Computational ad hominem detection

Pieter Delobelle, Murilo Cunha, Eric Massip Cano,
Jeroen Peperkamp and **Bettina Berendt**
KU Leuven, Department of Computer Science
Celestijnenlaan 200A, 3000 Leuven, Belgium
{pieter.delobelle,murilo.cunha, eric.massipcano}@student.kuleuven.be
and {jeroen.peperkamp, bettina.berendt}@kuleuven.be

Abstract

Fallacies like the personal attack—also known as the ad hominem attack—are introduced in debates as an easy win, even though they provide no rhetorical contribution. Although their importance in argumentation mining is acknowledged, automated mining and analysis is still lacking. We show TF-IDF approaches are insufficient to detect the ad hominem attack. Therefore we present a machine learning approach for information extraction, which has a recall of 80% for a social media data source. We also demonstrate our approach with an application that uses online learning.

1 Introduction

Debates are shaping our world, with them happening online more than ever. But for these debates—and their offline variants as well—to be valuable, their argumentation needs to be solid. As Stede and Schneider (2018, Sec. 1.3) recognize, studying fallacies is crucial for the understanding of arguments and their validity. The ad hominem fallacy, the personal attack, is one of the more prevalent fallacies. Despite its common occurrence, a personal attack can be quite effective and might shape the course of debates.

In online discussion fora, these attacks are often unwanted for their low rhetorical quality. These debates are watched by dedicated members of those fora, so-called moderators. They follow entire discussion threads and flag any unwanted posts; which can take up a lot of their time and the discussion might have already panned out. Automated flagging could significantly improve the quality of debates and save moderators a lot of time.

When developing such an automated system, the variety and ambiguity of ad hominem attacks can be difficult to cope with. These attacks range from simple name calling (*i.e. "You're stupid"*), abusive attacks (*i.e. "He's dishonest"*) to more complex circumstantial attacks (*i.e. "You smoke yourself!"*) (Walton, 1998). Detecting all these varieties is quite challenging, since there can even be discussion about some of those labels amongst humans.

We hereby focus in this paper on detecting mainly two variants of the ad hominem fallacy: name calling and abusive attacks. To realize this automated system, we present a recurrent neural model to detect ad hominem attack in a paragraph, and we experiment with various other models to compare them. Finally, we look into the issues related to crowd sourcing of additional labeled paragraphs through an application as a web demo.

This article is structured as follows: Section 2 covers related work on ad hominem fallacies, detecting those fallacies, and crowd sourcing data sets. Section 3 will then review the components used in our approach, which is then further discussed in Section 4. Section 5 outlines the used data set, the training setup, and baseline models; afterwards the results are discussed in Subsection 5.3. Finally, Section 6 concludes this work and discusses future improvements.

2 Related work

The study of argumentations has a long history, with *Rethoric* by Aristotle being one of the more traditional works. In the second book, he discussed the concept of *Logos*, the argument or reasoning pattern in a debate. More recently, reasoning and argumentations are studied in the field of natural language processing (NLP). The subfield of argumentation mining focusses on extracting arguments and their relations (i.e. graphs) from texts (Stede and Schneider, 2018).

Ad hominem attacks The work of Walton (1998) describes at the structure of ad hominem attacks in great depth, from a non-computational view. In addition to this, the work also analyzes

different subtypes of ad hominem attacks. The simplest form is the direct ad hominem, and an example of a more complex attack is guilt by association. Similarly, programmer and venture capitalist Paul Graham introduces a hierarchical view of discussions, with name-calling and ad hominem attacks as the lowest layers (Graham, 2008). Although both discussions of ad hominem fallacies and debating in general are an important aspect to keep in mind, neither of them discuss automated detection of ad hominem fallacies.

Mining ad hominem fallacies Habernal et al. (2018) discusses methods to detect name calling, a subset of ad hominem attacks. Their work is focussed on how and where these fallacies occur in so-called discussion trees, of which online fora and social media are examples. But they also look into two models for identifying these fallacies: firstly, a two-stacked bi-directional LSTM network, and secondly, a convolutional neural network. Their analysis on the occurrence of those fallacies is an important contribution, and their brief attempt at classifying the fallacies is an important baseline for our work.

Sourcing data Sourcing data from public fora, such as Reddit, is used by other works in the field of NLP, like for hateful speech detection (Saleem et al., 2017) or agreement amongst participants in a discussion on a community on Reddit called *ChangeMyView* (Musi, 2018). Habernal et al. (2018) also collect their data from this community, since each post is expected to be relevant to the discussion. Moderators dedicate their time to flag and remove those posts. One of those flags is that a post attacks another person, which is included in the data set assembled by Habernal et al. (2018).

Crowdsourcing the labeling is another option; either by paying the participants (Hube and Fetahu, 2018) or by providing a service in return, like a game (Habernal et al., 2017).

3 Components of the classifier

This section will review current techniques for sentence representation in Subsection 3.1. Recurrent neural networks, which are used for the classifier in this paper, are covered in Subsection 3.2.

3.1 Sentence representation

Word2vec Word2Vec (Mikolov et al., 2013; Goldberg and Levy, 2014) offers a way to vectorize words whilst also encoding meaning into the vectors. The vector representation of the word "cats" would be similar—measured for example by the cosine similarity—to the vector representation of the word "dogs" but different than the vector representation of the word "knowledge". This also allows arithmetic operations to take place (Mikolov et al., 2013). For example:

$$\vec{w}(\text{“}king\text{”}) - \vec{w}(\text{“}man\text{”}) + \vec{w}(\text{“}woman\text{”}) \approx \vec{w}(\text{“}queen\text{”}) \quad (1)$$

The vectors are obtained by maximizing the likelihood of predicting a determined word (or term), given other surrounding ones. Thus, a vectorized paragraph yields a matrix of $l \times w$, where l is the length of each vector for a particular word (arbitrarily chosen) and w are the number of words for that paragraph. The first input of our model is a vectorized version of each paragraph from our dataset, where every element of the vector represents each word on that paragraph. This vector is mapped to a pre-trained Word2Vec model from GoogleNews, which has vector representations of 300 values for 3 million words, names, slang and bi-grams. Even though the paragraphs do not have the same length, the amount of words is equal for all of them and the empty values of shorter paragraphs are masked.

Par2vec Doc2vec or par2vec is extremely similar to Word2Vec. The difference is that a sentence or paragraph is represented as a vector, instead of a single word. The vector values are adjusted by maximizing the likelihood of a word (or term), given the surrounding words (or terms) with an adjustment for the discrepancy between the likelihood and actual value. Doc2Vec generates a vector of size l, which can be arbitrarily chosen. The second input of our model generates a vector representation of the paragraph itself (Le and Mikolov, 2014)

POS tagging Part-Of-Speech (POS) tagging applies a tag to each word in a particular sentence. For example, words can be tagged as "noun", "adjective", or "verb". For verbs, the tags can also encode the tense, and further information can be contained. These labels have been used successfully in NLP tasks (Hube and Fetahu, 2018). In this work, the POS tagging is done by the Python

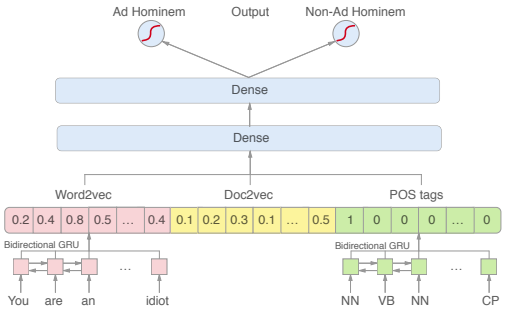


Figure 1: Graph of the combined neural network which gave the best results.

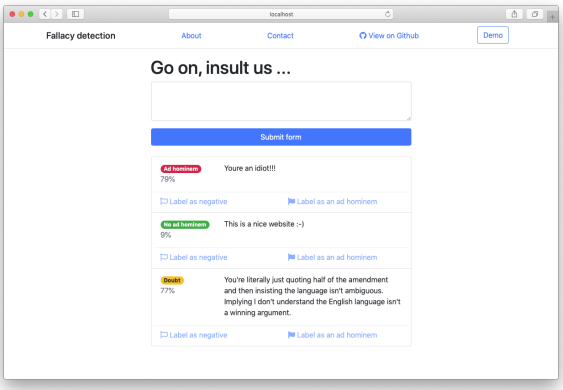


Figure 2: Screenshot of the web app with some example sentences.

library NLTK (Bird et al., 2009), which uses the Penn Treebank tagset.

3.2 RNN sentence encoding

Recurrent neural networks (RNNs) have successfully been used in sequence learning problems (Lipton et al., 2015), for example machine translation (Sutskever et al., 2014; Luong et al., 2015), and language modeling (Kim et al., 2015). RNNs extend feedforward neural networks by introducing a connection to adjacent time steps. So recurrent nodes are not only dependent on the current input $\mathbf{x}^{(t)}$, but also on the previous hidden states $\mathbf{h}^{(t-1)}$ at a time t.

$$\mathbf{h}^{(t)} = \sigma\left(W\mathbf{x}^{(t)} + W'\mathbf{h}^{(t-1)} + b\right) \qquad (2)$$

Some applications, for instance sentence modeling, can benefit not only from past, but from future input as well. For this reason, bidirectional recurrent neural networks were developed (Schuster and Paliwal, 1997).

The recurrent nodes can also be adapted by introducing memory cells. This forms the foundation for long short-term memory (LSTM) nodes (Hochreiter and Schmidhuber, 1997). By leaving out the memory cell, but maintaining the introduced gating mechanism, a gated recurrent unit (GRU) is created.

4 Model architecture

In this section, our approach will be discussed in detail. Subsection 4.1 will go in depth about our approach and its architecture. Finally, Subsection 4.2 illustrates how the classifier can be used in a web demo with online training.

4.1 Approach

Our approach is based on an RNN, as is illustrated in Figure 1. The Word2Vec vectorization is sent into a Bidirectional GRU. The POS tagging vectorization is also sent into a Bidirectional GRU. Both GRU layers consist of 100 recurrent cells each, with a ReLU activation. Lastly, the Doc2Vec vectorization output is already a vector, so we don't need to manipulate it to concatenate it with the other 2 vectors. Consequently, we concatenate the 3 previously mentioned output vectors in one single vector.

This vector is fed into 2 consecutive fully connected layers with a ReLU activation function. The last layer is also a fully connected layer but with a sigmoid activation that represents the probability that the input paragraph includes an ad hominem attack.

Even though the network uses masking on the inputs, an upper word limit L is introduced. Paragraphs with more words are restrained to the this limit L. The following section will also analyze how different limits affect the performance. These networks are then trained with the AdaDelta optimizer (Zeiler, 2012) with the default learning rate $l_r = 1.0$ and binary cross entropy as the loss function. Each of them was trained on a NVIDIA K80 GPU in one hour. In addition, class weights were used to tackle the imbalanced data.

4.2 Web demo

To demonstrate the classifier, a web application is built. It uses the same implementation of the classifier in Keras (Chollet and others, 2015), which is made available through a REST API with Flask. The front end is a Vue.js application.

Table 1: Illustration of how fallacies in the middle of a paragraph are contributing less to the overall output of the model, even though the attack itself is the same.

Sentence	Confidence
Augmented recurrent neural networks, and the underlying technique of attention, are incredibly exciting. You're so wrong and a f*cking idiot! We look forward to seeing what happens next.	0.39
You're so wrong and a f*cking idiot! Augmented recurrent neural networks, and the underlying technique of attention, are incredibly exciting. We look forward to seeing what happens next.	0.79

The web application also supports online learning and labeling of paragraphs. Each queried paragraph has two buttons to label the input, after which the backend saves the feedback and optionally retrains the network. However, as will be discussed in Section 5.3, this approach can actually worsen the accuracy of the classifier.

5 Evaluation

To correctly compare different models, the data set is split into a training and test set. All models are evaluated on a withheld test set (Flach, 2012). In this case, the test set contains over 8k labeled paragraphs. In two instances, memory issues forced us to train and evaluate on a smaller data set. These issues can be mitigated by streaming smaller batches of data, but this was made less of a priority since the provisional results were in favor of the RNN network. A further breakdown of the data collection and processing is discussed in Subsection 5.1.

Our model is compared to different baselines, which are reviewed in Subsection 5.2.

5.1 Data set

Our models were trained on a data set that was initially collected by Habernal et al. (2018). This data set is in essence a database dump of a Reddit community called *Change My View*, which focusses on online debating. In this context, ad hominem attacks are unwelcome and thus removed by moderators. The data set contains these labels amongst other things. The authors analyzed this data set extensively to make sure the labels were correct, in part by relabeling a subset by crowd-sourced workers.

However, our goal is different than that of Habernal et al. (2018): we classify each post individually, without taking any context about the discussion into account. Habernal et al. (2018) focusses on what this context—which they call a discussion tree—means for the occurrence of an attack. For this reason, we decided to not use the filtered dataset with only discussing trees that end up in ad hominem attacks. Instead we used the database dump and applied our own data cleaning.

Reddit allows the use of text formatting with Markdown (i.e. *bold* or _italics_). These were filtered, and more complex tags like links were removed, while still preserving the text associated with the link. Finally, the Markdown format also allows citations, which were commonly used to quote sentences of other posts. Since these citations could contain ad hominem attacks, they are removed as well.

5.2 Baselines

We compare our approach to multiple baselines. One of them is a CNN approach by Habernal et al. (2018), whilst the others are baselines we consider without recurrent layers.

1. **SVM**$_a$: our first model is based on a TF-IDF vectorizer and an SVM classifier (Flach, 2012). The TF-IDF vectorizer uses the top 3000 words from the test set. The SVM classifier is a linear SVM.

2. **SVM**$_b$: this model also uses a linear classifier, but the features are based on word representations. Instead of the TF-IDF vectors for a paragraph, the 300 most occurring words from the training set are used. For each of these 300 words, the TF-IDF value is replaced by a weighted word embedding. So for words that don't occur in a sentence or paragraph, all elements of this vector are zero. Otherwise the word representation is scaled by the TF-IDF value of that word. In total, this yields an array of 300 words by 300

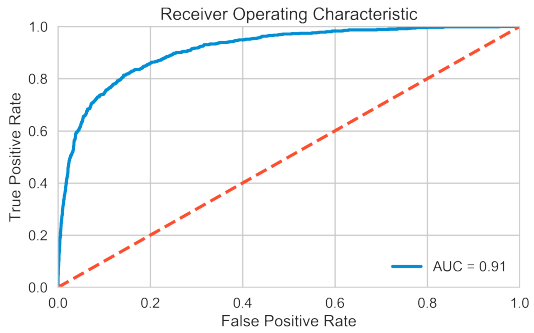
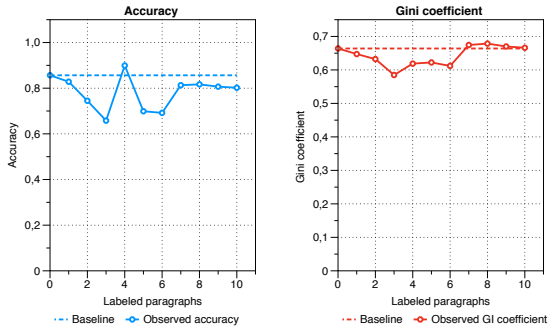


Figure 3: ROC curve of the best performing model, based on the test set.



(a) Accuracy (b) Gini coefficient

Figure 4: Influence of online training on three metrics. The dashed line is the best performing model before any additional online learning.

vector values. This approach is an extension of Kenter et al. (2016), which used a bag-of-words approach.

Other approaches—like as averaging all vectors—don't perform as well (Le and Mikolov, 2014).

3. **NN**: the NN approach continues with the above described vectorizing and uses a neural network for classification instead of an SVM. The output is formed by two sigmoid-activated neurons after one fully connected layer.

5.3 Results

Table 2 compares the models discussed in Section 3. The models are compared based on several metrics on a test set of 8531 paragraphs with 726 ad homimems. (Haghighi et al., 2018). Figure 3 show the ROC curve for the RNN model.

The best performing model is the RNN with word2vec, doc2vec, and POS tag features. This model scored best on recall R, the Gini coefficient GI, and the F1 score. The accuracy is slightly higher for the SVM model with word2vec features, due to the class imbalance.

These results—and in particular the difference between the our RNN model and both SVM_a and SVM_b—show that a TF-IDF approach is insufficient to distinguish most ad hominems from neutral comments, as the Gini coefficients indicate. The recurrent neural network incorporates sequential information, which has a positive effect on all metrics. Using longer input lengths L might allow to classify longer paragraphs at once, but this has a negative impact on the classification results. A possible reason of this is discussed in Subsecion 5.4.

5.4 Limitations

An issue is that the output of the model, which can be interpreted as a confidence scale between 0 and 1, is strongly influenced by the position of an ad hominem attack. This is illustrated in Table 1.

The web demo features a feedback button to label the input, and also allows to train the model directly with this new input. Since only the input sentence is used, this can cause an issue, namely *catastrophic forgetting* (McCloskey and Cohen, 1989). In this case, the model forgets all previously learned weights and instead of the intended increase in accuracy, it decreases.

Figure 4 illustrates how the online training of 10 paragraphs affects three metrics. This experiment is executed on the same test set and the baseline—the original model—is indicated as well. This clearly illustrates that when all parameters are taken into account, the model slowly degenerates, so it clearly highlights an issue with online learning.

6 Conclusion and further work

In this paper, we presented a machine learning approach to classify ad hominem fallacies. This model is based on two sequence models: a bidirectional GRU neural network for a sequence of word representations and another similar network for POS tags. The outputs of these two networks are combined with an additional feature, a paragraph representation, and fed into a fully connected neural network. This approach yields better results than a TF-IDF approach, which doesn't take any sequence information into account.

During the writing of this paper, a novel representation model based on transformers is

Table 2: Comparison of different models. All models are trained on 70% of the dataset and evaluated on the remaining 30%, unless annotated with an asterisk (*). In this case, 3k paragraphs of the dataset were used.

Model	L	ACC	R	GI	F1
CNN (Habernal et al., 2018)		0.810			
SVM_a		0.88819	0.29967	0.28208	0.42519
SVM_b		**0.90044**	0.34519	0.29812	0.37455
NN (word2vec)*		0.81667	0.52066	0.38331	0.43299
NN (word2vec and doc2vec)*		0.71222	0.69421	0.40923	0.39344
RNN (word2vec, doc2vec, POS tags)	150	0.83523	0.72853	0.57371	0.43006
	200	0.85647	**0.80256**	**0.66406**	**0.48854**
	300	0.81755	0.74614	0.57034	0.41049
	400	0.84480	0.69284	0.55177	0.35214

published (Devlin et al., 2018). This multilingual model could be used as an alternative for Word2vec, which has been critiqued for gender bias (Bolukbasi et al., 2016). Another posibility is ELMo (Peters et al., 2018), which takes the entire sentence into account before assigning an embedding to each word.

Finally we also discussed the issue of how the position of an attack changes the output. A possible solution would be to add attention to the RNN layer. This attention mechanism grants the network access to historical hidden states, so not all information has to be encoded in a single fixed-length vector (Bahdanau et al., 2014; Hermann et al., 2015). Continuing in this direction, it would also be possible to use hierarchical attention on both the word and sentence level (Yang et al., 2016).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings.

François Chollet and others. 2015. *Keras*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Peter Flach. 2012. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press.

Yoav Goldberg and Omer Levy. 2014. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv:1402.3722 [cs, stat]*. ArXiv: 1402.3722.

Paul Graham. 2008. How to Disagree.

Ivan Habernal, Raffael Hannemann, Christian Pollak, Christopher Klamm, Patrick Pauli, and Iryna Gurevych. 2017. Argotario: Computational Argumentation Meets Serious Games. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Copenhagen, Denmark. Association for Computational Linguistics.

Ivan Habernal, Henning Wachsmuth, Iryna Gurevych, and Benno Stein. 2018. Before Name-Calling: Dynamics and Triggers of Ad Hominem Fallacies in Web Argumentation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 386–396, New Orleans, Louisiana. Association for Computational Linguistics.

Sepand Haghighi, Masoomeh Jasemi, Shaahin Hessabi, and Alireza Zolanvari. 2018. PyCM: Multiclass confusion matrix library in Python. *Journal of Open Source Software*, 3(25):729.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Christoph Hube and Besnik Fetahu. 2018. Neural Based Statement Classification for Biased Language. *CoRR*, abs/1811.05740.

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-Aware Neural Language Models. *arXiv:1508.06615 [cs, stat]*. ArXiv: 1508.06615.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Zachary C. Lipton, John Berkowitz, and Charles Elkan. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv:1506.00019 [cs]*. ArXiv: 1506.00019.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. *arXiv:1508.04025 [cs]*. ArXiv: 1508.04025.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Elena Musi. 2018. How did you change my view? A corpus-based study of concessions' argumentative role. *Discourse Studies*, 20(2):270–288.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

Haji Mohammad Saleem, Kelly P. Dillon, Susan Benesch, and Derek Ruths. 2017. A Web of Hate: Tackling Hateful Speech in Online Social Spaces. *arXiv:1709.10159 [cs]*. ArXiv: 1709.10159.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Manfeld Stede and Jodi Schneider. 2018. *Argumentation Mining*. Morgan & Claypool Publishers, San Rafael, United States.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [cs]*. ArXiv: 1409.3215.

Douglas Walton. 1998. *Ad Hominem Arguments (Studies in Rhetoric & Communication)*. The University of Alabama Press, Alabama.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.