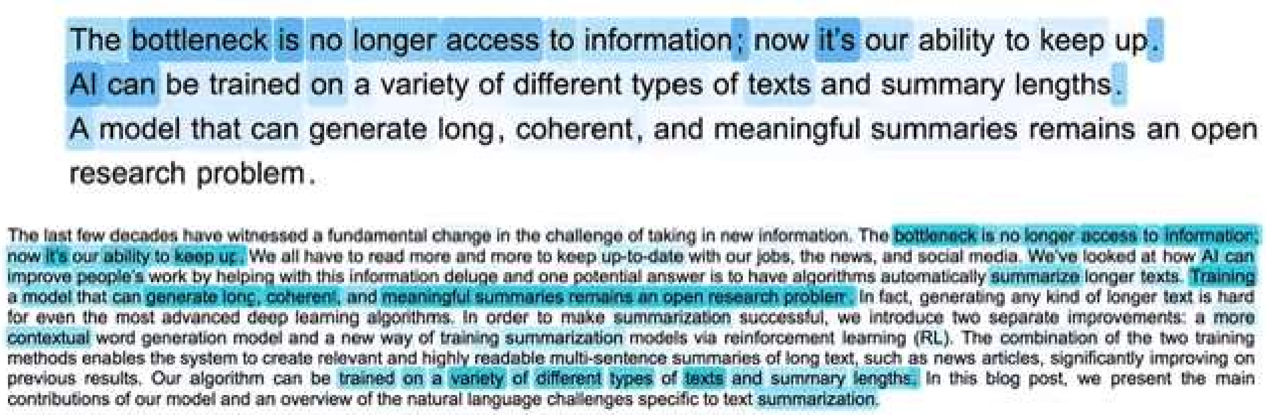


Illustration of the model generating a summary.

Image source

The authors use a biLSTM encoder to read the input and an LSTM decoder to generate the output. Their main contributions are a new intra-attention strategy that attends over the input and the continuously generated output separately, and a new training method that combines standard supervised word prediction and reinforcement learning.

Intra-attention strategy

The goal of the proposed intra-attention strategy is to avoid the repetitions in the output. To achieve this, they use temporal attention when decoding to look at previous segments of the input text, before deciding which word will be generated next. This forces the model to use different parts of the input in the generation process. They also allow the model to access the previous hidden states from the decoder. These two functions are then combined to choose the best next word for the output summary.

Reinforcement learning

To create a summary, two different people will use different words and sentence orders, and both summaries will probably be considered as valid. Thus, a good summary does not necessarily have to be a sequence of words that match a sequence in the training dataset as much as possible. Knowing this, the authors avoid the standard teacher forcing algorithm, which minimizes the loss at each decoding step (i.e. for each generated word), and they rely on a reinforcement learning strategy that proves to be an excellent choice.

Great results for an almost end-to-end model

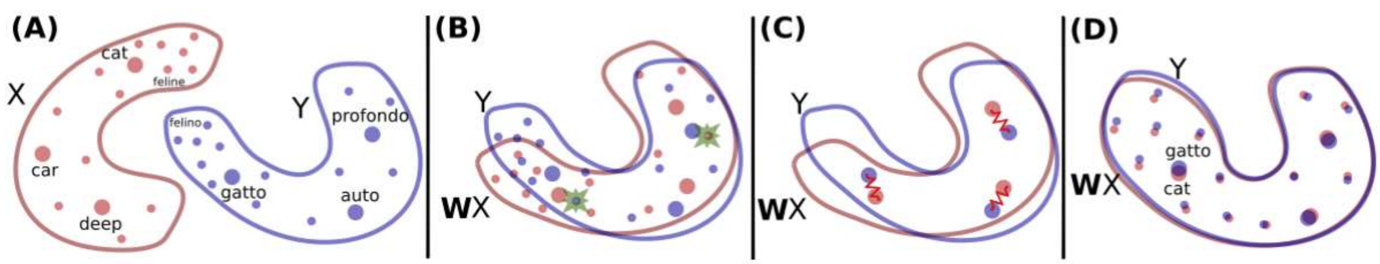
The model was tested on the **CNN/Daily Mail dataset** and achieved state-of-the-art results. A specific experimentation with human evaluators showed, in addition, an increase in human readability and quality. These results are impressive given such basic pre-processing: the input texts are tokenized, lowercased, the numbers are replaced with "0" and some specific entities of the dataset are removed.

A first step towards fully unsupervised machine translation?

Bilingual lexicon induction, that is, identifying word translation pairs using source and target monolingual corpora in two languages, is an old NLP task. Automatically induced bilingual lexicons help in other NLP tasks such as **information retrieval** and **statistical machine translation**. However, the approaches rely most of the time on some kind of resource, typically an initial bilingual lexicon, which is not always available or easy to build.

With the success of word embeddings, the idea of cross-lingual word embeddings appeared, were the goal is to align embedding spaces instead of lexicons. Unfortunately, the first approaches also rely on bilingual lexicons or parallel corpora. In their work, **Conneau et al.** (2018) present a very promising approach that does not rely on any specific resource, and outperforms state-of-the-art supervised approaches on several language pairs for the tasks of word translation, sentence translation retrieval, and cross-lingual word similarity.

The method developed by the authors takes as input two sets of word embeddings trained independently on monolingual data and learns a mapping between them such that translations are close in the common space. They use unsupervised word vectors trained on Wikipedia documents with fastText. The following image illustrates the key idea.



Building the mapping between two word embedding spaces.

Image source

The X distributions in red are the embeddings for English words and the Y distributions in blue the ones for Italian words.

First, they use **adversarial learning** to learn a rotation matrix *W* that is going to perform a first raw alignment. They basically train a **Generative Adversarial Network** (GAN), following the proposition made by **Goodfellow et al.** (2014). To have an intuitive idea of how GANs work, I recommend you **this excellent post by Pablo Soto**.

To model the problem in terms of adversarial learning, they define the discriminator to have the role of determining, given some elements randomly sampled from *WX* and *Y* (see second column in the picture above), to which language each one of them belongs. Then, they train *W* to prevent the discriminator from making good predictions. This seems to me very clever and elegant, and the direct results are pretty nice.

After that, they apply two more steps to refine the mapping. One to avoid the noise that rare words introduce in the mapping computation. The other one to build the actual translations, mainly using the learned mapping and a distance measure.

The results in some cases are impressive regarding the state-of-the-art. For example, in the case of English-Italian word translation, they outperform the best average precision for 1.500 source words by nearly 17% in the P@10 case.

	English to Italian			Italian to English		
	P@1	P@5	P@10	P@1	P@5	P@10
Methods with cross-lingual supervision						
Mikolov et al. (2013b) †	33.8	48.3	53.9	24.9	41.0	47.4
Dinu et al. (2015)†	38.5	56.4	63.9	24.6	45.4	54.1
CCA†	36.1	52.7	58.1	31.0	49.9	57.0
Artetxe et al. (2017)	39.7	54.7	60.5	33.8	52.4	59.1
Smith et al. (2017)†	43.1	60.7	66.4	38.0	58.5	63.6
Procrustes - CSLS	44.9	61.8	66.6	38.5	57.2	63.0
Methods with cross-lingual supervision (Wiki)						
Procrustes - CSLS	63.7	78.6	81.1	56.3	76.2	80.6
Methods without cross-lingual supervision (Wiki)						
Adv - Refine - CSLS	**66.2**	**80.4**	**83.4**	**58.7**	**76.5**	**80.9**

English-Italian word translation average precisions.

Image source

The authors claim that their method can be used as a first step towards unsupervised machine translation. It would be great if that is the case. In the meantime, let's see how far this new promising method can go.

Specialized frameworks and tools

There are a lot of generic DL frameworks and tools, some of them widely used, such as **TensorFlow**, **Keras** or **PyTorch**. However, specific open source NLP oriented DL frameworks and tools are just emerging. This has been a good year for us because some very useful open source frameworks have been made available to the community. Three of them caught my attention in particular, that you might also find interesting.

AllenNLP

The **AllenNLP framework** is a platform built on top of PyTorch, designed to easily use DL methods in semantic NLP tasks. Its goal is to allow researchers to design and evaluate new models. It includes reference implementations of models for common semantic NLP tasks such as semantic role labeling, textual entailment and coreference resolution.

ParlAI

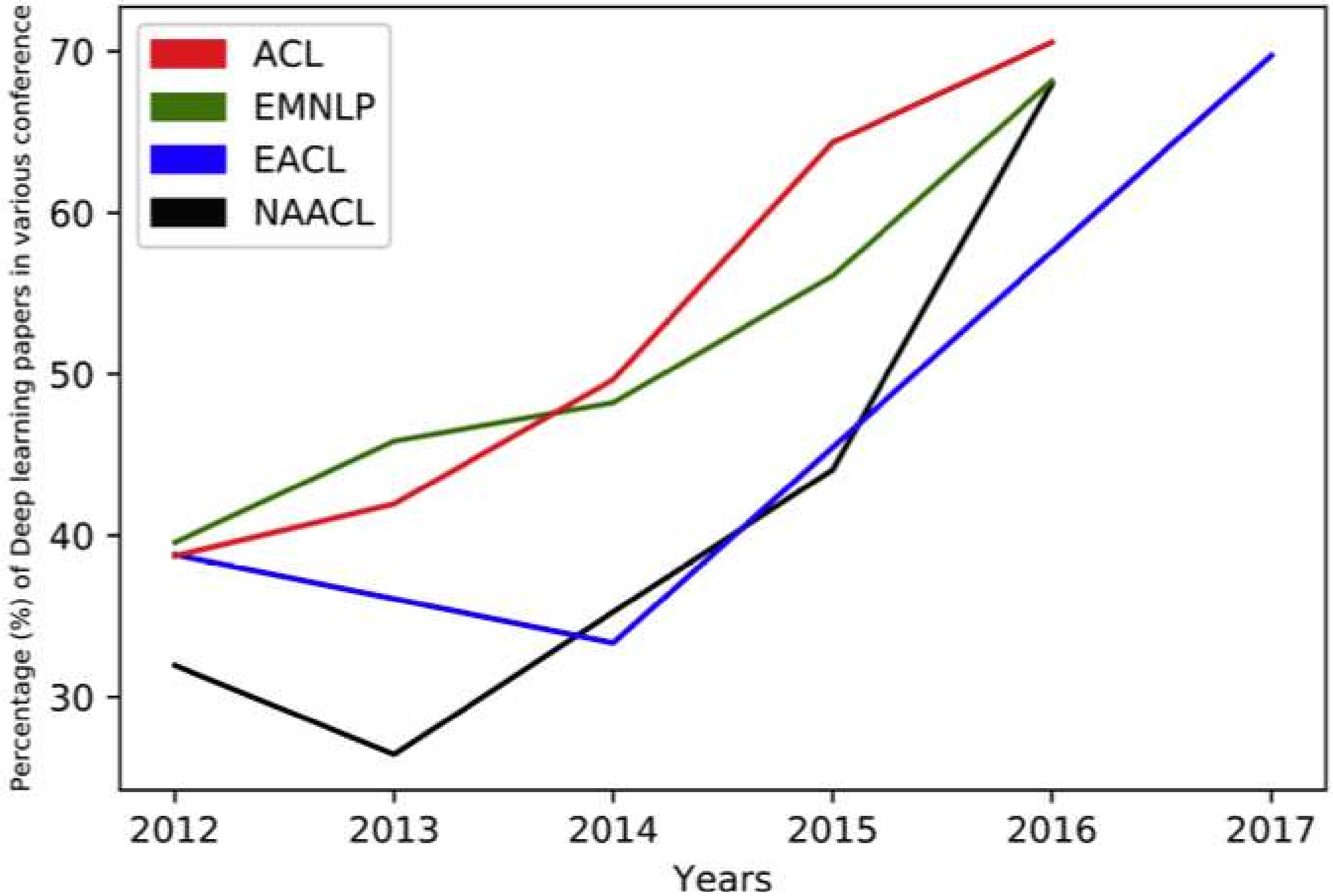
The **ParlAI framework** is an open-source software platform for dialog research. It is implemented in Python and its goal is to provide a unified framework for sharing, training and testing of dialog models. ParlAI provides a mechanism for easy integration with Amazon Mechanical Turk. It also provides popular datasets in the field and supports several models, including neural models such as memory networks, seq2seq and attentive LSTMs.

OpenNMT

The **OpenNMT toolkit** is a generic framework specialized in sequence-to-sequence models. It can be used in particular to perform tasks such as machine translation, summarization, image to text, and speech recognition.

Final thoughts

The constant increment of DL techniques used for NLP problems is undeniable. A good indicator of this is the variation of the percentage of deep learning papers in key NLP conferences such as **ACL**, **EMNLP**, **EACL** and **NAACL**, over the last years.



Percentage of deep learning papers.

Image source

However, works on true end-to-end learning are just beginning to emerge. We are still dealing with some classic NLP tasks to prepare the dataset, such as cleaning, tokenization or unification of some entities (e.g. URLs, numbers, e-mail addresses, etc.). We also use generic embeddings, with the drawback that they fail to capture the importance of specific domain terms, and also that they perform poorly for multi-word expressions, a key issue that I find over and over in the projects that I work on.

The latest advancements have been great for DL applied to NLP. I hope that the future brings more end-to-end learning works and that the specific open source frameworks get more developed. Please feel free to share with us in the comments section your opinion about these works and frameworks, and also those that you liked this year and that I do not mention here.

Further reading

For more information about the deep learning methods in NLP research today, I strongly recommend you the excellent paper "**Recent Trends in Deep Learning Based Natural Language Processing**" by Young et al. (2017).

Another interesting reading is the report from the seminar "**From Characters to Understanding Natural Language (C2NLU): Robust End-to-End Deep Learning for NLP**" by Blunsom et al. (2017), were researchers on NLP, computational linguistics, deep learning and general machine learning have discussed about the advantages and challenges of using characters as input for deep learning models instead of language-specific tokens.

To have a comparative perspective between models, I can recommend you a very interesting **comparative study of CNN and RNN for NLP**, by Yin et al. (2017).

To have an intuitive idea of how GANs work, you can read **this excellent post by Pablo Soto**, that presents the major advancements in Deep Learning in 2016.

For a detailed explanation of word embeddings, I suggest you to read **this great post by Gabriel Mordecki**. It is written in a didactic and entertaining way, and explains the different methods and even some myths about word embeddings.

Finally, Sebastian Ruder wrote a pretty nice and exhaustive post about word embeddings in 2017, that you might find useful: **About Word embeddings in 2017: Trends and future directions**.

Oh… and if you liked this post, you should **subscribe to our newsletter** so you don't miss the ones to come :)

Bibliography

- **From Characters to Understanding Natural Language (C2NLU): Robust End-to-End Deep Learning for NLP** Phil Blunsom, Kyunghyun Cho, Chris Dyer and Hinrich Schütze (2017)
- **BB_twtr at SemEval-2017 Task 4: Twitter Sentiment Analysis with CNNs and LSTMs** Mathieu Cliche (2017)
- **Word Translation without Parallel Data** Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, Hervé Jégou (2018)
- **Generative adversarial nets** Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio (2014)
- **Distributional structure** Zellig Harris (1954)
- **OpenNMT: Open-source toolkit for neural machine translation** Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart and Alexander M Rush. (2017)

- **Multiplicative lstm for sequence modelling** Ben Krause, Liang Lu, Iain Murray and Steve Renals (2016)
- **Parlai: A dialog research software platform** Alexander H Miller, Will Feng, Adam Fisch, Jiasen Lu, Dhruv Batra, Antoine Bordes, Devi Parikh and Jason Weston (2017)
- **Linguistic Regularities in Continuous Space Word Representations** Tomas Mikolov, Scott Wen-tau Yih and Geoffrey Zweig (2013)
- **Glove: Global vectors for word representation** Jeffrey Pennington, Richard Socher and Christopher D. Manning (2014)
- **Learning to Generate Reviews and Discovering Sentiment** Alec Radford, Rafal Jozefowicz and Ilya Sutskever (2017)
- **A Simple Regularization-based Algorithm for Learning Cross-Domain Word Embeddings** Wei Yang, Wei Lu, Vincent Zheng (2017)
- **Comparative study of CNN and RNN for Natural Language Processing** Wenpeng Yin, Katharina Kann, Mo Yu and Hinrich Schütze (2017)
- **Recent Trends in Deep Learning Based Natural Language Processing** Tom Younga, Devamanyu Hazarikab, Soujanya Poriac and Erik Cambriad (2017)

24 Comments **Tryolabs Blog** 1 **Login**

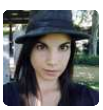
Recommend 20 Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name



Vered Shwartz • a year ago

Thanks for the intersting summary!

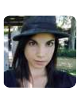
Petty comment: the source of the first image is actually this blog post about word representations I wrote two years ago:
https://veredshwartz.blogsp...
It has been used in the paper without crediting the original source.

3 • Reply • Share ›

Javier Couto → Vered Shwartz • a year ago

Thank you Vered and sorry for the wrong reference, I thought that the image was created by the authors of the article (!) We are going to fix the source. It's a beautiful image, by the way, very intuitive to understand embeddings!

• Reply • Share ›



Vered Shwartz → Javier Couto • a year ago

Thanks Javier! and no worries, you obviously couldn't have known that.

1 • Reply • Share ›

DGDA • a year ago

This article is really helpful for students to have a plan to study NLP and DL. Thanks for your precise summary and essential knowledge.

2 • Reply • Share ›

Kaustubh Kunte • a year ago

Very informative post . Thank you.

2 ^ | v • Reply • Share ›

Steven • a year ago

Really enjoyed the post!

2 ^ | v • Reply • Share ›

Marie Corradi • a year ago

Really great article, thanks! Another link to fix, from Yang et al. (2017), should be this: http://www.aclweb.org/antho...

2 ^ | v • Reply • Share ›

Tryolabs Mod → Marie Corradi • a year ago

Thanks Marie! Fixed now!

^ | v • Reply • Share ›

Alexander Wolf • a year ago

Awesome article! Eager to read more into some of these papers and use the new NLP tools.

2 ^ | v • Reply • Share ›

Todor Mihaylov • a year ago

Great summary of the current trends in NLP! Very well written article. Something to fix is the the reference to "Paulus et al" - it should be https://arxiv.org/abs/1705.....

2 ^ | v • Reply • Share ›

Tryolabs Mod → Todor Mihaylov • a year ago

Thanks for the kind words Todor! The reference should be fixed now :)

^ | v • Reply • Share ›

Naveen Kumar, India • 7 months ago

Thanks for the post, I started research in NLP and DL, I could understand what major things happend during 2017. and all together your article is super!!!!

1 ^ | v • Reply • Share ›

Javier Couto → Naveen Kumar, India • 6 months ago

Many thanks for your feedback!

^ | v • Reply • Share ›

Michele Stecca • a year ago

Congrats, this is really a great article!

1 ^ | v • Reply • Share ›

Andy Tsang Chun • a year ago

Great summary, thanks!

1 ^ | v • Reply • Share ›

Erick Fonseca • a year ago

Very interesting and extensive!
I'd mention, however, that polysemy may be more of a problem than OOV words for embedding models.

1 • Reply • Share ›

Javier Couto → Erick Fonseca • a year ago

Absolutely, yes, the posts by Sebastian Ruder and Gabriel Mordecki explain a little bit that. And researchers like Omer Levy have shown that the semantics of the learned relations with these methods is not really clear. Embedding models are very helpful but far from perfect.

1 • Reply • Share ›

Nick P • a year ago

I'd be interested to know if you have any views on DL NLP techniques being used to predict multi class output where each example is a large document.

1 • Reply • Share ›

Javier Couto → Nick P • a year ago

Hi Nick, do you mean multi class classification of long texts?

• Reply • Share ›

Nick P → Javier Couto • a year ago

Yes. Specifically thinking about ideas for classifying a legal case outcome based on a large document describing the details of the case (with a
large training dataset). Initially I was thinking that just averaging the word vectors in a document using an embedding trained in a legal domain (interesting what you said about transfer with embeddings), but I think all the detail will be lost. I'm thinking a sequence model with the words converted to vectors might work? It's really ground not well trodden so I was keen to know if you had any ideas? BoW and TF-IDF just seem to throw away so much useful signal in the data.

• Reply • Share ›

Javier Couto → Nick P • a year ago

It's hard to say without knowing the specific details, but it seems to me that you probably want to develop something more complex than a simple classifier, whether you use DL or classic ML techniques. It's pretty straightforward to say if a document's subject is sports, music or literature. But if you want to train a model that learns the outcome of a legal case by "understanding" the details of the case, it might be really hard, if not impossible. If the outcome can be learned from lexicalized concepts (things like "the jury stated..."), then yes, you can try a CNN for example. But if, as a human, to predict the outcome you need a lot of world-knowledge (legal stuff, geographical and social knowledge, etc.), then you're right about a BoW approach: you *need* to introduce this knowledge into your model if you expect to have a decent performance. Take a look at this paper: https://www.ijcai.org/proce.... It's for short texts but the main idea is there: to conceptualize a text as a set of relevant concepts using a large taxonomy knowledge base. Maybe that can help you.

Reply • Share ›

Nick P → Javier Couto • a year ago

That's fantastic, thanks. It's good that you concur with my general idea for an approach. A lot of world knowledge is most likely needed, but I guess what is missing from the manual way of predicting outcomes - if just done by a legal expert - is that they may not effectively have a database of specific legal precedents easily available which a model may be able to learn given enough examples. I could imagine some sort ensemble approach might work well. A single model may be forced to generalise but in reality it would be good if in some scenarios if the model was well fitted to the data. For instance judge x happens to be more lenient was a case type of y. I'll have a read of the paper - thanks!

Reply • Share ›

Nick P • a year ago

Fantastic read. Thanks.

1 • Reply • Share ›

Barney Pell • a year ago

Great post, thanks!

1 • Reply • Share ›

What to read next

The major advancements in Deep Learning in 2018

How we built a stand-in robot for remote workers using IoT and computer vision

How Machine Learning is reshaping Price Optimization

Get in touch

Do you have a project in mind?

We'd love to e-meet you!

YOUR NAME

YOU@EMAIL.COM

COMPANY

TELL US ABOUT YOUR PROJECT

☐ *Subscribe to receive news and blog updates*

CONTACT US

ABOUT

About us
What we do
ML Consulting

OUR WORK

Clients
Products
Brochure

COMMUNITY

Blog
Open Source
Careers

CONTACT

UY Phone: (598) 2716 8997
hello@tryolabs.com

OFFICES

US: 156 2nd Street, SF.
UY: Rambla Gandhi 655/701, MVD.

SOCIAL