Machine Learning Project-2: Finding meaning in autogenerated text

Erik Börve, Gonçalo Gomes, Marcus Henriksbø
erik.borve@epfl.ch, goncalo.cavacogomes@epfl.ch marcus.henriksboe@epfl.ch
Department of Computer Science, EPFL, Switzerland

Abstract

Understanding the intent of spoken utterances can be a difficult task, not only for humans on occasions, but even more so for computers. In this project we aim at doing to later by implementing a pipeline that combines automatic speech recognition (ASR) and natural language processing (NLP) with the concepts that where introduced during the course. Utilizing different ASR and NLP techniques we where able to construct suitable feature spaces from labeled audio files that could be used to train classifiers and identify the intent of the user. Using a TF-IDF embedding with the Wave2vec ASR and logistic regression resulted in a correct intent classification with $\sim 87\%$ accuracy.

I. Introduction

The subject of spoken language understanding (SLU) concerns extracting meaning from some spoken utterance. As one can assume, a large application within this field is for different voice assistance that aim to convert some spoken user query to a certain action. This is a task consisting of multiple parts. The first, and perhaps largest part, is to extract the intent of the user. This can be illustrated using the following example sentence "*Activate all the lights in the entire house*", which corresponds to the user action "*SwitchLightOn*". However, one can also distinguish that the full intent of this sentence is more nuanced and for example also specifies the location of the lights in question. Hence, performing intent classification is no arbitrary task.

Considering the whole pipeline, translating the audio signal to text requires the use of some automatic speech recognition algorithm (ASR). This process will introduce some word error rate (WER) which further complicates the classification problem. This process applied to the previous example sentence can be illustrate as seen below in figure 1. In this case the used ASR algorithm was the *TDNNF-LFMMI* model which resulted in the translation "*w. lights in the in her house*" with a WER $= 0.625$.

As seen in this example a high WER can significantly distort the ground truth sentence which naturally results in a more challenging classification problem. To successfully identify the intent of the user it is hence vital construct an intent classification pipeline that, not only minimizes the WER, but also is robust to different WER. The focus of this project is hence, not only to find a well performing classifier for this task, but also to investigate the impact of WER by performing classification on both ground truth and ASR sentences. This was accomplished by applying a range of natural language processing (NLP) and machine learning (ML) techniques which are detailed further in the following sections.

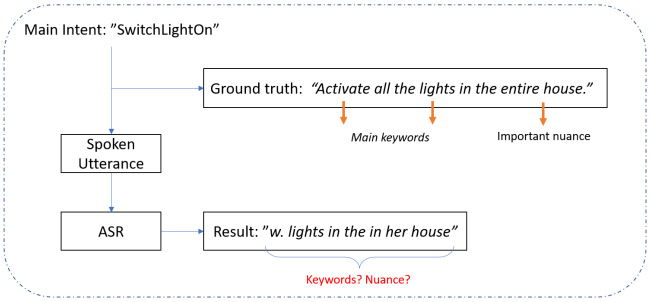

Figure 1: Illustration of classification problem.

II. Data analysis and Processing

The data that was used in this project consisted of 1660 audio files with corresponding ground truth transcripts and labels. Each transcript corresponded to a single sentence of some length and the corresponding label was a member of a set of 6 different actions. The exact content is described further in the paper "*Spoken Language Understanding on the Edge* by *SNIPS* [1].

A. Preprocessing

To improve the performance of the different text embedding methods different common NLP methods where tested and applied. First, all uppercase and special characters where either replaced or removed. Each sentence could then be tokenized into unigrams. Further, methods utilizing lemmatization and removing stopwords was studied. The idea of lemmatization is to reduce certain words to a common stem, e.g *goes→go* and the idea of removing stopwords is to remove some words if they exists in some predefined set of common words [2]. A visualization of the impact of applying these methods to the previous example sentence can be seen in the figure 2.

However, as can be noted in the displayed example some subtle meanings can be removed in this process. One should therefore note that e.g removing stopwords also could have a

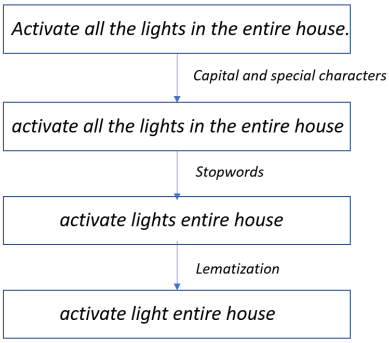

Figure 2: Illustration of the text pre-processing process.

negative impact on the classification performance and should be used sparingly.

B. Text embedding

Since each feature initial consist of an array of strings it is necessary to first represent these in a format that is suitable for a classification algorithm. Selecting the most appropriate method to obtain an appropriate text representation is a large field within NLP and in this case 3 conceptually different methods where studied.

As an initial approach the more simple *Term frequency-Inverse Document Frequency* (TF-IDF) method was studied. This approach weights the amount of times a word appears in a sentence with the length of the sentence, as well as taking into account how many times this word is present through out the entire corpus. For a token i in a sentence j this is hence expressed as,

$$X_{i,j}^{\text{TF-IDF}} = \frac{\text{Nr of } i \text{ in } j}{\text{Total tokens in j}} log\left(\frac{\text{Nr of sentences}}{\text{Nr of sentences containing } i}\right)$$

This results in a vector for each sentence where each entry corresponds the TF-IDF embedding of each unique word in the corpus. Hence, for a corpus with N sentences and n unique words the feature space will be of size $(N \times n)$. Depending on the variance of used words this usually results in a quite spares space. Additionally, since this method only takes into account different frequencies there is no semantic or contextual information that is being preserved [3].

On the other hand utilizing the *Word2vec* (W2V) text embedding we are able to capture the semantic meaning of words in the corpus. This is done by training a neural network to predict the context that a certain word will appear in, as outlined in "*Distributed Representations of Words and Phrases and their Compositionality*" by Mikalov.T. et al [4]. Each word in the corpus can then be represented by some n-dimensional vector, for which the euclidean norm will be similar for words that appear in a similar context. Since the text representation is trained based on the context this representation will have some form of semantic meaning (e.g the vector representation of "dog" will ideally have a small euclidean distance to "cat"). Further, by summing the Word2vec representation of over each word in a sentence, it is possible to express each utterance as a single vector of size n,

$$X_j^{\text{W2V}} = \sum_{i=1}^{I_j} \frac{X_i^{\text{W2V}}}{I_j}$$

where I_j is the number of words in each sentence j. In addition,since the corpus of this data set is relatively small we opted to use pre-trained models. In this case the "*gigaword-100*" model was used which is trained on the corpus of Wikipedia and returns a 100-dimensional vector representation of each word.

Lastly, to investigate the impact of a text embedding with conceptual meaning, we applied the BERT-model (*Bidirectional Encoder Representations from Transformers*) which is formally outlined in the paper *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"* by Delvin J. et al [5]. All though this method relies heavliy on recurrent neural networkds,which can be considered to be outside of the scope of this course, the fundamental difference between the BERT embedding and the W2V embedding is that the W2v variant will generate a fixed embedding for each token regardless of its contextual meaning. Using BERT e.g the word "*fair*" can have a different encoding depending on if its used in a transcript from a court of law or on a cosmetics product.

C. Automatic Speech Recognition

The purpose of an ASR algorithm is to convert some audio file to a corresponding text transcript with an as low WER as possible. For this project 2 different methods where studied. These text transcripts where then similarly process by one of the different text embedding models to generate a feature for each transcript.

The first and relatively less involved model was the *TDNNF-LFMMI* model which combines the "*TDNNF*" model developed in "*Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks*" by Povey D et al and the "*LFMMI*" model developed in "*Multitask adaptation with Lattice-Free MMI for multi-genre speech recognition of low resource languages* " by Motlicek .P et al [6], [7]. These models utilize deep neural network concepts which are out of the scope of this course and an accurate description is available in the above articles.

The second and more involved studied model was the "*Wave2Vec-U*"-model which utilizes generative adversarial neural networks (GAN) for training and is hence unsupervised. The exact algorithm however, can again be considered to be outside the scope of this course but a detailed description is available in the paper "*Unsupervised Speech Recognition*" by Baevski A et al [8].

D. Feature augmentation

Before performing the classification task the available data was augmented in an attempt to mitigate any potential bias and improve the prediction performance. Utilizing the 3 different text embedding methods resulted in a feature space $\mathbf{X}$ of dimension $(N \times n)$ where in the TF-IDF case $n =$ "Nr. unique words in data set",W2V $n = 100$ and Bert $n = 768$. As an initial approach the Bert and W2V features where standardized to have zero mean and the TF-IDF features where normalized.

After doing the speech recognition, we notice that there were some deviations from the true words in the sentences produced by the automatic speech recognition engine (e.g: ight istead of light). We were concerned if this deviation would affect the natural language processing techniques in producing a congruent feature space, especially for text embeddings that tacked into account the contextual and semantic similarity of the words to produce the feature space such BERT and Word2Vec. In an attempt to mitigate this issue and at the same time reduce the WER, an auto correction algorithm was introduced. The auto correction was performed by first creating a dictionary "(D_g)" containing all tokens "(t)" in the ground truth. Using the Jaccard similarity (J) between each token in the ASR transcript "A_ij" and each word in the ground truth dictionary $D_g(t)$ it was then possible to get some metric of their character based similarity. This was computed as,

$$J(A_{i,j}, D_g(t)) = \frac{|A_{i,j} \cap D_g(t)|}{|A_{i,j} \cup D_g(t)|} = \frac{\text{"Nr. of similar characters"}}{\text{"Total Nr. of characters"}}$$

Maximizing $J(A, D_g)$ then gave the most similar token of the ground truth which could be used to replace its misspelled version in the ASR transcript.

These auto corrected sentences was then similarly feed to the corresponding text embedder and processed exactly as mentioned previously.

III. Method

As outlined in "*Text Classification Algorithms: A Survey*" by Kamran Kowsari et al some of the most popular algorithms for performing classification include support vector machines (SVM), Naive Bayes (NB) and different versions of recurrent neural networks (RNN) such as "long short term memory" (LSTM) [9]. As RNNs where out of the scope of this course we opted for a more simple feed forward neural network. In summary the studied methods where:

- Logistic regression
  As suggested in the above article this was a suitable linear classifier.
- Naive Bayes
  For the NB classifier we decided on the Gaussian Naive Bayes version, as this supported the negative values produced by our Word2Vec text embedding.
- SVM
  The kernel used was the standard method in the sklearn library, the "radial basis function" (RDF). RDF was since it has been shown to hold higher predicive ability than a linear kernel in some cases [10].
- Neural networks
  For tuning the parameters, in this case the hidden layers and neurons, iterations in the interval (40*n, 20*m) was tried with n and m ranging from 0 to 25, following the framework presented by Heaton Research [11].

IV. Result

The main results regarding both classifying intent and the impact of WER in ASR are described below. The presented accuracy measurements displayed in tables I,II and III where computed as an average over 100 iterations with a 90/10 train/test split being shuffled each iteration. The best classifier respective of each text embedding technique is in bold.

A. Ground truth transcripts

Utilizing the existing transcribed data (i,e $WER = 0$) we found the performance in classifying intent for different chosen text embeddings and classifiers as shown in table I.

Table I: Results for text embedding and classification on ground truth data.

	Logistic Regression	SVM	MLP (400, 100)	Gaussian Naive Bayes
TF-IDF	0.95	**0.96**	0.95	0.77
W2V	0.90	0.84	**0.92**	0.73
BERT	**0.93**	0.87	0.91	0.60

For TF-IDF we found the SVM classifier provided the best accuracy overall. The three highest scoring combinations of text embedding and classifier were TF-IDF using SVM, W2V using MLP and Bert using Logistic regression.

B. ASR transcripts

Similarly, performing the same analysis for the ASR transcriptions gave the results shown below in tables II and III. Table II shows the results using the *TDNNF-LFMMI* model ($WER = 0.56$) and table III shows the results using the *Wave2vec* model ($WER = 0.36$).

Table II: Results for text embedding and classification on TDNNF-LFMMI transcripts.

	Logistic Regression	SVM	MLP (400, 100)	Gaussian Naive Bayes
TF-IDF	**0.6815**	0.6775	0.63	0.48
W2V	0.57	0.53	**0.58**	0.45
BERT	**0.6**	0.4	0.49	0.39

Table III: Results for text embedding and classification on Wave2vec transcripts.

	Logistic Regression	SVM	MLP (400, 100)	Gaussian Naive Bayes
TF-IDF	0.83	**0.86**	0.84	0.64
W2V	**0.79**	0.76	0.77	0.586
BERT	**0.78**	0.72	0.76	0.54

We clearly see the ground truth data performing better than the ASR transcribed data, as expected. The score difference between the two ASR also deviate quite strongly, where we see the Wave2Vec performing considerably better.

With the word error rate in mind, we see that the Wave2Vec, performing better on WER also leads to higher accuracy on all the combinations of text embedding and classifiers.

As seen in table IV we have better score results in general for BERT and Word2Vec text embeddings using the autocorrection. As mention before the main reason for this is that both of this NLP techniques have in consideration the meaning and semantic similarity of the words so even the slightest semantic deviation of the keywords in the ASR transcript could change completely the entire meaning of the sentence and for that reason this models preformed better after mitigating this issue.

We also calculated the WER on the autocorrected data relative to the ground truth data in order to try to have a sense of how well the data was translated, we see that it yielded a score of 0.32, meaning the autocorrected version was 0.04 more accurate than that without.

Table IV: Results for text embedding and classification on Wave2Vec transcripts with autocorrection.

	Logistic Regression	SVM	MLP (400, 100)	Gaussian Naive Bayes
TF-IDF	0.78	**0.87**	0.81	0.68
W2V	**0.82**	0.77	0.79	0.58
BERT	**0.81**	0.76	0.79	0.60

Looking at the confusing matrix for the different best performing classifiers and text embedding combinations it was possible to analyze the trends in miss-classification.Here figure 3 displays the confusion matrix for TF-IDF using SVM corresponding to the best performing classifier. Additionally, Appendix I displays this same figure for a plethora of combinations. Considering all of these figures it was not possible to distinguish any clear trend in different classes being more difficult to identify.

C. Hyper-parameter Tunning

We tuned the models that were giving us the best results for each NLP technique, in order to try achieve maximum

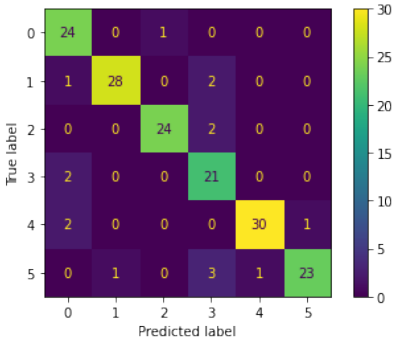

Figure 3: Confusion matrix for TF-IDF and SVM classifier, without the autocorrected dataset.

accuracy scores for both the train/test ASR data with and without autocorrection. The values are present in table V

Table V: Accuracies and tunned hyperparameters (HP) of the best models for the ASR data with and without autocorrection.

	No Autocorrection		Autocorrection	
	HP	Accuracy	HP	Accuracy
TFIDF - SVM	C = 2.78 degree = 3	0.87	C = 1 degree = 3	0.87
WORD2VEC - LGR	C = 0.06	0.79	C = 0.16	0.82
BERT - LGR	C = 0.53	0.78	C = 0.6	0.83

V. Conclusion

According with the results presented above, we can see that TFIDF is the NLP technique that is giving the best accuracy's in general. This might at first be a surprise since it is the simplest text embedding out of the 3. Initially, it would be expected that since it doesn't take into consideration the semantic or contextual meaning of the sentences, this lack of information would be seen as a disadvantage for TFIDF's classification task compared to Word2Vec or BERT, but turns out that this is not the case at all. A reason for it could be the fact that for the ground truth data, the sentences are very similar to each other and for that reason, there is no big advantage to try get information in the meaning or semantic similarity in the sentences to help in the predictions. In addition, to that for the ASR transcripts, the sentences suffer a deviation from the ground truth, influencing the semantic and meaning of the sentence in general, making it difficult to try get information from it.

Regarding the WER it would seem that a lower value in general gives a better performing classifier. However, since it is more important to accurately transcribe the keywords then some other words this metric can be considered to be somewhat misleading. A further investigation could be to identify the keywords of each sentence and calculate the WER solely for these words.

References

[1] A. S. . et al, "Spoken language understanding on the edge," 2019. [Online]. Available: https://arxiv.org/pdf/1810.12735.pdf

[2] D. K. . et al, "An interpretation of lemmatization and stemming in natural language processing," 2021. [Online]. Available: https://www.researchgate.net/publication/348306833_An_Interpretation_of_Lemmatization_and_Stemming_in_Natural_Language_Processing

[3] J. Ramos, "Using tf-idf to determine word relevance in document queries," 2003. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1424&rep=rep1&type=pdf

[4] T. M. . et al, "Distributed representations of words and phrases and their compositionality," 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

[5] J. D. . et al, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/pdf/1810.04805.pdf

[6] D. Povey, G. Cheng, Y. Wang, K. Li, H. Xu, M. Yarmohammadi, and S. Khudanpur, "Semi-orthogonal low-rank matrix factorization for deep neural networks," *Interspeech*, pp. 3743–3747, 2018.

[7] S. Madikeri, P. Motlicek, and H. Bourlard, "Multitask adaptation with lattice-free mmi for multi-genre speech recognition of low resource languages," *Interspeech 2021*, pp. 4329–4333, 1998.

[8] A. Baevski, W.-N. Hsu, A. Conneau, and M. Auli, "Unsupervised speech recognition," *NeurIPS*, 2021. [Online]. Available: https://arxiv.org/pdf/2105.11084.pdf

[9] K. K. . et al, "Text classification algorithms: A survey," *Information*, 2019. [Online]. Available: https://arxiv.org/pdf/1904.08067.pdf

[10] S. S. Keerthi and C.-J. Lin, "Basymptotic behaviors of support vector machines with gaussian kernel," 2003. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.880&rep=rep1&type=pdf

[11] H. Research, "Choosing number of neurons and hidden layers," 2017. [Online]. Available: https://www.heatonresearch.com/2017/06/01/hidden-layers.html

Appendix I

This appendix displays additional plots of confusion matrices for more combinations of classifiers,text embedders and datasets. All ASR transcripts where created using Wave2Vec.

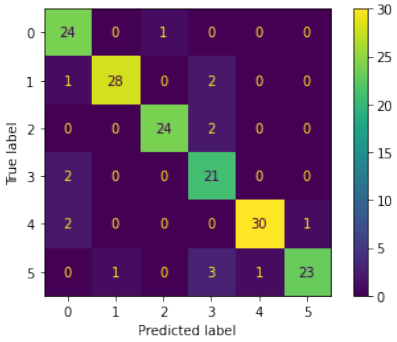

Figure A1: Confusion matrix for TF-IDF and SVM classifier, without the autocorrected dataset.

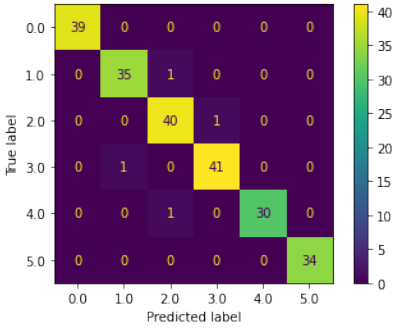

Figure A2: Confusion matrix for TF-IDF and SVM classifier, with the ground truth data.

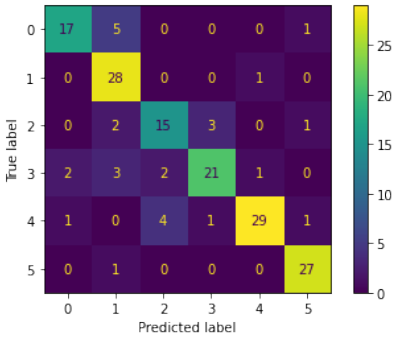

Figure A3: Confusion matrix for W2V and LGR classifier, with the autocorrection of ASR.

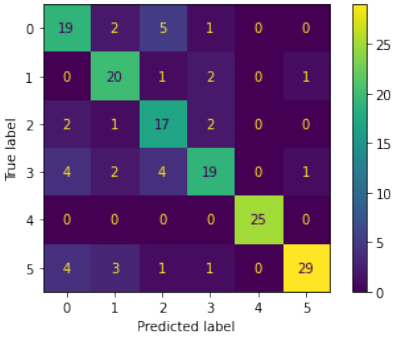

Figure A4: Confusion matrix for W2V and LGR classifier, without the autocorrection of ASR.

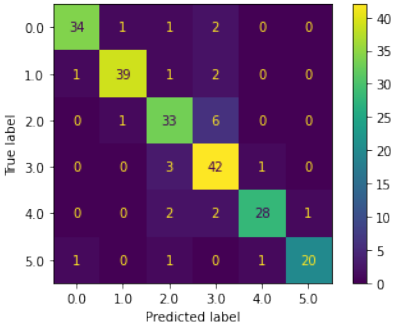

Figure A5: Confusion matrix for W2V and LGR classifier using the ground truth data.

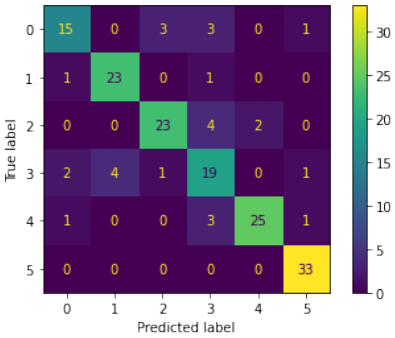

Figure A6: Confusion matrix for BERT and LGR classifier, with the autocorrection of ASR.

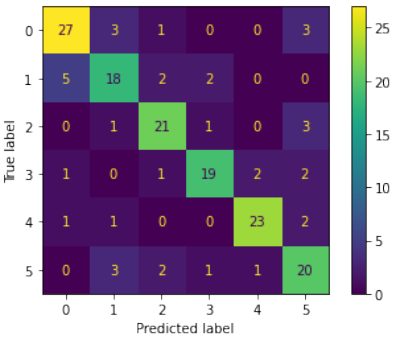

Figure A7: Confusion matrix for BERT and LGR classifier, without the autocorrection of ASR.

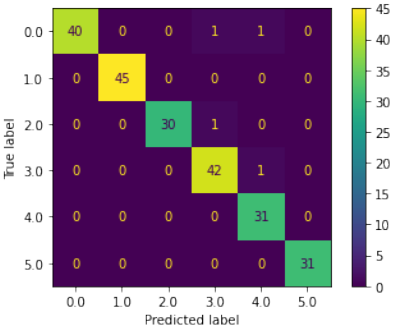

Figure A8: Confusion matrix for BERT and LGR classifier using ground truth data.