Inducing Symbolic Rules from Entity Embeddings using Auto-encoders

Thomas Ager, Ondřej Kuželka, Steven Schockaert

School of Computer Science and Informatics, Cardiff University
{AgerT,KuzelkaO,SchockaertS1}@cardiff.ac.uk

Abstract. Vector space embeddings can be used as a tool for learning semantic relationships from unstructured text documents. Among others, earlier work has shown how in a vector space of entities (e.g. different movies) fine-grained semantic relationships can be identified with directions (e.g. more violent than). In this paper, we use stacked denoising auto-encoders to obtain a sequence of entity embeddings that model increasingly abstract relationships. After identifying directions that model salient properties of entities in each of these vector spaces, we induce symbolic rules that relate specific properties to more general ones. We provide illustrative examples to demonstrate the potential of this approach.

1 Introduction

In this paper, we consider the problem of how we can learn symbolic rules from unstructured text documents that describe entities of interest, e.g. how we can learn that thrillers tend to be violent from a collection of movie reviews. Obtaining meaningful and interpretable symbolic rules is important in fields like exploratory data analysis, or explaining classifier decisions, as they can be interpreted easily by human users.

A straightforward approach might be to directly learn rules from bag-of-words representations of documents. However, such an approach would typically lead to a large number of rules of little interest, e.g. rules pertaining more to which words are used together rather than capturing capturing meaningful semantic relationships. Our approach instead builds on the method from [6], which induces an entity embedding from unstructured text documents. Their method finds directions which correspond to interpretable properties in a vector space, labelled using adjectives and nouns that appear in the text collection. In particular, these directions induce a ranking of the entities that reflects how much they have the corresponding property. For example, in a space of wines, a direction may be found that corresponds to the property of being "Tannic", allowing us to rank wines based on the number of tannins.

In order to obtain symbolic rules, we first derive a series of increasingly general entity embeddings using auto-encoders (see Section 3). To induce rules from embeddings, we link properties derived from those embeddings together.

As an example, below is one of the rules we have derived using this method:

$$\texttt{IF Emotions AND Journey THEN Adventure} \tag{1}$$

Using a set of symbolic rules that qualitatively describe domain knowledge is a promising approach to generate supporting explanations. Explanations of classification decisions can give valuable insight into why a system produces a result. For example, in fields such as medicine it is important for experts to verify the predictions of a system and justify its classification decisions [7, 9]. In the domain of movies, we may have a situation where the synopsis or reviews mention the words "Emotions" and "Journey", from which the system could derive that it is probably an "Adventure" movie and use rule (1) as a supporting explanation. We note that the ideas presented in this paper may also be directly useful for explaining predictions of some kinds of deep neural networks.

The rest of the paper explains how we use unsupervised methods to learn rules such as (1). In Section 2, we recall the method from [6] for identifying interpretable directions in entity embeddings. Subsequently in Section 3 we detail how we build on this method using stacked denoising auto-encoders, and how we induce rules that explain the semantic relationships between the properties that we discover. In Section 4 we qualitatively examine these properties and rules, and in Section 5 we place our work in the context of related work. Finally, in Section 6 we provide our conclusions.

2 Learning Interpretable Directions

In this section, we recall the method from [6] that learns a vector space representation for the entities of a given domain of interest, such that salient properties of the domain correspond to directions in the vector space. The method proceeds in several steps, detailed next.

From bags-of-words to vectors. We use a text collection where each document describes an entity. For example, if the entities are movies, a collection of movie reviews. We first learn a vector space of entities using classical multidimensional scaling (MDS), which takes a dissimilarity matrix as input. MDS is commonly used in cognitive science to generate semantic spaces from similarity judgements that are provided by human annotators. It outputs a space where entities are represented as points and the Euclidean distance between entities reflects the given dissimilarity matrix as closely as possible. It was empirically found to lead to representations that are easier to interpret than the more commonly used singular value decomposition method [5]. To obtain a suitable dissimilarity matrix, we quantify how relevant each term is to an entity using Positive Pointwise Mutual Information (PPMI). PPMI scores terms highly if they are frequently associated with an entity but relatively infrequent over the entire text collection. We create PPMI vectors for each entity using the PPMI values for each word as the components of its vector, and calculate the dissimilarity between those vectors using the normalized angular difference. These dissimilarity values are then used as the input to MDS.

Identifying directions for frequent terms. To discover terms that correspond to interpretable properties in the MDS space, the nouns and adjectives that occur in sufficiently many reviews are used as the input to a linear Support Vector Machine (SVM). The SVM is trained to find the hyperplane that best separates the entities that contain the term at least once in their associated textual description. To accommodate class imbalance, we increase the cost of positive instances such that their weight is inversely proportional to how many times the term has occurred. To assess the quality of the hyperplane found by the SVM, we use Cohen's Kappa score [4] which evaluates how well the hyperplane separates positive/negative instances while taking class imbalance into account. We consider terms with a high Kappa score to be labels of properties that are modelled well by the MDS space. The direction corresponding to a given term/property is given by the vector perpendicular to the associated hyperplane. This vector in turn allows us to determine a ranking of the entities, according to how much they have the property being modelled. This ranking is obtained by determining the orthogonal projection of each entity on an oriented line with that direction. It is easy to see that if v is the vector modelling a given property, then entity e_1 is ranked before entity e_2 iff $e_1 \cdot v < e_2 \cdot v$. Another way to look at this is that entities are ranked according to their signed distance to the hyperplane.

Identifying saleint properties by clustering directions. It can sometimes be ambiguous as to what property each term is referring to. For example, it is unclear whether "mammoth" refers to the animal or an adjective meaning large. In this paper, we have chosen the number of clusters equal to the number of dimensions. To determine the cluster centers, we first select directions whose associated Kappa score is above some threshold T^+. We use the highest scoring direction as the center of the first cluster and find the most dissimilar direction to the first cluster's direction to get the centre of the second cluster. Continuing in this way, we repeatedly select the direction which is most dissimilar to all previously selected clusters. By doing so, we obtain a collection of cluster centres that capture a wide variety of different properties from the space. We then associate each remaining direction to its most similar cluster centre. In this step, we consider directions whose associated Kappa score is at least T^-, where typically $T^- < T^+$. Finally, we take the average of all directions in a cluster to be the overall direction for a cluster. The value of T^+ should be chosen as large as possible (given that the terms with the highest Kappa scores are those which are best represented in the space), while still ensuring that we can avoid choosing cluster centers which are too similar. Choosing the value of T^- represents a trade-off. A cluster of terms is often easier to interpret than a single term, which means that we shouldn't choose T^- to be too high. On the other hand, choosing T^- to be too low would result in poorly modelled terms being added to clusters. For example, we would not want to term "Bee" to be added to the cluster for "Emotional", even though the direction for "Bee" is closest to that cluster.

Note that as each cluster produced by the above procedure is associated with a direction, it induces a ranking of the entities. This gives us two ways to disambiguate which properties a term is referring to: the first being examining which terms it shares its cluster with e.g. we know that “Mammoth” refers to the adjective because it is shared with “Epic”, “Stupendous”, and “Majestic”, and the second being examining which entities score highly in the rankings for a cluster direction e.g. “Monster” defines a ranking in which “Frankenstein” and “The Wolfman” appear among the top ranked movies.

3 Inducing Rules from Entity Embeddings

In this section, we explain how we obtain a series of increasingly general entity embeddings, and how we can learn symbolic rules that link properties from subsequent spaces together.

To construct more general embeddings from the initial embedding provided by the MDS method, we use stacked denoising auto-encoders [16]. Standard auto-encoders are composed of an “encoder” that maps the input representation into a hidden layer, and a “decoder” that aims to recreate the input from the hidden layer. Auto-encoders are normally trained using an objective function that minimizes information loss (e.g. Mean Squared Error) between the input and output layer [2]. The task of recreating the input is made non-trivial by constraining the size of the hidden layer to be smaller than the input layer, forcing the information to be represented using fewer dimensions, or in denoising auto-encoders by corrupting the input with random noise, forcing the auto-encoder to use more general commonalities between the input features. By repeatedly using the hidden layer as input to another auto-encoder, we can obtain increasingly general representations. To obtain the entity representations from our auto-encoders, we use the activations of the neurons in a hidden layer as the coordinates of entities in a new vector space.

The main novelty of our approach is that we characterize the salient properties (i.e. clusters of directions) modelled in one space in terms of salient properties that are modelled in another space. Specifically, we use the off-the-shelf rule learner JRip [7] to predict which entities will be highly ranked, according to a given cluster direction, using as features the rankings induced by the clusters of the preceding space. To improve the readability of the resulting rules, rather than using the precise ranks as input, we aggregate the ranks by percentile, i.e $1\%, 2\%, ..., 100\%$, where an entity has a 1% label if it is among the 1% highest ranked entities, for a given cluster direction. For the class labels, we define a movie as a positive instance if it is among the highest ranked entities (e.g. top 2%) of the considered cluster direction. Using the input features of each layer and the class labels from the subsequent layer, these rules can be used to explain the semantic relationships between properties modelled by different vector spaces. We note that one drawback of discretizing continuous attributes is that the accuracy of the rules extracted from the network may decrease [14]. However, in our setting, interpretability is more important than accuracy, as we do not aim to use these rules for making predictions, but use them only for generating explanations and getting insight into data.

4 Qualitative Evaluation

We base our experiments on the movie review text collection of the 15,000 top scoring movies on IMDB[1] made available by [6]. To collect the terms that are likely to correspond to property names, we collect adjectives and nouns that occur at least 200 times in the movie review data set, collecting 17,840 terms overall. We share terms used for the property names across all spaces.

4.1 Software, Architecture and Settings

To implement the denoising auto-encoders, we use the Keras [3] library. For our SVM implementation, we use scikit-learn [11]. We have made all of the code and data freely available on GitHub[2]. We use a 200 dimensional MDS space from [6] as the input to our stack of auto-encoders. The network is trained using stochastic gradient descent and the mean squared error loss function. For the encoders and decoders, we use the tanh activation function. For the first auto-encoder, we maintain the same size layer as the input. Afterwards, we halve the hidden representation size each time it is used as input to another auto-encoder, and repeat this process three times, giving us four new hidden representations $\{Input : 200, Hidden : 200, 100, 50, 25\}$. We corrupt the input space each time using Gaussian noise with a standard deviation of 0.6. As the lower layers are closer to the bag-of-words representation and are higher dimensional, the Kappa scores are higher in earlier spaces, as it is easier to separate entities. We address this in the clusters by setting the high Kappa score threshold T^+ such that the number of terms we choose from is twice the number of dimensions in the space. Similarly, we set T^- such that 12,000 directions are available to assign to the cluster centres in every space.

4.2 Qualitative Evaluation of Induced Clusters

In Table 1, we illustrate the differences between clusters obtained using standard auto-encoders and denoising auto-encoders. Layer 1 refers to the hidden representation of the first auto-encoder, and Layer 4 refers to the hidden representation of the final auto-encoder. As single labels can lead to ambiguity, in Table 1 we label clusters using the top three highest scoring terms in the cluster. Clusters are arranged from highest to lowest Kappa score.

Both auto-encoders model increasingly general properties, but the properties obtained when using denoising auto-encoder properties are more general. For example, the normal auto-encoder contains properties like “Horror” and “Thriller”, but does not contain more general properties like “Society” and “Relationship”. Further, “Gore” has the most similar properties “Zombie” and “Zombies” in Layer 1, and has the most similar properties of “Budget” and “Effects” in Layer 4. By representing a category of movie where “Budget” and “Effects” are important, the property is more general.

[1] http://www.cs.cf.ac.uk/semanticspaces/

[2] https://github.com/eygrr/RulesFromAuto-encoders

Table 1. A comparison between the first layers and the fourth layers of two different kinds of auto-encoders.

Standard Auto-encoder		**Denoising Auto-encoder**	
Layer 1	**Layer 4**	**Layer 1**	**Layer 4**
horror: terror, horrific	horror: victims, nudity	gore: zombie, zombies	society: view, understand
thriller: thrillers, noir	documentary: perspective, insight	jokes: chuckle, fart	emotional: insight, portrays
comedies: comedy, timing	blood: killing, effects	horror: terror, horrific	stupid: flick, silly
adults: disney, childrens	suspense: mysterious, tense	emotionally: tragic, strength	gore: budget, effects
husband: wife, husbands	thriller: thrillers, cop	gags: zany, parodies	military: war, ship
relationships: intimate, angst	gory: gruesome, zombie	hindi: bollywood, indian	romance: younger, handsome
nudity: naked, gratuitous	beautifully: satisfying, brilliantly	touching: teach, relate	ridiculous: awful, worse
political: politics, nation	emotional: complex, struggle	scary: frightening, terrifying	government: technology, footage
smart: slick, sophisticated	laughed: laughing, loud	documentary: document, narration	awesome: chick, looked
creepy: sinister, atmospheric	charming: delightful, loves	adults: disney, teaches	political: country, documentary
laughed: humorous, offensive	hilarious: funny, parody	laughed: brow, laughter	relationship: relationships, sensitive
adventure: adventures, ship	scares: halloween, slasher	thriller: thrillers, procedural	horror: genre, dark
actions: reaction, innocent	funniest: funnier, gags	cgi: animated, animation	waste: concept, plain
cute: adorable, rom	emotions: respect, relationships	suspense: clues, atmospheric	army: disc, studio
british: england, accent	laugh: mom, crazy	dumb: mindless, car	combat: enemy, weapons
horrible: worse, cheap	filmmaker: approach, artist	political: propaganda, citizens	supporting: office, married
narrative: filmmaker, structure	drama: portrayed, portrayal	witty: delightfully, sarcastic	amazon: bought, copy
digital: dolby, definition	interviews: included, showed	laughing: outrageous, mouthed	study: details, detail
gory: graphic, gruesome	comedic: comedies, humorous	relationships: ensemble, interactions	land: water, super
romantic: handsome, attractive	emotionally: central, relationships	creepy: mysterious, eerie	chemistry, comedies, comedic

4.3 Qualitative Evaluation of Induced Symbolic Rules

Our aim in this work is to derive symbolic rules that can be used to explain the semantic relationships between properties derived from increasingly general entity embeddings. We provide examples of such rules in this section. Since the number of all induced rules is large, here we only show high accuracy rules that cover 200 samples or more. Still, we naturally cannot list even all the accurate rules covering more than 200 samples. Therefore we focus here on the rules which are either interesting in their own right or exhibit interesting properties, strengths or limitations of the proposed approach. The complete list of induced rules is available online from our GitHub repository[3].

For easier readability, we post-process the induced rules. For instance, the following is a rule obtained for the property "Gore" in the third layer of the network shown in the original format produced by JRip:

```
IF scares-L2 <= 6 AND blood-L2 <= 8 AND funniest-L2 >= 22
 => classification=+ (391.0/61.0)
```

In this rule, `scares-L2 <= 6` denotes the condition that the movie is in the top 6% of rankings for the property "`scares`" derived from the hidden representation of the second auto-encoder. We will write such conditions simply as "Scares$_2$". Similarly, a condition such as `funniest-L2 >= 22`, which indicates that the property is not in the top 22%, will be written as NOT Funniest$_2$. In this simpler notation the above rule will look as follows:

IF Scares$_2$ AND Blood$_2$ AND NOT Funniest$_2$ THEN Gore$_3$

This rule demonstrates an interpretable relationship. However, we have observed that the meaning of a rule may not be clear from the property labels that are automatically selected. In such cases, it is beneficial to label them by including the most similar cluster terms. For example, using the cluster terms below we can see that "Flick" relates to "chick-flicks" and that "Amazon" relates to old movies:

IF Flick$_2$ AND Sexual$_2$ AND Cheesy$_2$ AND NOT Amazon$_2$ THEN Nudity$_3$

Flick$_2$: {Flicks, Chick, Hot}
Amazon$_2$: {Vhs, Copy, Ago}

Rules derived from later layers use properties described by rules from previous layers. By seeing rules from earlier layers that contain properties in later layers, we can better understand what the components of later rules mean. Below, we have provided rules to explain the origins of components in a later rule:

IF Emotions$_2$ AND Actions$_2$ THEN Emotions$_3$
IF Emotions$_2$ AND Emotion$_2$ AND Impact$_2$ THEN Journey$_3$
IF Emotions$_3$ AND Journey$_3$ THEN Adventure$_4$

[3] https://github.com/eygrr/RulesFromAuto-encoders

We observe a general trend that as the size of the representations decreases and the entity embeddings become smaller, rules have fewer conditions, resulting in overall higher scoring and more interpretable rules. To illustrate this, we compare rules from an earlier layer to similar rules in a later layer:

```
IF Romance$_1$ AND Poignant$_1$ AND NOT English$_1$ AND NOT French$_1$
 AND NOT Gags$_1$  AND NOT Disc$_1$ THEN Relationships$_2$

IF Relationships$_2$ AND Emotions$_2$ AND Chemistry$_2$ THEN Romantic$_3$
IF Emotions$_2$ AND Compelling$_2$ THEN Beautifully$_3$
IF Warm$_2$ AND Emotions$_2$ THEN Charming$_3$
IF Emotions$_2$ AND Compelling$_2$ THEN Emotional$_3$
```

Rules in later layers also made effective use of a NOT component. Below, we demonstrate some of those rules:

```
IF Touching$_3$ AND Emotions$_3$ AND NOT Unfunny$_3$ THEN  Relationship$_4$
IF Laughs$_3$ AND Laugh$_3$ AND NOT Compelling$_3$ THEN  Stupid$_4$
IF Touching$_3$ AND Social$_3$ AND NOT Slasher$_3$ THEN  Touching$_4$
```

As the same terms were used to find new properties for each space, the obtained rules sometimes use duplicate property names in their components. As the properties from later layers are a combination of properties from earlier layers, the properties in later layers are refinements of the earlier properties, despite having the same term. Below, we provide some examples to illustrate this:

```
IF Emotions$_2$ AND Actions$_2$ THEN Emotions$_3$

Emotions$_2$: {Acted, Feelings, Mature}
Actions$_2$: {Control, Crime, Force}
Emotions$_3$: {Emotion, Issue, Choices}

IF Horror$_2$ AND Creepy$_2$ AND Scares$_2$ THEN Horror$_3$

Horror$_2$: {Terror, Horrific, Exploitation}
Creepy$_2$: {Mysterious, Twisted, Psycho}
Scares$_2$: {Slasher, Supernatural, Halloween}
Horror$_3$: {Creepy, Dark, Chilling}

IF Touching$_2$ AND Chemistry$_2$ THEN Touching$_3$
IF Touching$_2$ AND Emotions$_2$ THEN Touching$_3$
IF Compelling$_2$ AND Emotional$_2$ AND Suspense$_2$ THEN Compelling$_3$
If Romance$_2$ AND Touching$_2$ AND Chemistry$_2$ THEN Romance $_3$
IF Emotionally$_2$ AND Emotions$_2$ AND Compelling$_2$ THEN Emotionally$_3$
```

5 Related Work

The work presented in this paper differs from existing works in that it focuses on inducing rules which involve salient and interpretable features from unstructured text documents.

The existing neural network rule extraction algorithms can be categorized as either decompositional, pedagogical or eclectic [1]. Decompositional approaches derive rules by analysing the units of the network, while pedagogical approaches treat the network as a black box, and examine the global relationships between inputs and outputs. Eclectic approaches use elements of both decompositional and pedagogical approaches. Our method could be classified as decompositional, as we make use of the hidden layer of an auto-encoder. We will now describe some similar approaches and explain how our methods differs.

The algorithm in [10] is a decompositional approach that applies to a neural network with two hidden layers. It uses hyperplanes based on the weight parameters of the first layer, and then combines them into a decision tree. NeuroLinear [15] is a decompositional approach applied to a neural network with a single hidden layer that discretizes hidden unit activation values and uses a hyperplane rule to represent the relationship between the discretized values and the first layer's weights. HYPINV [13] is a pedagogical approach that calculates changes to the input of the network to find hyperplane rules that explain how the network functions.

The main difference in our work is that our method induces rules from properties derived from the layers of a network, rather than learning rules that describe the relationships between units in the network itself. Additionally, we focus on learning increasingly general entity embeddings from hidden representations rather than tuning network parameters such that weights directly relate to good rules.

Another recent topic that relates to our work is improving neural networks and entity embeddings using symbolic rules [8]. In [12] a combination of first-order logic formulae and matrix factorization is used to capture semantic relationships between concepts that were not in the original text. This results in relations that are able to generalize well from input data.

This is essentially the opposite of the task we consider in this paper: using embeddings to learn better rules. The rules that we derive are not intended to explain how the network functions but rather to describe the semantic relationships that hold in the considered domain. In other words, our aim is to use the neural network representations in the hidden layer as a tool for learning logical domain theories, where the focus is on producing rules that capture meaningful semantic relationships.

6 Conclusions

In this paper, we have shown how we can obtain increasingly general entity embeddings from stacked denoising auto-encoders, and how we can obtain rules from those embeddings that capture domain knowledge. We have qualitatively evaluated the obtained rules to demonstrate the semantic relationships that they capture. The results show the potential of the method for exploratory analysis of collections of unstructured text documents and explaining decisions of classifiers.

Acknowledgement. This work was supported by ERC Starting Grant 637277.

References

1. R. Andrews, J. Diederich, and A. B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, 1995.
2. Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends® in Machine Learning*, 2(1):1–127, 2009.
3. F. Chollet. Keras. https://github.com/fchollet/keras, 2015.
4. J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37, 1960.
5. J. Derrac and S. Schockaert. Enriching taxonomies of place types using Flickr. *Lecture Notes in Computer Science*, 8367:174–192, 2014.
6. J. Derrac and S. Schockaert. Inducing semantic relations from conceptual spaces: A data-driven approach to plausible reasoning. *Artificial Intelligence*, 228:66–94, 2015.
7. J. L. Herlocker, J. a. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. *Proceedings of the ACM conference on Computer supported cooperative work*, pages 241–250, 2000.
8. Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. Harnessing Deep Neural Networks with Logic Rules. *arXiv preprint*, pages 1–18, 2016.
9. W. B. Kheder, D. Matrouf, P.-M. Bousquet, J.-F. Bonastre, and M. Ajili. Statistical Language and Speech Processing. *Statistical Language and Speech Processing*, 8791:97–107, 2014.
10. D. Kim and J. Lee. Handling continuous-valued attributes in decision tree with neural network modeling. 1810:211–219, 2000.
11. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
12. T. Rocktäschel, S. Singh, and S. Riedel. Injecting logical background knowledge into embeddings for relation extraction. *Proceedings of the 2015 Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, 2015.
13. E. W. Saad and D. C. Wunsch. Neural network explanation using inversion. *Neural Networks*, 20(1):78–93, 2007.
14. R. Setiono, B. Baesens, and C. Mues. Recursive neural network rule extraction for data with mixed attributes. *IEEE Transactions on Neural Networks*, 19(2):299–307, 2008.
15. R. Setiono and H. Liu. Neurolinear: From neural networks to oblique decision rules. *Neurocomputing*, 17(1):1–24, 1997.

16. P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.