HLE Final Project: Detecting Discriminative Attributes

Human Language Engineering
Master in Artificial Intelligence
January 2023

Content

Abstract

In this report we present our work on SemEval 2018 Task 10 ¨Capturing Discriminative Attributes¨, a task which aims to detect semantic differences between concept pairs. After doing an analysis of the top performing teams' submissions we have implemented the best performing methods and experimented with combining them in several manners and training them with different classifiers. The features that we have extracted are word embeddings, distance and similarity measures computed from those embeddings and semantic knowledge obtained from ConceptNet knowledge graph. Finally, we show the results that we obtain and we declare them as not consistent with our initial expectations and also with other participants' results.

1. Introduction

Capturing discriminative attributes describes the process of identifying whether an attribute is able to help differentiate two concepts from each other. The problem was presented as a SemEval Task (#10) in 2018, allowing participants to discover complementary features of semantic models that extend beyond classical semantic similarity. Capturing both semantic similarities and differences is important, and as the authors of the challenge note, ”no model can claim to capture semantic competence if it does not, in addition to similarity, predict semantic differences between words” [1]. Further, they noted that there are several problems in similarity modelling, including low inter-annotator agreement, small dataset sizes, and differences in similarity due to linguistic context. Capturing the differences between entities can be applicable, for example, in enhancing conversational agents by choosing lexical items with contextually relevant differential features to create human-like dialogs or in machine translation, where explicitly taking into account semantic differences between translation variants can improve the quality of the output.

With this in mind, the authors presented the SemEval Challenge above. The primary goal was to design a system that assigns a binary label to a word-word-attribute triple, describing whether that attribute is a discriminatory feature among the two or not. More about the data and employed methodology can be found in the following sections.

2. Methodology

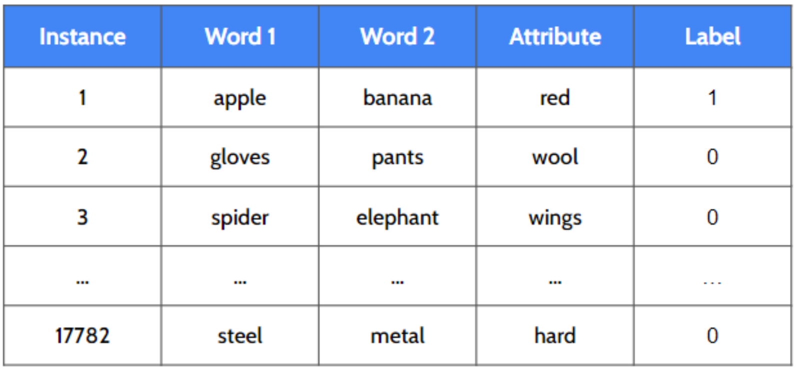
2.1. Overview

Around 40 teams participated in this SemEval task, and the methodologies they used, including selection of pre-processing techniques, feature extraction and classifier, varied greatly among them. For computing the features, participants used a large number of resources. Such resources can be divided into word embeddings (e.g., Word2Vec, GloVe, fastText) and knowledge base type resources (e.g., Word-Net, ConceptNet, Probase). It is worth mentioning that none of the participants used BERT like approaches to compute word embeddings, this will be issued in the following sections. As for the classifier, some of the most successful systems employed traditional machine learning algorithms such as SVMs [2], SVC [3] and Maximum Entropy Classifiers [4]. Other teams, on the other hand, decided to use deep learning systems such as neural networks [5] and XGB classifiers [6].

This task has been evaluated using the F1 score, a metric which computes the harmonic mean between precision and recall. The team which achieved the highest score was SUNNYNLP [2] with an F1 score of 0.75, using GloVe pre-trained word embeddings and entity-entity relation information from Probase knowledge base. Second and third teams achieved 0.74 and 0.73 F1 scores using a similar approach, but using a different knowledge base (ConceptNet) and a different classifier (XGBoost).

2.2. Dataset

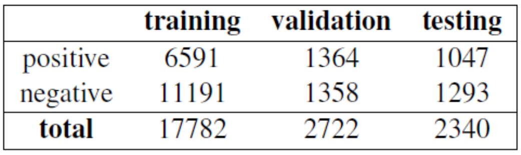
One of the integral parts of this challenge was the data: the words and attributes to be used to discriminate were not found in sentences or documents from a corpus. The data is found in a tabular structure, which has four columns: two words, an attribute, and a binary (0-1) label indicating whether this attribute is able to discriminate between these two words. In order for an attribute to be discriminative, the attribute must only be shared by one of the words. On the other hand, to not be discriminative, two cases can happen: that the attribute is shared by both of the words or is not shared by neither of them. In the table below there is one example exemplifying each case respectively (1st, 2nd and 3rd rows).



Instance	Word 1	Word 2	Attribute	Label
1	apple	banana	red	1
2	gloves	pants	wool	0
3	spider	elephant	wings	0
…	…	…	…	…
17782	steel	metal	hard	0

Figure 1: Visualisation of a subset of the dataset and its structure

All of the data used in this challenge (including the train-validation-test splits) was supplied by the challenge organisers. Below is a breakdown of the supplied datasets. Note that here the positive case describes an attribute that is able to discriminate between the two words (label=1), and the negative case means the opposite (label=0).



	training	**validation**	**testing**
positive	6591	1364	1047
negative	11191	1358	1293
total	17782	2722	2340

Table 1: dataset partition in train, validation and testing and overview of positive and negative example distribution

As mentioned in the introduction, dataset sizes are often a challenge in semantic modelling. To combat this, the challenge authors constructed the data from various different sources, including both manually verified cases as well as automatically generated data. They used three discrete methods for generating data; a) manual filtered triples generated from McRae Norms [7]; b) manually annotated features for word selected from a SimLex-999 wordbase (add source); and c) random matching of words and features (later annotated by the authors). A closer description of each data source can be read in the SemEval 2018 paper Section 2.2.

Due to vast differences in the methods participants chose to tackle this challenge, our choice was to perform an ablation study of various system configurations. This meant taking different implementations from challenge participants, mixing them up, and comparing how they performed in comparison to each other. During the study, we evaluated both parts of the system, the input features and the classifier. Altogether, 8 feature combinations and 3 different classifiers were tested, resulting in the evaluation of 24 different systems. They are described in more detail below. The code from this project can be consulted here[1].

2.3. Features

Regarding feature extraction, we divide the methods into two groups: word embeddings and knowledge base. We have selected them because top performing teams followed a similar approach and we wanted to experiment whether those methods, an alterations of them, yielded similar or better results. Figure below shows a summary of the pipeline of our model and its variations.

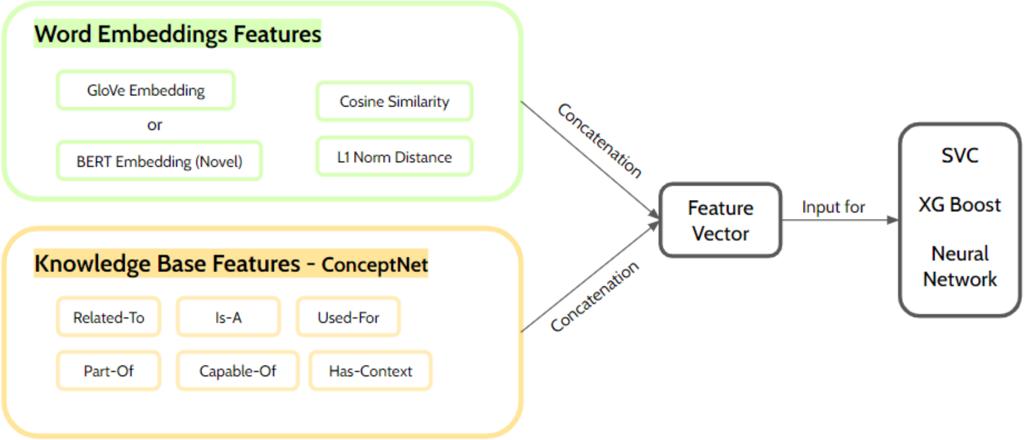


Figure2 : Pipeline of the model: feature extraction and classification

[1] https://github.com/luciaurcelay/Capturing-Discriminative-Attributes

2.3.1 Embeddings

One of the integral features for training the model was the conversion of the words to n-dimensional word embeddings. Initially, we computed the pre-trained word embeddings using GloVe embeddings, which have been greatly used since they were introduced in 2014. In the experiments, we tested the impact of using different dimensions: 50, 100, 200, 300, hoping that higher dimensional vectors would yield better results as they encode more information about the words. These were pre-trained, meaning that they could be retrieved with a simple word-embedding mapping. Not all of the different dimensions were used in the ablation study - instead, preliminary testing was done with between the four dimension embeddings on a vanilla XGBoost classifier to determine which were the best performing embeddings for our purpose. Empirical testing found this to be the maximum-dimension (300d) GloVe embeddings, and these features were used as the de-facto GloVe embeddings during the ablation study.

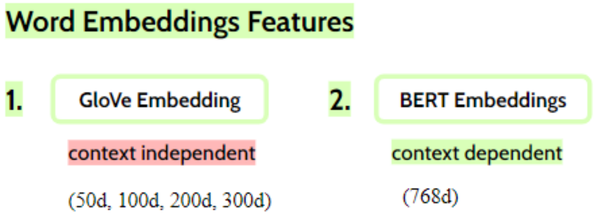


Figure 3:Word embeddings

The second, more empirical option was using embeddings from a pre-trained BERT model[2]. Although it was known beforehand that the purpose of BERT embeddings are to provide the data with contextuality, and that it might not be the appropriate model for passing through a sequence of context independent words, it was still felt that using BERT might be useful for empirical testing. Further, BERT embeddings were not mentioned by any of the top of the performing participants (understandable since this challenge was published in 2018), and this could provide us with an opportunity to test something novel. By introducing contextual embeddings, we hoped that it would help out with the disambiguation of words based on the other two from each instance. Later, we realised that computing BERT embeddings using just the 3 words from each instance is not a good approach. More on this in the discussion section.

2.3.2 Vector-space distances

Another important feature is using finding distances between words in their respective embedding vector spaces. The intuition behind this is that inherently different concepts should also have relatively different embeddings, and hence these differences should also be projected

[2] https://huggingface.co/bert-base-uncased

into the vector space. These differences could then be used to differentiate between words and their attributes.

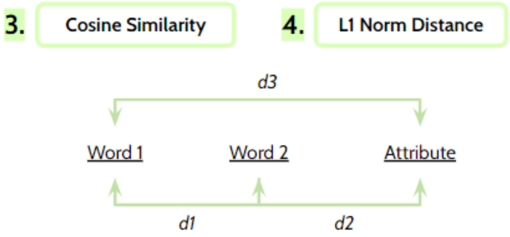


Figure 4: Similarity and distance metrics

The two different distance measures used were the L1 norm (Manhattan) distance and cosine similarity. L1 is a more simple distance measure, simply being a sum of absolute element differences, while cosine similarity takes into account the embedding vectors’ orientation in the vector space. In this work we compute three different distances: word1-word2, word1-attribute and word2-attribute.

2.3.3 Knowledge bases

The second kind of features is based on ontologies. There are several knowledge bases which encode the information of relationships between entities. For this work we used ConceptNet, a large semantic network that represents common sense knowledge about the relationships between words and concepts in natural language. One way to use ConceptNet to extract relationships between entities is to query the graph for edges that connect the entities. In this work, we defined a list of relationship types between entities and extracted features which indicate whether those relationships are present in word-word and word-attribute combinations.

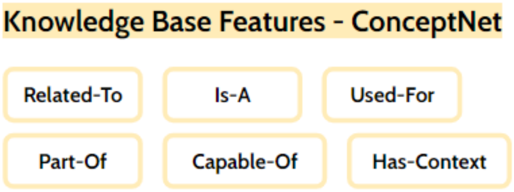


Figure 5: Relations extracted from ConceptNet knowledge graph

Figure above shows the list of the 6 different types of relationships analysed. To exemplify how ConceptNet works let's put the following example: to find relationships between the concepts "dog" and "animal", you could query the graph for edges that connect the node representing "dog" to the node representing "animal". The results of the query could include edges labelled "IsA" that indicate that a dog is a type of animal, as well as edges labelled "HasProperty" that indicate that dogs have certain properties that are associated with animals.

2.4. Classifiers

Three different classifiers were implemented and compared, all of them varying in their architectural structure. These were the XGBoost gradient boosting tree, a linear support vector classifier, and shallow multi-layer perceptron.

2.4.1 XGBoost

The first classifier implementation was with the XGBoost algorithm, which uses gradient boosted trees for predicting discriminative labels. It was used in X of the top 5 challenge submissions, and for this report it was seen as a suitable implementation as well. However, the hyperparameters were fine-tuned manually with a window search. These parameters were learning rate, maximum tree depth, and column subsample size (fraction of data columns taken for training of each individual tree). Each classifier was trained with the default number of estimators (100).

2.4.2 Linear Support Vector Classifiers

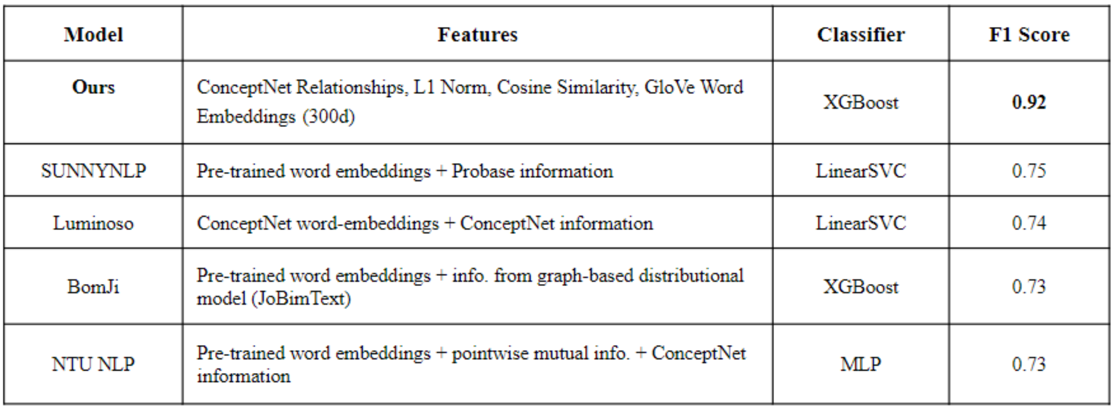
Another classifier used was LinearSVC, because it was the classifier which yielded the best results in the SemEval task. LinearSVC works by finding the hyperplane that maximally separates the data points of different classes, 0 and 1 in our case. We tested different parameters for this classifier in order to optimise it: C, kernel and gamma.

2.4.3 Multi-layer perceptron

Another classifier which showed good performance in the task was simple neural networks. In this work we implemented an MLP composed of 4 dense layers, ReLU activation functions, adam optimizer, dropout and binary cross entropy as loss function. We also considered using Convolutional Neural Networks but since the ‘simpler’ NN did not yield good results we did not continue this path.

3. Results

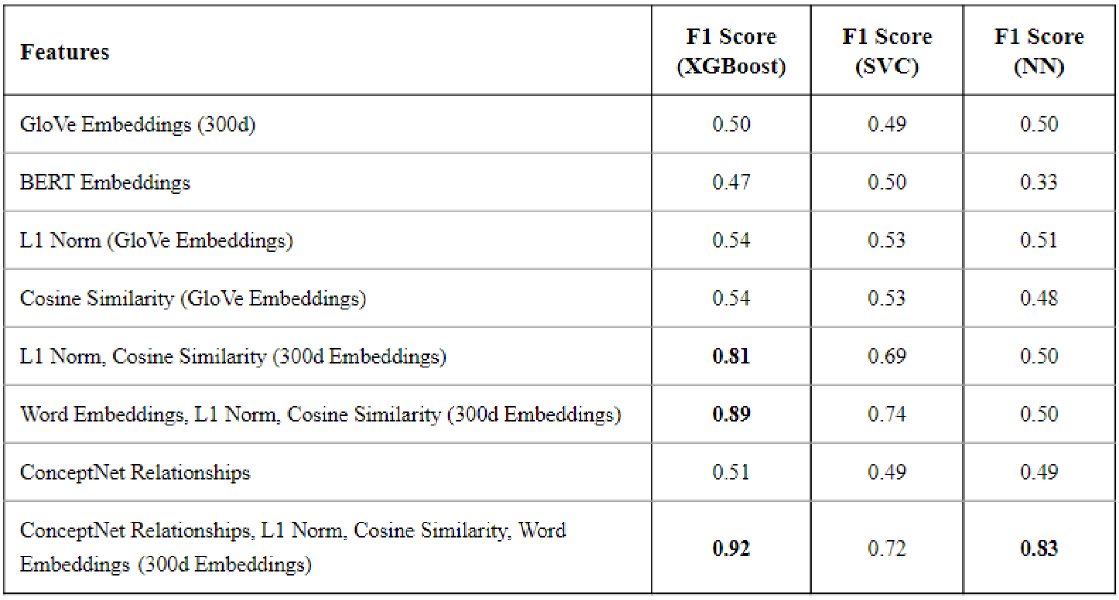
In this section we show the results that have been obtained by our model. The table below shows a comparison of the results of our best performing model with the best performance models by the top 4 teams of SemEval 2018 Task 10. As it can be seen, our model outperformed in terms of F1 score all the other methods, which is not really coherent due to the fact that it used similar features and same classifier as one of the approaches. This will be commented on in the following section.



Model	Features	Classifier	F1 Score
Ours	ConceptNet Relationships, L1 Norm, Cosine Similarity, GloVe Word Embeddings (300d)	XGBoost	**0.92**
SUNNYNLP	Pre-trained word embeddings + Probase information	LinearSVC	0.75
Luminoso	ConceptNet word-embeddings + ConceptNet information	LinearSVC	0.74
BomJi	Pre-trained word embeddings + info. from graph-based distributional model (JoBimText)	XGBoost	0.73
NTU NLP	Pre-trained word embeddings + pointwise mutual info. + ConceptNet information	MLP	0.73

Table 2: F1 score obtained by our model (first row) and top 4 performance from teams which participated in SemEval 2018 Task 10

In order to better assess the evaluation of our model, we performed an ablation study in which we used different combinations of features and classifiers. We started from the most simple case: feeding the classifier with just the embeddings extracted from the triplets. This experiment yielded similar results as random guessing. Then, we experimented with using the word embeddings extracted using BERT, which showed a similar performance in the case of SVC and XGBoost and significantly lower score when using neural networks.



Features	F1 Score (XGBoost)	F1 Score (SVC)	F1 Score (NN)
GloVe Embeddings (300d)	0.50	0.49	0.50
BERT Embeddings	0.47	0.50	0.33
L1 Norm (GloVe Embeddings)	0.54	0.53	0.51
Cosine Similarity (GloVe Embeddings)	0.54	0.53	0.48
L1 Norm, Cosine Similarity (300d Embeddings)	**0.81**	0.69	0.50
Word Embeddings, L1 Norm, Cosine Similarity (300d Embeddings)	**0.89**	0.74	0.50
ConceptNet Relationships	0.51	0.49	0.49
ConceptNet Relationships, L1 Norm, Cosine Similarity, Word Embeddings (300d Embeddings)	**0.92**	0.72	**0.83**

Table 3: Ablation study of our model. Comparison of the results obtained by different feature combinations and different classifiers

In the rest of the experiments, we introduced progressively the rest of features, including the L1 Norm, Cosine Similarity and relationships extracted from ConceptNet. The best results were obtained by training the classifier with all of the features which were extracted (GloVe embeddings + distances + knowledge base features), achieving a F1 score of 0.92. As for the classifiers, overall, SVC showed best performance and neural networks worst performance. These results will be discussed in the following section.

4. Discussion

As it was mentioned in the methodology section, the best performance from the teams that participated in SemEval was 0.75. Nevertheless, our model achieved an F1 score of 0.92 when using all features and XGBoost classifier. To us, these results are not consistent with those obtained by other teams, mainly because using similar features and same classifier gives very different results. Besides, there are other inconsistencies in our results. The original paper which proposes this SemEval task provides a baseline which only uses cosine similarity as a feature, which yields a F1 score of 0.61. In our case, neither using GloVe embeddings nor BERT, provides results as high, being in our case the score obtained similar to random guessing. We attribute these strange results to either some error in the data which has been used, on how it is splitted or in the feature extraction pipeline.

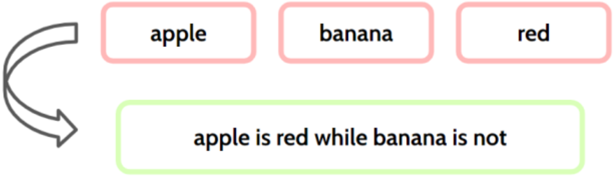


Figure 6: Words to sentence transformation

Besides, we believe that it is important to comment on the BERT embeddings which have been used in the experiments and how they have been used. BERT is a pre-trained transformer-based language model that is designed to understand the context of a word by looking at the words that come before and after it in a sentence. Because BERT is a contextual model, it cannot be used to extract embeddings for isolated words. The embeddings are generated by the model when it processes the word in context and it is not possible to extract an embedding for a word out of context. Initially, we believed that using the other 2 words of each instance to compute the embedding would serve as ‘context’, then, when seeing the results, we realised it was not a good approach to follow. Instead, we should have added context to the words by, for example, as it was mentioned in the project presentation, convert the words into a sentence as it is illustrated in the figure above. It is believed that this approach would be more suitable for the task.

5. Conclusion

In this work we have proposed a solution for SemEval 2018 Task 10 ‘Capturing Discriminative Attributes’, where we extract both word-embedding related features and knowledge base based features. Besides, we have experimented with different classifiers, such as LinearSVC, XGBoost and Vainilla Neural Networks. After analysing the results that we obtained we saw inconsistencies in the scores compared to what other teams achieved by using similar kinds of features. To the moment, it is unknown the cause of these inconsistencies but it will be looked into in depth. Besides, we also conclude that there are other ways to improve our model by, for example, ‘contextualising’ the words from each instance by constructing a sentence within them. By doing so, we can use contextual models such as BERT to model our data.

References

[1] Krebs, A. (n.d.). SemEval-2018 Task 10: Capturing Discriminative Attributes. ACL Anthology. https://aclanthology.org/S18-1117/

[2] Lai, S. (n.d.). SUNNYNLP at SemEval-2018 Task 10: A Support-Vector-Machine-Based Method for Detecting Semantic Difference using Taxonomy and Word Embedding Features. ACL Anthology. https://aclanthology.org/S18-1118/

[3] Speer, R. (n.d.). Luminoso at SemEval-2018 Task 10: Distinguishing Attributes Using Text Corpora and Relational Knowledge. ACL Anthology. https://aclanthology.org/S18-1162/

[4] Brychcín, T. (n.d.). UWB at SemEval-2018 Task 10: Capturing Discriminative Attributes from Word Distributions. ACL Anthology. https://aclanthology.org/S18-1153/

[5] Attia, M. (n.d.). GHH at SemEval-2018 Task 10: Discovering Discriminative Attributes in Distributional Semantics. ACL Anthology. https://aclanthology.org/S18-1155/

[6] Santus, E. (n.d.). BomJi at SemEval-2018 Task 10: Combining Vector-, Pattern- and Graph-based Information to Identify Discriminative Attributes. ACL Anthology. https://aclanthology.org/S18-1163/

[7] Cree, G. S. (2003). [PDF] Analyzing the factors underlying the structure and computation of the meaning of chipmunk, cherry, chisel, cheese, and cello (and many other such concrete nouns). | Semantic Scholar. https://www.semanticscholar.org/paper/Analyzing-the-factors-underlying-the-structure-and-Cree-McRae/f0ae61f7240293e056f5299dac6dc9d65669b247