Data expansion for semantic classification

Anna Liednikova, Claire Gardent

January 31, 2019

Contents

1 Introduction

This project was set within the framework of a collaboration with the ALIAE startup where the 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 three main modules:

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

In this project, we focus on the NLU step which consists in analysing the user utterance (does the user speaks about pain, physical activity etc.) and detecting at least one most probable intent of user message so that later the chatbot can choose the right strategy for fulfilling information.

ADD: ADD an example showing a user input and the expected representation ie intent + entity

A small manually annotated dataset (roughly 100 sentences for each category) was available. But that's not enough for learning a good NLU model and meet the variety of users possible inputs. Annotating sentence takes plenty of time, so some automated or semi-automated way should be chosen. So this project focuses on exploring automatic ways of extending the training data and learning NLU models from these extended data.

First, we do the literature review of approaches applied to similar problems. Later in the methodology part, we will cover four parts: building sentence representation model, evaluate it by clusterisation, train a classifier for labelling the data and application semi-supervised learning for extending the dataset. Next two our datasets and preprocessing routine will be described. Experiment part contains a quantitative and qualitative description of the models and final results. In the end, we will sum up the work have been done and give some assumption for future improvements.

2 Literature review

[?] MODIFIED: present an approach for automatically generating additional training data for event extraction systems. First, they trained a baseline classifier on available data. Then, they cluster external data to obtain cluster of paraphrases using the NewsSpike method introduced by [?]. 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.

3 Methodology

We follow [?] methodology. Instead of using [?]'s methods for identifying clusters of related sentences however, we explore different ways of representing sentences using deep learning approaches. We then apply clustering to the resulting sentence representations. Finally, we investigate the impact of the extended labeled data on classification.

3.1 Building Sentence Representations

We create sentence representations in two steps using machine and deep learning techniques. First, we map words to continuous representations. Second, we combine these word representations into sentence representations.

3.1.1 Word Representations

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

LDA. One of the most common methods for constructing thematic models is the Latent Dirichlet Allocation (LDA) [?]. This algorithm is based on the Dirichlet prior distribution and uses in its work the 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 allows you to detect implicit relationships between words based on polysemy. The LDA method performs soft clustering and assumes that each word in the document is generated by some latent topic, which is determined by a probability distribution on the set of all words of the text.

Finally each document with distribution vector over the topics while each topic is characterized by a distribution over words. For example, a shortcut for a topic is following

```
[(0.13819554, 'sleep'),
  (0.10817484, 'morning'),
  (0.10187448, 'sometimes'),
  (0.09861754,  'help'),
  (0.060275868,  'able'),
  (0.060111083,  'nap'),
...],
```

Word2Vec. This technology works in the following manner: word2vec takes a large text corpus as input and matches each vector with a word, producing the coordinates of the output words. First, he creates a dictionary, learning on the input text data, and then calculates a vector representation of the words. A vector representation is based on contextual intimacy: words that appear in the text next to identical words (and therefore, have a similar meaning) will have close coordinates of word vectors in a vector representation. The resulting word vectors can be used for natural language processing and machine learning.

$https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf$

ADD: example of w2v representation

Preprocessed tokenized sentences were used to create dictionary and were passed to W2V model as it is and in form of Bag-Of-Word to LDA model.

For selecting the best model we changed space dimension (topics for LDA and features for W2W). Preferred number of features for W2V model is 200. It's the start point for our model.

The good point of using this models is that they can be used for both word and sentence representation since one word can be treated as a small sentence.

3.1.2 Sentence Representations

We construct sentence representations out of the word representations described in the previous section using two types of neural networks: PositionwiseFeedForward [**?**] and BiLSTM ADD: bibref.

Here you need to explain the theory behind it. Explain how LSTM and the PositionwiseFeedForward build a sentence representation out of word embeddings.

Bi-LSTM Representations. 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 (Conneau et al., 2018)

3.2 Clustering Sentences

Using the sentence representations described in the previous section, we apply clustering to group together sentences that are similar. We compare three clustering algorithms:

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

We set the number of clusters to the number of intents (20) which allows using not only purity and Silhouette coefficients to evaluate the clusters but also homogeneity and completeness to get the idea how resulting groups 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.

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 barytsentera 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. 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.

$Ward, JoeH.(1963)."HierarchicalGroupingtoOptimizeanObjectiveFunction".JournaloftheAme$
$236244.doi : 10.2307/2282967.JSTOR2282967.MR0148188.$

Gaussian Mixture. REWRITE: In statistics, a mixture model is a probabilistic model for representing the presence of subpopulations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs. Formally a mixture model corresponds to the mixture distribution that represents the probability distribution of observations in the overall population without sub-population identity information.

A Gaussian mixture model 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.

$https://pdfs.semanticscholar.org/734b/07b53c23f74a3b004d7fe341ae4fce462fc6.pdf$

3.3 Classifying Sentences

After choosing the best model for word embedding we train classifier to label data with intents. The most common approach in similar task is to use SVC. Also we test another classical approach for classification is Random Forest Classifier. So, we create balanced train and test dataset and validate model by cross validation score.

SVC. 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.

Random Forest is an ensemble of Decision trees that comes to substitute a big and complex one. The weaknesses of Decision tree model is bad generalization and overfitting on small amounts on data. It became possible to deal with it by training every tree on a random subset of data and features in order to obtain decorrelated ones. The final prediction is provided as a voting result of the whole ensemble that helps to reduce the variance of the model.

3.4 Semi-supervised approach

First, we find the best clusterization for the forum data, later we classify each item and omit ones with low probability of elements at threshold Theta. In result some clusters will be empty from the beginning and some of them will be ommited due to small population.

Elements of clusters that are left will be labelled by the majority. Usually, they are pure. Later we select N labelled items for each category and extend train data with them to retrain word embedding and classificatin 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.

4 Datasets

4.1 Labelled Data

The initial dataset is created manually covering 20 main possible users intents. Each sentence represents one intent. For, example,

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 1. 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).

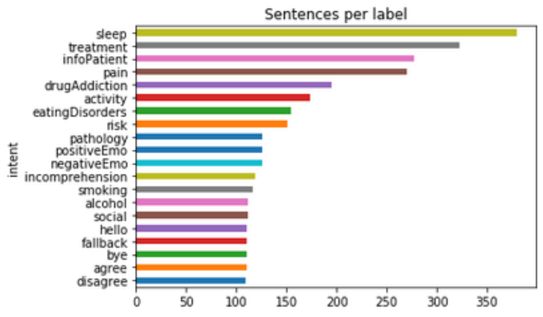


Figure 1: Label distribution in dataset

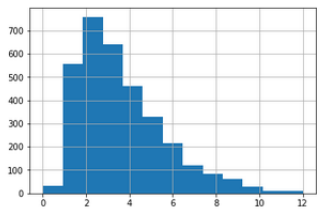


Figure 2: Sentence dist

4.2 Preprocessing routine

The textual data was segmented into sentences, tokenized and lemmatized using NLP libraries. Stop words were removed.

We compared two libraries, NLTK and SpaCy.

For example, in word tokenization they give different results that can influence not only simple statistics but meaning too. Some example of different tokenization can be seen in table 1. Though concating words to ones like 'flulike' or '35mg' or 'longterm' sometimes gives more robust and concrete meaning in case of big dataset, in our case it seems better to stay with SpaCy way of tokenization in order to have smaller and more simple vocabualary.

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 1: Tokenization comparison

For stopwords removing 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 like positive or negative emotion, social. Finally nltk one was selecting because of containing shorts like 'm' from 'am', 've' from 'have'. Final dictionary contained 1882 words. Also all numbers were changed to num. Chart (3) looks fine.

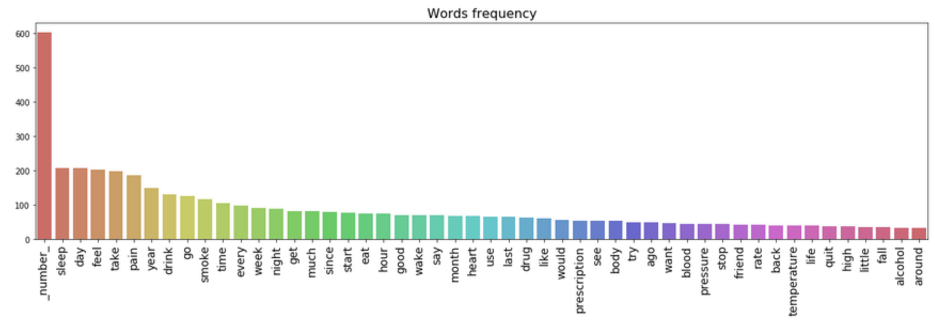


Figure 3: Words frequency

Next problem we faced is empty sentences. But there is not a lot of occurances, so their dropping don't change the dataset much.

intent	items
fallback	12
disagree	11
hello	4
agree	3
incomprehension	2
positiveEmo	1

These sentences are: 'what?', 'same that again', 'I am up', 'same', 'can you', 'do that', 'do this', 'Will do', 'no', 'no i will not', "No i don't", 'no', 'no i will not', "No i don't", "What's up?", "What's up?", 'he', 'a', 'd', 'i', 'm', 'o', 's', 't', 'y', 'I will', 'Will do', 'No', 'no', 'No it is not', "No I don't", 'No', "I'm here"

4.3 Forum data

We scraped 272,553 unique posts contained in each category.

To improve initial classifier we needed to extend dataset by much more data and more naturally constracted one. No test set was available at the beginning of the project. To solve this problem, we opted to created our own test set following Zhang et al (2015) and Sondhi et al (2010) example.

HealthBoards **??** is a medical forum web portal that allows patients to discuss their ailments. We scraped 272553 unique posts contained in each of 238 categories. Figure 4 show the dataset statistics. In average sentence contains 11+-7 words.

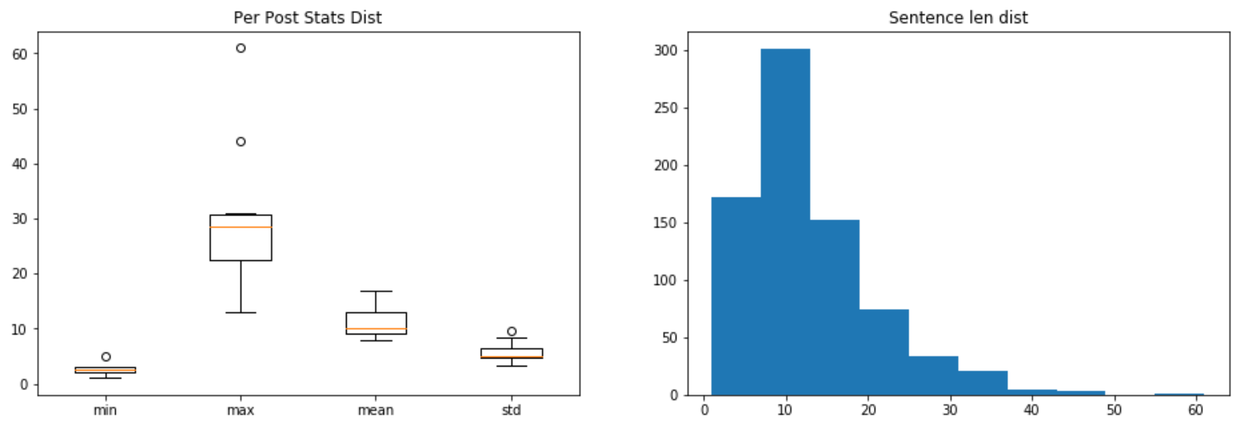


Figure 4: Text stat

Because our intents are not that precise as forum ones we should select particular boards with relevant information to be able gradually expand their context. Result is in table 2 We should start with the data that is most similar to the initial one, so we choose categories of boards that are similar to our intents.

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 2: Selected categories

Finally, the corpus consists of N sentences.

Also, data should be divided in subsets by increasing sentence length because of the difference in mean values for both datasets. So after tokenizing posts into sentences we calculate it's length. For the each iteration we should leave sentences with mean+std words in cleaned text.

5 Experiment

5.1 Clustering

5.1.1 Evoluation metrics

For every model following metrics and their average were calculated: purity score, Silhouette Coefficient, homogeneity and completeness scores. Later for each parameter average among clustering models calculated for each score in order to get both table and plot.

Also for each model confusion matrix created between cluster labels and initial 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.

5.1.2 W2V model

Comparison table

features num	purity	silhouette	homogeneity	completeness
10	0.183157	0.0680834	0.108901	0.12661
20	0.182854	0.0506986	0.111099	0.124805
30	**0.185477**	0.0683824	0.110799	0.132563
50	0.184569	0.0647393	**0.111876**	0.131964
100	0.182148	**0.0813837**	0.108022	0.138329
150	0.183661	0.0298524	0.106401	0.141179
200	0.176803	0.0581627	0.103708	0.141973
300	0.176097	0.0432732	0.106379	0.151118
400	0.169642	0.0626822	0.100455	0.148768
500	0.168533	0.0653733	0.0976706	0.146799
550	0.17176	0.0584935	0.0994422	**0.151185**
600	0.169138	0.0574187	0.0980455	0.150695

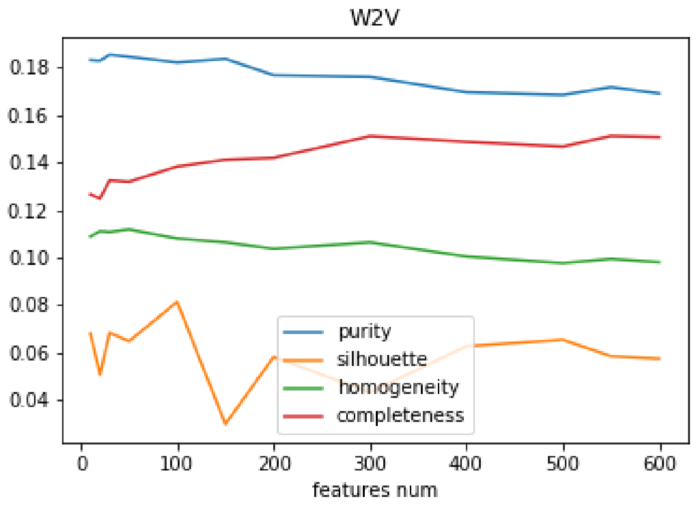
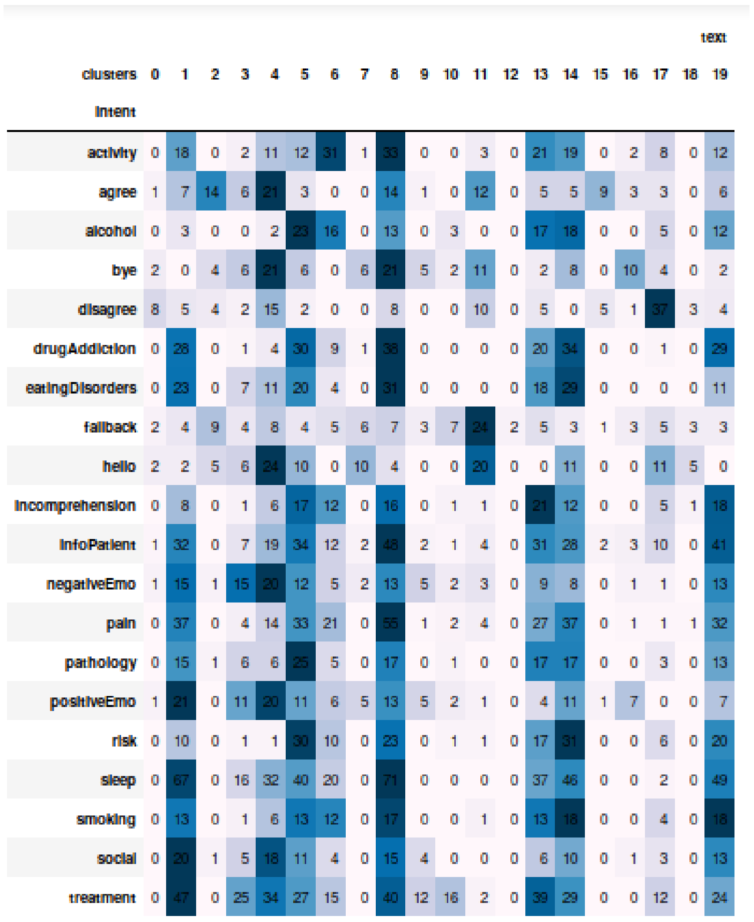


Figure 5: LDA scores

Later we will try both 10 and 100 features.



	text																			
clusters	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
intent																				
activity	0	18	0	2	11	12	31	1	33	0	0	3	0	21	19	0	2	8	0	12
agree	1	7	14	6	21	3	0	0	14	1	0	12	0	5	5	9	3	3	0	6
alcohol	0	3	0	0	2	23	16	0	13	0	3	0	0	17	18	0	0	5	0	12
bye	2	0	4	6	21	6	0	6	21	5	2	11	0	2	8	0	10	4	0	2
disagree	8	5	4	2	15	2	0	0	8	0	0	10	0	5	0	5	1	37	3	4
drugAddiction	0	28	0	1	4	30	9	1	38	0	0	0	0	20	34	0	0	1	0	29
eatingDisorders	0	23	0	7	11	20	4	0	31	0	0	0	0	18	29	0	0	0	0	11
fallback	2	4	9	4	8	4	5	6	7	3	7	24	2	5	3	1	3	5	3	3
hello	2	2	5	6	24	10	0	10	4	0	0	20	0	0	11	0	0	11	5	0
incomprehension	0	8	0	1	6	17	12	0	16	0	1	1	0	21	12	0	0	5	1	18
infoPatient	1	32	0	7	19	34	12	2	48	2	1	4	0	31	28	2	3	10	0	41
negativeEmo	1	15	1	15	20	12	5	2	13	5	2	3	0	9	8	0	1	1	0	13
pain	0	37	0	4	14	33	21	0	55	1	2	4	0	27	37	0	1	1	1	32
pathology	0	15	1	6	6	25	5	0	17	0	1	0	0	17	17	0	0	3	0	13
positiveEmo	1	21	0	11	20	11	6	5	13	5	2	1	0	4	11	1	7	0	0	7
risk	0	10	0	1	1	30	10	0	23	0	1	1	0	17	31	0	0	6	0	20
sleep	0	67	0	16	32	40	20	0	71	0	0	0	0	37	46	0	0	2	0	49
smoking	0	13	0	1	6	13	12	0	17	0	0	1	0	13	18	0	0	4	0	18
social	0	20	1	5	18	11	4	0	15	4	0	0	0	6	10	0	1	3	0	13
treatment	0	47	0	25	34	27	15	0	40	12	16	2	0	39	29	0	0	12	0	24

Figure 6: W2V confusion matrix

5.1.3 LDA model

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	0.16620016	0.149270	0.158548
600	0.254261	0.191729	0.171744	0.186001

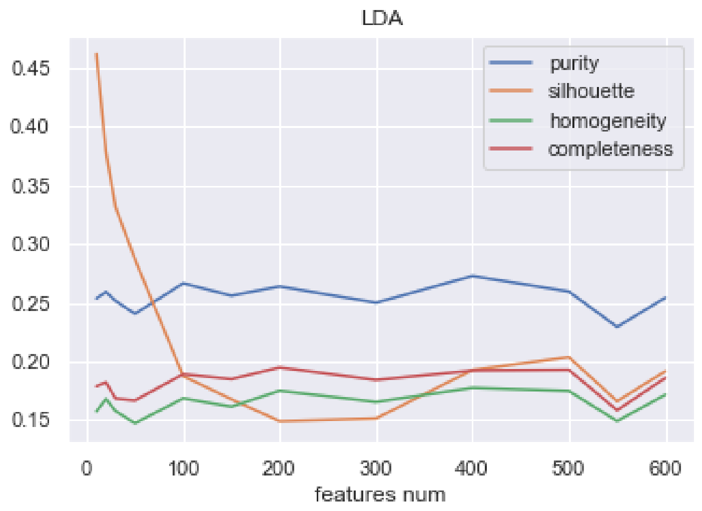
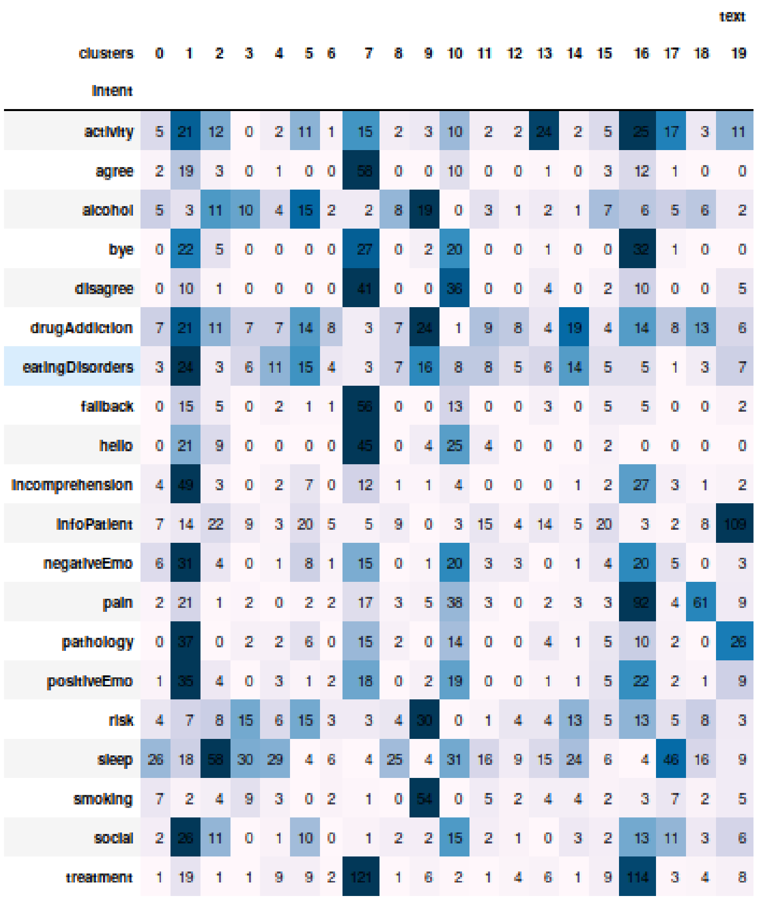


Figure 7: LDA scores

The best model - is LDA with 400 topics according to scores and with priority to purity and homogeneity ones.

Thanks to confusion matrix that is constructed based on cluster and intent labels we can make assumpsions about possible errors and difficulties and figure out what connections our model distinguish well. Though different clustering algorithms still captures different connection between intents, there are some common groups for all of them: fallback and hello; alcohol, drugAddiction and risk; social and emotions.



																				text
clusters	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
intent																				
activity	5	21	12	0	2	11	1	15	2	3	10	2	2	24	2	5	25	17	3	11
agree	2	19	3	0	1	0	0	58	0	0	10	0	0	1	0	3	12	1	0	0
alcohol	5	3	11	10	4	15	2	2	8	19	0	3	1	2	1	7	6	5	6	2
bye	0	22	5	0	0	0	0	27	0	2	20	0	0	1	0	0	32	1	0	0
disagree	0	10	1	0	0	0	0	41	0	0	36	0	0	4	0	2	10	0	0	5
drugAddiction	7	21	11	7	7	14	8	3	7	24	1	9	8	4	19	4	14	8	13	6
eatingDisorders	3	24	3	6	11	15	4	3	7	16	8	8	5	6	14	5	5	1	3	7
fallback	0	15	5	0	2	1	1	56	0	0	13	0	0	3	0	5	5	0	0	2
hello	0	21	9	0	0	0	0	45	0	4	25	4	0	0	0	2	0	0	0	0
incomprehension	4	49	3	0	2	7	0	12	1	1	4	0	0	0	1	2	27	3	1	2
infoPatient	7	14	22	9	3	20	5	5	9	0	3	15	4	14	5	20	3	2	8	109
negativeEmo	6	31	4	0	1	8	1	15	0	1	20	3	3	0	1	4	20	5	0	3
pain	2	21	1	2	0	2	2	17	3	5	38	3	0	2	3	3	92	4	61	9
pathology	0	37	0	2	2	6	0	15	2	0	14	0	0	4	1	5	10	2	0	26
positiveEmo	1	35	4	0	3	1	2	18	0	2	19	0	0	1	1	5	22	2	1	9
risk	4	7	8	15	6	15	3	3	4	30	0	1	4	4	13	5	13	5	8	3
sleep	26	18	58	30	29	4	6	4	25	4	31	16	9	15	24	6	4	46	16	9
smoking	7	2	4	9	3	0	2	1	0	54	0	5	2	4	4	2	3	7	2	5
social	2	26	11	0	1	10	0	1	2	2	15	2	1	0	3	2	13	11	3	6
treatment	1	19	1	1	9	9	2	121	1	6	2	1	4	6	1	9	114	3	4	8

Figure 8: LDA confusion matrix for gaussian mixture

5.1.4 GloVe pretrained

5.1.5 Google News W2V pretrained

We used word2vec google-news model with 300 dimensional feature space.

CV 0.418 + 0.057

5.1.6 CNN by word

From simple encoder, w2v, lda, w2v + lda

5.1.7 BiLSTM by word

From simple encoder, w2v, lda, w2v + lda

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 3: Layer choosing

So it's better to stay with one layer.

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 4: Dimension choosing

64 output size due to bigger difference in purity score
With maxpool on hidden layer

layers	purity	silhouette	homogeneity	completeness
64	0.272616	0.177766	0.218339	0.219109

Table 5: With maxpool on hidden layer

With maxpool on output

layers	purity	silhouette	homogeneity	completeness
64	0.256011	0.183609	0.200849	0.20462

Table 6: With maxpool on output

BiLSTM model with one 64-dimensional layer without pooling.

5.1.8 Overall comparison

model	features	purity	silhouette	homogeneity	completeness
W2V	10	0.192032	0.077404	0.108761	0.127672
W2V	100	0.184266	0.065381	0.109944	0.144074
LDA	400	0.272718	0.192777	0.177547	0.192270
BiLSTM	64	**0.281683**	0.22728	0.231666	0.232176

Table 7: Overall comparision

5.2 Classifier

5.2.1 Evoluation 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.

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

Another approach is counting AUC (Area Under the Curve) there the curve is the ROC one that shows trade-off between precision and recall having False Positive Rate on X-axis and True Positive Rate (Recall /Sensitivity) in Y-axis. It represents the capability of the model to distinguish classes.

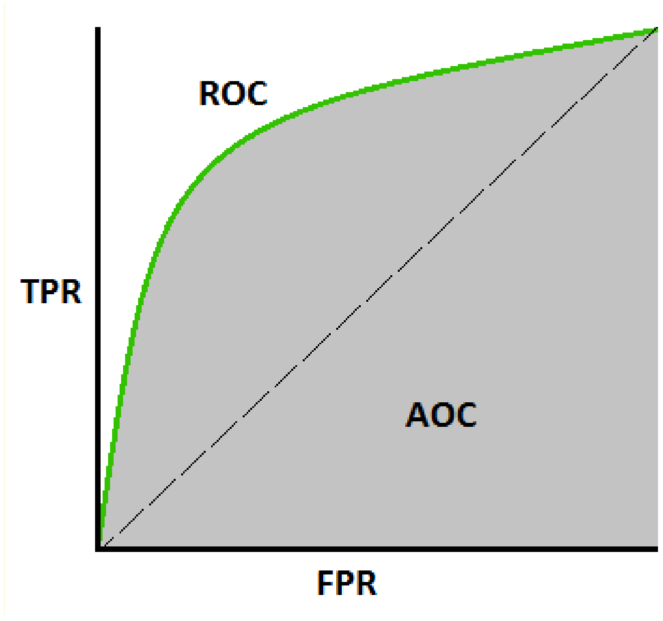


Figure 9: AUC-ROC curve

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

5.2.2 Results for each model

5.3 Semi-supervised learning

6 Conclusion

7 References

WardJ.H.Hierarchicalgroupingtooptimizeanobjectivefunction//J.oftheAmericanStatisticalAssoci