# A Neural Network Component for Knowledge-Based Semantic Representations of Text

**Alejandro Piad-Morffis**[1], **Rafael Muñoz**[2], **Yudivián Almeida-Cruz**[1], **Yoan Gutiérrez**[3], **Suilan Estevez-Velarde**[1], and **Andrés Montoyo**[2]

[1]School of Math and Computer Science, University of Havana, Cuba
{sestevez,yudy,apiad}@matcom.uh.cu
[2]Department of Languages and Computing Systems, University of Alicante, Spain
[3]U.I. for Computer Research (IUII), University of Alicante, Spain
{montoyo,ygutierrez,rafael}@dlsi.ua.es

## Abstract

This paper presents Semantic Neural Networks (SNNs), a knowledge-aware component based on deep learning. SNNs can be trained to encode explicit semantic knowledge from an arbitrary knowledge base, and can subsequently be combined with other deep learning architectures. At prediction time, SNNs provide a semantic encoding extracted from the input data, which can be exploited by other neural network components to build extended representation models that can face alternative problems. The SNN architecture is defined in terms of the concepts and relations present in a knowledge base. Based on this architecture, a training procedure is developed. Finally, an experimental setup is presented to illustrate the behaviour and performance of a SNN for a specific NLP problem, in this case, opinion mining for the classification of movie reviews.

## 1 Introduction

In recent years, the increase in the volume of available data has provided both new techniques and new challenges for discovering relevant knowledge. The huge amount of information produced daily makes it impossible for humans to manually build organized representations of all this information. On the other hand, the surplus of information enables the design of statistical learning techniques which scale better with large data. Deep learning approaches have successfully obtained state-of-the-art results for many learning problems, from image recognition to natural language parsing. Instead of handcrafted representations designed by experts, deep learning automatically builds representations from raw data that are suitable for a given machine learning problem.

When deep learning is used, we can often identify structures inside a neural network, which can be explained as high-level concepts that are automatically discovered during the learning process (Bengio, 2012). For instance, in the image recognition domain, often internal neurons or groups of neurons can be identified to recognize common visual features such as textures or patterns or even specific objects (Le, 2013). Hence, the neural network is able to build high-level concepts from low-level input (i.e., pixels). This ability to discover higher-level features is one of the main strengths of deep learning and representational learning in general (Bengio, 2012). However, one of the main challenges of deep learning is interpreting the internal representations of a neural network in terms that can be understood by humans and mapped to clearly defined domain concepts (Montavon et al., 2017).

To address this challenge, our proposal is to“persuade” a neural network to build representations that can be interpreted in terms of a formal conceptualization (such as a knowledge base). This intuitive approach, as opposed to hand-crafting features, would still allow the neural network to learn the best representation of the input, while enabling a better interpretation and explanation of the whole learning process.

Our approach involves designing specific neural network components —Semantic Neural Networks (SNNs)— that are trained to learn how to map raw input to specific domain concepts that are extracted from a convenient knowledge base. These components could then be included in larger deep learning architectures, providing a semantic representation of the input that the rest of the network could learn to use for solving a specific problem.

https://doi.org/10.26615/978-954-452-056-4_105

In this work, SNNs have been designed to represent the relevant concepts of a knowledge base as part of the artificial network architecture. Hence, during the whole process, the structures that map to specific concepts and relations are clearly identifiable inside the network. This process attaches a semantic meaning to the network architecture, which is useful for debugging and understanding the learning dynamics. Furthermore, the SNN is trained to learn the specific instances in the knowledge base and their relations. This way, the SNN not only encodes abstract concepts, but also true facts about instances of those concepts. Preliminary source code is available online for the research community.[1]

When used as a component of a larger deep learning architecture, a SNN that is trained for a specific knowledge domain can be seen as a semantic representational component. Its input consists of a low-level representation of data, (for example, words or entities), and its output consists of an implicit representation of this data expressed in terms of the learned domain. This representation can be seen as a type of embedding that maps raw input to a semantic space defined by the concepts and relations of the learned knowledge base. The SNN is used as a representational layer in a larger neural network, for a natural language processing problem in which the semantic representation induced by the learned knowledge base is expected to be a good representation.

The paper is organized as follows. Section 2 presents a review of related works and relevant concepts in the domain of representational learning, with an emphasis on different semantic representations of natural language. In section 3 the architecture and training procedure for the SNN is formalized and explained. Section 4 presents a brief experimental setup designed to illustrate the behavior and performance of a SNN in a specific natural language problem. Finally, section 5 discusses the main contributions of this proposal, whereas section 6 presents the final conclusions of the research and highlights possible future lines of development.

## 2 Related Works

Building semantic representations of raw input data, specifically for natural language text, is a common task for many machine learning problems. In this section we present different approaches to the design of semantic representations for natural text.

**Network-Based Approaches** for semantic representations usually consist of defining some similarity metric based on the relations of terms in some knowledge base, interpreted as a graph. It is based in the assumption that words which are connected by short paths in a knowledge base should have similar semantics. WordNet is commonly used as a knowledge base where different semantic relations among words can be exploited for defining similarity metrics. Using WordNet (Miller, 1995), several semantic similarity metrics are defined by exploring the graph structure of the knowledge base, mostly depending on the graph structure of words, such as *Hirst-St-Onge* (Hirst and St-Onge, 1998) and *Leacock-Chodorow* (Leacock and Chodorow, 1998). In this direction, other researchers include information content formulae to measure appearances of terms in a corpus, such as the *Resnik* metric (Resnik, 1999).

**Corpus-Based Similarity Metrics** are defined by some measure of the co-occurrence of terms in a corpus of natural text. The intuitive idea is that words which co-occur within the same context must have similar semantics. One such metric is PMI-IR (*point mutual information - information retrieval*) (Turney, 2001), which considers the information content of each pair of words $(w_i, w_j)$, measured as the relative number of co-occurrences of $w_i$ and $w_j$ in a document, with respect to the individual count of occurrences of each word. Another corpus-based similarity metric based is ESA (*explicit semantic analysis*) (Gabrilovich and Markovitch, 2007), which considers Wikipedia as a corpus for building a co-occurrence matrix of words. A similar approach is HAL (*hyperspace analogue to language*) (Lund and Burgess, 1996), which also builds a co-occurrence matrix, but only considering words within a small window.

**Dimensionality Reduction Techniques** such as PCA (*principal component analysis*) (Martinsson et al., 2011) can be interpreted as a projection from the BOW (bag of words) or TF-IDF (term frequency - inverse document frequency) space to a semantic space. An interesting recent approach, that mixes ideas from the previous tech-

[1]`https://github.com/knowledge-learning/snn`

niques, is the family of word embedding algorithms (Mikolov et al., 2013). In word embeddings, similar to PCA, each word is mapped to a vector which encodes the semantics of the word. The embedding is chosen such that a word's vector contains an implicit representation of the probabilistic distribution of the word's context in a given corpus. To achieve this, a neural network is trained to predict, given a word $w_i$'s embedding, the probability that some other word $w_j$ appears in a small window centered around $w_i$. In this sense, word embeddings can be seen as a generalization of corpus-based metrics, whereby the best representation is learned from the data, rather than handcrafted. Even though word embeddings don't explicitly model specific semantic relations (such as hypernymy, or synonymy), it has been shown that several interesting semantic relations get encoded in specific directions in the embedding space, enabling the solution of analogue inference queries (Schnabel et al., 2015).

**Entity Embeddings** are a specific type of embedding technique that encodes the context of entities in a knowledge base. Several metrics can be used to define the notion of "context similarity" when using a knowledge base for entity embedding. For example, embeddings can be designed such that a particular direction $d_r$ is associated with each particular relation of the knowledge base, such that $e_i + d_r \approx e_j$ whenever $e_i$ and $e_j$ are related by $r$. These formulations allow a semantic meaning to be attached to a particular algebraic operation and properties, and enable a whole new field of study that finds the "meaning" of, say, other directions $d$ which are orthogonal to or linearly dependent on a specific relation. Entity embeddings have been extended to encode also the hierarchical structure of knowledge bases (Hu et al., 2015) and mixed with word embeddings for tasks such as entity disambiguation (Yamada et al., 2016).

Word and entity embeddings in general are promising approaches to deal with learning semantic representations of data. Moreover, recent research deals with finding ways to exploit the structure of these representations to explain why a specific answer is output by a neural network. Being able to explain neural networks is a first step towards designing accountable machine learning systems that humans can trust for solving the most crucial problems (e.g.medical diagnosis or legal advice). By carefully designing the learning criteria and structure of embeddings, it is conceivable that a semantic representation can be interpreted in terms of a formal conceptualization defined *a priori*.

## 3 Semantic Neural Networks

We define a Semantic Neural Network (SNN) as an artificial neural network architecture that encodes knowledge. Two main semantic elements are encoded from the Knowledge Base. First, the graph structure of the Knowledge Base (i.e. entity classes and their relations) is directly represented in the architecture of a SNN. Second from a Knowledge Base $KB$, the information about which instances belong to which entity classes and their specific relations are encoded into the weights of the SNN. By design, the architecture of the SNN is built to represent each specific entity class and relation in a clearly recognizable structure (a set of neurons with a pre-designed connection pattern). This allows a semantic meaning to be attached to an activation of the SNN in terms of the classes and relations defined in the Knowledge Base.

The purpose of the SNN is to provide a semantic representation of the input data that can be used as component inside a larger deep learning architecture, to solve a related learning problem $L$. If the knowledge represented in $KB$ is useful for solving the problem $L$, then the representation provided by the SNN should be richer than plain bag-of-words or general purpose embeddings. With the same computational power (same number of parameters), a deep learning architecture using a SNN is expected to achieve equal or better performance than using other representations not specifically designed to exploit the knowledge in $KB$. Furthermore, the SNN architecture provides a semantic explanation for the model's predictions.

A SNN is built based on a specific Knowledge Base $KB$ which is of interest for solving a related learning problem $L$. Figure 1 shows a schematic representation of the process for constructing, training and using an SNN in an arbitrary learning problem $L$. First, given the problem $L$ to solve, a relevant Knowledge Base $KB$ is chosen. The entity classes and relations of $KB$ are used to define the architecture of the SNN (Section 3.1). Then, training instances are

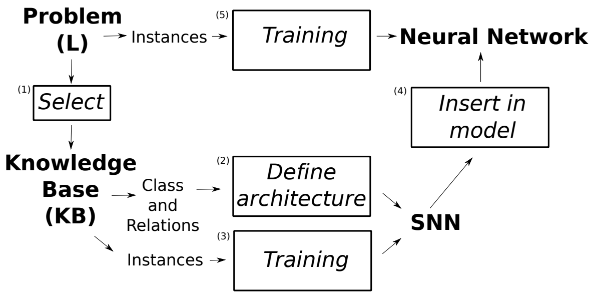

Figure 1: The process for constructing and using a SNN for a specific learning problem ($L$) and a suitable Knowledge Base ($KB$). The SNN is first defined and pre-trained based on $KB$, and then used in a larger neural network trained specifically for $L$.

extracted from $KB$ and used to pre-train the SNN weights (Section 3.2.1). Afterwards, the SNN is included in a deep learning model (which can contain other components such as extra layers). Finally, this larger model is trained on instances from $L$ using standard optimization techniques and loss functions suitable for the problem $L$ (Section 3.2.2).

Different Semantic Neural Networks can be trained on different Knowledge Bases ahead of time and reused for many related problems. In this sense, SNNs are similar to pre-trained word embeddings, since a SNN trained for a commonly used Knowledge Base (i.e., DBPedia or WordNet) could be useful in solving different problems. However, pre-trained SNNs can (and should) be fine-tuned on a specific problem $L$ after deciding a convenient deep learning architecture for this problem.

### 3.1 Architecture of the Semantic Neural Network

The architecture of a Semantic Neural Network (SNN) is composed of several instances of two simple structures: **entity blocks** and **relation blocks**. For each class of the knowledge base, an entity block is created, and for each relation, a corresponding relation block.

An **entity block** is a computational graph with a single input variable and a single output variable. The input dimension and shape is determined by the specific application, a sensible default consists of a single one-hot encoding layer when using a bag-of-words representation, but an alternative input could be a general-purpose word embedding representation. The output dimension is a fixed size dense vector of small dimension (e.g., 10). The input and output are connected by a linear matrix operator. Additionally, a one-dimensional indicator neuron is connected to the output layer through a dot product operator with a sigmoid activation function. The purpose of the indicator is to signal when the activation of the output is large in absolute value. This is interpreted as the importance of the corresponding concept in a particular input text.

A **relation block** is a similar computational graph, but with an input variable whose size is twice the entity output size. Thus, the relation input shape corresponds with the output shape of the two entity blocks that will be connected. The outputs from each incoming entity block are concatenated, forming a single vector, which is then connected through a linear matrix operator to the output variable. An identical indicator neuron is connected to the output variable.

The overall architecture of an SNN consists of several entity blocks, one for each class in the knowledge base, and several relation blocks, one for each relation defined. The entity blocks are all connected to a single input (e.g., a bag of words representation). Every relation block is connected to the respective outputs of the entity blocks that represent the classes for which the relation is defined. Figure 2 shows a schematic representation of an example SNN built from a knowledge base in the cinematic domain (specifically the Internet Movie Database, IMDB).

The input size of the SNN is the size of some vocabulary chosen before hand. This vocabulary should include the common terms in the knowledge domain(s) of interest. An additional input dimension can be added to account for unknown words (those not present in the vocabulary at the moment of training).

### 3.2 Training the Semantic Neural Network

The training procedure for a SNN is divided in two phases, each of which solves a different learning problem. In the first phase, which we call "structured pre-training", the parameters of the entity and relation blocks are adjusted. The learning objective for this phase consists of predicting to which instance of the knowledge base the sample of natural text refers. In the second phase, in order to deal with the original natural language process-

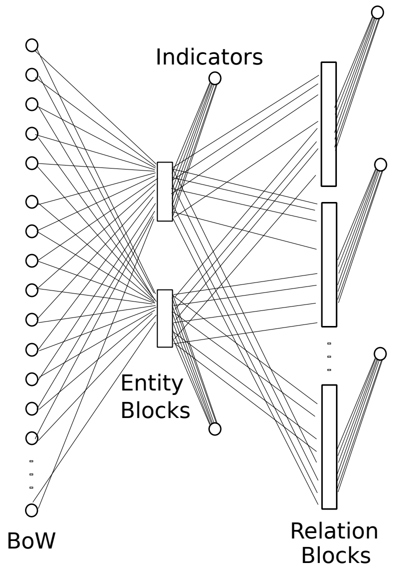

Figure 2: Schematic representation of a Semantic Neural Network, for an example knowledge domain in the film industry. The input layer consists of a bag of words (BOW) representation. The middle layer contains all the entity blocks (Movie and Person in this example) and the final layer contains all the relation blocks. The indicator outputs are one-dimensional sigmoid neurons.

ing problem, a standard training is performed. In this phase, the parameters of the rest of the network are adjusted. The internal parameters of the SNN can either be frozen, or optionally, adjusted together with the rest of the network in a post-optimization phase.

#### 3.2.1 Structured Pre-Training

During the structured pre-training step, a knowledge base is used to extract random instances and adjust the SNN parameters. This knowledge base is the same as that chosen for designing the architecture, i.e., to choose the entities and relations which are represented in the SNN blocks. A random instance of this knowledge base is a tuple $(e_i, r, e_j)$ which asserts the relation $r$ between entities $e_i$ and $e_j$. In this instance, $e_i$ and $e_j$ are assumed to have a natural text label associated (e.g., a name, description, tag, etc.).

The SNN learns to represent natural text in a manner which resembles the entity extraction process. Note that no natural text is associated to the label of the relation. Hence, in order for the SNN to accurately predict that relation $r(e_1, e_2)$ is true for a specific pair of entities $e_1$, $e_2$, but false for another pair of entities, the "list" of pairs of entities that hold in each relation must be encoded in the $W_r$ weights of the corresponding relation block. Furthermore, in order to differentiate the relations, the SNN needs to learn to differentiate the entities implicitly because there is no requisite to define the specific learning objective, namely representing different entities and their corresponding encodings.

#### 3.2.2 Unstructured Training

After the pre-training phase is completed, standard training proceeds. In this phase, given the characteristics of the learning problem, the training is performed. For example, if the problem consists of text classification (i.e., opinion mining, topic detection, etc.), then the training examples consist of pairs of single-hot encoded text and the corresponding class label. The exact parameters of the training depend on the problem characteristics, and are thus left unspecified in this proposal. During this training phase, the parameters of the SNN are frozen, i.e., they remain unchanged throughout the training procedure. The reason for this is that the parameters of the SNN have already been adjusted, and the rest of the parameters are randomly initialized. Therefore, allowing the training algorithm to change the SNN parameters would destroy the learned mappings.

After the standard training, a final phase of fine-tunning can be optionally performed. In this final phase, all the parameters, both those of the SNN and those of the rest of the network, are allowed to change. In this phase, only small parameter updates are expected to happen, since the network should have converged in the previous phase.

## 4 Experimental Analysis

To analyze the behavior of the SNN, we selected a classic NLP problem and a suitable knowledge base. The selected problem is opinion mining in movie reviews, using the dataset from Pang and Lee (2004). The corpus contains 1000 positive and 1000 negative movie reviews written in English. For building the knowledge base, raw data from IMDB[2] was processed obtaining a graph representation with 2 classes (*Person* and *Movie*), 11 relations and a total 27,044,985 tuples. Table 1 summarizes the statistics of the knowledge base.

The SNN obtained from the IMDB knowledge base has the structure shown in figure 3. The input

[2]https://datasets.imdbws.com

| **Classes** | **Instances** |
|---|---|
| *Person* | 2 854 359 |
| *Movie* | 2 361 769 |
| **Relations** | |
| *actor* | 6 531 498 |
| *actress* | 4 561 176 |
| *archive_footage* | 160 467 |
| *archive_sound* | 1 643 |
| *cinematographer* | 1 016 444 |
| *composer* | 1 030 405 |
| *director* | 2 871 640 |
| *editor* | 946 097 |
| *producer* | 1 646 903 |
| *production_designer* | 221 315 |
| *self* | 4 739 853 |
| *writer* | 3 256 270 |

Table 1: Summary statistics of the knowledge base built from IMDB data.

size is 267,178 (number of unique words in a standard English dictionary), the output size of each entity block is 10, and of each relation block is 20. The total number of indicator outputs is 14 (total number of concepts present in the IMDB knowledge base). Hence, the total number of trainable parameters in the SNN is 5,350,873.

The pre-training phase is executed for 10 epochs, each one with 100 batches, and each batch comprising 100 training examples. This cycle was repeated three times, first only with entities, then only with relations, and finally with both entities and relations. Hence, a total of 300,000 training examples were used in the pre-training phase. The final validation accuracy was 0.976 for the entities cycle, 1.00 for the relations cycle and 0.987 for the combined cycle. Since there are far fewer different relation types than actual entities, convergence is expected earlier when only training with relations.

### 4.1 Evaluating in the Opinion Mining Problem

The performance of the pre-trained SNN was evaluated by including it as an internal component in a larger neural network. This neural network was applied to the original problem of classifying movie reviews. The architecture of this extended neural network consists of a single-hot encoding input which is connected to the SNN. The output of the SNN is connected to three sequential dense

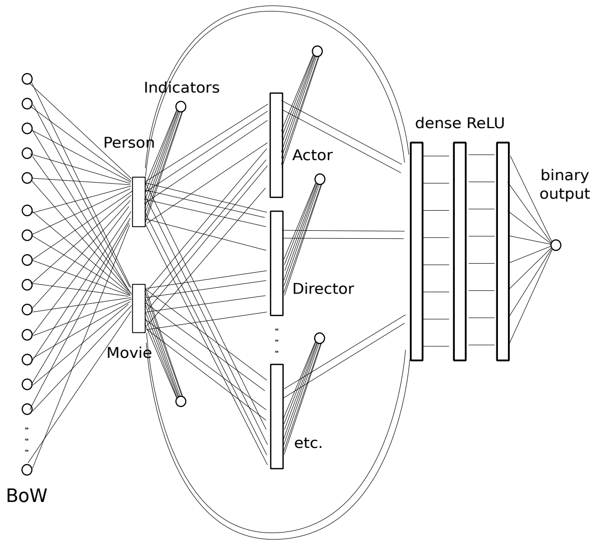

Figure 3: Schematic representation of the Semantic Neural Network trained in IMDB. Not all relation blocks are shown, given space constraints.

layers with ReLU activation. A final dense layer with sigmoid layer was used, since the movie reviews problem is a binary classification problem. This network had an additional 141,361 trainable parameters.

Training was performed only on these new parameters for 10 epochs, using 100 batches each one with 100 samples per epoch. After this process, the SNN parameters were unfrozen, and a fine-tune training was performed with all 5,491,981 parameters.

This final training was performed for 50 epochs, until convergence was achieved. The validation accuracy obtained at this point was 0.702. Finally, an independent test set was used to measure test accuracy, obtaining 67.82% precision in the movie reviews problem.

For comparison purposes, a fully-connected feed forward neural network, with 4 dense ReLU layers and a total of 5,494,681 trainable parameters was also trained from scratch in the movie review corpus. This network achieved a test accuracy of 64.47% in the same test set used for testing the SNN. Hence, with roughly the same number of trainable parameters the SNN obtains a small, but statistically significant advantage ($p \approx 10^{-39}$). Results are summarized in Table 2.

## 5 Discussion

The experimental results obtained show that using a SNN provides some benefits over a stan-

| Model | Parameters | Accuracy |
|---|---|---|
| SNN + 3 ReLU | 5,491,981 | **67.82** |
| 4 Dense ReLU | 5,494,681 | 64.47 |

Table 2: Comparison of performance in the opinion mining problem between different SNN-based architectures.

dard architecture. No comparison has been performed with more advanced architectures, such as Long Short Term Memory (LSTM) or Convolutional (CNN) networks. However, the purpose of SNNs is not to compete with, but rather to complement existing architectures with a new type of component that is knowledge-aware. Hence, the fact that the SNN performs effectively when combined with more traditional architectures encourages its use with other state-of-the-art deep learning components.

The most significant advantage of using a SNN is that it provides a semantic interpretation of the network's behavior. The one-dimensional indicators used for guiding the SNN training could potentially act as a signaller for the high-level concepts that are being activated in any given example. By looking at the activation patterns inside the SNN, it may be possible to obtain an explanation of the network's prediction for a given example, in terms of the concepts in the knowledge base learned. Figure 4 presents an illustrative example in the context of the NLP problem presented in Section 4.

Another advantage of SNNs is that once pretrained with a given knowledge base, they can be reused in many different architectures. In a sense, SNNs can be seen as feature extractors or representational components that are associated with a given knowledge domain. It is also conceivable to use multiple SNNs trained in different knowledge domains in the same neural network. This provides a strategy for knowledge integration, when different and possible incompatible knowledge domains are considered relevant for a particular problem. This opens the door to new strategies for transfer learning, from a more semantic perspective, where the transferred or reused knowledge can be tied to a particular domain.

In our experimental setup we used a small natural language dataset, since our purpose was not to obtain state-of-the-art results in the opinion mining problem, but rather to illustrate the design of the SNN and explain its main architectural characteristics. We demonstrated that the SNN model provides effective support to learning approaches for resolving NLP problems.

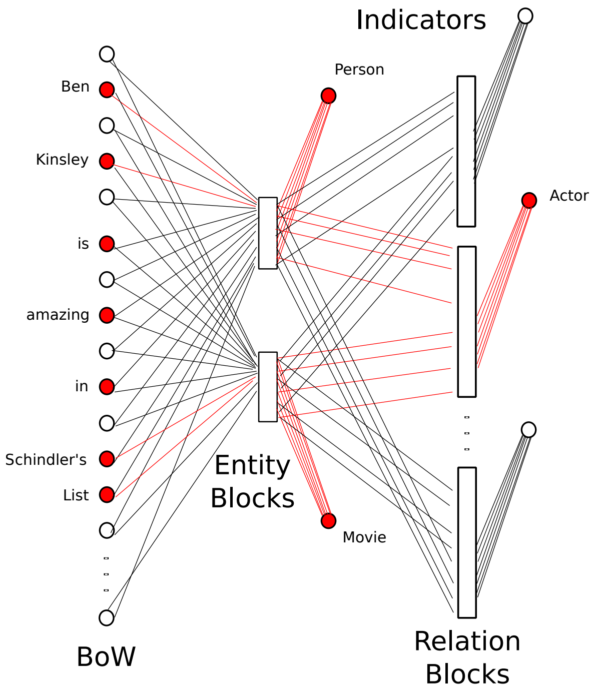

Figure 4: Illustrative example of a Semantic Neural Network activation in response to a particular input.

## 6 Conclusions and Future Work

In this paper, we present Semantic Neural Networks (SNNs), which allow the inclusion of explicit semantic knowledge in traditional neural networks without affecting performance. This knowledge is extracted from available knowledge bases, built by domain experts. We considered that the SNN is a first step towards raising the abstraction level in neural networks and building semantically-aware architectures that can be self-explained. A possible line of future research consists of combining multiple SNNs for multiple knowledge domains in a single problem, and studying how the different representations output by each network can be merged. Further, in the current design, the core of the SNN is a simple linear operation. In future works, we will explore other, more complex operations, which will potentially improve accuracy. It is also necessary to compare the performance of SNN-powered architectures with other state-of-the-art deep learning architectures, such as LSTMs, CNNs and different embedding strategies. Finally, the SNN internal structure has a semantic meaning attached to each neuron block. This opens the door for the design of interpretation models that can automatically output an explanation of a neural networks response in terms of human-defined concepts.

## Acknowledgments

This research has been supported by a Carolina Foundation grant in agreement with University of Alicante and University of Havana. Moreover, it has also been partially funded by both aforementioned universities, the Generalitat Valenciana and the Spanish Government through the projects SIIA (PROMETEU/2018/089), LIVING-LANG (RTI2018-094653-B-C22) and INTEGER (RTI2018-094649-B-I00).

## References

Yoshua Bengio. 2012. Deep Learning of Representations for Unsupervised and Transfer Learning. In *JMLR: Workshop and Conference Proceedings*. volume 27, page 17–37.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *IJCAI International Joint Conference on Artificial Intelligence*. https://doi.org/10.1145/2063576.2063865.

Graeme Hirst and David St-Onge. 1998. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database* 305(April):305–332. https://doi.org/citeulike-article-id:4893262.

Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric P Xing. 2015. Entity hierarchy embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Stroudsburg, PA, USA, volume 1, pages 1292–1300. https://doi.org/10.3115/v1/P15-1125.

Quoc V Le. 2013. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pages 8595–8598.

Claudia Leacock and Martin Chodorow. 1998. Combining Local Context and WordNet Similarity for Word Sense Identification. *In: WordNet: An electronic lexical database.* (January 1998):265–283. https://doi.org/citeulike-article-id:1259480.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, and Computers* https://doi.org/10.3354/ame01683.

Per Gunnar Martinsson, Vladimir Rokhlin, and Mark Tygert. 2011. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis* https://doi.org/10.1016/j.acha.2010.02.003.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)* pages 1–12. https://doi.org/10.1162/153244303322533223.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2017. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* .

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*.

Philip Resnik. 1999. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research* 11:95–130. https://doi.org/10.1613/jair.514.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. pages 298–307.

Peter D Turney. 2001. Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. *Lecture Notes in Computer Science* https://doi.org/10.1007/3-540-44795-4_42.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation http://arxiv.org/abs/1601.01343.