Uppsala University and Gavagai at CLEF eRISK: Comparing Word Embedding Models

Elena Fano[1,2][0000−0003−1597−2040], Jussi Karlgren[2,3][0000−0003−4042−4919], and Joakim Nivre[1][0000−0002−7873−3971]

[1] Uppsala University
[2] Gavagai, Stockholm
[3] KTH Royal Institute of Technology, Stockholm

Abstract. This paper describes an experiment to evaluate the performance of three different types of semantic vectors or word embeddings—random indexing, GloVe, and ELMo—and two different classification architectures—linear regression and multi-layer perceptrons—for the specific task of identifying authors with eating disorders from writings they publish on a discussion forum. The task requires the classifier to process texts written by the authors in the sequence they were published, and to identify authors likely to be at risk of suffering from eating disorders as early as possible. The data are part of the eRISK evaluation task of CLEF 2019 and evaluated according to the eRISK metrics. Contrary to our expectations, we did not observe a clear-cut advantage using the recently popular contextualized ELMo vectors over the commonly used and much more light-weight GloVe vectors, or the more handily learnable random indexing vectors.

Keywords: Semantic vectors · Word embeddings · Author classification

1 eRISK and the Challenge of Sequence Classification

This paper describes an experiment on classifying the risk that individual authors on the Reddit discussion forum suffer from eating disorders, based on their writings. The challenge consists in sequentially processing texts written by the authors in the chronological order they were produced and to identify authors at risk based on those texts. The task is a continuation with slight modification from several years of experimentation of the eRISK evaluation lab of the CLEF conference: the experimental data consist of writings by more than a thousand authors, with on average several hundreds of texts per author over a lengthy period of time. About ten per cent of the authors have been identified to suffer from eating disorders. This is a specific case of text classification, in that more information is made available over time and that the objective is both to be accurate and to detect illness as early in the sequence as possible. The evaluation

metrics have been formulated to penalise missed cases, false positives, and late detection [13].

Most authors in the test set discuss a broad range of innocuous topics unrelated to self-harm and eating disorders. Many authors discuss eating disorders without themselves being afflicted, or even to discuss how they overcame their ailments and no longer suffer from them. To some extent, the hypotheses of the challenge task is that even other writings may reveal personality traits or social context of relevance for a diagnosis, but mostly, the task is about identifying relevant texts among many less relevant ones, and to do so quickly, since waiting incurs a penalty.

2 Previous Work

In 2017, CLEF (Conference and Labs of the Evaluation Forum) introduced a new laboratory, with the purpose to set up a shared task for Early Risk Prediction on the Internet (eRisk). The first edition was mainly meant as a trial run to chart the specific challenges and possibilities of this task.

The first full-fledged shared task was launched in 2018. In what follows, we will go over some of the strategies used by the teams that submitted a system for Task 2, detection of anorexia, focusing on the approached most similar to ours.

Roughly, the solutions can be divided into traditional machine learning approaches and other approaches based on different types of document and feature representations, but many teams used a combination of both. Some researchers also came up with innovative solutions to deal with the temporal aspect of the task.

A common theme was to focus on the difference in performance between manually engineered (meta-)linguistic features and automatic text vectorization methods. For example, contributions of [24] and [20] both dealt with this research question. For a more detailed description of [24] see below. The other team used a combination of over 50 linguistic features for two of their models, and doc2vec [11], which is a neural text vectorization method, for the other three. When they submitted their 5 runs, they used the feature-based models alone or in combination with the text vectorization models, but they report that they did not submit any doc2vec model alone because of the poor performance shown in their development experiments.

Probably the most specific challenge of this task was building a model which could take the temporal progression into account. One of the teams that obtained the best scores, the UNSL team [6], built a time-aware system which used algorithms invented specifically for this task. Among the other teams, [19] use an approach that bears some resemblance to our system. They stacked two classifiers, the first one which predicted what they call the "mood" of the texts (positive or negative), and the second which was in charge of making a decision given this prediction. The main difference is that they were operating with a chunk based system, so they had to build models of different sizes to be able

to make a prediction without having seen all the chunks, whereas our second classifier operates on a text-by-text basis. Furthermore, their first models uses Bayesian inversion on the text vectorization models, whereas we used a feed-forward neural network with LSTMs.

Other notable approaches were to look specifically at sentences which referred to the user in the first person [15], or to build different classifiers that specialized in accurately predicting positive cases and negative cases [2]. If one of the two models' output rose above a predetermined threshold of confidence, that decision was emitted; if none of the models or both of them were above the threshold, the decision was delayed. Another team used latent topics to help in classification and focused on topic extraction algorithms [14].

The FHDO team [24] employed a machine learning approach rather similar to ours for some their models. They submitted five runs to Task 2, and they obtained the best score in three out of five evaluation measures. Models three and four were regular machine learning models, whereas models one, two and five were ensemble models that combined different types of classifiers to make predictions. This team used some hand-crafted metadata features for their first model, for example the number of personal pronouns, the occurrence of some phrases like "my therapist", and the presence of words that mark cognitive processes.

Their first and second models consisted of an ensemble of logistic regression classifiers, three of them based on bags of words with different term weightings and the fourth, present only in their first model, based on the metadata features. The predictions of the classifiers were averaged and if the result was higher than 0.4 the user was classified as *at risk*. These models did not obtain any high scores, contrary to other models submitted by this team.

Their third and fourth models were convolutional neural networks (CNN) with two different types of word embeddings: GloVe and FastText. The GloVe embeddings were 50-dimensional, pre-trained on Wikipedia and news texts, whereas the FastText embeddings were 300-dimensional, and trained on social media texts expressly for this task. The architecture of the CNN was the same for both models, with one convolutional layer and 100 filters. The threshold to emit a decision of risk was set to 0.4 for model 3 and 0.7 for model 4. Unsurprisingly, the model with larger embedding size and specifically trained vectors performed best, reporting the highest recall (0.88) and the lowest ERDE_{50} (5.96%) in the 2018 edition of eRisk. ERDE stands for Early Risk Detection Error and is an evaluation metric created to track the performance of early risk detection systems (see Section 4).

The fifth model presented in [24] was an ensemble of the two CNN models and their first model, the bag of words model with metadata features. This model obtained the highest F1 in the shared task, namely 0.85, and came close to the best scores even for the two ERDE measures.

3 Experimental Conditions and Processing Pipeline

We have two experimental foci: the representation of lexical items, and the classification step given such representations.

3.1 Semantic Vectors or Word Embeddings

We represent lexical items in the posts under analysis as word embeddings, vectors of real numbers, under the assumption that a vector space representation allows for generalisation from the lexical items themselves to a more conceptual level of semantics. By allowing the classification scheme to relax the representation to include near neighbours in semantic space, we hope to achieve better recall than otherwise were possible. Semantic vectors are convenient as a learning representation, allowing for aggregation of distributional context, but if used blindly, risk bringing in contextual information of little or even confounding relevance. In general, semantic spaces built from similar data sets with similar aggregation parameters should represent the same information and the actual aggregation process is of less importance, but implementational details may have effects on the usefulness of the semantic space. Parameters of importance have obviously to do with size and selection of data set, but also how the distributional context is defined, the dimensionality or level of compression of the representation, weighting of items based on their information content, and how rare or previously unseen items are treated. In these experiments we compare three semantic vector models: Random Indexing which is used in commercial applications; GloVe, which is used in a broad range of recent academic experiments; and the recently published ELMo, which has shown great promise to provide a better and more sensitive representation for general purpose application.

Random Indexing Random indexing is based on the Sparse Distributed Memory model formulated by Pentti Kanerva [10] which is intended to be both neurophysiologically plausible and efficiently implementable for large streaming data. Random indexing is built for seamless online learning without explicit compilation steps, and is based on a fixed-dimensional representation of typically around 1000 dimensions. The vectors are built by simple operations: each lexical item is assigned a randomly generated sparse *index vector* and an initially empty *context vector*. The latter is populated for each lexical item by, for each observed occurrence of it, adding in index vectors of items observed within a context of interest such as a window of preceding and succeeding items. If the objective of the semantic space is to encode synonymy or other close semantic relations, a window of two preceding and succeeding items is used as a context. Preceding and succeeding items are kept separate to preserve sequential information in the representation, implemented by applying separate permutations for preceding and succeeding items [23]. In the present experiments, we use a large 2000-dimensional semantic space trained on several years of social and news media by Gavagai for inclusion in their commercial tools [22]. Vectors are normalised

to length 1 and items that are not found in the vocabulary are represented with empty vectors and thus do not contribute to the classification.

GloVe Global Vectors (or GloVe for short) are semantic vectors which are built to provide downstream processes with handy access to lexical cooccurrence data from large data sets [17]. The vectors are populated with data from cooccurrence within a 15-word window, thus providing a more associative relation between items than the random indexing model above. The quality of the vectors has proven useful for a wide range of tasks and GloVe vectors have in recent years been used as a standard way of achieving a conceptual generalisation from simple words in text. There are several GloVe vector sets that can be retrieved at no cost, and in these experiments we chose a 200-dimensional set provided by the Stanford NLP group trained on microblog data which we judged to be the closest fit to the data under analysis.[4] Items that are not found in the vocabulary are replaced with a stand-in vector populated with values from a normal distribution where the mean and standard deviation are obtained from all available vectors.

ELMo Semantic vector models in general produce vectors that are intended to encode information from language usage in general (or language usage in the training set). They do not accommodate to the specific task at hand and are trained on large amounts of previous knowledge. Recent approaches try to address the challenge of domain and task accommodation more explicitly by combining a previously trained general representation with a more rapid learning process on the data set under analysis. For linguistic data, ELMo (Embeddings from Language Models) proposed by [18] is one such model. ELMo representations are different from traditional semantic vectors in that individual vectors are generated for each token in the data under analysis, based on a large pretrained language model represented in a richer three-level representation trained on sentence-by-sentence cooccurrences. The ELMo processing model incorporates a character-based model which means that no items will be out of vocabulary: previously unseen items inherit a representation based on the similarity of their character sequence to other known items. We use the AllenNLP python package to generate vectors [7]. Each lexical item is represented by an average of the three ELMo layers in one 1024-dimensional vector and they are passed in, sentence by sentence to the classifier.

Baseline representation As a baseline we use randomly initialized word embeddings obtained from the Keras embedding layer. First a tokenizer is used to obtain a list of all lexical items in the training set. Only the top most common 10,000 words are considered for the classification task, and they are converted into 100-dimensional word vectors generated by Keras. These vectors thus contain no information about previous usage of the lexical items.

[4] https://nlp.stanford.edu/projects/glove/

3.2 Classifier

The first step in our processing pipeline involves building a *text classifier*. Texts are classified to be written either by authors with eating disorders or by authors without eating disorders. This is in keeping with the underlying hypothesis above, that some characteristics of authors with eating disorders may be discernible even in texts about other topics. Text classification is done with a Recurrent Neural Network (RNN) implemented with Long Short-Term Memory cells (LSTMs). Recurrent neural networks are neural architectures where the output of the hidden layer at each time step is also used as input for the hidden layer at the next time step. This type of processing model is particularly suitable for tasks that involve processing of sequences, for example sentences in natural language. LSTM cells retain information over longer distances than regular RNN cells [8]. Our neural architecture consists of an embedding layer, two hidden layers of size 100 and a fully connected layer with one neuron and sigmoid activation (as illustrated in Figure 3.2). The embedding layer differs according to which type of representations we use for each model, whereas the rest of the model is equivalent for all of our neural models. The output layer with a sigmoid activation function makes sure that the network assigns a probability to each text instead of a class label. We set the maximum sentence length to 100 words and the vocabulary to 10,000 words in order to make the training process more efficient.

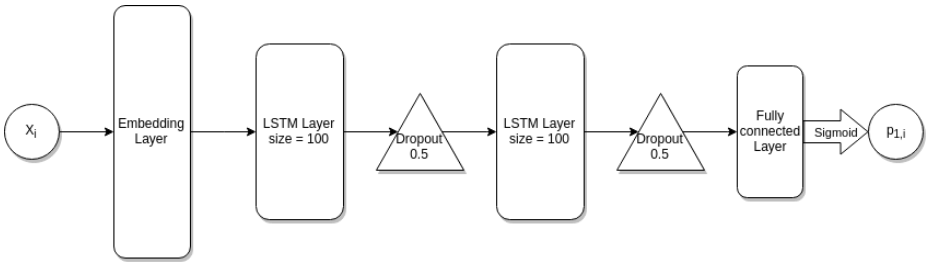

Fig. 1. The diagram illustrates the architecture of the text classifier.

This recurrent neural network takes care of the text classification task: it outputs the probability that each text belongs to the 1 (at risk) class. The output of the text classifier is passed on as input to a second *author classifier* in a feature vector composed of the following elements:

- The number of texts seen up to that point, min-max scaled to match the order of magnitude of the other features
- The average score of the texts seen up to that point
- The standard deviation of the scores seen up to that point
- The average score of the top 20% texts with the highest scores

– The difference between the average of the top 20% and the bottom 20% of texts.

We experimented with two architectures for the author classifier: logistic regression and multi-layer perceptron. Logistic regression is a linear classifier that uses a logistic function to model the probability that an instance belongs to the default class in a binary classification problem. A multi-layer perceptron, on the other hand, is a deep feed-forward neural network, and therefore a non-linear classifier. We tested their performance by feeding each architecture with identical input from the text classifier. We varied different hyperparameters such as embedding size, hidden layer size, number of layers, vocabulary size, etc., to find the best combination, also taking practical issues such as training time into account. One important factor to keep in mind is that we wanted to compare word embedding methods, so it was desirable to have the same (or very similar) settings for all models. During our development phase we found that often a hyperparameter setting that worked well for one model was not ideal for another model, and compromises had to be made.

3.3 Practical Considerations

For the implementation we use Sci-kit learn [16] and Keras [3], two popular Python packages that support traditional machine learning algorithms as well as deep learning architectures, and we use NLTK for preprocessing purposes [1]. We pre-processed the documents in the same way for all our runs: we used the stop-word list provided with the package Natural Language Toolkit, but we did not remove any pronouns, as they have been found to be more prominent in the writing style of mental health patients. We replaced URLs and long numbers with ad hoc tokens and the Keras tokenizer filters out punctuation, symbols and all types of blank space characters.

We only took into consideration those messages where at least one out of the text and title fields was not blank. Similarly, at test time we did not process blank documents, instead we repeated the prediction from the previous round if we encountered an empty document. If any empty documents appeared in the first round, we emitted a decision of 0, following the rationale that in absence of evidence we should assume that the user belongs to the majority class.

The text classifier is trained on the training set with a validation split of 0.2 using model checkpoints to save the models at each epoch, and early stopping based on validation loss. Two dropout layers are added after the hidden LSTM layers with a probability of 0.5. Both early stopping and dropout are intended to avoid overfitting, given that the noise in the data makes the model more prone to this type of error.

For the author classifier, we experimented with different settings for logistic regression and the multi-layer perceptron. For the logistic regression classifier, we used the SAGA optimizer [4]. We used balanced class weight to give the minority class (the positive cases) more importance during training. For the multi-layer perceptron, we used two hidden layers of size 10 and 2.

Since we need to focus on early prediction of positive cases and on recall, precision in our system tends to suffer. In order to improve precision as much as possible, we experimented with different cut-off points for the probability scores to try to reduce the number of false positives as much as possible. We ended up using a high cut-off probability of 0.9 for a positive decision, because we found that this did not affect our recall score too badly, and it did help improve precision. We made the practical assumption that a good balance between precision and recall would more useful in a real-life setting than really good scores on the early prediction metrics.

4 Evaluation Metrics

Precision and Recall Precision and recall are calculated over only the positive items in the test set and they are combined into the F_1 score in the traditional way.

ERDE Originally proposed by the eRisk organisers in 2016 [12] and applied in every year since, the Early Risk Detection Error (ERDE) score takes into account both the correctness of the decision and the number of texts needed to emit that decision. ERDE assigns each classification decision—in this case, identifying a user to be ill or healthy—by a system an editorially determined cost: c_{fn} for false negatives, c_{fp} for false positives, c_{tn} for true negatives, and c_{tp} for true positives. The true positive score c_{tp} is weighted by a latency cost factor $lc(o, k)$, where k is the number of texts seen by the system before a decision is made and o is a parameter to control for how many texts are considered to be acceptable or expected before a decision is made. The $lc(o, k)$ factor increases rapidly after o texts have been seen. The objective is to minimise this score. In the 2019 evaluation cycle, c_{tn} was set to 0, c_{fn} to 1, c_{fp} to the relative frequency of the positive items in the test set, c_{tp} to 1, and o variously to 5 and 50, shown as $ERDE_5$ and $ERDE_{50}$ respectively.

Latency Proposed by Sadeque et al [21], the latency measure is the *median* of the number of documents seen by the system until it makes a determination that a user is at risk. This is only computed for true positives, and thus carries no penalty for false or missed positives. The latency score is can be reformulated as a *speed* factor which is used to rescore the raw F_1 score to a latency-weighted $F_{latency}$ score. A system which identifies positive items from their first writing will have $F_1 = F_{latency}$.

Ranking based metrics: P@10, nDCG@10, nDCG@100 The participating systems were required to rank the users in order of assessed risk, and then the precision of that list was measured at 10 items, and compared to a perfect ranking at 10 and at 100 using the normalised discounted cumulative gain measure (nDCG) [9].

5 Results

5.1 Official Results

We submitted 5 runs to the official test phase, the maximum number allowed for each team. They are listed in Table 1. A total of 13 teams successfully completed at least one run in eRISK. Some teams stopped processing texts before the stream of 2000 texts was exhausted. Unfortunately, due to a processing error, our submissions were among them: we only processed the first round of texts and emitted our decisions based on that. Our official scores are thus not based on the entire test material but are an extreme case of early risk detection, based on the first text round only. The results are given in Tables 2 and 3.

Run ID	Vector type for text classifier	Author classifier
0	Baseline	Logistic Regression
1	Baseline	Multi-layer Perceptron
2	GloVe	Logistic Regression
3	GloVe	Multi-layer Perceptron
4	Random indexing	Multi-layer Perceptron

Table 1. Summary of the models used in the 5 runs submitted to the eRisk 2019 shared task.

team	run	P	R	$F1$	$ERDE_5$	$ERDE_{50}$	$latency_{TP}$	$speed$	$latency\text{-}weighted\ F1$
UppsalaNLP	0	.32	.44	.37	.06	.06	**1**	**1**	.37
UppsalaNLP	1	.36	.39	.37	.06	.06	**1**	**1**	.37
UppsalaNLP	2	.34	.42	.38	.06	.06	**1**	**1**	.38
UppsalaNLP	3	.39	.30	.34	.07	.07	**1**	**1**	.34
UppsalaNLP	4	.40	.42	.41	.06	.06	**1**	**1**	.41

Table 2. An extract of the results table provided by the organizers for the decision-based evaluation, concerning our team UppsalaNLP.

The model with the best performance was run 4, with random indexing vectors and a multi-layer perceptron. This holds for both the decision-based evaluation and the ranking-based evaluation. The only exception to this is the best

team	run	1 writing			100 writings		
		$P@10$	$NDCG@10$	$NDCG@100$	$P@10$	$NDCG@10$	$NDCG@100$
UppsalaNLP	0	.6	.59	.47	.6	.59	.47
UppsalaNLP	1	.4	.31	.40	.4	.31	.40
UppsalaNLP	2	.5	.38	.42	.5	.38	.42
UppsalaNLP	3	.7	.65	.45	.7	.65	.45
UppsalaNLP	4	.8	.75	.52	.8	.75	.52

Table 3. An extract of the results table provided by the organizers for the ranking-based evaluation, concerning our team UppsalaNLP.

recall score, obtained by the baseline model with a logistic regression classifier. In development experiments we found that the random indexing model had the least number of false positives and that the multi-layer perceptron balances precision and recall well. We believe that the GloVe model, in combination with the multi-layer perceptron, is too conservative to give a good performance after only one round of texts, whereas the random indexing model strikes a better balance early on in the data stream.

Compared to the other submissions, our scores for the decision-based evaluation were excellent in terms of latency, speed, and $ERDE_5$, since we always made our decisions at the first possible time. On most other evaluation parameters our offical scores were ranked in the lower third, if compared to the best scores of the other participants. For the ranking results, given in Table 3, the results were more respectable (although due to the processing error, they did not change as more data was processed).

5.2 Continued Experimentation

After the official testing period was over, the organizers made the test set available to the participating teams. This allowed us to carry out continued experiments, including ELMo which was not practicable during the official training period due to lengthy processing times. Table 4 shows the performance of our models on the official test set. We used a script provided by the organizers to evaluate precision, recall, F_1, and ERDE, so the results should be comparable to the official ones. These results should be comparable to the results reported in the table, because they are obtained under the same testing conditions. We found that once the processing error was sorted out, we were able to produce scores on par with the top participants: our best F_1 score on the test set was 0.68, whereas the best F_1 in the shared task was 0.71, and we obtained a recall of 0.9 which is close to the best score of 1.0, obtained by a team that heavily sacrificed precision. For ERDE_5 and ERDE_{50} more than one team shared the first place with the same non-perfect scores of respectively 0.06 and 0.03. These values are likely rounded up the the nearest percentage point, and if we do the same thing with our continued results, we actually obtain an ERDE_5 of 0.04 for all the vector representations in using the logistic regression model, and an ERDE_{50} of 0.02 for our GloVe/ELMo and logistic regression models. More de-

tails about these further experiments can be found in a comprehensive report by Fano [5].

		Baseline	Rand. Ind.	GloVe	ELMo	Best official score
LSTM classifier	Accuracy	96.17	96.57	**96.79**	96.67	
Logistic regression classifier	Accuracy	**96.33**	95.8	93.76	95.64	
	Precision	0.35	0.34	0.4	**0.41**	0.77
	Recall	0.89	0.89	**0.9**	**0.9**	1.0
	F_1	0.5	0.49	0.55	**0.56**	0.71
	$ERDE_5$	4.01	4.22	4.49	**3.64**	6
	$ERDE_50$	2.68	2.58	2.45	**2.27**	3
Multi-layer perceptron classifier	Accuracy	97.76	97	97.48	**98.13**	
	Precision	0.49	0.64	**0.68**	0.65	0.77
	Recall	**0.77**	0.63	0.67	0.7	1.0
	F_1	0.6	0.63	**0.68**	0.67	0.71
	$ERDE_5$	**5.08**	6.51	6.98	6.81	6
	$ERDE_50$	**3.46**	4.46	4.3	3.85	3

Table 4. The results of our experiments on the test set. The scores are reported in percentages, except precision, recall and F_1 which are reported in decimal points.

6 Lessons Learnt

We found that in the continued experiments, the model with GloVe embeddings and the multi-layer perceptron classifier had the best precision, without sacrificing recall. ELMo vectors did not make much of a difference for the multi-layer perceptron condition, but held a slight edge on the generally lower performing logistic regression classifier. In general, the benefit of using knowledge from the generalised vector models was relatively small. Compared to the baseline, the three pre-trained models show a better balance between precision and recall, and they also show worse ERDE scores, which are a symptom of a more conservative behavior, especially in the early phases.

Regarding the difference between the logistic regression and multi-layer perceptron classifiers, we could detect a clearer trend on the test set than we had on the development set. We had already observed that logistic regression seemed to lead to worse precision scores, but on the test set we could also determine that it also gave rise to better ERDE scores. This result can be explained as follows: if the system incurs in many false positives, it will likely also be able to correctly identify the true positives, and zooming in on many true positives early on also leads to good ERDE scores.

The more far-reaching conclusions that can be drawn from our experiments is that the choice of representation and classifier does have some effect on the

results, and that the chronological aspect of this task made clear the compound effect of learning curves and robustness of the combination of the two: more conservative models which are likely to perform better in the long run suffer from not daring to pronounce a decision early in the sequence.

References

1. Bird, S., Klein, E., Loper, E.: Natural language processing with Python: analyzing text with the natural language toolkit. O'Reilly Media, Inc. (2009)
2. Cacheda, F., Iglesias, D.F., Nóvoa, F.J., Carneiro, V.: Analysis and experiments on early detection of depression. In: Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum (2018)
3. Chollet, F., et al.: Keras. https://keras.io (2015)
4. Defazio, A., Bach, F.R., Lacoste-Julien, S.: SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. Computing Research Repository (CoRR) (2014), http://arxiv.org/abs/1407.0202
5. Fano, E.: A comparative study of word embedding methods for early risk prediction on the Internet. Master's thesis, Uppsala University (2019)
6. Funez, D.G., Ucelay, M.J.G., Villegas, M.P., Burdisso, S.G., Cagnina, L.C., Montes-y Gómez, M., Errecalde, M.L.: Unsls participation at erisk 2018 lab. In: Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum. vol. 2125 (2018)
7. Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N.F., Peters, M., Schmitz, M., Zettlemoyer, L.S.: Allennlp: A deep semantic natural language processing platform. In: arXiv (2017)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8) (1997)
9. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems (TOIS) **20**(4), 422–446 (2002)
10. Kanerva, P., Kristoferson, J., Holst, A.: Random indexing of text samples for latent semantic analysis. In: Proceedings of the 22nd Annual Meeting of the Cognitive Science Society (CogSci). vol. 22 (2000)
11. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International conference on machine learning. pp. 1188–1196 (2014)
12. Losada, D.E., Crestani, F.: A test collection for research on depression and language use. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction - 7th International Conference of the CLEF Association (2016)
13. Losada, D.E., Crestani, F., Parapar, J.: Overview of eRisk 2019: Early Risk Prediction on the Internet. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction. 10th International Conference of the CLEF Association (2019)
14. Maupomé, D., Meurs, M.J.: Using topic extraction on social media content for the early detection of depression. In: Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum (2018)
15. Ortega-Mendoza, R.M., López-Monroy, A.P., Franco-Arcega, A., y Gómez, M.M.: Peimex at erisk2018: Emphasizing personal information for depression and anorexia detection. In: Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum (2018)

16. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12** (2011)
17. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP) (2014)
18. Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics (2018)
19. Ragheb, W., Moulahi, B., Azé, J., Bringay, S., Servajean, M.: Temporal mood variation: at the clef erisk-2018 tasks for early risk detection on the internet. In: CLEF: Conference and Labs of the Evaluation. p. 78. No. 2125 (2018)
20. Ramiandrisoa, F., Mothe, J., Benamara, F., Moriceau, V.: Irit at e-risk 2018. In: E-Risk workshop. pp. 367–377 (2018)
21. Sadeque, F., Xu, D., Bethard, S.: Measuring the latency of depression detection in social media. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM). ACM (2018)
22. Sahlgren, M., Gyllensten, A.C., Espinoza, F., Hamfors, O., Karlgren, J., Olsson, F., Persson, P., Viswanathan, A., Holst, A.: The Gavagai living lexicon. In: Proceedings of the Language Resources and Evaluation Conference (LREC). ELRA (2016)
23. Sahlgren, M., Holst, A., Kanerva, P.: Permutations as a means to encode order in word space. In: Proceedings of The 30th Annual Meeting of the Cognitive Science Society (CogSci) (2008)
24. Trotzek, M., Koitka, S., Friedrich, C.M.: Word embeddings and linguistic metadata at the clef 2018 tasks for early detection of depression and anorexia. In: Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum. vol. 2125 (2018)