Metaphor Detection by Deep Learning and the Place of Poetic Metaphor in Digital Humanities

Chris Tanasescu (Margento), Vaibhav Kesarwani, Diana Inkpen

School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Ontario, Canada
margento.official@gmail.com, vkesa079@uottawa.ca, diana.inkpen@uottawa.ca

Abstract

The paper presents the work that has been done as part of the *Graph Poem* project in developing metaphor classifiers, now by deep learning methods (after previously having developed rule-based and machine learning algorithms), and a web-based metaphor detection tool. After reviewing the existing work on metaphor in natural language processing (NLP), digital humanities (DH), and artificial intelligence (AI), we present our own research and argue in favor of adopting data-intensive approaches, developing NLP classifiers, and applying graph theory (and particularly networks of networks) in computational literary or poetry analysis, while also highlighting the relevance of such work to DH, NLP, and AI in general.

keywords: poetry; metaphor; natural language processing; deep learning; digital humanities; graph theory applications; networks of networks

Introduction

The concept behind the *Graph Poem* project was initially a creative writing and generative one, writing, assembling, (post)digitally or X-algorithmically generating and/or expanding poetry corpora based on commonalities between poems. Poems are nodes in a network graph while edges represent commonalities initially drawn manually (MARGENTO 2012) and for which later on computational tools have been developed. The creative and generative purposes have thus naturally switched to critical and analytical ones, and thus the project joined a more general trend in digital humanities (DH) regarding—in what was called the second and third wave in DH ((Berry 2011))—a transition from classification or retrieval to more creative approaches and tools, or moreover, a fusion of the two. The latter was foregrounded in a significant way by N. Katherine Hayles and Jessica Pressman (Hayles and Pressman 2013) who underscored the complementarity if not congruence between these two sides of the coin in advancing a pun-like desideratum: making; critique. And this is perhaps particularly relevant to poetry and literature in general as it came against the background of a proposed new subject: comparative textual media. In the specific case of our project, textual-medium-related concerns are occasioned by the more general critical approach to data. Our tenet is that generally speaking for digital tools to be relevant—in poetry computational analysis and not only—they need to be trained and applicable to substantial amount of data, and while it is naturally debatable what magnitude would that specifically entail in the case of poetry, and since big data is at the same time a concept that has features and implications that are hardly applicable to literature, there is indeed possible to adapt another subject-relevant concept to poetry: data intensive research (Critchlow and Kleese van Dam 2013). Such an approach entails working with as large amounts of data as possible (and we will refine this statement in a bit) while also applying methods and developing tools that are specifically sensitive to data in quantitative and qualitative fashions. We therefore looked for large available databases and archives, and settled on the Poetry Foundation browser that contains tens of thousands of poems and has been manually annotated for various poetic features ranging from topic to form to period and region. The quantitative component is hence obvious, and that is part of our contribution since other work in the field has involved significantly smaller datasets, at the scale of sometimes even one hundred poems ((Kao and Jurafsky 2012) & (Dalvean 2013), with otherwise significant results in poetry automated analysis). While in the existing work it is true that the magnitude of the explored data is perhaps relative and not necessarily consequential in terms of the quality of the results, this aspect becomes of the essence in developing deep learning approaches, as is the case of the work presented in this paper. As will see below, deep learning proves to provide better results than other methods, but in order for this method to work, the amount of data has to be increased beyond certain limits.

The issue of data and data intensive research is also of great importance for the *Graph Poem* project as a whole. The focus is on poetry corpora and the features of the assembled network graphs—such as connectivity, the existence of cliques and of cut vertices, percolation, etc.—tell us significant things about a specific corpus or corpora (be it a random selection, a volume, a whole oeuvre or several oeuvres of several poets) and are obtainable depending on the quantity, quality, and computational analysis of data, since the output of our poetic-feature-focused tools (the processing of form, topic, style, etc.) is the input data for the graph visualization and analysis component. Graph theory has never been to our knowledge deployed in poetry, but network graphs have been used in other literature-focused chapters of DH and computational linguistics, specifically in social network analysis as applied in modeling fiction characters (Agarwal, Kotalwar, and Rambow 2013), see (Vala et al. 2016) for a comprehensive literature review) but also in structure-based clustering of novels (Ardanuy and Sporleder 2014) and in generating narrative (Sack 2012). Besides the inevitable genre-related differences between these applications and the ones pertaining to the *Graph Poem* project, there is another significant distinction: all of the graphs in fiction analysis are plain (single-layer) networks whereas the ones assembled and analyzed in the latter are multiplex networks (or multigraphs). Multiplex networks are graphs in which vertices are connected by different types (layers) of edges, therefore a particular case of networks of networks that behave in certain critical respects differently from plain networks (D'Agostino and Scala 2014) and whose importance for the study of literature has been explored recently ((Tanasescu and Tanasescu 2018) forthcoming, an article deploying multiplex networks in translation studies). The vertices in our network graphs are connected therefore in various layers, and analyzing those graphs involves processing and classifying the features in these layers independently and then analyzing the whole as a multiplex network with its specific characteristics.

One of these layers is metaphor. In our previous work we have developed a classifier combining rule-based and machine learning methods, and in this paper we are presenting the deep learning tool deploying convoluted neural networks we have developed for the same purpose but obtaining better results.

Metaphor has been though touched on in a number of recent DH publications that are, if not developing tools, posing theoretical questions regarding for instance metaphorical thinking in media studies analysis (Graham and Brook 2016), the place of metaphor in digital hermeneutics and as embedded in an interface (Armaselu and van den Heuvel 2017) or the role of metaphors in shaping public opinion (Núñez et al. 2017). But perhaps the most elaborate metaphor-focused DH project to this day is the *Metaphor Map of English* at the University of Glasgow (Hamilton, Bramwell, and Hough 2016) that visualizes the use of metaphor in the Historical Thesaurus with links between various categories, such as People or Travel and Travelling, all taken from the Thesaurus and falling under three general classes, external, mental, and social world. While this project involves an impressive database and interface, it does not work as an automatic classifier, and it is limited to visualizing and extracting data from the Thesaurus, with no option to run the tool on any other data. A poetry search for instance will produce a table of examples of metaphors from the Historical Thesaurus alongside their category, (the source texts) year of occurrence, and strength (strong or weak).

But the only other DH project that has aimed to process poetic metaphor computationally is POEMAGE. The project has actually started as a visualization system for exploring the sonic topology of a poem (McCurdy et al. 2016) in the same period when the *Graph Poem* team began working on a rhyme classifier (Tanasescu, Paget, and Inkpen 2016), but has more recently turned to considering metaphor processing as well. Unlike the *Graph Poem,* though, POEMAGE does not involve creating a tool that will algorithmically identify and visualize metaphor (Coles 2017) but getting at metaphor in an oblique way by pointing the human reader to places in the text the machine is uncertain of in terms of pronunciation and meaning and thus signalling potential metaphor occurrences.

In our previous work on metaphor we researched what has been done in this respect in machine learning and NLP, and since we were the first to approach metaphor in poetry (and literature in general), we turned to what had been accomplished in processing metaphor in non-literary texts and drew on what was usable in poetry as well. Turney's (Turney et al. 2011) notion of tracking down abstract-concrete discrepancies as indication of possible occurrences of metaphors became a feature in our model (combined with concrete category overlap, (Assaf et al. 2013), and ConceptNet, (Speer and Havasi 2012)), and one of our alternative computations of metaphor expanded on the part of speech tagging advanced by Neuman (Neuman et al. 2013). The expansion involved updating one of their sequences and adding two more (noun-verb and verb-verb on top of their adjective-noun and noun-verb-noun with two subcategories, copulative and regular verb) and deploying word embeddings. Word embeddings are a novel way of representing words as vectors aimed at redefining the high dimensional word features into low dimensional feature vectors by preserving the contextual similarity in the corpus (Kesarwani et al. 2017).

The abovementioned structural sequences were developed into a rule-based method, and then, after researching another major contribution to metaphor natural language processing Shutova et al (2016) we also developed a statistical model and compared the results (and it turned out, quite predictably, that the latter works significantly better). We also had to develop our own corpus and do manual annotation for training our model: 720 sentences out of 1500 extracted only for type 1 metaphor [noun-copulative verb-(determiner)-noun] with two annotators and an arbiter deciding on disagreements.

But this was not our only training data. We combined our dataset with two other sets, with the notations TroFi (Birke and Sarkar 2006) and Shutova (Mohammad et al 2016), while the abbreviation we have picked for ours is PoFo. The results for PoFo + TroFi + Shutova were significantly better than those on PoFo alone, which proved an interesting twofold point, namely that the abovementioned data intensive approach improves outcomes indeed, and that in detecting metaphor in poetry, non-poetry data are as helpful as poetry data.

We also reached in certain other respects a converse conclusion, that is, that tools trained on poetry data are sometimes better for non-poetry tasks than the ones trained on non-poetry data. We trained our own word embeddings on the PoFo poems, which we did not use for metaphor detection, as we wanted the embeddings to be rooted in corpora that contained as few metaphors as possible. The options in terms of available generally used embeddings were GloVe (Pennington et al., 2014) and word2vec (Mikolov et al. 2013) vectors, yet we chose the former as they had been shown to work better for many lexical-semantic tasks (Pennington et al., 2014). Still, these word embeddings of ours turned out to be better than the GloVe ones, and we deployed them in our poetic diction processing tasks. We are currently working on comprehensive technical evaluations, but the superiority of our own embeddings is already intuitively obvious for all examples we have run our tool on as compared to GloVe.

Word	Score
me	0.738
passion	0.735
my	0.733
life	0.729
dream	0.727
you	0.718
always	0.711
wonder	0.709
i	0.708
dreams	0.707
mind	0.706
friends	0.704
true	0.703
loves	0.700
feel	0.698
happy	0.698
fun	0.697
kind	0.696
soul	0.695
she	0.695

Table 1: LOVE in the GloVe model

The results in Table 1 and Table 2 show the words related to the word *love* in the GloVe and PoFo models in decreasing order of similarity score. It can be seen that the words in the GloVe model are more conversational (pronouns like *me*, *my*, *you*, *I* and *she* reinforce this) and less thematic, whereas the words from the PoFo model are more consistently—semantically and logically—related to the query *love*.

Convolutional neural networks (CNN)

A convolutional neural network (LeCun et al. 1998) contains one or more convolutional layers, pooling or fully connected, with nonlinear activation functions like ReLU (Nair and Hinton 2010) or tanh applied to the results. In a traditional feedforward neural network, each input neuron is connected to each output neuron in the next layer. That is also called a fully connected layer, or affine layer. In CNNs, instead, we use convolutions over the input layer to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output. Each layer applies different filters, typically hundreds or thousands, and combines their results. During the training phase, a CNN automatically learns the values of its filters based on

Word	Score
joy	0.791
sorrow	0.783
hope	0.781
desire	0.764
grief	0.759
despair	0.742
delight	0.737
pleasure	0.730
beauty	0.730
pain	0.729
bliss	0.729
hate	0.716
pity	0.714
true	0.709
comfort	0.706
shame	0.702
passion	0.701
faith	0.697
fear	0.697
hunger	0.695

Table 2: LOVE in the PoFo model

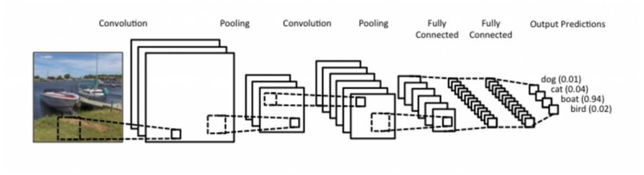


Figure 1: Layers in a Convolutional neural network.

the task. For example, in image classification, a CNN may learn to detect edges from raw pixels in the first layer, then use the edges to detect simple shapes in the second layer, and then use these shapes to deter higher-level features, such as facial shapes in higher layers. The last layer is then a classifier that uses these high-level features[1].

Instead of image pixels, the input to most NLP tasks are sentences or documents represented as a matrix. Each row of the matrix corresponds to one token, typically a word, but it could be a character. That is, each row is vector that represents a word. Typically, these vectors are word embeddings (low-dimensional representations) like word2vec (Mikolov et al. 2013) or GloVe (Pennington, Socher, and Manning 2014), but they could also be one-hot vectors that index the word into a vocabulary. For a 10 word sentence using a 100-dimensional embedding we would have a 10x100 matrix as our input[1].

In what follows we will describe our metaphor deep learning classifier and our web-based application for metaphor detection.

[1]http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Deep Learning Classification

The first requirement for deep learning classification is lots of examples / data points (by the thousands). Therefore, when we were able to collect more data (around 4870 instances) with metaphor (poetry and non-poetry), we experimented with Convolutional Neural Networks (CNN) (Kim 2014) to examine whether we can get any gains in F-score when compared to the standard machine learning classifiers. Figure 2 shows the details of the CNN text classifier schema.

We used the Keras (Chollet and others 2015) deep learning framework with a Tensorflow (Abadi et al. 2015) backend and used a local GPU to accelerate the training process. The parameters that we tested on are given in Table 3.

Parameter	**Range**
Inputs	103 - 106
Input activation function	ReLU, TANH
Hidden layers	1 - 4
Neurons in 1st layer	6 - 306
Output activation function	Softmax, Sigmoid
Dropout	0 - 0.9
Outputs	2
Epochs	20 - 1000
Loss function	Cat./Binary Cross Entropy
Optimizer	ADAM
Batch size	20 - 200

Table 3: Range of parameters tested.

The results of our experiments are given in Table 4. The best result, i.e., F-score 0.833 for metaphor and F-score 0.744 for the non-metaphor class was seen with epochs 300, batch 70, neurons 206 and inputs 103. Though we tested on hundreds of combinations of hyper-parameters, only the top results are being reported here for brevity.

It can be observed that with the same training and test set, CNN performed significantly better than SVM and other machine learning algorithms. The best F-score for metaphor class was 0.781, seen with SVM with Puk kernel. For CNN, we get a gain of 5.2% and we get a high F-score of 0.833. For the non-metaphor class, KNN obtained a F-score of 0.711. For CNN, we get a gain of 3.3% and we get a higher F-score of 0.744. For both classes, the performance is better with CNN.

The major drawback for the CNN classification (when compared to the other machine learning algorithms) is that a lot of data points are needed for training. Consequently, though we got better results for PoFo+TroFi+Shutova dataset collectively, results on the rest of the datasets individually were worse than the SVM results for both the classes. We empirically observed that, in our case, anything less than 3,000 instances was insufficient for training the CNN and the results are much worse (close to the baseline of 50%). If we are able to overcome this soft threshold of 3,000 data points, deep learning classification works appreciably better.

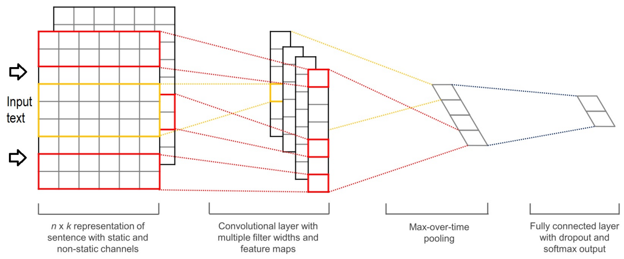

Input text
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 2: Schema diagram of the CNN text classifier.

Web Application for Generic Metaphor Detection

We used our best performing machine learning model (SVM with F-score 0.781) and developed a web application for Multi-line metaphor detection. This web application works for not just poetry, but also for any natural language text. For this application, we have not used POS tag sequence for Type 1 metaphor, instead we use Stanford NLP parser dependencies given below to extract the potential word pairs:

- nsubj
- dobj
- nsubjpass
- acl:relcl

Moreover, the pre-trained model (SVM with F-score 0.781) used for prediction in these two applications is serialized to decrease the execution time. It normally takes 10 - 12 seconds for execution. If serialization is not used, execution time can be as high as 40 seconds.

The application accepts multi-line text and outputs line-by-line result for the analysis. There can be multiple word-pairs for each line that are analysed for metaphoric intent. Figure 3 shows a screenshot of the web application. The poem (excerpt) (Lorde 2000) entered in the application is given below:

Poem Title : Afterimages (excerpt)
Author : Audre Lorde

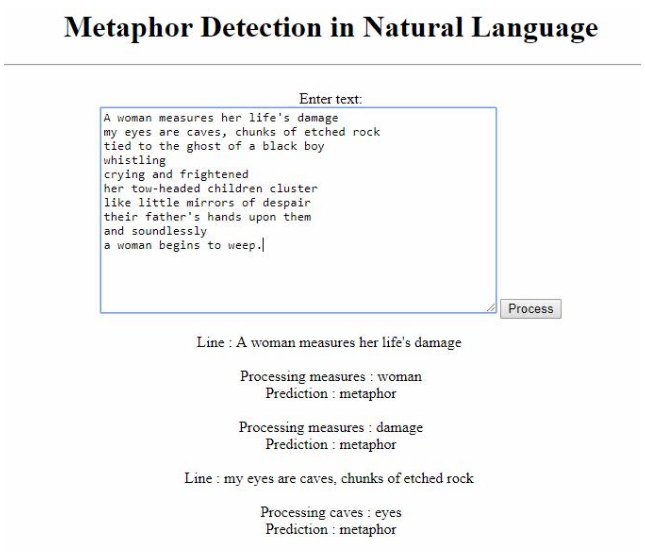
A woman measures her life's damage
my eyes are caves, chunks of etched rock
tied to the ghost of a black boy
whistling
crying and frightened
her tow-headed children cluster
like little mirrors of despair
their father's hands upon them
and soundlessly
a woman begins to weep.

The complete result from the application is given below:

	metaphor			literal		
Parameters	Precision	Recall	F-score	Precision	Recall	F-score
e:200 b:5 n:202 i:102	0.812	0.795	0.804	0.698	0.720	0.709
e:200 b:50 n:202 i:102	0.810	0.826	0.818	0.727	0.704	0.715
e:100 b:150 n:202 i:102	0.805	0.833	0.819	0.732	0.694	0.712
e:100 b:70 n:202 i:102	0.811	0.850	0.830	0.754	0.699	0.725
e:100 b:70 n:206 i:103	0.823	0.840	0.831	0.748	0.725	0.736
e:250 b:70 n:206 i:103	0.837	0.823	0.830	0.737	0.756	0.746
e:300 b:70 n:206 i:103	0.831	0.836	**0.833**	0.748	0.740	**0.744**

Table 4: Top results for CNN classification tested on variable hyper-parameters where e denotes epochs, b denotes batch size, n denotes the number of neurons in 1st layer and i denotes the number of inputs. All other hyper-parameters remain constant, input activation : RELU and output activation : SOFTMAX.

```
Line : A woman measures her life's damage
Processing measures : woman
Prediction : metaphor
Processing measures : damage
Prediction : metaphor

Line : my eyes are caves, chunks of etched rock
Processing caves : eyes
Prediction : metaphor

Line : her tow-headed children cluster
Processing cluster : children
Prediction : metaphor

Line : like little mirrors of despair
Processing like : mirrors
Prediction : metaphor

Line : their father's hands upon them
Processing hands : father
Prediction : literal

Line : a woman begins to weep.
Processing begins : woman
Prediction : literal

Time taken : 15.814 secs
```



Metaphor Detection in Natural Language

Enter text:

```
A woman measures her life's damage
my eyes are caves, chunks of etched rock
tied to the ghost of a black boy
whistling
crying and frightened
her tow-headed children cluster
like little mirrors of despair
their father's hands upon them
and soundlessly
a woman begins to weep.
```

```
Line : A woman measures her life's damage

Processing measures : woman
Prediction : metaphor

Processing measures : damage
Prediction : metaphor

Line : my eyes are caves, chunks of etched rock

Processing caves : eyes
Prediction : metaphor
```

Figure 3: Screenshot of the multi-line metaphor detection web application. The full text is not captured in this screenshot.

Conclusion and Future Work

While work has been done in artificial intelligence and in digital humanities (DH) on processing poetry computationally, see for instance the repository of digital-pedagogy-relevant poetry projects put together by Chuck Rybak (Rybak 2016) this is the first initiative in assembling a comprehensive holistic conglomerate of poetry algorithms and tools. These tools are meant for both poetry analysis and creative writing/generative purposes and tackle the multifaceted features of the genre consistently and coordinately, from topic to form to diction and style. For the latter, so far we have developed metaphor classifiers that deploy rule-based, statistical (machine learning), and deep learning methods, and we have launched a web-based metaphor natural language processing (NLP) application. While metaphor has been tackled in NLP before, the focus of that research has never been poetic metaphor (or literary tropes in general), and nor have any DH projects set as (part of) their goal(s) developing poetic metaphor processing tools. In our own work in metaphor deep learning classification we have established that deep learning, and particularly convoluted neural networks (CNN) output better results than the other previously deployed methods, given that the amount of data fed to the CNN is big enough, which also gave us the opportunity to verify the validity of the data intensive approach we have adopted generally in working on the overarching *Graph Poem* project. Our future work will involve a generic metaphor classifier that will not be dependent on any syntactical pattern and integrating the resulting web-based application into the overall network graph analysis and visualization tool.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Goodfellow, I.; Harp, A.; Irving, G.; Isard, M.; Jia, Y.; Jozefowicz, R.; Kaiser, L.; Kudlur, M.; Levenberg, J.; Mané, D.; Monga, R.; Moore, S.; Murray, D.; Olah, C.; Schuster, M.; Shlens, J.; Steiner, B.; Sutskever, I.; Talwar, K.; Tucker, P.; Vanhoucke, V.; Vasudevan, V.; Viégas, F.; Vinyals, O.; Warden, P.; Wattenberg, M.; Wicke, M.; Yu, Y.; and Zheng, X. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Agarwal, A.; Kotalwar, A.; and Rambow, O. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *IJCNLP*, 1202–1208.

Ardanuy, M. C., and Sporleder, C. 2014. Structure-based clustering of novels. In *CLfL@ EACL*, 31–39.

Armaselu, F., and van den Heuvel, C. 2017. Metaphors in digital hermeneutics: Zooming through literary, didactic and historical representations of imaginary and existing cities. *Digital Humanities Quarterly* 11(3).

Assaf, D.; Neuman, Y.; Cohen, Y.; Argamon, S.; Howard, N.; Last, M.; Frieder, O.; and Koppel, M. 2013. Why "dark thoughts" aren't really dark: A novel algorithm for metaphor identification. In *Computational Intelligence, Cognitive Algorithms, Mind, and Brain (CCMB), 2013 IEEE Symposium on*, 60–65. IEEE.

Berry, D. M. 2011. The computational turn: Thinking about the digital humanities. *Culture Machine* 12.

Chollet, F., et al. 2015. Keras. `https://github.com/fchollet/keras`.

Coles, K. 2017. Getting at metaphor. paper presented at the digital humanities 2017 conference.

D'Agostino, G., and Scala, A. 2014. *Networks of networks: the last frontier of complexity*. Springer.

Dalvean, M. 2013. Ranking contemporary american poems. *Digital Scholarship in the Humanities* 30(1):6–19.

Graham, E., and Brook, S. S. 2016. The printing press as metaphor. *Digital Humanities Quarterly* 10(3).

Hamilton, R.; Bramwell, E.; and Hough, C. 2016. Mapping metaphor with the historical thesaurus: a new resource for investigating metaphor in names.

Hayles, N. K., and Pressman, J. 2013. *Comparative textual media: Transforming the humanities in the postprint era*. University of Minnesota Press.

Kao, J., and Jurafsky, D. 2012. A computational analysis of style, affect, and imagery in contemporary poetry. In *CLfL@ NAACL-HLT*, 8–17.

Kesarwani, V.; Inkpen, D.; Szpakowicz, S.; and Tanasescu, C. 2017. Metaphor detection in a poetry corpus. In *Proceedings of the Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, 1–9.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lorde, A. 2000. *The collected poems of Audre Lorde*. WW Norton & Company.

MARGENTO. 2012. *Nomadosophy*. Max Blecher Press.

McCurdy, N.; Lein, J.; Coles, K.; and Meyer, M. 2016. Poemage: Visualizing the sonic topology of a poem. *IEEE transactions on visualization and computer graphics* 22(1):439–448.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Neuman, Y.; Assaf, D.; Cohen, Y.; Last, M.; Argamon, S.; Howard, N.; and Frieder, O. 2013. Metaphor Identification in Large Texts Corpora. *PLOS ONE* 8(4):1–9.

Núñez, A.; Gerloff, M.; Do Dinh, E.-L.; Rapp, A.; Gehring, P.; and Gurevych, I. 2017. A wind of change-shaping public opinion of the arab spring using metaphors. paper presented at the digital humanities 2017 conference.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global Vectors for Word Representation. In *Proc. EMNLP*, volume 14, 1532–1543.

Rybak, C. 2016. Poetry. *Digital Pedagogy in the Humanities: Concepts, Models, and Experiments*.

Sack, G. 2012. Character networks for narrative generation. In *Intelligent Narrative Technologies: Papers from the 2012 AIIDE Workshop, AAAI Technical Report WS-12-14*, 38–43.

Speer, R., and Havasi, C. 2012. Representing General Relational Knowledge in ConceptNet 5. In *Proc. LREC*, 3679–3686.

Tanasescu, R., and Tanasescu, C. 2018. Translator networks of networks. the case of *Asymptote* journal. *Complexity Theory (K. Marais and R. Maylaerts, Eds.)*.

Tanasescu, C.; Paget, B.; and Inkpen, D. 2016. Automatic classification of poetry by meter and rhyme. In *FLAIRS Conference*, 244–249.

Turney, P. D.; Neuman, Y.; Assaf, D.; and Cohen, Y. 2011. Literal and Metaphorical Sense Identification through Concrete and Abstract Context. In *Proc. EMNLP*, 680–690. Association for Computational Linguistics.

Vala, H.; Dimitrov, S.; Jurgens, D.; Piper, A.; and Ruths, D. 2016. Annotating characters in literary corpora: A scheme, the charles tool, and an annotated novel. In *LREC*.