Words learning using deep structures

Martin Holecek
mholecek91@volny.cz
AA2 Final Project (A.A. 2014-15)
June 15, 2015

1 Text and machine learning

1.1 Representation

To represent text, the basic choices are:

- Attribute forgetting - basic techniques such as bag of words
- Gramatic models
- Vectorization of words and complex ideas - like, for example letting the net learn gramatics on itself

1.1.0.1 Bag of Words Bag of words is a representation, that transforms a document to an integer-valued vector (with the length of dictionagy size) by assigning to each word the number of occurences in the document. It is also commonly used together in a combination with naive Bayes and Support vector machines (comparsion is for example in [WM12]).

To note the features of BoW:

- The representation is sparse and forgets the gramatical and temporal structure of a document.
- Normalization on such a representation include TF-IDF technique, which divides the word by total count across all (training) documents.
- This technique can be used also on bigrams [Wal06] or images[FFP05, CDF+04].
- Computational tricks for speedup involve, for example, using a hashing function to create the representation [WDA+09].
- Usage in latent models include techniques called LDA (Latent dirichlet allocation) and LSI (Latent semantic indexing).

1.1.0.2 Vectorization of words ("word2vec")

In the work [MSC+13], fast process to turn a word in a vector is examined, such that similar meanings are supposed to be groupped together. This idea can be applied even to phrases and longer texts, as in [LM14]. Another view of the data is to use character by character codes and letting the network learn the grammar by itself - the work [ZL15] uses transformation of a character to a column of image and then applies convolutional neural networks.

1.2 Usage

The basic usage of a text processing is to classify a text, to extract usefull information from a text or to cluster texts to label some data (for later fast document retrieval or for finding similar documents).

Clustering can serve for example as an sentient analysis, like the article [dSG], that uses sequential representation of data and special dictionary for words and characters.

1.2.0.3 To cluster text data

[HS11] uses RBM to find similar documents based on word count (TF-IDF) with the additional ideas on how to make the codes binary and another pretraining phase when all the RBM layers are unfolded and trained together.

[LB08] is another example on using RBMs to directly classify text in the form of bag of words. The ideas in this paper are also about using discriminative version of RBM (based on infering another probability) to use for unsupervised and semisupervised learning.

[CXWS12] use faster version of denoising autoencoders (where the pretraining can be done in one iteration, because the nonlinear function is separated from the autoencoder), uses bag of words.

[RS08] proposes a deep network specifically designed to process word counts (compact representation from bag of words and then tricks including poisson regressor).

2 The Idea

Gramatic models would require different plugins for different languages and bag of words needs big input space (as much as the number of words) and does not preserve word order.

This idea to try comes from the phrase "seinsctits hvae descroveid, taht a hmaun can raed txet, wrhee the biginenngs and egnidns are the smae, but the mdidels are mgeland in a crzay way" (with a hint to try ones own mother toungue to read faster).

This gives exact representation of a word - just the frequency of characters and beginning and ending character.

To translate it to neural network - alphabet of 26 characters gives 52 neurons per word. First 26 are frequencies (the most frequent from the dictionary being 7 times 's' due to english constructions like "uselessnessess") and the last 26 are just sum of vectorized first and last character (putting just the characters there would tell the network, that z is 26 times a and thats misleading).

The exact function is presented below:

$r : \mathscr{A}^\infty \to \mathscr{B} \subset \mathbb{R}^{2\cdot|\mathscr{A}|} : word \in \mathscr{A}^\infty \to r(word) \in \mathscr{B} :$
$r(word) = \left(\sum_{c\in|\mathscr{A}|} vec(c) \cdot \sum_{i=0}^{len(word)} \delta_{vec(word_{[i]}),vec(c)}, vec(word_{[0]}) + vec(word_{[len(word)]})\right)$

$vec : \mathscr{A} \to \mathbb{R}^{|\mathscr{A}|} : vec(c)_i = \delta_{i,ord(c)}$

$ord(c) : \mathscr{A} \to \mathbb{R} : |\{x \in \mathscr{A}, x <_{\mathscr{A}} c\}|$ (lexicographical order of character c)

$\delta_{a,b}$: 1 if a=b,0 otherwise ... kronecker delta

$word_{[i]}$: ... i-th character of word

$len(word)$ ... length of a word

$\mathscr{A}$ ... set of characters (ordered by $<_{\mathscr{A}}$ and terminated by the greatest element $\infty_{\mathscr{A}}$), $\mathscr{A}^\infty$ ... set containing all possible words made from alphabet.

Figure 1: The word "machine" encoded and visualized as grayscale image 1x52.

2.1 Justification:

1. uniqueness - from an english dictionary consisting of 109581 words, the collisions are just 7:

   - doorstops doorposts
   - kleig klieg

- noncasual noncausal
- organization's organizations
- regains reginas
- snakes sneaks
- teazles teazels

(To try another language - from a Czech dictionary consisting of 300000 words, the number of collisions are 43.)

2. seemingly nice properties - every word has a fixed length, the codes are mostly sparse (not many words using all 26 characters of alphabet) and we should be able to reconstruct the word from its code. Even when the code is altered a bit (this could be an idea for next analysis - how big is the distance between existing words and how much can we perturb the codeword without translating him to another word).

3. philosophical:

   - The humans can read it that way and even for a human, the collision - words are difficult to seprate, requiring more attention reading (thats why they are presented above). Interesting idea for infield work could be to record childs mistakes in primary schools in reading lessons to see, if the word being read is similar to the written word; in meaning of this encoding space similarity.
   - It is okay to allow this small percentage of collisions, if we have lots of data
   - If we consider human being a level that we would like to achieve, then lets use a representation, the human "obviously" uses

3 The process

3.1 Architectural considerations

Later, there will be description of so far used architectures, now lets just try to summarize some concepts and decisions.

At this point we have various sequences of words and so we can use any technique for sequential data. Lets obtain the first results on

simple approach and let more complicated models (long short term memory, recurrent neural networks, echo state networks) for later. Also it is needed to say, that now we can use also convolutional methods, because each paragraph can be translated to image 52 x number-of-words.

3.1.0.4 Different possibilities The representation of data is already forcing some architectural decisions in advance. The representation of data is sparse. That means, that autoencoders could work well (the dimensionality of the data might be lower than in our representation), denoising autoencoders would need a gaussian type of input perturbing (zero - replacing would waste resources replacing zeros by zeros). Another idea to perturb the input is to use thesarus and replace words with synonyms.

The representation is also suited for convolutional networks. Possible variations include rearranging of the codeword - this matters more here than in another architecture, because convolution does not see the whole input at the same time.

Normalization of the data can be also done differently. The divisor can be the most frequent used character, or every character can have its own normalization. It might not be advised to use binary RBMs for pretraining, because shorter words will be everytime clamped to zero. (But the possibility to use RBM after an autoencoder can be reasonable.)

3.1.0.5 Moving window Lets say, that we will select a constant number of words (in the scripts it will be 52 words) at the input to the network and then let it process the data in the following matter.

From each "row" of original data (paragraph of variable number of words) we will feed the network $NumberOfWords/WindowSpeed$ times - according to the specified moving window speed (extreme case being speed of one producing as many inputs as there are words in the paragraph). If the paragraph is shorter, than the window length, we will think of the paragraph as if it was repeated again at the end (sufficient number of times to fill the window).

Needed to say, that this way we obtain lots of inputs easily (that all do have the same output, if we are for supervised learning). Moreover, in the end we can benefit from effects known for ensemble methods, because we can just average the predicted output over all generated inputs from a single paragraph.

3.1.0.6 The data of interest If we build a deep neural network with stacked autoencoders at the beginnning (with pretraining phase) and logistic regression layer at the output (with finetuning phase, for supervised classification tasks), we will be interested not only in the output, but also in the values from the layer before. If we let the autoencoder to have really tight number of neurons, we can force the network, to actually paint the input to lowerdimensional space for us.

3.2 Architecture overview

According to these considerations, the following neural network models were tested so far:

3.2.1 Convolutional neural network

Convolutional neural network is an architecture, that replaces fully connected layers (known from standart feedforward architectures) with a convolutional layer and sub-sampling layers.

Convolutional layer produces specified number of so called "feature maps". Each feature map is a result of applied convolution to the input. The whole process of applying convolution can be also seen as repeated use of small, fully connected neural network over the input.

Subsampling operation reduces the dimensionality of inputs, examples of used subsampling are max-pooling and averaging.

The set of all convolution matrices are parameters of one layer. The strength of this architecture is in the attempt to reduce number of neurons by repeated use of the same, smaller ones. Usage of convolutional neural networks is dominated by image operations - for example image classification [KSH12] and text recognition [WWCN12].

3.2.2 Autoencoders

An autoencoder is fully connected feedforward layer. Such layers can be stacked onto each other and pretrained in such a way, that every such layer gets a prespecified number of epochs and its own output (decoder) to try to reconstruct its own input (in a way thats best for supplied cost function). The types and ideas of using autoencoders include:

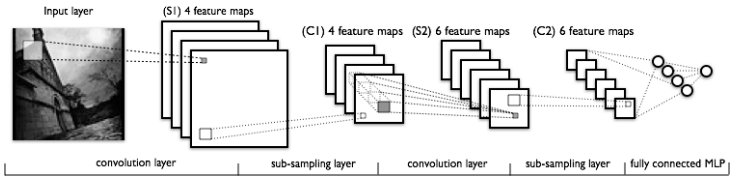

Figure 2: Convolutional neural network graphic explanation. (Image source: http://deeplearning.net/tutorial/lenet.html)

3.2.2.1 Denoising The idea behind deonising autoencoder is to randomly perturb the input each time and to let the autoencoder to try to decode information even from perturbed input. This idea should force the network to not be sensitive to input perturbations. A variant, marginalized denoising autoencoder [CXWS12], decouples the activation function from the reconstruction and enables faster pretraining.

3.2.2.2 Contractive The contractional autoencoder adds the following term $\|J_f(x)\|_F^2$ to the cost function to force more "contracted" representation of the inputs. The computation of the term can be simplified in case of sigmoid activation function [RVM+11].

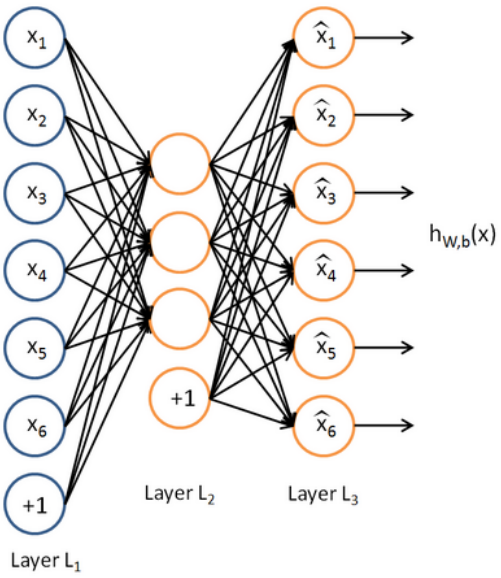

Figure 3: Autoencoder diagram (source of image)

3.2.2.3 Sparse The sparse autoencoder is using various techniques to keep the activation values of hidden units sparse, for example keeping just a specified number of highest activation values and resetting the rest to zero [MF13].

To summarize, the possibilities and choices to pretrain autoencoder include: contractional, sparse, denoising (marginalized), separate decoding weights

vs using transposed matrix of weight inputs, additional pretraining phase on unrolled layers; using more or less neurons than in the previous layer. Using separate decoding weights is not advised in case of not-augmented cost function (just eucleidian or crossentropy without, for example, contractive term), because at such architecture, the autoencoder might fail to be pretrained and outputs only average over values.

3.3 The implementation

The data are read from csv files using custom C++ library and fed to theano-based ([BLP+12, BBB+10]) neural network programmed in python.

The directory "wordylib" contains the dll module used to read texts from csv format and push them to python interface through cdll module. For first testing of the code, the neural network library FANN (http://leenissen.dk/) was used and interfaced to be able to process big datasets, the code is attached in the directory "FANN" (provided just for example, because theano is more suited to bigger data thanks to GPU support).

Theano itself is a python library translating formal computational graphs into C++ code or CUDA code and supports automatic differentiation. For theano there are provided (deeplearning.net) documentation pages and numerous tutorials - the code used to run the test cases is based on these modified tutorials.

3.4 The test cases

3.4.0.4 DBpedia The DBpedia dataset is a collection of formatted texts describing a certain entry from DBpedia. The entry belongs to one of 14 categories (Company, EducationalInstitution, Artist, Athlete, OfficeHolder, MeanOfTransportation, Building, NaturalPlace, Village, Animal, Plant, Album, Film, WrittenWork) and the ID of this category is also provided to be predicted from the text.

Totally there are 630000 rows, classes are evenly distributed (45000 rows for each class). Can be used for supervised learning and unsupervised classification with the possibility to see how classes are distributed. (The dataset is the also used in [ZL15], originally downloadable from https://drive.google.com/open

The dataset is stored in csv format with UTF8 encoding. Only preprocessing done was to convert to ASCII format.

3.4.0.5 Orders and cancelling The second case to test is a database of service orders (service helping students with courses), described by a text input from the orderer and the corresponding information if the order was executed or cancelled. The point of interest is, if we can see - based on what and how the applicants write - if the order is likely to be cancelled or not. This case might be, needed to say, difficult, if none such prediction actually exist. There are totally 22520 orders, 16225 orders were executed and 6295 were cancelled (about one fourth of the orders was cancelled).

The order's texts are generally shorter than dbpedia entries.

4 Experiments and results

The test models were 3 - smaller convolutional neural network, larger convolutional neural network and basic autoencoder.

Smaller convolutional neural network:

- convoltional layer - 20 frames, kernel 5x5
- Max Pooling - 2x2
- convoltional layer - 30 frames, kernel 5x5
- Max Pooling - 2x2
- convoltional layer - 20 frames, kernel 5x5
- Max Pooling - 2x2
- Fully connected sigmoid layer, 500 neurons
- ...followed by logistic regression to apropriate number of classes.

Bigger convolutional neural network:

- convoltional layer - 30 frames, kernel 5x5
- Max Pooling - 2x2
- convoltional layer - 40 frames, kernel 5x5

- Max Pooling - 2x2
- convoltional layer - 50 frames, kernel 5x5
- Max Pooling - 2x2
- Fully connected sigmoid layer, 1000 neurons
- ...followed by logistic regression to apropriate number of classes.

Autoencoder with 20 pretraining epochs without contraction or corruption:

- 3000 neurons, sigmoid
- 4000 neurons, sigmoid
- 3000 neurons, sigmoid
- 2000 neurons, sigmoid
- 1000 neurons, sigmoid
- 500 neurons, sigmoid
- 200 neurons, sigmoid
- 100 neurons, sigmoid
- ...followed by logistic regression to apropriate number of classes.

4.1 Results

With all possible adjustments to all parameters, the models failed so far, to predict the orders dataset. No signs of any improvement, network just kept guessing, no hyperparameter tuning or trying different models did help Also even looking at the representation in later layers, after running autoencoder, there was not clear why it assigns some values. Given a second look that it did work on all the other datasets, maybe the target values are, in fact, not predictable from the text. So lets concern us only with the performance and settings on dbpedia dataset.

All the models used the said moving window, which moves at a speed of 47 words (that is 52 window width minus 5 size of convolutional layer). Future tests could include a moving window of speed 1 leading to big number of inputs. (In addition, convolutional nets can be run in normailzed and unnormalized versions.) Learning rate was chosen to be 0.1, L2 regularization parameter to be 0.0001 in all cases. For autoencoder, the number of pretraining epochs were 20 with a learning rate of 0.001.

Batch sizes were chosen to be 361, size of training dataset was 649800 and testing and validation sets 238260 items each.

Convolutional neural networks proved, that the model can learn, progress (in a week of training only on CPU, not GPU) from initial 90% errors to

- 51.54% validation error, 52.76% test error for bigger convolutional network
- and 53.80% validation, 55.23% for smaller network on dbpeadia dataset

That means, that the original representation has the capacity to improve with more neurons. To compare to the best accuracy score of 98.40% on dbpedia dataset ([ZL15]), it needs to be said, that the work uses roughly hundred times more frames for convolution and two times more layers (6 covolutional, 3 fully connected).

Interestingly, autoencoder's cost function (average of cross entropy of the reconstruction) went right down on the first layer, but on the succesive layers, its descent was slower. Finally, to inspect the autoencoders ability to group similar results, we can look at a sentence and select sentences, that are close and far away in eucleidian norm of the hidden layer:

Lets inspect the close and distant sentences to this sentence (first one in the dbpedia dataset):

- "A Walk in the Sun is a World War II war film released in 1945 based on the novel by Harry Brown who was a writer for Yank the Army Weekly based in England..."

Three closest sentences are:

- "Dinglewood is a neighborhood/subdistrict located at the southern edge of Downtown Columbus Georgia. In it is the tallest building in Columbus the Aflac Tower. It is also home to the famous Dinglewood

Pharmacy which serves in the opinions of the city's residents the city's best scrambled hot dog..." (distance: $3.78E-05$)

- "A Walk to Beautiful is a 2007 American documentary film produced and distributed by Engel Entertainment about women who suffer from childbirth injuries in Ethiopia. In 2007 it premiered in film festivals and was chosen for the International Documentary..." (distance: $2.94E-06$)

- "A Walk in the Woods: Rediscovering America on the Appalachian Trail is a 1998 book by travel writer Bill Bryson describing his attempt to walk the Appalachian Trail with his friend Stephen Katz. The book is written..." (distance: 0 to machine float precision)

Some of the distant sentences (distance > 7) :

- She Married for Love is a 1914 silent comedy film featuring Oliver Hardy.

- The Scroafa River is a tributary of the Archita River in Romania.

Note, that the autoencoder groupped together (with a zero distance) the sentence that starts with the same suffix "A walk in the" and has "write" as substring.

References

[BBB+10] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation. 3.3

[BLP+12] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 3.3

[CDF+04] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. *Workshop on statistical learning in computer vision, ECCV*, 1(1-22):1–2, 2004. 1.1.0.1

[CXWS12] Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. *CoRR*, abs/1206.4683, 2012. 1.2.0.3, 3.2.2.1

[dSG] Cıcero Nogueira dos Santos and Maıra Gatti. Deep convolutional neural networks for sentiment analysis of short texts. 1.2

[FFP05] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531 vol. 2, June 2005. 1.1.0.1

[HS11] Geoffrey Hinton and Ruslan Salakhutdinov. Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science*, 3(1):74–91, 2011. 1.2.0.3

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 3.2.1

[LB08] Hugo Larochelle and Yoshua Bengio. Classification using discriminative restricted boltzmann machines. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 536–543, New York, NY, USA, 2008. ACM. 1.2.0.3

[LM14] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. 1.1.0.2

[MF13] Alireza Makhzani and Brendan J. Frey. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2013. 3.2.2.3

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013. 1.1.0.2

[RS08] Marc' Aurelio Ranzato and Martin Szummer. Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 792–799, New York, NY, USA, 2008. ACM. 1.2.0.3

[RVM+11] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 833–840, 2011. 3.2.2.2

[Wal06] Hanna M. Wallach. Topic modeling: Beyond bag-of-words. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 977–984, New York, NY, USA, 2006. ACM. 1.1.0.1

[WDA+09] Kilian Q. Weinberger, Anirban Dasgupta, Josh Attenberg, John Langford, and Alexander J. Smola. Feature hashing for large scale multitask learning. *CoRR*, abs/0902.2206, 2009. 1.1.0.1

[WM12] Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012. 1.1.0.1

[WWCN12] Tao Wang, David J Wu, Andrew Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012. 3.2.1

[ZL15] Xiang Zhang and Yann LeCun. Text understanding from scratch. *CoRR*, abs/1502.01710, 2015. 1.1.0.2, 3.4.0.4, 4.1