Deep Neural Network for Learning to Rank Query-Text Pairs

Baoyang Song[*1,2]

[1]Department of Computer Science, École Polytechnique
[2]Data & AI Lab, BNP Paribas

February 27, 2018

Abstract

This paper considers the problem of document ranking in information retrieval systems by Learning to Rank. We propose ConvRankNet combining a Siamese Convolutional Neural Network encoder and the RankNet ranking model which could be trained in an end-to-end fashion. We prove a general result justifying the linear test-time complexity of pairwise Learning to Rank approach. Experiments on the OHSUMED dataset show that ConvRankNet outperforms systematically existing feature-based models.

1 Introduction

This paper considers the problem of document ranking in information retrieval systems by Learning to Rank. Traditionally, people used to hand-tune ranking models such as TF-IDF or Okapi BM25 [Manning et al., 2008] which is not only time-inefficient but also tedious. Learning to Rank, on the other hand, aims to fit automatically the ranking model using machine learning techniques. In recent years, Learning to Rank draws much attention and quickly becomes one of the most active research areas in information retrieval. A number of supervised and semi-supervised ranking models has been proposed and extensively studied. We refer to [Liu, 2011] for a detailed exposition.

Though successful, these Learning to Rank models are mostly "feature based". In other words, given a query-document pair (q, d), the inputs to ranking models are vectors of form $v = \Phi(q, d)$ where Φ is a *feature extractor*. Some widely used features are such as TF-IDF similarity or PageRank score. However, feature based models suffers from many problems in practice. On the one hand, feature engineering is generally non-trivial and requires many trial-and-error before finding distinctive features; on the other hand, the computation of $\Phi(q, d)$ could be very challenging in a real use case. For instance, in the popular Elasticsearch

---

[Gormley and Tong, 2015], there is no direct way to calculate only IDF (though the product TF·IDF is readily available).

In this paper, we show how raw query-document pairs could be directly used to fit an existing feature-based ranking model. We develop ConvRankNet, a strong Learning to Rank framework composed of: (1) Siamese Convolutional Neural Network (CNN) encoder, a module designed to, given query q and two documents d_i, d_j, extract automatically feature vectors $\Phi(q, d_i)$ and $\Phi(q, d_j)$ and (2) RankNet, a successful three-layer neural network-based pairwise ranking model. We prove also a general result justifying the linear test-time complexity of pairwise Learning to Rank approaches. Our experiments show that ConvRankNet improves significantly state-of-the-art feature based ranking models.

2   Related Work

Our approach is based on the pairwise Learning to Rank approach [Liu, 2011].

Pairwise approach is extensively studied under supervised setting. As its name suggests, it takes a pair of documents d_i and d_j as input and is trained to predict if d_i is more relevant than d_j. Joachims [2002] uses "clickthough log" to infer pairwise preference and trains a linear Support Vector Machines (SVM) on the difference of feature vectors $v(q, d_i) - v(q, d_j)$. Burges et al. [2005] introduce a three-layer Siamese neural network with a probabilistic cost function which can be efficiently optimized by gradient descent. Instead of working with non-smooth cost function, Burges et al. [2007] propose LambdaRank which model directly the gradient of an implicit cost function. Burges [2010] introduces LambdaMART which is the boosted tree version of LambdaRank. LambdaMART is generally considered as the state-of-the-art supervised ranking model.

Under semi-supervised setting, however, there is considerably fewer work. Szummer and Yilmaz [2011] make use of unlabeled data by breaking the cost function C into two parts: C_l that depends only on labeled data and C_u that depends only on unlabeled ones. They report a statistically significant improvement over its supervised counterpart on some benchmark datasets.

Recent years, CNN [Krizhevsky et al., 2012] achieves impressive performance on many domains, including Natural Language Processing (NLP) [Kim, 2014]. It is shown that CNN is able to efficiently learn to embed sentences into low-dimensional vector space on preserving important syntactic and semantic aspects. Moreover, CNN is able to be trained in an end-to-end fashion, *i.e.* little preprocessing and feature engineering are required. Therefore, people attempt to adapt CNN to build an end-to-end Learning to Rank model.

Severyn and Moschitti [2015] combine a CNN with a pointwise[1] model to rank short query-text pairs and report state-of-the-art result on several Text Retrieval Conference (TREC) tracks. Though successful, their approach has several drawbacks: (1) both query and document are limited to a single sentence; (2) the underlying Learning to Rank approach is pointwise, which is rarely used in practice. Moreover, it is difficult to take advantage of pairwise preference provided by clickthough log and thus not practical for a real use case; (3) the add of "additional features" to the join layer is questionable. Indeed, the method

---

[1] Pointwise approach treats the feature vector of query-document pairs $\phi(q, d)$ independently. and considers a regression or multi-class problem on the relevance r.

is claimed to be end-to-end, but additional features could be so informative that the feature maps learned by CNN do not play an import role.
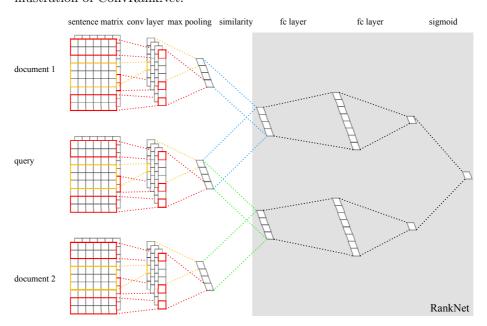
We are thus motivated to generalize the idea in Severyn and Moschitti [2015] and to build a real end-to-end framework whose underlying ranker is a successful Learning to Rank model.

3   ConvRankNet

Throughout the rest of paper, we denote $\mathcal{Q}$ the set of queries and $\mathcal{D}$ the set of documents. Given $q \in \mathcal{D}$, note $\mathcal{D}_q \subset \mathcal{D}$ the set of documents which "match"[2]q. For $d_i, d_j \in \mathcal{D}_q$, we write $d_i \succ d_j$ if d_i is more relevance than d_j ($d_i \prec d_j$ is defined similarly) and $d_i \sim d_j$ if there is a tie. Note further $p : \mathcal{D}_q \times \mathcal{D}_q \to \{-1, 0, 1\}$ the pairwise preference such that

$$p(d_i, d_j) = \begin{cases} -1, & \text{if } d_i \prec d_j \\ 0, & \text{if } d_i \sim d_j \\ +1, & \text{if } d_i \succ d_j \end{cases} \tag{1}$$

In the following we describe our system ConvRankNet for ranking short query-text pairs in an end-to-end way which consists of (1) Siamese CNN Encoder, a module designed to extract automatically feature vectors from query and text and (2) RankNet, the underlying ranking model. Figure 1 gives an illustration of ConvRankNet.

Figure 1: ConvRankNet structure.

---

[2]For example, $\mathcal{D}_q$ could be the set of documents sharing at least one token with q.

3.1 Siamese CNN Encoder

The Siamese CNN Encoder extracts feature vectors. As shown in Figure 1, the encoder consists of three sub-networks sharing the same weights (a.k.a. Siamese network [Bromley et al., 1994]). It is made up of the following major components: sentence matrix, convolution feature maps, activation units, pooling layer and similarity measure.

Sentence Matrix Given a sentence $s = w_1 \ldots w_N$, the sentence matrix $S \in \mathbb{R}^{N \times d}$ is such that each row is the embedding of a word (or n-gram) into a d-dimensional space by looking up a pre-trained word embedding model.

Convolution Feature Maps, Activation and Pooling Convolutional layer is used to extract discriminative patterns in the input sentence. A $2d$-filter of size $m \times d$ is applied on a sliding window of m rows of S representing m consecutive words (or n-grams). Note that the filter is of the same width d as the sentence matrix, therefore, a column vector $v \in \mathbb{R}^{N+m-1}$ is produced. Formally, the i-th component of c is such that

$$v_i = f * S_{i:i+m-1} = \sum (f \odot S_{i:i+m-1}) + b \tag{2}$$

where b is a bias. An non-linear activation unit is applied element-wise on v which permits the network to learn non-linearity. A number of activation units are widely used in many settings, in the scope of ConvRankNet, the rectified linear (ReLU) function is privileged. The output of activation unit is further passed to a max-pooling layer. In other words, v is represented by $\|\mathrm{ReLU}(v)\|_\infty$. In practice, a set of filters of different size $\{f_1, \ldots, f_n\}$ are used to produce feature maps $\{v_1, \ldots, v_n\}$. Each v_i is passed individually through the activation unit and max pooling layer so that we have a vector

$$\{\|\mathrm{ReLU}(v_1)\|_\infty, \ldots, \|\mathrm{ReLU}(v_n)\|_\infty\} \tag{3}$$

in the end.

Similarity Mesure Given $q \in \mathcal{Q}, d_i, d_j \in \mathcal{D}_q$, the encoder produces three vectors v_q, v_{d_i} and v_{d_j} respectively. In order to feed RankNet, two feature vectors $\Phi(v_q, v_{d_i})$ and $\Phi(v_q, v_{d_j})$ need to be further computed. Severyn and Moschitti [2015] introduce a similarity matrix M and defines $\Phi(q, d) = v_q^T M v_d$. However, such a choice is difficult to be fitted in a modern deep learning framework. In ConvRankNet, we choose instead

$$\Phi(v_q, v_d) = (v_q - v_d)^2 \tag{4}$$

where the square is element-wise.

The output of Siamese CNN Encoder, $\Phi(v_q, v_{d_i})$ and $\Phi(v_q, v_{d_j})$, are then piped to a standard RankNet. We privilege RankNet for its simple implementation and yet impressive performance on benchmark datasets. Our idea, however, is applicable to any pairwise ranking model.

3.2 RankNet

Proposed in [Burges et al., 2005], RankNet quickly becomes a popular ranking model and is deployed in commercial search engines such as Microsoft Bing. It is well studied in the literature. For sake of completeness, however, we describe briefly here its structure.

For $d_i, d_j \in \mathcal{D}_q$, suppose that there exists a deterministic target probability $\bar{P}(d_i, d_j) = \mathbb{P}(d_i \succ d_j) := \bar{P}_{ij}$ such that

$$\bar{P}_{ij} = \frac{1 + p(d_i, d_j)}{2} \in \{0, 0.5, 1\}. \tag{5}$$

The objective is to learn a posterior probability distribution P that is "close" to $\bar{P}$. A natural measure of closeness between probability distribution is the binary cross-entropy

$$C(d_i, d_j) = -\bar{P}_{ij} \log P_{ij} - (1 - \bar{P}_{ij}) \log(1 - P_{ij}) := C_{ij}, \tag{6}$$

which is minimized when $P = \bar{P}$. The posterior P is modeled by the Bradley-Terry model

$$P(d_i \succ d_j) = \frac{1}{1 + \exp(-s_{ij})} \tag{7}$$

$$P(d_i \prec d_j) = 1 - P(d_i \succ d_j) = \frac{1}{1 + \exp(s_{ij})}. \tag{8}$$

where s is a score function $s : \mathcal{D} \to \mathbb{R}$ and $s_i = s(d_i)$ and $s_{ij} = s(d_i) - s(d_j)$.

Under this assumption, the loss function (6) further becomes

$$C_{ij} = -\bar{P}_{ij} s_{ij} + \log(1 + \exp(s_{ij})) \tag{9}$$

C_{ij} being convex, it is straightforward to optimize it by gradient descent.

Since $d_i \succ d_j$ iff. $d_j \prec d_i$, without loss of generality we suppose that for (d_i, d_j) we always have $d_i \succeq d_j$. Moreover, Burges et al. [2005] show that training on ties makes little difference. Therefore, we could consider only document pairs (d_i, d_j) such that $d_i \succ d_j$.

3.3 Time Complexity

In this section we discuss the time complexity of general pairwise Learning to Rank models (and in particular, ConvRankNet).

In pairwise approach, we generally consider a bivariate function $h : \mathcal{D}_q \times \mathcal{D}_q \to \mathbb{R}, (d_i, d_j) \mapsto h(d_i, d_j)$ such that $d_i \succ d_j$ iff. $h(d_i, d_j) > 0$. h is then used to construct the cost function.

It is clear that the training time complexity is $\mathcal{O}(|\mathcal{D}|^2)$ since every pair (d_i, d_j) such that $p(d_i, d_j) \neq 0$ has to be considered. One may infer that the test cost is also quadratic since we have to evaluate h on the collection of all possible pairs on test data and construct a consistent total order (For example, $h(d_1, d_2) > 0, h(d_2, d_3) > 0$ induces the total order $d_1 \succ d_2 \succ d_3$).

However, we argue that under a very loose assumption, a linear time[3] $\mathcal{O}(|\mathcal{D}|)$ actually suffices for constructing the total order. First recall the following result in graph theory:

---

[3]Here we do not take into account the sort of score, which is in general $\mathcal{O}(n \log n)$. However, if the score is of fix precision, on could use *e.g.* Radix sort to achieve linear time complexity.

Lemma 3.1. *The* topological sort r *of a directed graph* $\mathcal{G}$ *is an* order of vertices *such that all edges of* $\mathcal{G}$ *go from left to right in the order. It is shown that*

- $\mathcal{G}$ *is a directed acyclic graph (DAG) iff. there exists a topological sort on* $\mathcal{G}$ *[Skiena, 2008].*

- *the topological sort is unique iff. there is a directed edge between each pair of consecutive vertices in the topological order (*i.e. $\mathcal{G}$ *has a Hamiltonian path) [Sedgewick and Wayne, 2011].*

We have then the following result:

Theorem 3.2. *Suppose that the hypothesis* h *has no tie,* i.e. $h(d_i, d_j) \in \mathbb{R}_+^*$. *If there exists* $\psi : \mathbb{R} \to \mathbb{R}, f : \mathbb{R} \to \mathbb{R}$ *such that* $h(d_i, d_j) = \psi \circ (f(d_i) - f(d_j))$ *and* $h(d_i, d_j) > 0$ *iff.* $f(d_i) > f(d_j)$, *then the total order defined by* $f(\cdot)$ *on* $\mathcal{D}_q$ *is the same as that of* $h(\cdot, \cdot)$ *on* $\mathcal{D}_q \times \mathcal{D}_q$.

Proof. Consider the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ induced by $h(\cdot, \cdot)$, *i.e.* $(i, j) \in \mathcal{E}$ iff. $h(d_i, d_j) > 0$. Remark first that $\mathcal{G}$ is a DAG. If not, there exists $d_{i_1} \ldots d_{i_n} \in \mathcal{D}$ such that $d_{i_1} \succ d_{i_2} \succ \ldots \succ d_{i_n} \succ d_{i_1}$. Then $f(d_{i_1}) > f(d_{i_2}) > \ldots > f(d_{i_n}) > f(d_{i_1})$, a contradiction. By Lemma 3.1, there exists a topological sort. Without loss of generality note the sort $d_q = \{d_1, \ldots, d_n\}$. Since h has no tie, d_q is an Hamiltonian path, thus the topological sort is furthermore unique. It is easy to see that r it is nothing but the sort with respect to $f(\cdot)$. $\square$

Corollary 3.2.1. *ConvRankNet has linear test time.*

4 Experimental Results

In this section we evaluate ConvRankNet on standard benchmark datasets and compare it with standard RankNet and LambdaRank.

4.1 Datasets

Since ConvRankNet is an end-to-end model, we need datasets to which we have access to raw query and documents.

To the best of our knowledge, OHSUMED dataset[4] is the only freely available dataset. Subset of MEDLINE (a database on medical publications), it consists of 106 queries on 348566 medical documents during 1987-1991. The relevance of 16140 query-document pairs are provided by human assessors on three levels: non-relevant (n), partially relevant (p) and definitely relevant (d). In particular, non-relevant pairs are explicitly provided. For historical reasons, each query-document pair is judged independently by up to 3 assessors. To avoid ambiguity, the highest relevance label is taken in our experiments.

To compare with classical feature-base models, we also use a synthesized version (where only feature vectors $\Phi(q, d)$ are accessible) of OHSUMED which is included in Microsoft's LETOR 3.0 dataset [5]. As in LETOR 3.0, we partition raw OHSUMED dataset into 5 folds as shown in Table 1.

---

[4]`http://mlr.cs.umass.edu/ml/machine-learning-databases/ohsumed/`
[5]`https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/`.

Table 1: Partition of OHSUMED dataset.

	S_1	S_2	S_3	S_4	S_5
Query id	1-21	22-42	43-63	64-84	85-106

4.2 Experimental Setup

All models are implemented in PyTorch framework.

In general, query and documents are not of the same length. Though PyTorch uses dynamic graph and is capable of handling texts of various lengths, one could only train the network one query-document-document triple a time. In order to perform batch training, both query and document are truncated to 100 words with zero-padding.

We use ConceptNet Numberbatch[6]Speer et al. [2016] as the default word embedding. Part of the ConceptNet open data project, ConceptNet Numberbatch consists of state-of-the-art semantic vectors that can be used directly as representation of word meanings. It is built using an ensemble that combines data from ConceptNet, word2vec, GloVe, and OpenSubtitles 2016, using a variation on retrofitting. Several benchmarks show that ConceptNet Numberbatch outperfoms word2vec and GloVe. ConceptNet Numberbatch includes not only vectorization of single word but also that of some bigrams and trigrams. Bigrams and trigrams are semantically more informative. For example, "la carte" is clearly better characterized than "la" and "carte" separately. To exploit n-grams, we use a greedy approach in mapping text to vector matrix, *i.e.* we extend as long as possible the n-gram to map to vector. For example, the following toy "sentence" `hello world peace` would be segmented as [`hello world`, `peace`], even though [`hello`, `world peace`] is also possible. One possible extension to our work is then to find the semantically optimal segmentation of sentence. In our network, unknown words are mapped to a random vector and zero padding is mapped to zero vector.

Three different filters of size $3 \times 300, 4 \times 300, 5 \times 300$, with 10 copies each, are used for the convolution layer so that the input for RankNet is a 30-dimensional vector. In order to prevent overfitting, during training stage a drop-out layerSrivastava et al. [2014] with $p = 0.5$ is used after the max-pooling layer.

RankNet, LambdaRank and ConvRankNet are all trained for 500 epochs with learning rate $0.00001, 0.001, 0.001$ respectively. All tests were performed on a Ubuntu 16.04.4 server with $2 \times$ Xeon 3.2GHz CPU, 256GB RAM and Tesla P100 16GB GPU. Cross-validations are run in parallel with the help of GNU Parallel [Tange, 2011].

Normalized Discounted Cumulative Gain at truncation level k (NDCG@k) is used as the evaluation measure. NDCG@k is a multi-level ranking quality measure widely used in previous work. It ranges from 0 to 1 with 1 for the *perfect ranking*. We refer to [Manning et al., 2008] for a detailed presentation.

Table 2: 5-fold cross-validation of NDCG@k on test set.

method	NDCG@1	NDCG@2	NDCG@3	NDCG@4	NDCG@5
ConvRankNet	0.5479	0.5265	**0.5204**	**0.5241**	**0.5204**
LambdaRank	0.5677	**0.5267**	0.4942	0.4884	0.4780
RankNet	**0.5737**	0.5362	0.5128	0.4898	0.4746

method	NDCG@6	NDCG@7	NDCG@8	NDCG@9	NDCG@10
ConvRankNet	**0.5179**	**0.5109**	**0.5139**	**0.5122**	**0.5132**
LambdaRank	0.4681	0.4604	0.4552	0.4553	0.4503
RankNet	0.4648	0.4608	0.4560	0.4493	0.4461

4.3 Results

5-fold cross-validation is performed on both datasets. Table 2 reports NDCG@k at all truncation levels. It is clear that ConvRankNet outperforms RankNet and LambdaRank especially for large truncation level k.

Table 3: p-value of two-tailed Wilcoxon signed-rank test.

	RankNet	LambdaRank	ConvRankNet
RankNet	-	0.575	0.021
LambdaRank	-	-	0.012
ConvRankNet	-	-	-

A two tailed Wilcoxon signed-rank test [Dalgaard, 2008] is performed on these values. As we can see from Table 3, the improvement of ConvRankNet over RankNet and LambdaRank is statistically significant at 5% level. Moreover, according to [Liu, 2011], ConvRankNet also outperfoms systematically existing methods.

5 Discussion and Conclusions

In this paper, we proposed ConvRankNet, an end-to-end Learning to Rank model which directly takes raw query and documents as input. ConvRankNet is shown to have linear test time and thus applicable in real-time use cases. Our results indicate that it outperforms significant existing methods on OHSUMED dataset.

Future work could aim to study the generalization of the underlying RankNet module to other stronger neural network based model (such as LambdaRank).

---

[6]https://github.com/commonsense/conceptnet-numberbatch

Acknowledgements

References

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Tie-Yan Liu. *Learning to Rank for Information Retrieval*. Springer, 2011. ISBN 978-3-642-14266-6.

Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2015. ISBN 1449358543, 9781449358549.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775067. URL `http://doi.acm.org/10.1145/775047.775067`.

Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 89–96, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102363. URL `http://doi.acm.org/10.1145/1102351.1102363`.

Christopher J. Burges, Robert Ragno, and Quoc V. Le. Learning to rank with nonsmooth cost functions. In P. B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 193–200. MIT Press, 2007. URL `http://papers.nips.cc/paper/2971-learning-to-rank-with-nonsmooth-cost-functions.pdf`.

Christopher J. C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. Technical report, Microsoft Research, 2010. URL `http://research.microsoft.com/en-us/um/people/cburges/tech_reports/MSR-TR-2010-82.pdf`.

Martin Szummer and Emine Yilmaz. Semi-supervised learning to rank with preference regularization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 269–278, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2063620. URL `http://doi.acm.org/10.1145/2063576.2063620`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th*

International Conference on Neural Information Processing Systems - Volume 1, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. URL http://dl.acm.org/citation.cfm?id=2999134.2999257.

Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL http://arxiv.org/abs/1408.5882.

Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3621-5. doi: 10.1145/2766462.2767738. URL http://doi.acm.org/10.1145/2766462.2767738.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 737–744. Morgan-Kaufmann, 1994. URL http://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neura

Steven S. Skiena. *The Algorithm Design Manual*. Springer Publishing Company, Incorporated, 2nd edition, 2008. ISBN 1848000693, 9781848000698.

Robert Sedgewick and Kevin Wayne. *Algorithms*. Addison-Wesley Professional, 4th edition, 2011. ISBN 032157351X, 9780321573513.

Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. *CoRR*, abs/1612.03975, 2016. URL http://arxiv.org/abs/1612.03975.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

O. Tange. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, Feb 2011. URL http://www.gnu.org/s/parallel.

Peter Dalgaard. *Introductory Statistics with R*. Statistics and Computing. Springer, New York, second edition, 2008. ISBN 978-0-387-79053-4. doi: 10.1007/978-0-387-79054-1. URL http://dx.doi.org/10.1007/978-0-387-79054-1.