Power to the People: Using Deep Learning to Predict Power Relations

Angela Kong
Stanford University
akong2@stanford.edu

Michelle Lam
Stanford University
mlam4@stanford.edu

Catherina Xu
Stanford University
yuex@stanford.edu

Abstract

In this project, we use deep learning to predict power relations between individuals based on email dialogue. We present several email representations that combine lexical and non-lexical features, including number of recipients and email length. Then, we construct and tune a multi-layer neural net, RNN-LSTM, and CNN to generate predictions that are tested against a formal organizational hierarchy. Experiments conducted using the Enron Email Dataset show that our best model is able to consistently outperform existing machine learning frameworks for the task.

1 Introduction

What does the language you use with someone else say about your relationship with them? There are often inherent power dynamics in human interactions, which researchers have shown are expressed through language and dialog behavior [10]. In particular, the corporate environment is one social context in which power dynamics have a clearly defined structure and are embedded into much of the interactions between individuals. Thus, this domain presents an interesting case study on how language and power interact, especially because in this setting, the power status between two interacting individuals is often imbalanced and the expression of superiority is culturally normalized.

The predominant use of email as a form of business correspondence has motivated researchers to analyze email metadata and textual correspondence in order to better understand employee behavior. While email is an incomplete record of the relationship between individuals, it is critical to the day-to-day functioning of a company and is an exceedingly common avenue of communication – especially between subordinates and their superiors. Detecting power relations in emails not only has immediate implications for a corporation's workflow, but also has larger significance. For example, in recent years, the increasing accessibility of technology has led to the unfortunate "proliferation of websites and online communities that disseminate extremist propaganda" [6]. In these unsupervised forums where the identity of the participants are unknown, it is useful for intelligence agencies to infer the power structure of these communities and directly target those powerful members that have an expansive influence on other members of the community.

In formulating our experiments, we consider multiple data representations of both lexical and non-lexical features. We use several neural network models to make our predictions, including a single-layer neural network, multi-layer LSTM, attention-based LSTM, and CNN. Compared to a previous researcher Prabhakaran, who ran a model on the thread level with a 73.03% accuracy, we achieve 81.8% test accuracy with our CNN model using both lexical and non-lexical features, and a 79.7% test accuracy with our CNN model using lexical features and non-lexical features extracted from the thread level.

2 Background and Related Work

Previous work has investigated corporate email corpora for social structures including power relations. Gilbert studied all textual exchanges between pairs of employees from differing ranks of the corporate hierarchy and used an SVM model to identify predictive words and phrases for a superior or subordinate employee [4]. While Gilbert's work focuses mostly on identifying predictive phrases using statistical methods, we seek to extend previous work that made stronger use of machine learning techniques to identify phrases indicative of relative power.

Bramsen similarly investigated all emails between given pairs to model social power [2]. This work statistically extracted vocabulary and grammar patterns indicative of social power, used groups of these n-grams as features, and fed them to machine learning classifiers that would identify whether the speech was from superior-to-subordinate, subordinate-to-superior, or peer-to-peer. The best result found was 78.9% accuracy using SVMs.

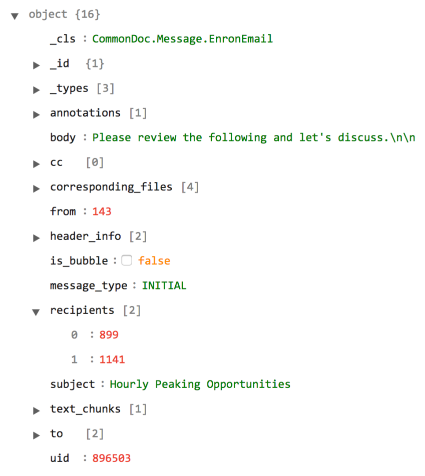
Prabhakaran used an SVM-based supervised learning system to classify the power relationship between two individuals in a single message thread. The best model, which utilized a mixture of lexical (n-grams) and non-lexical features, achieved an accuracy of 73.03%. The most relevant non-lexical features include the average number of recipients, requests for action, thread initiation, words per message, and number of messages. We include a subset of these features in our models to assess their impact in the deep learning context [10].

3 Approach

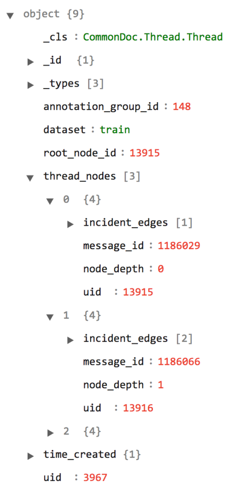
3.1 Dataset

Gold Standard for Enron Organizational Hierarchy

We obtained our data from annotated versions of the Enron Email Database. Apoorv Agarwal graciously agreed to send us the dataset he produced in the form of a BSON database [1]. The database contained data and metadata at the employee, email thread, and individual email level. An example email object (left) and thread object (right) is depicted below:



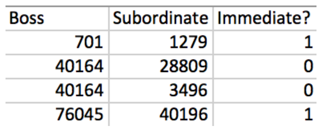
```
▼ object {16}
    _cls : CommonDoc.Message.EnronEmail
  ▶ _id  {1}
  ▶ _types [3]
  ▶ annotations [1]
    body : Please review the following and let's discuss.\n\n
  ▶ cc   [0]
  ▶ corresponding_files [4]
    from : 143
  ▶ header_info [2]
    is_bubble : ☐ false
    message_type : INITIAL
  ▼ recipients [2]
       0  : 899
       1  : 1141
    subject : Hourly Peaking Opportunities
  ▶ text_chunks [1]
  ▶ to   [2]
    uid  : 896503
```



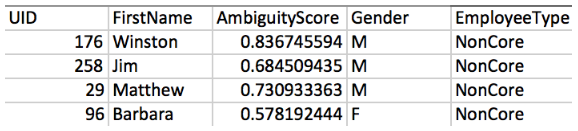
```
▼ object {9}
    _cls : CommonDoc.Thread.Thread
  ▶ _id  {1}
  ▶ _types [3]
    annotation_group_id : 148
    dataset : train
    root_node_id : 13915
  ▼ thread_nodes [3]
    ▼ 0  {4}
      ▶ incident_edges [1]
        message_id : 1186029
        node_depth : 0
        uid  : 13915
    ▼ 1  {4}
      ▶ incident_edges [2]
        message_id : 1186066
        node_depth : 1
        uid  : 13916
    ▶ 2  {4}
  ▶ time_created {1}
    uid  : 3967
```

Enron Email Dataset

Prabhakaran provided us with power relation, gender, and employee type annotations in the form of CSV files. The power relation CSV contains all 13,724 known superior and subordinate employee ID pairs. Snippets from the raw CSV files are included below:



Boss	Subordinate	Immediate?
701	1279	1
40164	28809	0
40164	3496	0
76045	40196	1



UID	FirstName	AmbiguityScore	Gender	EmployeeType
176	Winston	0.836745594	M	NonCore
258	Jim	0.684509435	M	NonCore
29	Matthew	0.730933363	M	NonCore
96	Barbara	0.578192444	F	NonCore

Pre-processing

We investigate three possible formulations of the power relation classification task:

1. **Per-Email**: For each email in which there is at least one (sender, recipient) pair with a valid labelled power relation, we randomly select a (sender, recipient) pair to evaluate in order to avoid overlap between the train, test, and dev sets. Using the email text, we predict the power relation between the sender and recipient: (superior sender, subordinate recipient) or (subordinate sender, superior recipient). We also incorporate non-lexical features, including the number of recipients and number of words in the email. We divided the 25,006 valid emails into train (60%), dev (20%), test (20%) by random sampling. There are 9,607 (superior sender, subordinate recipient) pairs and 15399 (subordinate sender, superior recipient) pairs.
2. **Per-Thread**: For a given thread *t*, there are a set of messages *m* that are exchanged in the thread. We consider all pairs in this thread who interact within the email thread *t*. For each pair (*p1*, *p2*), *p1* and *p2* are considered interacting pairs if either *p1* is the sender and *p2* is one of the recipients or vice versa. Using all email text in the thread, we predict the power relation between the sender and recipient. We also include the following non-lexical features: *AvgRecipients*, *AvgTokens*, *FirstPos*, and *RemovePerson*. We considered also including response rate, since it seems intuitive that e-mails sent from a superior should have higher precedence in immediate response. However, we realized that higher-level representatives often send mass announcements which do not warrant or expect a response. Therefore, we exclude this potentially confounding feature. The below chart details the statistics for the thread representation.

Table 1: Thread Data Statistics

Description	Total
# of Threads	36,196
# of Interactive Pairs	355,797
# of Interactive Pairs Hierarchically Related	18,253

3. **Grouped**: For each (sender, recipient) pair in the labelled power relation dataset, we use the text of all emails between these individuals and predict the power relation between the sender and recipient. There are 3,755 total valid pairs.

Vector representation. When including non-lexical features in our existing feature representation for RNN and CNN, we consider each scalar feature value as an extra word in each email. For example, for an existing email *d* represented as a *k*-dimensional vector **v**, we construct a *(k+4)*-dimensional vector **v**´ by appending the value for each of the feature values (*AvgRecipients*, *AvgTokens*, *FirstPos*, and *RemovePerson*) to **v**. Since **v** is a vector space representation of the email as the whole, we felt this design choice made the most sense by appending more information about the document. We then design three main categories of models and apply them to the above classification tasks.

3.2 Models

Model 1: Baseline

One-layer feedforward NN. For our baseline, we run a feed-forward neural network with one linear layer and a nonlinearity:

$$a = relu(xW_1 + b_1)$$
$$\hat{y} = softmax(a) \text{ (Predictions)}$$

where x represents the input vector created from an email. The neural activations are used to compute the probabilities of each of the two power orderings, and the loss is calculated and minimized using standard stochastic gradient descent.

Multi-layer feedforward NN. In order to capture additional non-linearities in the data, we also evaluate a two (left figure) and three (right figure) layer network, as denoted below:

$$h = tanh(xW_1 + b_1)$$
$$\hat{y} = softmax(hW_2 + b_2) \text{ (Predictions)}$$

$$h_1 = tanh(xW_1 + b_1)$$
$$h_2 = tanh(h_1W_2 + b_2)$$
$$\hat{y} = softmax(h_2W_3 + b_3) \text{ (Predictions)}$$

For the bag of words model, an index is assigned to each word in the email to create x. We pad x with zeroes when the email length is shorter than the maximum email length. When using GloVe vectors, we sum the individual word vectors of an email to create x [9].

Model 2: Recurrent Neural Network

LSTM. Next, we use a Recurrent Neural Network (RNN) architecture with Long Short-Term Memory (LSTM) units for our language model. These LSTM cells guard against the problem of vanishing and exploding gradients that is common to ordinary RNNs [5]. The incorporation of *gates* (input gate, forget gate, output gate) renders additional flexibility to the model so that it can process sequences of input in which notable events may be separated by arbitrary, unknown lengths of time. The equations defining these cells are outlined below:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) \text{ (Input gate)}$$
$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) \text{ (Forget gate)}$$
$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) \text{ (Output/Exposure gate)}$$
$$\tilde{c}_t = tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \text{ (New memory cell)}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \text{ (Final memory cell)}$$
$$h_t = o_t \circ tanh(c_t)$$

For each of the labelled entities of the three data formats described earlier, we concatenate the email text and tokenize by words and by sentences. For each sentence, we retrieve all GloVe pre-trained word vectors (up to a max-word threshold, a parameter which we adjust), and then we sum the sentence vectors to produce a (num_words, word_vector_dimension) matrix for each email [9]. We pad the sentences to account for differing-length sentences across emails. At each timestep, we provide one row of this matrix as input to our RNN.

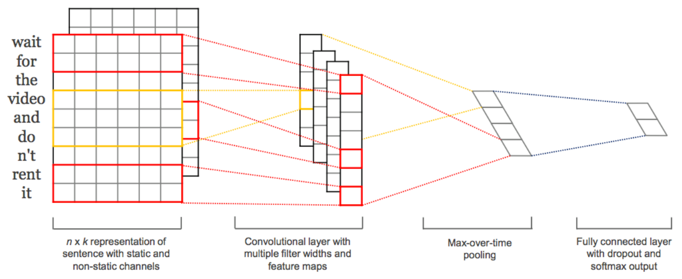
Then, using the output probabilities of our RNN LSTM model after all of the timesteps, we feed these results to a softmax layer to compute the softmax cross-entropy loss between the predicted probabilities and the true labels. We then use the Adam (Adaptive Motion Estimation) Optimizer, a gradient descent optimizer algorithm that computes adaptive learning rates for each parameter, to minimize this loss [8].

LSTM with attention. We also add local attention to our existing RNN LSTM cells [3]. Attention mechanisms allow our neural network to examine the input sequence (up to a given window size) more holistically in order to determine which inputs are most important at a given point in time.

LSTM with multiple layers. Furthermore, we extend our LSTM model with multiple layers of stacked LSTM RNN cells (sequentially composed cells) to generate a multi-layered recurrent neural network.

Model 3: CNN

Finally, we train a simple CNN with one layer of convolution on top of word vectors obtained from an unsupervised neural language model (GloVe). We use a single channel architecture in our model, which is a slight simplification of [7], as shown below:



wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1: Taken from "Convolutional neural networks for sentence classification" [7]

An email with length n (padded to the length of the longest email) is represented as $x_{1:n} = x_1 \oplus x_2 \oplus .. \oplus ...x_n$, where $\oplus$ represents the concatenation operator. Here, $x_i \in R^k$ represents the k-dimensional word vector corresponding to the i-th word in the email. A convolution operation involves a filter $w \in R^{(hk)}$, which is applied to a window of h words to produce a new feature. A feature is generated from a window of words $x_{i:i+h-1}$ by $f_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + \mathbf{b})$.

The filter is applied to each possible window of words in the sentence $x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}$ to produce a feature map $\mathbf{f}$ of R^{n-h+1}. Next, we max-pool the result of the convolutional layer, as represented as $(\hat{f_i}) = max(\mathbf{f})$ as the feature corresponding to this filter.

In our model we use varying filter sizes (three, four, and five words at a time), combining each feature from each filter to form our feature vector on the final layer. We then employ dropout on this feature vector, which prevents hidden neurons from co-adapting by randomly dropping a fraction *p* of the hidden units.

4 Experiments

4.1 Experimental Design

For each of the three defined tasks (Per-Email, Per-Thread, and Grouped), we apply various versions of the three aforementioned models. We conduct hyperparameter tuning by training our models on our train set, evaluate the performance of the trained model on a development set with respect to a range of hyperparameter values, and then use the optimal hyperparameter values to evaluate our model on our test set. Some examples of the hyperparameters tuned using this process were learning rate, number of hidden layers, number of neural network layers, attention length, dropout probability, filter size, and maximum sentence length.

4.2 Evaluation techniques

We compare our predictions against the labels provided by the Enron Email Dataset. For each of our models, we calculate accuracy, precision, recall, and F_1 scores to evaluate the performance of our neural network classifiers. The latter three metrics are incorporated because the processed dataset contains unequal quantities of data samples for each label. To maximize the amount of data used in training, development, and test, we do not equalize the two categories.

4.3 Results

In Table 2, the results of our experiments are summarized according to model, feature format, and problem formulation.

Table 2: Test Accuracies

Model	**Per-Email**	**Per-Email + non-lex features**	**Grouped**	**Grouped + non-lex features**
3-layer NN	65.6	67.7	63.0	69.7
RNN-LSTM	62.4	67.4	71.0	73.9
CNN	**75.0**	**78.7**	**80.0**	**81.8**

Model 1: Baseline

We test the bag of words vector representation on the 3-layer neural network, establishing the baseline at around 59% train and 57.4% test accuracy, with poor precision, accuracy, and recall (Precision: 50.4%, Recall: 50.0%, F_1: 50.2%).

For GloVe representations, the 3-layer neural network consistently outperforms the single and double layer networks, with accuracies hovering at around 57-61% across all representations. This could be due to the increased capability to capture non-linearities. Adding non-lexical features leads to accuracy increases for both representations, especially for Grouped pairs with an improvement of 6.7%. This may be because more information is captured with average email length and average number of words per email, across several emails, than statistics on a single message. The 3-layer model performs best with the Grouped representation and non-lexical features, achieving a test accuracy of 69.7% (Precision: 67.3%, Recall: 67.3%, F_1: 67.0%). The graph for is shown below, with red denoting train and purple denoting test accuracies and losses (run on 100 epochs):

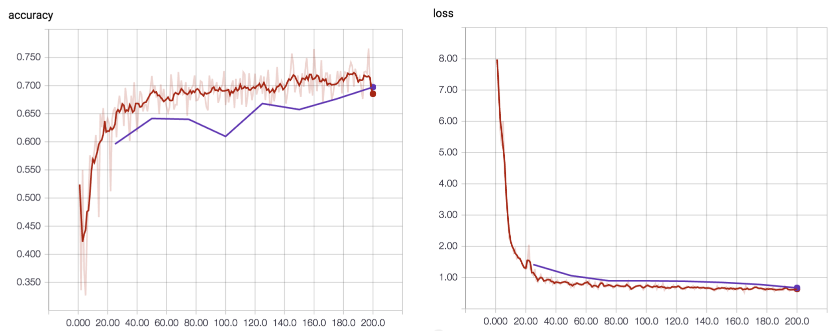


Figure 2: 3-layer NN, Grouped + non-lexical features
Accuracy and Cross-Entropy Loss

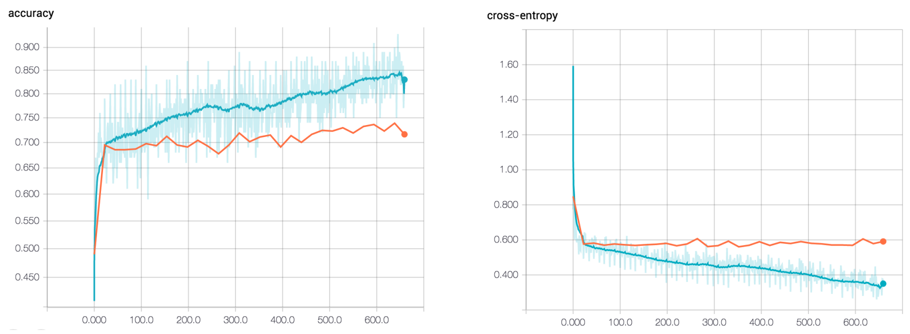


Figure 3: RNN-LSTM, Grouped + non-lexical features
Accuracy and Cross-Entropy Loss

Model 2: Recurrent Neural Network

Overall, the RNN model had higher performance than the baseline, and the simple LSTM model has similar or slightly higher performance than those incorporating attention mechanisms or multiple layers. The incorporation of non-lexical features significantly improves the model's performance on both the Per-Email and Grouped tasks. For the Per-Email task with GloVe vectors and non-lexical features, LSTM achieves 67.4% accuracy (Precision: 65.8%, Recall: 66.9%, F_1: 64.2%), and LSTM with attention and multi-layer LSTM achieved nearly identical results. For the Grouped task with lexical and non-lexical features, the highest-performing model is the LSTM attention model with 73.9% accuracy (Precision: 72.0, Recall: 71.6, F_1: 71.7), followed closely by basic LSTM and multi-layered LSTM. These results are illustrated in Figure 3, where blue denotes train scores and orange denotes test scores.

Model 3: CNN

We found that the CNN consistently outperforms the other models, as exemplified most prominently when non-lexical features from the email are incorporated. Because of the promising performance of this particular model, we decided to explore further by incorporating non-lexical features found at the thread-level. We display our results with various data representations below, run on 30 epochs, where the model is evaluated on the test set after every 100 steps:

Table 3: CNN Results

Model	Per-Email	Per-Email + non-lex features	Thread	Thread + non-lex features
CNN	75.0	78.7	71.9	79.7

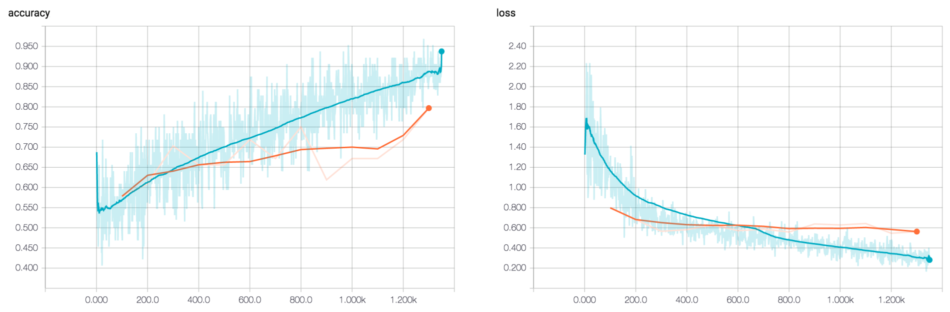


Figure 4: CNN, Per-Thread + non-lexical features
Accuracy and Cross-Entropy Loss

Qualitative Results

By examining some correctly- and incorrectly-classified emails, we see a few examples of how our model recognizes traits associated with power in Table 4. For example, as demonstrated in Row 1, brief, direct emails are associated with emails sent by superiors; as illustrated in Row 2, words such as "report", "attached", and gratuitous phrases like "please let me know" and "if you have any questions" are more strongly correlated with emails from subordinates. However, our models did not perform well with emails as found in Row 3 that express authority in nuanced ways. A semantic understanding that the email is making a command would help a system to perceive the sender's dominance, but we cannot always pick up these more subtle signals.

Table 4: Example email classifications

Email text	**Predicted**	**True**	**Correct?**
Why are we bothering with loc, I thought we had moved past that?	Superior sender, Subordinate recipient	Superior sender, Subordinate recipient	correct
Tammie, attached is the Global Risk Management Operations update for Louise's weekly report [...] If you have any questions, please let me know. Thanks, Brian	Subordinate sender, Superior recipient	Subordinate sender, Superior recipient	correct
Kevin, please call asap. While I have been working through the theory on how to make this happen, but I don't know the details on which pipelines and which accounts.	Subordinate sender, Superior recipient	Superior sender, Subordinate recipient	incorrect

5 Conclusion

Main Takeaways

Through our experiments, we found that CNN is the optimal model for this task. Initially, this may seem counterintuitive since the CNN's architecture is designed for image recognition and is thus location-invariant. However, CNN is very efficient in its representation - its filters are able to capture n-grams where n is up to 5. With such a large vocabulary, other models cannot compute larger than 4-grams without becoming computationally expensive. Thus, we hypothesize that CNN's success stems from its use of learned filters to capture rich textual features in an efficient manner.

In addition, we found that our models generally achieve higher accuracies using the Grouped representation. This is as expected due to the larger amount of lexical data available for each training example.

Through our experiments, we found that non-lexical features such as the average number of recipients and email length are strongly indicative of power relations, which is consistent with Prabhakaran's findings using a supervised model. This was especially useful when used in conjunction with the Grouped representation.

Future Work

An interesting avenue of future work might be to evaluate how well our model could perform on a four-way classification to capture non-immediate versus immediate power relations in addition to the direction of power.

Furthermore, we observe that since pronouns and full names are often contained in the email, especially in greeting and signature lines, our model might simply learn from the names of the interaction. For example, if the CEO with name ”Johnson” sends out many emails with his name in the signature, then the model may incorrectly assign a larger weight to the token ”Johnson”. In the future, greetings and signatures should be removed from all emails to avoid overfitting. We can also use named entity recognition to remove all names from email text or replace all of these names with a generic <NAME> tag to prevent our models from learning power in terms of the identities represented in our particular dataset.

Finally, we would like to see whether the traits learned from the Enron Email Dataset can be generalized to other domains. We hope to broaden our scope to other email datasets such as the Avocado Research Email Collection or datasets from online forums or discussion boards in order to enhance our model's understanding of power relations.

Acknowledgments

We would like to thank Arun Chaganty, our project mentor, as well as Vinodkumar Prabhakaran and Apoorv Agarwal for allowing us to build upon their datasets. A special thank-you to Vinodkumar for meeting with us to provide feedback, and to the instructors of CS 224N.

References

[1] Apoorv Agarwal et al. “A comprehensive gold standard for the Enron organizational hierarchy”. In: *ACL ’12 Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* (2012). DOI: `http://dl.acm.org/citation.cfm?id=2390706`.

[2] Philip Bramsen et al. “Extracting social power relationships from natural language”. In: *HLT ’11 Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies* (2011), pp. 773–782. DOI: `http://dl.acm.org/citation.cfm?id=2002570`.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *ICLR 2015: International Conference on Learning Representations* (2015). DOI: `https://arxiv.org/abs/1409.0473`.

[4] Eric Gilbert. “Phrases That Signal Workplace Hierarchy”. In: *CSCW ’12 Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work* (2012), pp. 1037–1046. DOI: `http://dl.acm.org/citation.cfm?id=2145359`.

[5] Sepp Hochreiter and Jurgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* (1997), pp. 1735–1780. DOI: `http://dl.acm.org/citation.cfm?id=1246450`.

[6] D. Janbek and P. Prado. “Rethinking the role of virtual communities in terrorist websites”. In: *Combating Terrorism Exchange* (2012), pp. 23–32. DOI: `https://globalecco.org/rethinking-the-role-of-virtual-communities-in-terrorist-websites`.

[7] Y. Kim. “Convolutional neural networks for sentence classification”. In: *EMNLP* (2014), pp. 1746–1751. DOI: `http://emnlp2014.org/papers/pdf/EMNLP2014181.pdf`.

[8] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *ICLR 2015: International Conference on Learning Representations* (2015). DOI: `https://arxiv.org/pdf/1412.6980.pdf`.

[9] Jeffrey Pennington, Richard Socher, and Christopher D. 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. DOI: `http://www.aclweb.org/anthology/D14-1162`.

[10] Vinodkumar Prabhakaran and Owen Rambow. “Predicting Power Relations between Participants in Written Dialog from a Single Thread”. In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (2014), pp. 339–344. DOI: `http://www.aclweb.org/anthology/P14-2056`.