machine learning & knowledge extraction

MDPI

Article

Enhancing Domain-Specific Supervised Natural Language Intent Classification with a Top-Down Selective Ensemble Model

Gard B. Jenset [1,*] and Barbara McGillivray [2,3]

[1] Abaka Holdings Ltd. 1, London SW1E 5HX, UK
[2] The Alan Turing Institute, London NW1 2DB, UK; bmcgillivray@turing.ac.uk
[3] Theoretical and Applied Linguistics, Faculty of Modern and Medieval Languages, University of Cambridge, Cambridge CB3 9DB, UK
* Correspondence: gjenset@gmail.com

Received: 1 March 2019; Accepted: 16 April 2019; Published: 18 April 2019

Abstract: Natural Language Understanding (NLU) systems are essential components in many industry conversational artificial intelligence applications. There are strong incentives to develop a good NLU capability in such systems, both to improve the user experience and in the case of regulated industries for compliance reasons. We report on a series of experiments comparing the effects of optimizing word embeddings versus implementing a multi-classifier ensemble approach and conclude that in our case, only the latter approach leads to significant improvements. The study provides a high-level primer for developing NLU systems in regulated domains, as well as providing a specific baseline accuracy for evaluating NLU systems for financial guidance.

Keywords: natural language understanding; dialogue systems; multi-classifier; word embeddings; domain adaptation; conversational artificial intelligence; financial domain; long short-term memory

1. Introduction

A production dialogue system, or conversational Artificial Intelligence (AI) system, is typically built around effective machine learning models [1,2] and probabilistic resources derived from large textual corpora [3,4]. However, domain-specific applications present a challenge for Natural Language Processing (NLP). On the one hand, general NLP resources and models will tend to omit or under-represent domain-specific vocabulary and patterns [5]. On the other hand, constructing domain-specific models and resources without sufficient data is challenging [6,7]. Furthermore, dialogue systems used in an industry or production context will typically have to handle a much higher number of intents (a classification scheme for user inputs) than what is used in research [8]. In this paper, we focus specifically on the Natural Language Understanding (NLU) part of building a dialogue system, i.e., the process of categorising natural language user inputs into actionable intents in a dialogue system (e.g., "what is AI" for inputs where the user asks what AI is).

A good NLU capability is an important factor in creating a satisfactory user experience in a production dialogue system. Predicting intents for NLU in dialogue systems can be viewed as a supervised classification task. Training probabilistic models to predict such intents has been aided in recent years by combinations of deep neural network models [2,6] and distributed word representations in the form of word embeddings [3,4]. Despite these improvements, building a system that performs adequately in a production role remains a substantial challenge, even when we focus on the NLU component only and leave aside the dialogue state management [9].

Applications of dialogue systems in highly-regulated domains face the potential added challenge of providing accurate predictions for compliance reasons. An important part of that is to properly

understand what the user is asking for in the first place, in other words correctly identifying the user intent. Hence, there is naturally a strong impetus to improve the quality of the NLU continuously.

This challenge can be approached in different ways that are not mutually exclusive: improve the training data, the word embeddings, the algorithm or the model architecture. In this article, we present the results of a series of experiments comparing word embedding improvements with an architecture improvement approach, in the context of a financial guidance NLU system.

The experiments show that with reasonable amounts of high-quality training data, the word embedding post-processing techniques we tested offer no advantages. Conversely, a top-down selective Multi-Classifier System (MCS) ensemble model offers a significant improvement over our baseline.

2. Related Work and Motivation

Building an NLU system for conversational AI aimed at a production environment is a much more complicated task than simply training a supervised, deep learning multi-class classification model, where each intent corresponds to a class. More realistically, the task entails managing a large number of variables that might conceivably have an impact on the resulting ability to understand the intent of the user. These variables include, but are not limited to, the amount of training data, the quality of the training data, the number of intents relative to the amount of training data, the conceptual overlap (or distinctness) of intents, the suitability and quality of the word embeddings, the choice of neural network architecture and its parameters. Each step in the list above comes with its own challenges and research context. Hence, research related to the task at hand can be found across a number of sub-fields in NLP research. The present overview does not aim to be an exhaustive review, but is rather a highlight of some of the salient recent findings in the published literature.

As mentioned above, there are at least two key components in contemporary statistical NLU systems: the deep neural network model itself (to which we will return) and the word embedding used to encode the input to the model in a manner that captures broad similarities between similar words. Due to their ability to represent high-quality word vectors from large amounts of data [3], word embeddings have become the go-to tool for many NLP tasks. These representations capture both semantic and syntactic aspects of word representations [10]. The techniques for creating these word vectors follow the distributional hypothesis [11] by capturing distributional regularities [4], so that the distributional semantic and syntactic similarities encoded in the word vectors represent the properties that arise from multiple co-occurrences in a large training corpus. As a result, these representations tend to treat similarity very broadly, e.g., conflating synonyms and antonyms [12]. Moreover, the generality or specialization of the representations is a reflection of the data used for training the representation.

Thus, a highly pertinent topic in the context of constructing an NLU system is, as mentioned above, collecting sufficient data. Authors of previous work discussed some of the issues with collecting sufficient training data in an industry context, specifically how to collect good-quality crowd-sourced data [8]. They distinguished between scenario-based and paraphrase-based crowd-sourcing collection jobs, where the former consists of a described scenario to the crowd-source worker and the latter consists of an instruction to paraphrase an input, and concluded that both result in data of similar quality. In line with their recommendations, we collected training data from Amazon's Mechanical Turk (AMT), using a mixture of scenario and paraphrasing prompts.

For the NLU application we consider here, the role of training data is potentially two-fold:

1. Train a supervised classifier.
2. Train a word embedding.

Both tasks can be data intensive, and previous work discussed some of the challenges with developing domain-specific word embeddings, specifically in the bio-medical domain [13]. Somewhat surprisingly, the authors found that a larger training corpus does not automatically result in a better word embedding. With a corpus of 20,000 sentences, they found that corpus quality, pre-processing, choice of embedding architecture and hyper-parameters all played a role in the quality of the final

embedding. However, that was nevertheless still a larger corpus than what was available in our project. Furthermore, although we wanted word embeddings that would perform well in the domain-specific task at hand, we also wanted a good general coverage. The reason for this is the nature of the task: an NLU system should be able to handle (at least at a very general level) generic or out of domain inputs, as well as domain-specific ones. An alternative is to adapt an existing, out of domain word embedding for the domain in question [7]. The authors showed that adapting an existing word embedding, by fine-tuning it with in-domain training data, provides very good results. For these reasons, we chose to use an available pre-trained word embedding based on Twitter data [4], since that provides a good approximation to general English for an informal, digital communication channel.

Regarding the choice of neural network architecture, we chose a Long Short-Term Memory (LSTM) architecture. LSTM was a natural choice since these models have become increasingly popular in NLP research [14]. That does not, however, imply that the LSTM architecture is intrinsically better suited to NLP tasks than other architectures such as Convolutional Neural Networks (CNNs). In fact, neural network architectures have generally a very similar performance on NLP tasks, all else being equal [15]. Our own, internal preliminary exploration of model architectures reached the same conclusion, with no accuracy score being significantly better than others. A broad overview of the field describes CNN architectures as "promising" for NLP applications that are relevant to our task, whereas recurrent architectures are described as displaying "very strong" results for language modelling [2]. Based on these considerations, we chose the LSTM architecture. We kept the model architecture and hyper-parameters constant, so that we would be able to better compare the model performance, with respect to the effect of the word embeddings themselves, and the overall model architecture, i.e., single model vs. multi-classifier ensemble models.

The question of improving word embeddings is a pertinent one, since in recent years a number of light-weight post-processing techniques for improving the quality of the word embeddings have been presented. In general, these light-weight post-processing methods apply existing linguistic resources, such as FrameNet [16] or WordNet [17], to fine-tune the word vector weights found in the word embeddings based on the information in the lexical resources. This approach has been shown to improve performance in NLP tasks that are based on word embeddings, such as detecting synonyms [18]. In our case, the aim is not to use the word embeddings directly to solve a task, but rather to employ them for encoding inputs for the input layer of an LSTM classifier. Such post-processing techniques can also be used to improve performance in down-stream tasks that use word embeddings [12]. The general idea is also an attractive one from an applied perspective: if a light-weight post-processing technique, injecting knowledge from a lexical or linguistic resource, can improve general word embeddings for a domain-specific task, then that would help solve some of the problems related to data scarcity in domain-specific applications [6,7]. In principle, this might extend more generally to any application area of word embeddings for NLP tasks where data are scarce, for example historical linguistics [19,20], although the historical application scenario is outside the scope of the present article, and further research would be required to test this hypothesis.

Conversely, Multi-Classifier Systems (MCS) offer an alternative way to improve the performance of intelligent systems [21]. With complex and high-dimensional data, combining multiple heterogeneous or homogeneous classifiers into an ensemble of classifiers can improve results by smoothing out classification errors related to small training sets, overcoming problems related to local optima, or by providing a broader set of hypotheses regarding the optimal classification decision than what can be represented in a single classifier ([21], p. 5). However, an MCS approach will often lead to increased complexity in terms of architecture and evaluation [21]. This, in turn, can lead to a significant amount of hidden technical debt in the form of data and resource dependencies and configuration issues [22], which is an important practical trade-off consideration for any production implementation of an NLU system.

3. Experiments

Our starting point was an LSTM deep neural network multi-class classifier, combined with generic word embeddings and a crowd-sourced, manually-verified set of sentences, on the order of magnitude of 10,000–20,000. From this starting point, we tested a number of approaches and compared them in terms of accuracy and macro F1 score. Specifically, we tested different ways of post-processing the word embedding and compared it to an MCS approach. Twenty percent of the annotated data were randomly selected and set aside for testing. The remaining 80% used as a training set were over-sampled to account for different numbers of training sentences for different intent labels. Table 1 shows a short dialogue illustrating four domain-specific user inputs from the training data, along with their intents.

Table 1. Training data examples of domain-specific user inputs along with corresponding intents.

Input	Intent
Hi	greeting
Tell me about ISAs	what_is_isa
Can you help me open an account?	open_saving_account
How much do I need to save for my retirement?	calculate_ideal_pension_contribution

3.1. The Models

All the LSTM models were implemented in Python 3 with the Keras library [23], running TensorFlow as a backend [24]. The models were all deep neural networks with an embedding layer as input, two hidden layers, two drop-out layers and an output layer using a softmax activation function. Following previous work [15], model parameters and structure were kept the same for all models. Although this means that some scope for optimization was lost, it also makes it easier to compare results across models.

3.2. The Baseline

The model setup described in Section 3.1 corresponds to the one deployed as the core of a production NLU system at the time. We chose to use the accuracy from this model as our baseline, since our aim was to provide results that were not only statistically significant, but also of practical business value. The baseline accuracy of the model was 0.929 (95% confidence interval: [0.92, 0.94]).

3.3. Retrofitting

Word embeddings are special cases of vector space representations for words, learned from the distributional information about words in large corpora. At a theoretical level, this approach is inspired by the so-called distributional hypothesis in linguistics [11], where the meaning or relatedness of words is considered an empirical property derived from usage, specifically from a word's typical context of co-occurrence with other words. At a practical level, the idea is attractive since it allows us to estimate relatively easily word similarity at scale, so that a machine learning algorithm can learn that the words *dog* and *cat* are similar (both refer to domestic animals), without an explicit manual encoding of this relationship.

The alternative, manual approach, is found in various lexical resources where hand-crafted ontologies encode this type of information. For example, the WordNet lexical database [17], as available in the NLTK Python package [25], holds information explicitly specifying that the words for dogs and domestic cats are related via a hierarchical graph structure, since both are hyponyms of the category "domestic animal". The FrameNet database [16] extends this idea to consider also the context words with which they typically appear.

Already from the examples above, we can see a difference between the manual approach and the distributional approach. For instance, a distributional resource such as a word embedding might

indicate that *cat* is in fact more similar to *dog* than *dog* is to *puppy*. Conversely, WordNet will correctly indicate that the word *dog* is actually conceptually more similar to *puppy* than it is to *cat*. The reason is simply that in a training corpus, the word *puppy* might be used in a slightly different context compared to *cat* and *dog*, and the word embedding lacks the real-world knowledge that a puppy is in fact a type of dog.

The retrofitting approach [18] combines the two approaches outlined above. Retrofitting is a post-processing step applied to pre-trained word embeddings in order to improve the quality of the embedding. Specifically, this is achieved by creating a graph derived from information in a lexical database, where the vectors for words that are in the vocabulary of the lexical database are adjusted via an iterative optimisation procedure, so that the output is new word vectors where the weights are more similar for similar words in the lexical database. The approach is applicable to word embeddings trained with any algorithm, and the authors have made the Python code available online. The original retrofitting code was in Python 2. For our experiments, it was converted to Python 3.

Retrofitting has been shown to be a useful post-processing step for improving synonym handling and semantic relatedness [26,27]. A limitation of the retrofitting algorithm is that from a distributional perspective synonyms and antonyms are often considered similar by virtue of occurring in similar contexts. Further work in this area of research has resulted in more complex algorithms that also take into consideration antonyms from a lexical resource [28], as well as adjustments for words that are not in the lexicon [12].

In our experiments, we decided to use the original retrofitting algorithm for several reasons. Retrofitting has been shown to yield good results with respect to producing a specialised word embedding that is optimised for relatedness. In our case, we are predominantly concerned with the relatedness in general, rather than the more nuanced synonym-antonym distinction. Finally, from the perspective of a software startup company working according to agile principles, starting with a simple, Minimum Viable Product (MVP) approach [29] to validate the basic hypothesis is a natural step.

The sections below briefly outline the approaches followed for retrofitting word embeddings and training different types of models. We ran a number of experiments involving retrofitting word embeddings with out-of-the box lexical resources, various types of domain-specific empirically-constructed resources for related words and, finally, the MCS ensemble approach, where no retrofitting was involved. The two broad approaches to training a model, viz. enriching a word embedding and training models solely based on the training data and the word embeddings, are schematically illustrated in Figures 1 and 2, respectively.

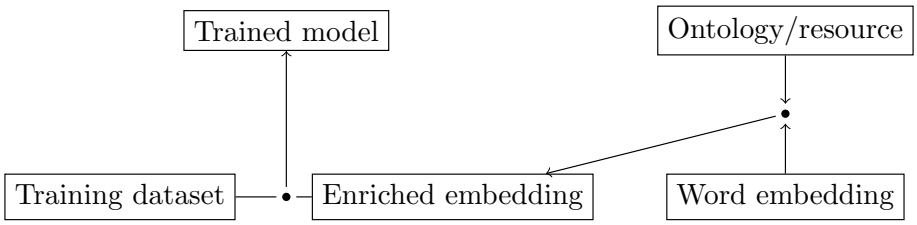

Figure 1. Model training with retrofitting of word embeddings. The word embeddings are adjusted based on an ontology or other resource. The result is then combined with training data to train the final model.

3.3.1. Retrofitting with Out-of-the-Box Resources

As a first step, we trained two models where the word embedding had been post-processed using FrameNet and WordNet resources. The purpose of this step was to serve as an additional baseline to compare retrofitting with domain-specific ontologies, and thereby control for any potential differences arising from how the domain-specific ontology was created.

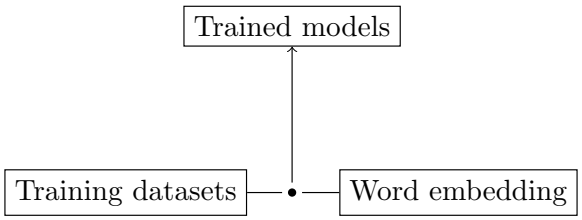

Figure 2. Model training with the Multi-Classifier System (MCS) ensemble approach. No adjustment takes place. One model is trained per sub-domain, as well as one meta-classifier is trained in this fashion.

3.3.2. Retrofitting with Bespoke Domain Ontologies

We created two different resources for injecting specialised information into our word embeddings before the model training stage. We defined "domain ontology" in the following manner:

- A collection of words (tokens),
- that are typical or indicative of an intent label,
- grouped on the basis of intent labels.

A very simple example is shown below for the intent "what is AI":

```
"what_is_ai": ["definition", "artificial", "intelligence", "define", "ai"]
```

The logic behind this is that the sentences in the training data already contain information about similarities with respect to our domain, by virtue of being grouped by intent labels. In practice, this was achieved by calculating TF-IDF scores [30] for all word tokens in the training data, where we treated each intent label as a document. For each intent label, we treated all tokens with a TF-IDF score above the median for that intent as being both related and sufficiently distinctive of the label for our purposes. The resulting data were saved in a format appropriate for use as input to the retrofitting step. This file was used directly as input for a retrofitting experiment ("Retrofitted empirical lexicon" in Table 2).

Table 2. Accuracy and macro F1 scores for the different approaches investigated. Accuracy values statistically significantly improving on the baseline are in bold.

System	Accuracy	Avg. Macro F1
Baseline	0.929	0.922
Retrofitted with FrameNet	0.929	0.918
Retrofitted with WordNet	0.922	0.914
Retrofitted empirical lexicon	0.929	0.922
Retrofitted with manually-edited synonyms	0.926	0.915
MCS without grouping	0.910	0.904
MCS with grouping	**0.971**	0.949

Secondly, we created a manually-edited version of the resource above ("Retrofitted with manually-edited synonyms" in Table 2). This file was also used directly as input for a retrofitting experiment.

For each of the domain resources, we created a retrofitted word embedding and used that as the basis for training a classifier.

3.4. Ensemble Approach

There are two broad paradigms for creating ensemble machine learning models: model choice and model overproduction [21,31]. In the former approach, a top-level model is used to select the likely best model for the task at hand. In the latter, several models are invoked, and each produces an

output that is typically either used as input to a meta-classifier or as input to some voting algorithm. For the MCS ensemble testing, we settled for testing the simpler, model selection approach, following the MVP logic outlined above [29].

We tested two types of MCS ensembles based on the logic above. In the first case, we split our intent labels based on prefix. In our intent system, each intent label had a semantically-meaningful prefix indicating a sub-domain. For instance, intents dealing with general user inputs (e.g., greetings or small talk) start with "G", intents dealing with pensions start with "P", intents dealing with personal finance start with "PF", and so on. For each of these prefixes, we trained a separate, specialised model that learned to distinguish between the individual intents within the group. Then, we trained a separate top-level model that learned to associated inputs with the intent group ("MCS without grouping" in Table 2).

In the second case, we minimized between-group confusion by merging three of the most general intent groups into a joint group that we labelled "core". As above, we trained a separate top-level model that learned to associate inputs with the intent group ("MCS with grouping" in Table 2).

The training data for each sub-model were prepared following a majority bias approach, so that each sub-model was predominantly trained on data specific to the relevant sub-domain, but with very small amounts of data for all other intents present. This ensured that each model could produce scores and outputs for all intents, but with a bias towards those included in the sub-domain.

4. Results and Discussion

The results of the experiments are listed in Table 2, which provides accuracy and F1 scores for each model. None of the accuracy values for the models using retrofitted word embeddings were statistically significantly different from the baseline. This was tested with a binomial test, specifically the `binom.test()` function in R [32], alpha = 0.05. This holds for both the experiments using the out-of-the box WordNet and FrameNet resources, as well as the various domain-specific resources we created, including the manually-edited domain-specific resources. This suggests that the lack of improvement was not directly caused by the resources per se, but can more likely be attributed to the retrofitting step itself.

For the MCS experiments, the MCS ensemble without grouping (i.e., using each original intent prefix as a group with its own sub-model) performed slightly worse than all the other models. In fact, it is just below the 95% confidence interval of the baseline model. This suggests that simply splitting the intents into groups in itself leads to a loss of information that outweighs any gains from the clustering. The only model to beat the baseline was the MCS with grouping, where some of the intent labels had been grouped in such a manner that they shared a sub-model. This approach proved statistically significantly better than the baseline. We did not attempt to combine the two approaches, an exercise we leave for future studies.

There are several interesting insights to be drawn from these results. Regarding the value of retrofitting for down-stream NLP tasks, we found that this does not hold for our task. Of course, this says nothing about the value of retrofitting for tasks that rely on the word embedding directly (e.g., for similarity or relatedness tasks) or for other down-stream NLP tasks. In the case of using the retrofitted word embedding as the embedding layer for predicting intents in an NLU scenario, we could detect no benefits either for accuracy or macro F1. This in turn points to two wider lessons. First, other researchers have shown that word embeddings themselves can be surprisingly unstable, even for high-frequency words [33,34]. Secondly, it is possible that the light-touch retrofitting algorithm is simply too subtle for our purposes and that the effects are ephemeral compared to the process of training the LSTM classifier with the training data. Lastly, we should consider the effect of the amount of training data itself. It is possible that we simply had too much training data to benefit from the retrofitting and that the effect would have made itself felt with a smaller amount of data. This remains a question for further research.

Regarding the MCS approach, we found that the best MCS was in fact one in which we explicitly encoded some information about the domain in the form of a grouping structure and then trained a series of specialised sub-models. Interestingly, simply taking the original intent groups at face value did not improve the results. Only when we used human expert judgement to merge some groups did we see a statistically-significant improvement. Effectively, this meant that the ensemble was constructed in such a way that the meta-model would restrict choices by selecting a sub-model that was heavily biased towards the set of intents associated with it. We hypothesize that the success of this approach can be explained in part by our domain-specific data. Previous exploration of previous models trained on similar data showed that there was a tendency for between-group errors (i.e., errors between the different sub-domains). The restricting nature of the selective ensemble model we presented here might thus have acted to prevent such between-group errors. Hence, the high accuracy reported with this model is probably a combined function of both the model approach, as well as the domain-specific nature of the data on which it was trained.

Although this manual intervention might seem like a drawback from the perspective of training highly-efficient classifiers from data alone, we argue that this is actually an advantage. There is a growing discussion on the lack of transparency that statistical NLP models represent [35,36], a concern that is also echoed in machine learning generally [37–40]. As black boxes, such models offer little in the way of insights as to how language functions. Whether this should be the main concern of NLU models is debatable. However, from a regulatory perspective, there are clear benefits pertaining to greater transparency in how NLU models operate, meaning that the linguistic concerns have at least partially an industry counterpart.

On a theoretical level, it is interesting to note that our best-performing MCS model relies on understanding sentences on different levels: at a group or macro level, as well as at an intent or micro level. This seems to us to correspond at least partly to the structuralist linguistic notion of levels of analysis [41], where language could be understood as composed of a series of discrete levels (phonological, morphological, syntactic, etc.), each of which could be analysed separately in its own right, while simultaneously providing the bases for the next, higher level. We suggest that this model could provide, at least partially, a theoretical linguistic motivation for the MCS model in question. A similar argument has been put forth by others, who argued that the common "bag of words" model used in NLP (and criticised by some linguists for being too simple) can in fact be theoretically motivated in terms of theoretical discourse analysis [42]. We believe that these themes, namely transparency, explicit modelling of knowledge that we as humans take for granted and models that increasingly are motivated by linguistic insights, will be important factors in developing more sophisticated NLP models in the future, both for academic research and for industry applications.

5. Conclusions and Future Work

In this paper, we approached the task of building and optimizing an NLU system from an applied, industry angle. This involved performing tests with a larger number of NLU intents than what is often used for academic research [8] and aiming to improve the accuracy and F1 scores in a particular, regulated domain. Specifically, in light of common constraints such as a lack of large amounts of data, we compared and evaluated two approaches to improving the scores: post-processing pre-trained word embeddings using the retrofitting approach [18] versus switching to an MCS approach [21].

By designing a two-level ensemble classifier around the implicit semantic classification of our intent scheme, we managed to significantly improve the accuracy of our classifier. In contrast, the various post-processing approaches to the word embeddings produced results that were statistically indistinguishable from the baseline.

We offered an analysis of the results in terms of the metaphor of levels of linguistic analysis [41] and argued that by treating the NLU classifier proper as a multi-component (possibly hierarchical) level, we can move towards the transparency that others have called for in computational linguistics and NLP applications [35,36].

In summary, for our task of domain-specific intent label classification, we achieved a substantial and robust, statistically-significant result by means of an MCS approach that outperformed retrofitting modifications to the word embeddings for our domain-specific data. This attests to the importance of taking a holistic view (including considerations of the data themselves), over and above individual specific improvements to algorithms and resources, when developing domain-specific industry NLU applications. In addition, we established baseline accuracy and F1 scores for this specific task in the domain of financial guidance.

Author Contributions: Conceptualization, G.B.J. and B.M.; methodology, G.B.J. and B.M.; software, G.B.J.; validation, G.B.J.; formal analysis, G.B.J.; resources, G.B.J.; data curation, G.B.J.; writing, original draft preparation, G.B.J.; writing, review and editing, G.B.J. and B.M.; visualization, G.B.J. and B.M.

Funding: This work was supported by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Bengio, Y.; Ducharme, R.; Vincent, P.; Jauvin, C. A neural probabilistic language model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.
2. Goldberg, Y. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.* **2016**, *57*, 345–420. [CrossRef]
3. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
4. 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), Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
5. Nooralahzadeh, F.; Øvrelid, L.; Lønning, J.T. Evaluation of Domain-specific Word Embeddings using Knowledge Resources. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018), Miyazaki, Japan, 7–12 May 2018.
6. Neuraz, A.; Llanos, L.C.; Burgun, A.; Rosset, S. Natural language understanding for task oriented dialog in the biomedical domain in a low resources context. *arXiv* **2018**, arXiv:1811.09417.
7. Zhang, Y.; Li, H.; Wang, J.; Cohen, T.; Roberts, K.; Xu, H. Adapting Word Embeddings from Multiple Domains to Symptom Recognition from Psychiatric Notes. *AMIA Summits Transl. Sci. Proc.* **2018**, *2017*, 281–289.
8. Kang, Y.; Zhang, Y.; Kummerfeld, J.K.; Tang, L.; Mars, J. Data Collection for Dialogue System: A Startup Perspective. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers), New Orleans, LA, USA, 1–6 June 2018; Volume 3, pp. 33–40.
9. Yang, X.; Chen, Y.N.; Hakkani-Tür, D.; Crook, P.; Li, X.; Gao, J.; Deng, L. End-to-end joint learning of natural language understanding and dialogue manager. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA, USA, 5–9 March 2017; pp. 5690–5694.
10. Mikolov, T.; Yih, W.t.; Zweig, G. Linguistic regularities in continuous space word representations. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Atlanta, GA, USA, 9–14 June 2013; pp. 746–751.
11. Harris, Z.S. Distributional structure. *Word* **1954**, *10*, 146–162. [CrossRef]
12. Vulić, I.; Glavaš, G.; Mrkšić, N.; Korhonen, A. Post-Specialisation: Retrofitting Vectors of Words Unseen in Lexical Resources. *arXiv* **2018**, arXiv:1805.03228.
13. Chiu, B.; Crichton, G.; Korhonen, A.; Pyysalo, S. How to train good word embeddings for biomedical NLP. In Proceedings of the 15th Workshop on Biomedical Natural Language Processing, Berlin, Germany, 12 August 2016; pp. 166–174.
14. Sundermeyer, M.; Schlüter, R.; Ney, H. LSTM neural networks for language modeling. In Proceedings of the Thirteenth Annual Conference of the International Speech Communication Association, Portland, OR, USA, 9–13 September 2012.
15. Yin, W.; Kann, K.; Yu, M.; Schütze, H. Comparative study of CNN and RNN for natural language processing. *arXiv* **2017**, arXiv:1702.01923.

16. Baker, C.F.; Fillmore, C.J.; Lowe, J.B. The berkeley framenet project. In Proceedings of the 17th International Conference on Computational Linguistics-Volume 1, Montreal, QC, Canada, 10–14 August 1998; pp. 86–90.
17. Miller, G.A. WordNet: A lexical database for English. *Commun. ACM* **1995**, *38*, 39–41. [CrossRef]
18. Faruqui, M.; Dodge, J.; Jauhar, S.K.; Dyer, C.; Hovy, E.; Smith, N.A. Retrofitting Word Vectors to Semantic Lexicons. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, CO, USA, 31 May–5 June 2015; pp. 1606–1615.
19. Jenset, G.B.; McGillivray, B. *Quantitative Historical Linguistics: A Corpus Framework*; Oxford University Press: Oxford, UK, 2017.
20. McGillivray, B.; Hengchen, S.; Lähteenoja, V.; Palma, M.; Vatri, A. A computational approach to lexical polysemy in Ancient Greek. *Digit. Scholarsh. Humanit.* **2019**, in press.
21. Woźniak, M.; Graña, M.; Corchado, E. A survey of multiple classifier systems as hybrid systems. *Inf. Fusion* **2014**, *16*, 3–17. [CrossRef]
22. Sculley, D.; Holt, G.; Golovin, D.; Davydov, E.; Phillips, T.; Ebner, D.; Chaudhary, V.; Young, M.; Crespo, J.F.; Dennison, D. Hidden technical debt in machine learning systems. In Proceedings of 29th Annual Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; pp. 2503–2511.
23. Chollet, F. Keras. 2015. Available online: https://github.com/fchollet/keras (accessed on 17 April 2019).
24. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Available online: tensorflow.org (accessed on 17 April 2019).
25. Bird, S.; Klein, E.; Loper, E. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*; O'Reilly: Sebastopol, CA, USA, 2009.
26. Kiela, D.; Hill, F.; Clark, S. Specializing word embeddings for similarity or relatedness. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 2044–2048.
27. Yu, Z.; Cohen, T.; Wallace, B.; Bernstam, E.; Johnson, T. Retrofitting word vectors of mesh terms to improve semantic similarity measures. In Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis, Austin, TX, USA, 5 November 2016; pp. 43–51.
28. Mrkšić, N.; Vulić, I.; Séaghdha, D.Ó.; Leviant, I.; Reichart, R.; Gašić, M.; Korhonen, A.; Young, S. Semantic specialisation of distributional word vector spaces using monolingual and cross-lingual constraints. *arXiv* **2017**, arXiv:1706.00374.
29. Duc, A.N.; Abrahamsson, P. Minimum viable product or multiple facet product? The Role of MVP in software startups. In *Agile Processes, in Software Engineering, and Extreme Programming. XP 2016*; Vol. 251, Lecture Notes in Business Information Processing; Sharp, H., Hall, T., Eds.; Springer: Cham, Switzerland, 2016; pp. 118–130.
30. Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. *J. Doc.* **1972**, *28*, 11–21. [CrossRef]
31. Roli, F.; Giacinto, G.; Vernazza, G. Methods for designing multiple classifier systems. In *International Workshop on Multiple Classifier Systems*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 78–87.
32. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2017.
33. Antoniak, M.; Mimno, D. Evaluating the stability of embedding-based word similarities. *Trans. Assoc. Comput. Linguist.* **2018**, *6*, 107–119. [CrossRef]
34. Wendlandt, L.; Kummerfeld, J.K.; Mihalcea, R. Factors Influencing the Surprising Instability of Word Embeddings. *arXiv* **2018**, arXiv:1804.09692.
35. Manning, C.D. Computational linguistics and deep learning. *Comput. Linguist.* **2015**, *41*, 701–707. [CrossRef]
36. Church, K.W. Emerging trends: I did it, I did it, I did it, but.... *Nat. Lang. Eng.* **2017**, *23*, 473–480. [CrossRef]
37. Ribeiro, M.T.; Singh, S.; Guestrin, C. Why should I trust you?: Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144.
38. Biran, O.; Cotton, C. Explanation and justification in machine learning: A survey. In Proceedings of the IJCAI-17 Workshop on Explainable AI (XAI), Melbourne, Australia, 20 August 2017; pp. 8–13.

39. Arras, L.; Horn, F.; Montavon, G.; Müller, K.R.; Samek, W. "What is relevant in a text document?": An interpretable machine learning approach. *PLoS ONE* **2017**, *12*, e0181142. [CrossRef] [PubMed]
40. Holzinger, A. Introduction to machine learning and knowledge extraction (MAKE). *Mach. Learn. Knowl. Extr.* **2017**, *1*, 1–20. [CrossRef]
41. Koerner, E.F. Bloomfieldian Linguistics and the Problem of "Meaning": A Chapter in the History of the Theory and Study of Language. *Jahrbuch für Amerikastudien* **1970**, *15*, 162–183.
42. Roe, G.; Gladstone, C.; Morrissey, R. Discourses and Disciplines in the Enlightenment: Topic modeling the French Encyclopédie. *Front. Digit. Humanit.* **2016**, *2*, 8. [CrossRef]