Data expansion for semantic classification

Anna Liednikova
Supervisor: Claire Gardent

February 5, 2019

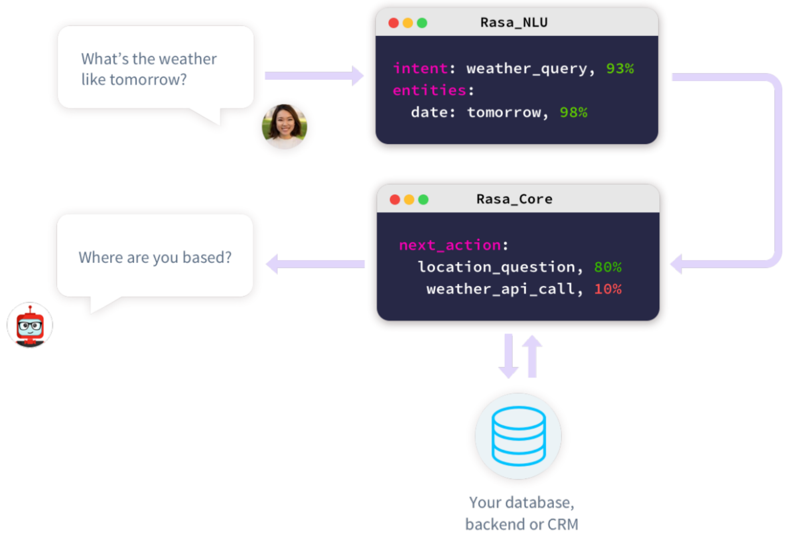
Contents

1 Introduction

This project was set within the framework of a collaboration with the ALIAE startup whose aim is to develop a chatbot to collect information from clinical patients. The main idea is to replace strictly defined surveys by a more natural conversation in order to let users express themselves freely so that more information could be collected.

The chatbot consists of two main modules:

- NLU (Natural Language Understanding): interpreting the user input
- Dialog Managment: deciding how to respond to the user input and generating the system response


What's the weather like tomorrow?
Rasa_NLU
Where are you based?
Rasa_Core
Your database, backend or CRM

```
intent: weather_query, 93%
entities:
  date: tomorrow, 98%
```

```
next_action:
  location_question, 80%
   weather_api_call, 10%
```

Figure 1:

In this project, we focus on natural language understanding and consider a simplified notion of understanding where each user input is mapped to an "intent". Given a pre-defined and finite set of intents (Does the user speaks about pain, physical activity etc.), the aim of the NLU module is to assign each user input an intent[1]. The detected intent is then used, in conjunction with additional information extracted from the previous dialog interactions, to determine how to respond (cf. Figure 1[2]).

2 Goal and Motivations

The NLU module is a classifier which given a user input, predicts the corresponding intent. Our goal is to improve this classifier starting from a small training set (roughly 100 utterance/intent pair for each intent) and automatically expanding this labelled data. To this end, we developed a semi-supervised data expansion technique which permits expanding the initial data with utterances extracted from a medical forum. We show that the expanded data permits improving the classifier by 0.05 accuracty points.

This report is structured as follows. Section 3 briefly reviews the state of the art and situates our proposal with respect to related work. Section 4 presents the methodology used for data expansion. Section 5 describes the datasets used. Section 6 reports on the experimental setting and on the results. Section 7 concludes with pointers for further research.

3 Related Work

To use supervised text classification methods, it is important to have a dataset of a reasonable size in order to make their training efficient. When only a small set of labelled documents is available, the semi-supervised learning becomes a way-out and helps to resolve tasks by comparing the distributions between labelled and unlabelled items.

[FLWH18] presented an approach for automatically generating additional training data for event extraction systems. First, they trained a baseline classifier on available gold data. Then, they split external data into clusters of paraphrases using the NewsSpike method introduced by [ZHDCZ15]. They then label the clusters using the baseline model trained on the initial dataset of labelled data. Combining the new labelled data and the original one, they then retrained the event extractor and achieved significant improvement of F1 score.

[1]We make the simplifying (and incorrect) assuption that the user input contains a single intent.

[2]The figure actually shows a more complex notion of understanding where the user utterance is mapped not only to an intent but also to a set of entities. In this work, we focus on how to improve intent detection and leave entity detection for future research.

There are many models that encode sentences into fixed-size representations. LDA and Word2Vec models have been extensively used and have shown their effectiveness on the continuation of a long time and a wide range of tasks[IPN16] [JWY$^+$17]. Due to their popularity, there are pre-trained models on Tweets and Google News data. Though, the research of [LLCC18] demonstrates that GloVe embeddings trained on the specific data produce better results in narrow field problems than the pre-trained ones that are better for more wide tasks. [CKL$^+$18] also showed the efficiency of BiLSTM-max model for capturing word context in encoding sentences.

SVM is considered to be a hands-on approach in textual classification [Tha18] that tends to perform better than other traditional models regardless of the word embedding type [LLCC18]. These classifiers are fast, reliable, work well in both linear and non-linear separation and can effectively deal with sparse data, and have a great ability to generalize knowledge in multidimensional spaces [Joa98]. Last two advantages especially important due to the common use of high dimension in sentence representation task and its sparse nature in case of LDA.

4 Methodology

Our methodology for data expansion falls into two main steps. First, we compare different methods for representing utterances using both clustering and classification (Section 4.1). Second, based on the best sentence representations, we devise a semi-supervised approach which relies on clustering and classification to gradually increment the size of the labelled data and iteratively train new classifiers on the extended data (Section 4.2).

4.1 Building and Evaluating Sentence Representations

We investigate different ways of creating continuous representations for sentences and use clustering to determine which representation best support the identification of intents.

4.1.1 Creating Continuous Representations of Sentences.

We explore two ways of building sentence representations: Word2vec and LDA.

LDA. One of the most common methods for constructing thematic models is the Latent Dirichlet Allocation (LDA) [BNJ03] that models a document as a distribution of topics and a topic as a distribution of words. Here a document is a sentence. Hence a sentence is represented as a distribution of topics.

Let's consider a toy LDA model produces following topics:

```
[(0,
  '0.089*"_num_" + 0.025*"day" + 0.022*"pain" + 0.020*"take" +
   0.018*"year" + 0.017*"feel" + 0.015*"smoke" + 0.014*"sleep"'),
 (1,
  '0.024*"good" + 0.022*"get" + 0.021*"feel" + 0.021*"blood" +
   0.021*"want" + 0.020*"pressure" + 0.014*"understand" '),
 (2,
  '0.067*"go" + 0.048*"sleep" + 0.023*"_num_" + 0.021*"bed" +
   0.018*"fall" + 0.018*"hour" + 0.017*"wake" + 0.016*"asleep" '),
 (3,
  '0.040*"drink" + 0.030*"life" + 0.025*"much" + 0.020*"eat" +
   0.017*"end" + 0.016*"go" + 0.013*"time" + 0.012*"body" ')]
```

And here is an example representation for a sentence/

```
'I can go for months and at least a year or so without drinking',
[0.61565    , 0.03821149, 0.32365379, 0.02248472]
```

If we want represent just one word we can just treat it like a small sentence and have the same output.

```
'sleep',
[(0, 0.21281178), (1, 0.13255174), (2, 0.5765724), (3, 0.07806411)]
```

The LDA algorithm is based on the Dirichlet prior distribution and assumes a bag-of-words model - a model for analyzing texts that takes into account only the frequency of words, but not their order. This model is well suited for thematic modeling, since it permits detecting implicit relationships between words. The LDA method performs soft clustering and assumes that each word in the sentence is generated by some latent topic, which is determined by a probability distribution on the set of all words of the text.

Word2Vec. Word2Vec representations are low dimensional vectors also called embeddings which are built based on textual cooccurences and learned using shallow, two-layer neural networks [MCCD13]. Given a large corpus of text as input, Word2Vec produces a vector space where words that share common contexts in the input corpus will be mapped to vectors that are close to one another in the vector space.

To build sentence representations, we simply average the Word2Vec vectors of the content words present in that sentence.

Bi-LSTM. The general idea of BiLSTM layer is repeating the first recurrent layer in the network so that they are next to each other and then providing the straigh input sequence as input to the first level and an inverse copy of it for the second.

In more formal way, for the sequence of words $T\{w_t\}_{t=1,...,T}$ bidirectional LSTM computes a set of T vectors $\{h_t\}_t$. For $t[1,.,,,T]$, h_t is the combination of forward and backward LSTM, that read sentences in two opposite directions.

We experiment with two ways of combining a changing number $(h_1, ..., h_T)$ to form a vector of a fixed size, either by choosing the last hidden state h_T (BiLSTM-last), or either by selecting the maximum value over each dimension of the hidden units (max pooling) (Collobert and Weston, 2008) or by considering the average of the representations (mean pooling). The choice of these models is motivated by their demonstrated effectiveness in embedding models [CKL+18].

4.1.2 Clustering Sentences

Using the sentence representations described in the previous section, we apply clustering to group together sentences that are similar. We set the number of clusters to the number of intents (20) and we use (i) purity and Silhouette coefficients to evaluate the clusters and (ii) homogeneity and completeness to better assess how the resulting clusters correlate with initial intents. Also, for each model, we plot a confusion matrix with a background gradient to visualize how well clusters separate different intents. Based on these metrics and vizualisation, we determine which method produces representations that best support intent detection.

We compare three clustering algorithms:

- K-Means (KM)
- Agglomerative Clustering (AG)
- Gaussian Mixture (GM)

K-Means. The basic idea is to define K centroides one for each cluster. It is best to place them as far apart as possible. The next step will be to accept each point belonging to this data set and associate it with the precision of the centroid. If no point is under consideration, the first stage is completed. At the moment, we need to re-calculate K's new centroids, as the cluster barycenter that stems from the previous step. After these new centroids appear on us, a new binding must be made between the same points of the data set and the nearest new center of gravity. A loop is created. As a result of this cycle, we can see that k centroids change their location step by step until any changes are made. In other words, centroids are no longer moving.

The algorithm is aimed at minimizing the target function, in this case, the quadratic error function. Target function:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} (\|x_i^{(j)} - c_j\|)^2 \tag{1}$$

where $(\|x_i^{(j)} - c_j\|)^2$ is the chosen distance between the data point $x_i^{(j)}$ and center c_j, it is an indicator of the distance n of the data points from their respective cluster centers.

Centroid is defined as follows:

$$c_j = \frac{1}{N} \sum_{i=1}^{N} x_i^{(j)} \tag{2}$$

where $x_i^{(j)}$ is the point belonging to the cluster J, N - total number of points in the cluster.

Agglomerative Clustering is a kind of hierarchical clustering with 'bottom-up' approach when new clusters are created by combining smaller clusters and, thus, a tree is created from leaves to trunk [War63]. The main feature of the agglomerative hierarchical clustering method is that if we want to get a different number of clusters, we will not need to restart the algorithm, because the whole tree is calculated, and then we say that we want to cut it in some place.

Different linkage criterions can be used to calculate distance between sets of observation. Wards method minimizes the variance of the clusters being merged. This method is used for tasks with closely spaced clusters. The euclidian distance between clusters is taken as the increment of the sum of squares of the distances of objects to the center of the cluster, obtained as a result of their combination:

$$\Delta = \sum_{i} (x_i - \bar{x})^2 - \sum_{x_i \in A} (x_i - \bar{a})^2 - \sum_{x_i \in B} (x_i - \bar{b})^2 \tag{3}$$

At each step of the algorithm, these two clusters are combined, which lead to a minimal increase in variance.

Gaussian Mixture. In statistics, a mixed model is a probabilistic model for representing the presence of subpopulations in the general population in accordance with the mixture of distributions, without requiring that a separate observation belong to a specific subgroup.

A Gaussian mixture model [Rey15] represents a weighted sum of M Gaussian densities elements for D-dimensional continuous-valued data vectors:

$$p(x|\lambda) = \sum_{i=1}^{M} w_i g(x|\mu_i, \sum_{i}) \tag{4}$$

where each Gaussian densities element defined as

$$g(x|\mu_i, \sum_{i}) = \frac{1}{(2\pi)^{\frac{D}{2}} |\sum_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_i)' \sum_i (x-\mu_i)} \tag{5}$$

with mean vector μ_i and covariance matrix $\sum_i$. The mixture weights satisfy the constraint that $\sum_{i=1}^{M} w_i = 1$.

GMM parameters are estimated from training data using the iterative Expectation-Maximization (EM) algorithm or Maximum A Posteriori (MAP) estimation from a well-trained prior model.

4.2 Semi-Supervised approach to Data Expansion and Classification

Based on experimental results (cf. Section 6), we choose the best model for representing sentences. We then use these representations to iteratively expand the training data and train the corresponding classifiers based on the following procedure:

- **Step 1: Classification.** Apply the classifier trained on the currently available labelled data to a set of unlabelled sentences;

- **Step 2: Clustering.** Clusterize these unlabelled sentences into 200 clusters

- **Step 3: Filtering.** Use the probability and the label assigned by the classifier to each unlabelled sentence to :
  - Eliminate sentences whose probability is below a given threshold
  - Eliminate clusters with less than a set number of members
- **Step 4: Data expansion.** For each intent, add the n best labelled items to the training data.
- **Step 5: Classifier Training.** Train a new classifier using both the previous and the newly selected, labelled data.

4.2.1 Classification.

For classification, we use SVC (Support Vector Classifier). The main idea of the method is transfering the initial vectors into a space of higher dimension and searching for a separating hyperplane with the maximum gap in this space. Two parallel hyperplanes are built on both sides of the hyperplane separating the classes. The separating hyperplane will be the one that maximizes the distance to two parallel hyperplanes. The algorithm works under the assumption that the greater the difference or the distance between these parallel hyperplanes, the less will be the average classifier error. Due to the small size for dataset it's essential to create balanced training and test datasets.

The initial labelled dataset was provided by ALIAE (gold data). We split our gold dataset into train and test sets and systematically use the test set for evaluation and comparison of the classifiers learned on the different datasets produced during data expansion.

At each iteration during data expansion, we use the classifier trained on the data created at the previous iteration to label unlabelled sentences extracted from some external source.

4.2.2 Clustering.

To expand the training data, we retrieve utterances from a medical forum. The unlabelled data is then built based on these utterances after some filtering is applied to eliminate sentences that are either out of topic or too complex (cf. Section 1). Next we build representations for these unlabelled sentences using the techniques described in Section 1 and we apply clustering to create 200 clusters.

4.2.3 Filtering.

The SVC classifier assigns each unlabelled sentence a category (an intent) and a probability. We eliminate from the clusters all those sentences whose probability is below a given threshold. We then eliminate all clusters whose size is less than a set minimum.

For our subset we find the best clusterization by elbow method [BK14]. Later, having both labels for intent and cluster, we leave just that clusters that populated by items with maximum probability that allows us to cover all intent with new examples.

For each cluster we calculate number of intent representators and their overall distribution. We then keep only those clusters where some intent is represented more than in average. Elements of clusters that are left will be labelled by the majority.

4.2.4 Data Expansion.

Finally, we select N labelled items for each category/intent and extend the training data with them to update sentence representation and classification models.

We repeat this cycle until we won't be able to extend train dataset by additional samples or classifier score will stop to grow.

5 Datasets

For our task we have two datasets: a small, manually labelled dataset where each utterance is associated with a target intent; and a large, unlabelled dataset extracted from the web. Our goal is to automatically label this web data in order to expand the training set and to improve the NLU module of the ALIAE chatbot.

5.1 Labelled Data

The initial dataset was created manually by the Aliae startup and covers 20 main possible users intents. Each sentence represents one intent e.g.,

Text: After sleep, for 2-3 hours, I am better and then start feeling tired again

Intent: sleep

The distribution of labels is shown in Figure 2. There are in average, 3.4 words per sentence (min: 0, max: 12, std: 2.16367). The total number of sentences is 3305 and the vocabulary consists of 1882 content words (after removing stopwords).

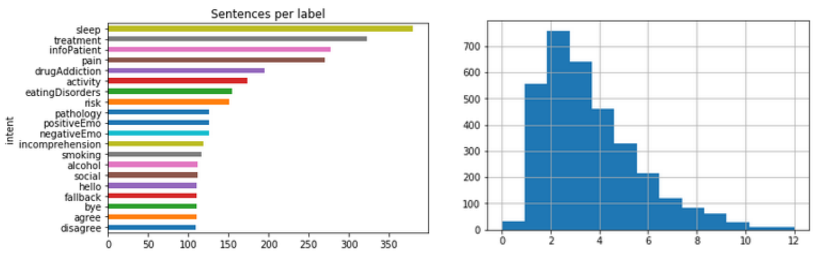


Figure 2: Left: Label distribution in dataset, Right: Distribution of sentence length

5.2 Forum data

To improve the initial classifier, we needed to extend the training dataset by much more data and more naturally constracted one. Looking for open medical dataset that can be used for this purpose, we found [ZHDCZ15] and [SGZH10] works that used the HealthBoards forum data for similar tasks. HealthBoards[3] is a medical forum web portal that allows patients to discuss their ailments.

We scraped 272553 unique posts labelled with 238 forum categories. Figure 3 shows the dataset statistics. In average, sentences contain 11 ± 7 words after preprocessing.

[3] `healthboards.com`

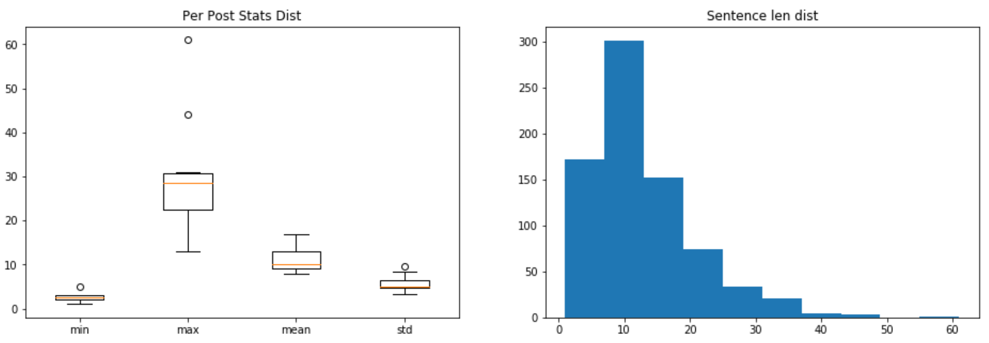


Figure 3: Text stat

This dataset has its own categorization so that some posts are more specialized and some are out of the scope of our intent labels. We should start with the data that is most similar to the initial one, so that's why we choose post categories hich are similar to our intents. This will help us to gradually expand the training data. So we will keep posts of categories provided in table 1.

Category of board	# of posts
digestive-disorders	5064
addiction-recovery	3644
sleep-disorders	1748
smoking-cessation	937
eating-disorder-recovery	762
chronic-pain	735
chronic-fatigue	662
stress	415
family-friends-addicts-alcoholics	312
pain-management	25

Table 1: Selected Forum categories

Finally, the corpus consists of 380014 sentences.

Also, for big number of iterations, the data should be divided in subsets by increasing sentence length because of the difference in mean values for both datasets. So after tokenizing tje posts into sentences we calculate their length. For each iteration, we only keep those sentences which have mean+std words after pre-processing (cf. Section 6.1).

6 Experiment

We now describe how the steps described in Section 4 were applied to the forum data.

6.1 Preprocessing

To build continuous representations of sentences, we only take into account the lemmatized form of the content words they contained. Hence a preprocesssing step is applied to segment the forum data into sentences, to tokenize and lemmatize the resulting sentences and to remove stop words.

For lemmatization we used SpaCy as it provides lemmas for every token. Next examples shows the transformation:

> Before: I can go for month**s** and at **least** a year or so without drink**ing**
> After: I can go for month and at little a year or so without drink

We compared two libraries, NLTK and SpaCy for tokenization, they give different results that can influence not only simple statistics but also meaning. Some example of different tokenizations can be seen in table 2. Although tokenizing words such as 'flulike' or '35mg' or 'longterm' sometimes gives more robust and concrete meaning, in our case it seems better to stay with SpaCy way of tokenization in order to have a smaller and simpler vocabulary.

NLTK	SpaCy
['i', 'wouldnt', 'go', 'to', 'sleep', 'until', 'like', '5', '6', 'or', '8am']	['i', 'would', 'nt', 'go', 'to', 'sleep', 'until', 'like', '5', '6', 'or', '8', 'am']
['that', 'is', 'totally', 'wrongheaded']	['that', 'is', 'totally', 'wrong', 'headed']
['i', 'am', 'in', 'the', 'process', 'of', 'tapering', 'from', 'suboxone', 'longterm', 'use']	['i', 'am', 'in', 'the', 'process', 'of', 'tapering', 'from', 'suboxone', 'long', 'term', 'use']
['i', 'had', 'an', 'onandoff', 'opiateopioid', 'habit', 'from', 'about', '2010']	['i', 'had', 'an', 'on', 'and', 'off', 'opiate', 'opioid', 'habit', 'from', 'about', '2010']
['i', 'have', 'flulike', 'pathologysymptom']	['i', 'have', 'flu', 'like', 'pathologysymptom']
['i', 'have', 'exerciseinduced', 'insomnia']	['i', 'have', 'exercise', 'induced', 'insomnia']
['i', 'm', 'supposed', 'to', 'take', '6', '35mg', 'tablets', 'a', 'day', 'but', 'i', 'have', 'taken', '20', 'today']	['i', 'm', 'supposed', 'to', 'take', '6', '35', 'mg', 'tablets', 'a', 'day', 'but', 'i', 'have', 'taken', '20', 'today']

Table 2: Tokenization comparison

For stopwords removal, there were three options: NLTK, SpaCy and the longest one. The last option was rejected due to containing words like 'want', 'stop', 'successfully' etc. that can be useful for detecting basic intents such as social (e.g., greetings) and positive or negative emotions. In the end, we decided to use NLTK because of containing shorts like 'm' from 'am', 've' from 'have'.

The final dictionary contained 1882 words. Also all numbers were changed to NUM. Figure 4 shows the distribution of word frequency.

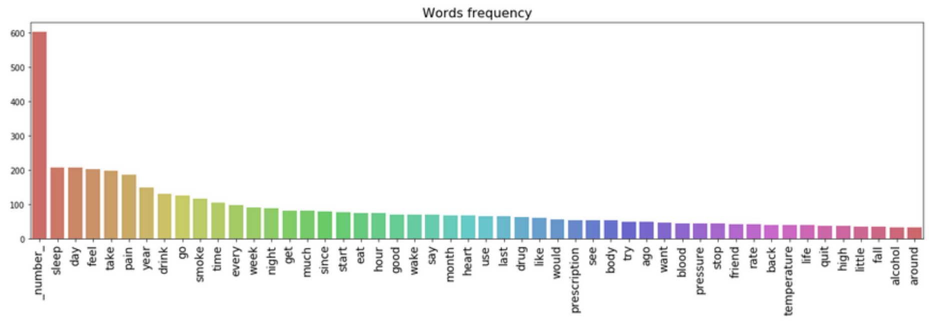


Figure 4: Words frequency

6.2 Clustering for sentence representation evaluation

We use clustering to assess the degree to which sentence representations support the correct classification i.e., the assignment of the correct intent. Also, clustering provides us with information about which classes will be easier to identify and which can mix with others. Using evalution metrics for clustering helps us to compare different ways of representing sentences and to chose the best one.

For each type of sentence representations described in Section 1, we split the forum data into 20 clusters which corresponds to the number of intents. To select the best model, we further experiment with different hyperparameters (number of topics for LDA and embedding size for Word2Vec).

6.2.1 Evaluation metrics

For every model, we compute the following metrics and their average: purity score, Silhouette Coefficient, homogeneity and completeness scores.

Also for each model, we create the confusion matrix between cluster labels and target intents.

Purity is calculated by assigning each cluster to the class that is most often found in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N.

$$purity(\Omega, C) = \frac{1}{N} \sum_{k} \max_{j} |\omega_k \cap c_j| \tag{6}$$

where $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$ is the set of clusters and $C = \{c_1, c_2, ..., c_J\}$ is the set of classes.

Silhouette Coefficient shows how much the average distance to objects of its cluster differs from the average distance to objects of other clusters. This value is in the range $\lfloor 0, 1 \rfloor$. Values close to -1 correspond to poor (scattered) clustering, values close to zero indicate that the clusters intersect and overlap each other, values close to 1 correspond to "dense" clearly selected clusters. Thus, the larger the silhouette, the more clearly highlighted the clusters, and they are compact, tightly grouped clouds of points.

Homogeneity and completeness. Homogeneity will be maximal if the cluster consists only of objects of one class. Completeness will be maximum if all objects from the class belong to only one specific cluster. Formally, these measures are also determined using the functions of entropy and conditional entropy, considering the partitioning of the sample as discrete distributions:

$$h = 1 - \frac{H(C|K)}{H(C)}, c = 1 - \frac{H(K|C)}{H(K)} \tag{7}$$

here is K the result of clustering, is C the true partitioning of the sample into classes. Thus, h measures how much each cluster consists of objects of one class, and c measues how much objects of one class belong to one cluster.

6.2.2 Word2Vec model

For Word2Vec representations, we compare results for embedding sizes ranging from 10 to 600.

Embedding Dim.	purity	silhouette	homogeneity	completeness
10	0.179526	0.0767427	0.106067	0.12561
20	0.185678	0.0873321	0.107696	0.132483
30	**0.189007**	0.0695026	0.110232	0.13267
50	0.180535	0.0441912	0.110117	0.131568
100	0.186788	0.0594827	**0.11196**	0.139803
150	0.184569	0.0427566	0.107914	0.141453
200	0.179425	0.0504081	0.103741	0.13929
300	0.174382	0.047393	0.102877	0.147216
400	0.175391	0.0520504	0.103353	**0.153824**
500	0.171659	0.0443621	0.0973479	0.153174
550	0.170449	**0.11301**	0.099802	0.133441
600	0.171357	0.0593728	0.0977992	0.151583

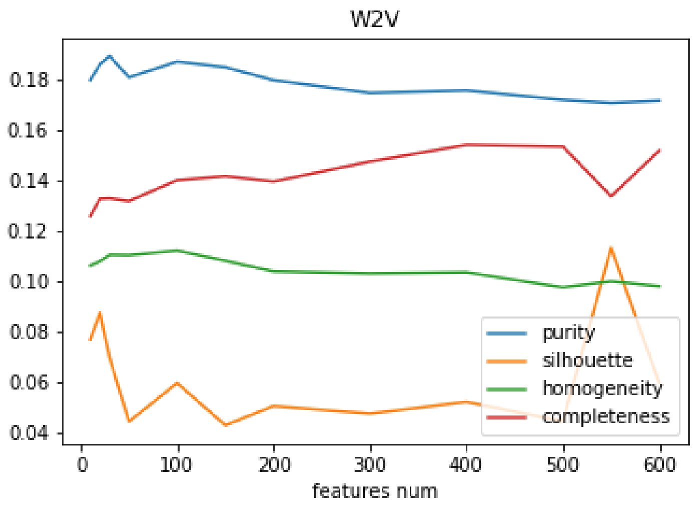


Figure 5: Word2Vec scores

Scores don't provide clear choice, so let's look at confusion matrixes of models with 30 (Figure 6) and 100 dimension vector (Figure 7) that have the highest scores for purity and homogeneity. As we can see from figures models tends to put most of the items into two general clusters: social-like (agree, bye, fallback, hello) and others. KMeans clusterization for W2V with 30 dimensions seems to have more separated groups, so it will be left for further selection.

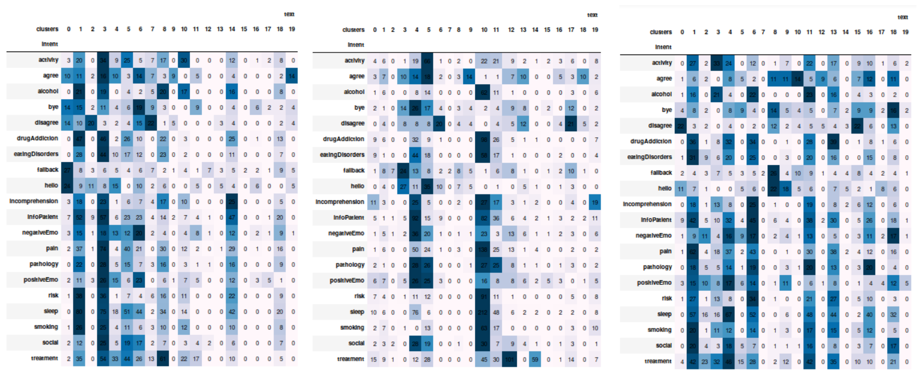


Figure 6: 30 dimension W2V confusion matrixes: agglomerative clustering, gaussian mixture, Kmeans

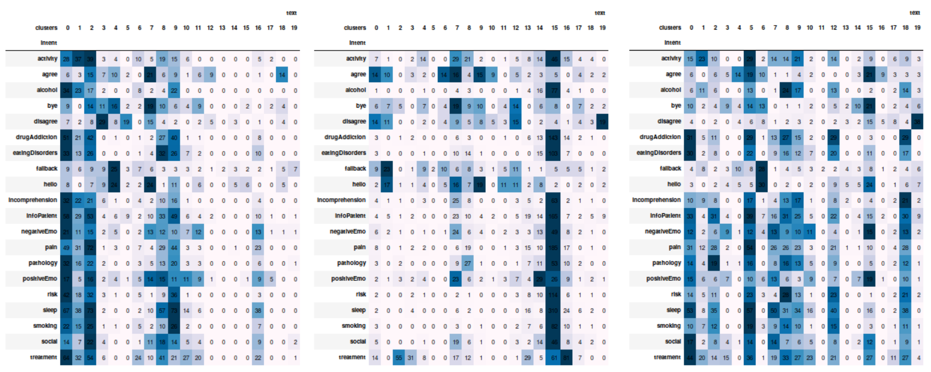


Figure 7: 100 dimension W2V confusion matrixes: agglomerative clustering, gaussian mixture, Kmeans

6.2.3 LDA model

For LDA representations, we iterated through different numbers of topics for LDA model to see which one should be used to represent our dataset. Table 3 shows the results.

num	purity	silhouette	homogeneity	completeness
10	0.253656	0.461589	0.157498	0.178928
20	0.259203	0.378276	0.168191	0.182203
30	0.251538	0.331069	0.157755	0.168448
50	0.240545	0.287032	0.147430	0.166824
100	0.266465	0.187596	0.168647	0.189221
150	0.256077	0.167888	0.161589	0.185054
200	0.263843	0.149212	0.175117	**0.194913**
300	0.250025	0.151526	0.165607	0.184314
400	**0.272718**	0.192777	**0.177547**	0.192270
500	0.259506	**0.203725**	0.174925	0.192831
550	0.229450	0.166200	0.149270	0.158548
600	0.254261	0.191729	0.171744	0.186001

Table 3: Comparison of LDA with different number of topics

The best model is LDA with 400 topics according to scores and with priority to purity and homogeneity ones as it's more important for us to have one intent in one cluster even if some intents will be present in several ones.

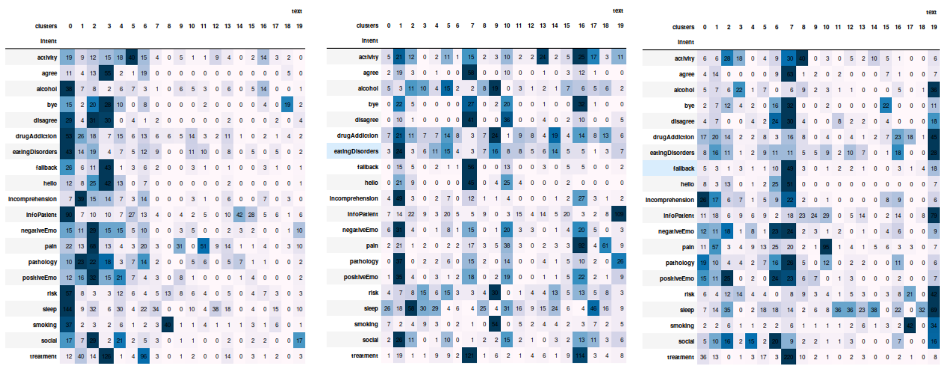


Figure 8: LDA confusion matrixes: agglomerative clustering, gaussian mixture, Kmeans

The confusion matrix between clusters and intent labels helps us make hypotheses about possible errors and difficulties and figure out what connections our model distinguishes well. Thus, although there are differences between the confusion matrices shown for the three clustering algorithms, they all show a reccurrent set of errors. In particular, the following intent are incorrectly clustered together: fallback and hello; alcohol, drugAddiction and risk; social and emotions.

6.2.4 BiLSTM by word

Following [CKL⁺18] we trained BiLSTM encoder model experimenting with different hyper-parameters such as the number of layers, the dimension of the word embedding and the type of pooling.

layers	purity	silhouette	homogeneity	completeness
1	**0.286369**	0.239979	**0.227069**	**0.232338**
2	0.262836	0.181804	0.211068	0.219279
3	0.238284	0.266460	0.195002	0.206703
4	0.218623	**0.439161**	0.164478	0.192599

Table 4: Comparison of models with different number of layers

So it's better to stay with one layer. Now for one layer, we experiment with different dimensions (layer size).

layers	purity	silhouette	homogeneity	completeness
32	0.270273	0.244798	0.214607	0.21474
64	**0.281683**	0.22728	0.231666	0.232176
128	0.271597	0.263519	**0.234111**	**0.235486**
256	0.270681	0.267634	0.229275	0.231675
512	0.277608	0.264844	0.231268	0.235075
1024	0.266707	0.260425	0.226291	0.229238
2048	0.268796	**0.278391**	0.228211	0.231041

Table 5: Dimension choosing

Finally, we chose 64 output size due to bigger difference in purity score.

The next step is to chose what encoder should we use: should the sentence representation be given by the last hidden layer of the LSTM, by mean or by max-pooling?

last layer	purity	silhouette	homogeneity	completeness
last hidden	0.281683	0.22728	0.231666	0.232176
mean hidden	0.255399	0.170045	0.194848	0.196056
max hidden	0.272616	0.177766	0.218339	0.219109

Table 6: Choosing pooling

Finally we got BiLSTM model with one 64-dimensional layer without pooling.

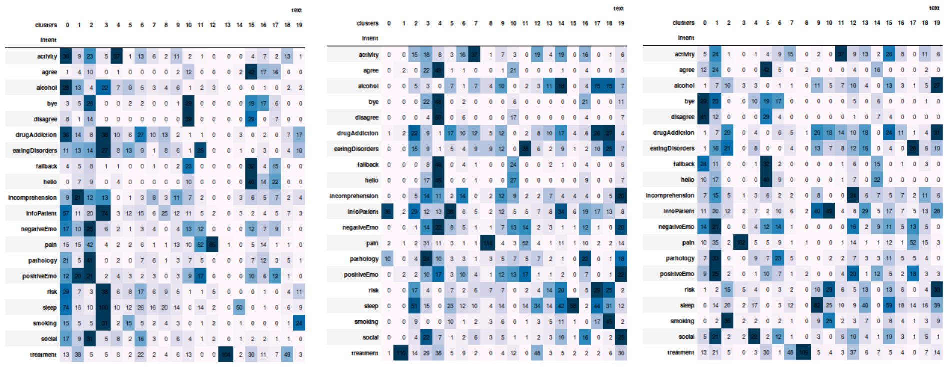


Figure 9: BiLSTM confusion matrixes: : agglomerative clustering, gaussian mixture, Kmeans

This model shows really good separatation for TREATMENT and PAIN labels but tends to group together utternance belonging to the AGREE, FALLBACK and HELLO intent on the one hand and to the BYE and DISAGREE intent on the other.

6.2.5 Overall comparison

Table 7 summarizes the results, focusing on the best hyper-parameters for each approach and taking into consideration pretrained GloVe and Google News Word2Vec embeddings. The last two models (LDA and BiLSTM) appear to provide the best results and will therefore be used for further steps.

model	features	purity	silhouette	homogeneity	completeness
Word2Vec	30	0.18901	0.0695026	0.110232	0.13267
GloVe	25	0.191125	0.003554	0.110058	0.147367
GloVe	200	0.186283	-0.026413	0.102103	0.137332
Google	300	0.201614	-0.016592	0.109450	0.156423
LDA	400	0.272718	0.192777	0.177547	0.192270
BiLSTM	64	**0.281683**	0.22728	0.231666	0.232176

Table 7: Overall comparison

6.3 Classifier

In the previous section, we compared different settings for building sentence representations using different methods (LDA, Word2Vec, Glove Embedding etc.) and experimenting with different hyper-parameters. We concluded that the best models were (i) LDA and (ii) biLSTM.

In this section, we now explore how these representations can be used to help guess the corresponding intents and further, how external (forum) data that is automatically labelled using the procedure described in Section 4.2 can help improve intent classification.

6.3.1 Evaluation metrics

A classical way to evaluate a classification model performance is using accuracy score. Sometimes, especially in case of unbalanced dataset, it can be misleading because it can represent that model learns just one class. Its better to look for a precision score as a measure of exactness and a recall score as a measure of completeness.

$$P = \frac{TP}{TP + FP} \tag{8}$$

$$R = \frac{TP}{TP + FN} \tag{9}$$

where TP - True Positives, FP - False Positives, FN - False Negatives.

If we want to get an idea how our classifier deals with both classes (in simple binary case) we should use F1 score that conveys the balance between the precision and the recall.

$$F1 = \frac{2 * P * R}{P + R} \tag{10}$$

There is a more clean and unambiguous way to describe the model behaviour that is a confusion matrix. The bigger values on the main diagonal the better as well as the more balanced small values on antidiagonal.

	Real positive	Real negative
Predicted positive	True Positive	False Positive
Predicted negative	False negative	True negative

Table 8: Confusion matrix template

6.3.2 Training on the Manually Labelled Dataset

The baseline classifier is trained and tested on the manually labelled data set using a 60/40 split ratio (65 items for training and 43 for testing). Both sets are balanced, so that each category is represented by the same number of items.

We use the SVC classification model and compare two kernels: rbf (C=10, gamma=1) and linear (C=10, gamma=0.001). For each model, we use grid search to identify the best parameters and we evaluate using 5 fold cross validation. Table 9 shows the results.

Model	Kernel	Train	Test	precision	recall	F1
LDA	rbf	0.524 + 0.00717	0.387 + 0.0341	0.308	0.316	0.287
	linear	0.371 + 0.00787	0.338 + 0.0431	0.238	0.241	0.215
BiLSTM	rbf	0.418 + 0.0193	0.373 + 0.0548	0.280	0.297	0.275
	linear	0.376 + 0.0147	0.343 + 0.0484	0.265	0.291	0.251

Table 9: CV score comparison

Although the rbf model model has higher scores, the linear one is less prone to overfitting as shown by the small difference between train and test errors. It was decided to try both and observe their behavior.

6.4 Semi Supervised Learning

We now report on the classifier results produced after each data expansion step (cf. Section 4.2).

On iteration 0, there is no data expansion and scores are provided for the baseline model trained on the manually labelled data.

For the first iteration, we created subset of external data with $N = 500$ sentences for each predicted intent, then splitted data into 200 clusters, a threshold for probability was chosen pretty low (0.12) in order to be able to extend all 20 categories. In result, we achieved 15 new items.

For the second iteration, we repeated the procedure choosing probability threshold as 0.1 and obtaining 13 new items.

BiLSTM model didn't succeded to provide expansion to all categories. It means that after labelling all sentences there was no candidates for some intents. So result only for LDA model are provided.

6.4.1 SVC with linear kernel

Iteration	Score	CV mean	CV std	precision	recall	F1
0	0.2651	0.337	0.0503	0.265	0.340	0.262
1	0.2895	0.323	0.0212	0.289	0.319	0.290
2	0.3174	0.333	0.0118	0.317	0.325	0.304

Table 10: Classifier scores comparison

label	precision				recall				f1		
activity	0.56	0.30	0.42		0.36	0.27	0.29		0.44	0.28	0.34
agree	0.23	0.16	0.05		0.19	0.28	0.12		0.21	0.21	0.07
alcohol	0.35	0.53	0.56		0.34	0.44	0.49		0.34	0.48	0.52
bye	0.14	0.33	0.53		0.24	0.41	0.38		0.18	0.36	0.44
disagree	0.14	0.14	0.37		0.40	0.08	0.38		0.21	0.10	0.38
drugAddiction	0.21	0.21	0.23		0.12	0.29	0.23		0.15	0.24	0.23
eatingDisorders	0.26	0.37	0.28		0.61	0.73	0.34		0.36	0.49	0.31
fallback	0.05	0.12	0.00		0.09	0.31	0.00		0.06	0.17	0.00
hello	0.05	0.00	0.21		0.17	0.00	0.43		0.07	0.00	0.28
incomprehension	0.49	0.37	0.35		0.20	0.26	0.20		0.29	0.31	0.25
infoPatient	0.33	0.49	0.49		0.25	0.34	0.44		0.29	0.40	0.46
negativeEmo	0.12	0.05	0.16		0.21	0.06	0.21		0.15	0.05	0.18
pain	0.70	0.60	0.70		0.81	0.46	0.81		0.75	0.53	0.75
pathology	0.02	0.05	0.05		0.50	0.09	0.29		0.04	0.06	0.08
positiveEmo	0.28	0.37	0.09		0.07	0.11	0.14		0.11	0.17	0.11
risk	0.02	0.12	0.07		0.08	0.16	0.11		0.04	0.13	0.08
sleep	0.44	0.42	0.49		0.47	0.50	0.54		0.46	0.46	0.51
smoking	0.60	0.44	0.49		0.68	0.59	0.66		0.64	0.51	0.56
social	0.14	0.14	0.26		0.20	0.21	0.31		0.16	0.17	0.28
treatment	0.19	0.58	0.56		0.80	0.75	0.15		0.30	0.67	0.24

Table 11: Comparison of classification scores for SVC with linear kernel

As we can see from Table 10 score on the gold test dataset, precision and F1 score steadily increasing, though values for CV score and recall are not stable. From Figure 10 we can see how confusion matrix became more diagonalized though some categories that were confusing from the very begining (agree, diasagree, fallback, hello, negativeEmo, positiveEmo, risk and alcohol). Coming back to section 6.2.3 we can see that these categories were considered to be in the same cluster, so on that early stage it was possible to predict such outcome. On the other hand such intents like social, incompresension, infoPatient, treatment became predicted more precise.

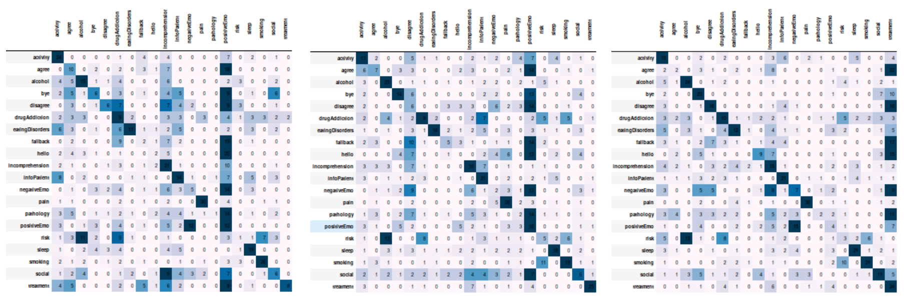


Figure 10: SVC confusion matrix for iteration 0, 1, 2.

6.4.2 SVC with rbf kernel

Zero iteration stands for the baseline model performance.

For the first iteration we created subset of external data with $N = 500$ sentences for each predicted intent, then splitted data into 1000 clusters, hreshold for probability was chosen pretty low (0.15) in order to be able to extend all 20 categories. In result, we achived 6 new items.

For the second iteration we used the same parameters, threshold for probability became lower(0.11) and in result 5 new items were obtained.

Here we provide grouped results about classifier scores chaging by iteration.

Iteration	Score	CV mean	CV std	precision	recall	F1
0	0.3186	0.371	0.0338	0.308	0.316	0.287
1	0.3523	0.350	0.0153	0.352	0.371	0.346
2	0.3167	0.332	0.0150	0.316	0.331	0.313

Table 12: Classifier scores comparison

label	precision				recall				f1		
	it 0	it 1	it 2		it 0	it 1	it 2		it 0	it 1	it 2
activity	0.44	0.49	0.37		0.40	0.54	0.34		0.42	0.51	0.36
agree	0.72	0.23	0.14		0.20	0.31	0.21		0.32	0.27	0.17
alcohol	0.51	0.56	0.47		0.44	0.41	0.39		0.47	0.47	0.43
bye	0.35	0.56	0.33		0.29	0.67	0.30		0.32	0.39	0.31
disagree	0.33	0.33	0.35		0.38	0.31	0.43		0.35	0.32	0.38
drugAddiction	0.09	0.26	0.28		0.13	0.39	0.29		0.11	0.31	0.28
eatingDisorders	0.33	0.47	0.33		0.45	0.67	0.25		0.38	0.52	0.28
fallback	0.00	0.26	0.14		0.00	0.12	0.12		0.00	0.17	0.13
hello	0.23	0.51	0.40		0.25	0.25	0.19		0.24	0.34	0.26
incomprehension	0.51	0.30	0.42		0.29	0.29	0.43		0.37	0.30	0.42
infoPatient	0.30	0.42	0.37		0.54	0.35	0.52		0.39	0.38	0.43
negativeEmo	0.12	0.05	0.14		0.19	0.06	0.18		0.14	0.05	0.16
pain	0.30	0.65	0.49		0.22	0.49	0.44		0.25	0.56	0.46
pathology	0.09	0.14	0.16		0.16	0.35	0.29		0.12	0.20	0.21
positiveEmo	0.09	0.14	0.16		0.17	0.20	0.20		0.12	0.16	0.18
risk	0.05	0.07	0.05		0.17	0.14	0.11		0.07	0.09	0.06
sleep	0.42	0.58	0.56		0.58	0.66	0.63		0.49	0.62	0.59
smoking	0.65	0.51	0.58		0.61	0.54	0.54		0.63	0.52	0.56
social	0.33	0.19	0.21		0.50	0.29	0.56		0.49	0.23	0.31
treatment	0.51	0.63	0.40		0.35	0.40	0.22		0.42	0.49	0.28

Table 13: Comparison of classification scores for SVC with rbf kernel

Though scores for the second iteration (Table 12) of data expansion falling down the confusion matrix anyway becoming more diagonalized (Figure 11) as well as F1 measure grows in comparision to the baseline. It's remarkable that such intents as drugAddiction, disagree succeded to gain precise just after first iteration. Some confusing groups like (agree, bye, diasgree, fallback, hello) or (positiveEmo and negativeEmo) or (risk and alcohol) still tends to stay together.

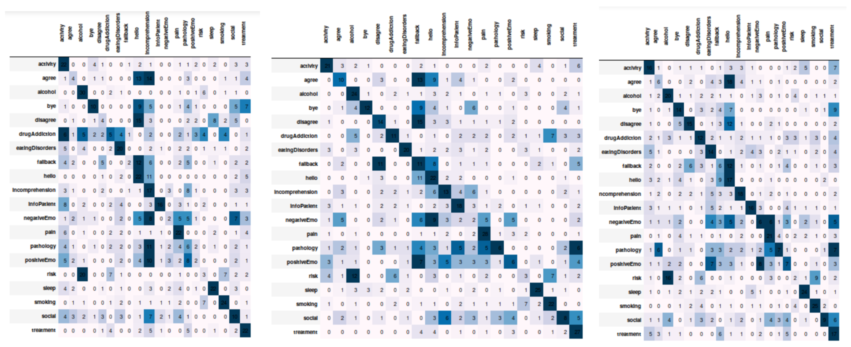


Figure 11: SVC confusion matrix for iteration 0, 1, 2.

7 Conclusion and future work

In this project we were dealing with task of semi-supervised text classification for expanding initial dataset. Our approch is to train a sentence representation model that will be able to provide clusters similar to initial classes. For this purpose we used LDA, W2V and BiLSTM models and evaluated them with the help of clustering scores. Later SVC and Kmeans models were trained to label external data and chose the representative examples to extend existing dataset and update models with them.

Our results shows how algorithm tends to converge and to be more precise though high rate of confusion between labels in intial classifier increases this confusion in futher data expansion. This confusion can be measured before going through semi-supervised routine in order to achive acceptable level or eliminate some intents. Also, the more data you have the higher chances to find more similar sentences to initial one so that possible context for each intent will grow gradually while diversified data can cause even more perplexity.

Future work will be devoted to improving word embedding model, trying one-shot and transfer learning model in order to improve classification score. Transfering from one-intent to multi-intent is preferable due to the fact that sentences are usually ambiguous than not. Another possible direction is developmenting a framework for adding new labels to the existing classifier.

References

[BK14] Purnima Bholowalia and Arvind Kumar. Article: Ebk-means: A clustering technique based on elbow method and k-means