Supervised Neural Clustering via Latent Structured Output Learning: Application to Question Intents

Iryna Haponchyk
DISI, University of Trento
Povo (TN), Italy
gaponchik.irina@gmail.com

Alessandro Moschitti
Amazon Alexa AI
Manhattan Beach, CA, USA
amosch@amazon.com

Abstract

Previous pre-neural work on structured prediction has produced very effective supervised clustering algorithms using linear classifiers, e.g., structured SVM or perceptron. However, these cannot exploit the representation learning ability of neural networks, which would make supervised clustering even more powerful, i.e., general clustering patterns can be learned automatically. In this paper, we design neural networks based on latent structured prediction loss and Transformer models to approach supervised clustering. We tested our methods on the task of automatically recreating categories of intents from publicly available question intent corpora. The results show that our approach delivers 95.65% of F1, outperforming the state of the art by 17.24%.

1 Introduction

Recent years have witnessed a vast spread of virtual assistants, such as Google Home, Siri and Alexa, which are based on the research areas of Conversational Agents and Question Answering. When designing such systems, the creation of classes of expected questions, aka intents, is essential for building the main states of a dialog manager. In particular, when an assistant is designed for a specific domain, a knowledge engineer needs to analyze typical user's questions, answered by human operators. This work would be greatly sped up, if the engineers could have questions clustered according to the different topics they ask for. For example, the following questions/requests from the intent dataset by Larson et al. (2019):

i want to switch to direct deposit
set up direct deposit for me
how do i go about setting up direct deposit

all have a common intent of making a *direct deposit*. Thus, the dialog designer will create this intent, if the cluster captures a large number of requests.

However, for being effective, the clustering algorithm must demonstrate a sufficient accuracy, which is often not the case for completely unsupervised methods. Thus, supervised clustering (Finley and Joachims, 2005), which exploits some training data of the target domain, e.g., previously designed clusters, to discover new clusters, is a viable approach. A seminal work on structured prediction was Latent Structural Support Vector Machines (LSSVM) by Yu and Joachims (2009). Recently, Haponchyk et al. (2018) have shown that LSSVM as well as the Latent Structured Perceptron (LSP) by Fernandes et al. (2014), originally designed for coreference resolution, were also effective, when provided with the appropriate node similarity function, for clustering questions into intents. These approaches used traditional feature engineering (question similarity) and a linear classifier, i.e., SVM, which can be highly improved by neural networks, and pre-trained Transformers, e.g., Devlin et al. (2019). Indeed, neural models enable representation learning, which can amplify the generalization ability of supervised clustering algorithms.

In this paper, we design neural supervised clustering (NSC) models using the structured prediction algorithms, LSSVM and LSP. These are based on a latent representation of clusters using graph structures, which are used to compute an augmented loss. The latter, in turn, is used together with the model score to globally select the max-violating constraint at each learning step. This is the clustering that maximally penalizes the current model, which is used for a model update. We apply the same idea by computing the margin loss for our neural model and then back-propagating it, as any other differentiable loss. The augmented loss does not depend on the neural model, thus our approach can be applied to arbitrary learning settings.

We applied NSC to two different question intent clustering tasks, defined by two datasets: IC&OOS (Larson et al., 2019), which is an intent classification corpus, and Quora Intent Corpus (Haponchyk et al., 2018). Another interesting contribution of our work is the creation of a training set for clustering from IC&OOS, which enables an effective training of NSC. Our corpus and software are available to the research community[1].

The comparative results of NSC using traditional CNN networks and Transformer models against traditional methods, e.g., spectral clustering, show an impressive boost in F1 of our NSC-BERT model, e.g., 95.65% vs. 78.38%, more than 17% of improvement over the best spectral clustering method. This accuracy enables the use of our approach for dialog applications and opens up further directions for other clustering tasks.

2 Related Work

This paper touches two main research areas: structured prediction, in particular with neural models, and intent clustering, which are described below.

Structured prediction has shown powerful machine learning algorithms for solving NLP tasks requiring complex output, e.g., syntactic parsing (Smith, 2011), coreference resolution (Yu and Joachims, 2009; Fernandes et al., 2014). This work has mainly regarded traditional frameworks, e.g., SVMs, CRF, perceptron. Only little work has been devoted to the integration of the above theory in neural networks (LeCun et al., 2006; Durrett and Klein, 2015; Weiss et al., 2015; Kiperwasser and Goldberg, 2016; Peng et al., 2018; Milidiú and Rocha, 2018; Xu et al., 2018; Wang et al., 2019), and, to the best of our knowledge, none to supervised clustering.

This is partially due to the fact that local solutions have usually produced optimal results. For example, in case of supervised clustering, it is difficult to design a loss function that captures the global information about the clusters. Work in neural coreference resolution, e.g., (Lee et al., 2017), uses simple losses, which deliver state-of-the-art results but do not strictly take into account the cluster structure. Secondly, this is also due to the complexity associated with adapting the methods from previous work to neural frameworks. For example, using ILP (Roth and Yih, 2004) for clustering inference in SPIGOT (Peng et al., 2018), which facilitates the backpropagation through argmax based on a projection onto the feasible set of structured outputs, would inevitably require reducing the computational overhead (Miyauchi et al., 2018).

On the line of research of question clustering, Wen et al. (2001) proposed to cluster queries with respect to a group of web pages frequently selected by users. Deepak (2016) describes a k-means like algorithm, MiXKmeans, that can cluster threads in Community Question Answering websites. These methods are unsupervised and, thus, are likely sensitive to setting the optimal number of clusters or to a heuristic adopted for the clustering criterion.

Also among the classification approaches, there are semi-supervised and mixed classification methods which advance on the use of vast amounts of unlabelled queries. Li et al. (2008) classify unlabeled queries using their proximity to labeled queries in a click graph. Beitzel et al. (2007) classify queries from logs into topics using supervised or unsupervised methods.

The following classification approaches address new emerging intents. Lin and Xu (2019) enable a neural model to detect unknown intents as outliers using a novelty detection technique. This model, however, does not have the capability to distinguish between different unknown intents. Xia et al. (2018) devise a capsule neural network able to discriminate between different emerging intents. Its zero-shot learning ability critically depends on the definition of a similarity between existing and new intents. Our approach does not hold any explicit representation of intents.

The recent work by Lin et al. (2020) proposes a deep intent clustering model which takes advantage of labeled data for discovering new user intents, but it requires the indication of the exact number of output clusters. Finally, Zhang et al. (2021) propose Deep Aligned Clustering, a semisupervised method, to discover new intents using limited knowledge over intent data. We believe their approach is completely compatible with ours, i.e., our supervised clustering models can be integrated in their approach to improve intent discovering.

3 Structured Output for Clustering

LSSVMs train a clustering function from a series of training examples $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, where $\mathbf{x}_i$ are input sets of elements, and $\mathbf{y}_i$ are structured outputs, i.e., gold clusters. This function applied to unseen elements $\mathbf{x}$ predicts their clusters $\mathbf{y}$.

3.1 Cluster inference

The clustering $\mathbf{y}$ of a set $\mathbf{x}$ is inferred over a fully-connected graph G, which nodes represent the el-

[1] https://github.com/iKernels/intent-qa

ements x_k of $\mathbf{x}$, and edges $\mathbf{e} = (x_i, x_j)$ – the pairwise links between them. The inference step consists in finding a maximum spanning forest $\mathbf{h}$ on G, e.g., using Kruskal's algorithm (Kruskal, 1956). The nodes appearing in the same connected component (tree) in $\mathbf{h}$ are placed together in the same cluster in $\mathbf{y}$ (deterministically obtained from $\mathbf{h}$).

3.2 Learning

The approach learns a linear scoring function which decomposes over the edges of $\mathbf{h}$:

$$s_{\mathbf{w}}(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \sum_{\mathbf{e}=(x_i,x_j)\in\mathbf{h}} \mathbf{w} \cdot \phi(\mathbf{e}), \qquad (1)$$

where $\phi(\mathbf{e})$ is a feature representation for edge $\mathbf{e}$, describing a pair of elements of $\mathbf{x}$. Graph structures $\mathbf{h}$ are incorporated as latent variables into the latent formulation of LSSVM. Haponchyk et al. (2018) adapt the latent structured perceptron (LSP) by Fernandes et al. (2014) to the graph structures $\mathbf{h}$ and apply the approach to question pairs, (q_i, q_j), to cluster sets of questions into different user intents; we compare to these methods.

4 Neural Structured Output Clustering

We propose a model for optimizing a structural clustering loss with neural networks.

4.1 Global max-margin objective

As a standard practice in structured prediction, our goal is to train a model with a scoring function s_θ such that the correct clustering $\mathbf{y}$ is scored higher than incorrect clusterings $\hat{\mathbf{y}}$. LSSVM optimizes an upper bound, E, on the structural loss Δ, which, in general terms, can be rewritten using the parameters θ as:

$$\begin{aligned}\Delta(\mathbf{y}, \hat{\mathbf{y}}(\theta), \hat{\mathbf{h}}(\theta)) \leq E(\mathbf{y}, \hat{\mathbf{y}}(\theta), \hat{\mathbf{h}}(\theta)) = \\ \max_{(\hat{\mathbf{y}},\hat{\mathbf{h}})\in Y\times H} [\Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}}) + s_\theta(\mathbf{x}, \hat{\mathbf{y}}, \hat{\mathbf{h}})] - \\ \max_{\mathbf{h}\in H} s_\theta(\mathbf{x}, \mathbf{y}, \mathbf{h}),\end{aligned} \qquad (2)$$

where $\hat{\mathbf{y}}(\theta)$ is an output of the model with its auxiliary latent structure $\hat{\mathbf{h}}(\theta)$; Y and H are the spaces of all possible clusters and latent trees; $\Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}})$ is a standard structural loss, measuring the difference between the gold $\mathbf{y}$ and the output $\hat{\mathbf{y}}$ clusters; and $(\hat{\mathbf{y}}(\theta), \hat{\mathbf{h}}(\theta)) = \underset{(\mathbf{y},\mathbf{h})\in Y\times H}{\operatorname{argmax}}\ s_\theta(\mathbf{x}, \mathbf{y}, \mathbf{h})$.

The right-hand side of Eq. 2 is essentially a margin-based objective with margin rescaled by the loss Δ. Its minimization forces maximum weighted incorrect $\hat{\mathbf{h}}$ score lower than the maximum weighted correct $\mathbf{h}$ by at least the value of the loss $\Delta(\mathbf{y}, \hat{\mathbf{y}}, \hat{\mathbf{h}})$ on that $\hat{\mathbf{h}}$ under θ parameters. Our neural model optimizes the objective E defined in Eq. 2. We use the loss Δ of Yu and Joachims (2009) based on computing edge mistakes in $\hat{\mathbf{h}}$, in which negative edge penalties are scaled with an r-parameter.

4.2 Differentiable scoring function

For optimizing E in Eq. 2, we define a scoring function that decomposes over the edges of $\mathbf{h}$:

$$s_\theta(\mathbf{x}, \mathbf{y}, \mathbf{h}) = \sum_{\mathbf{e}=(x_i,x_j)\in\mathbf{h}} net_\theta(\mathbf{e}). \qquad (3)$$

This enables the inference by Kruskal's algorithm, where the network net_θ activates on edge representations[2]. Eq. 3 indicates that our approach is applicable to any network net_θ.

Our work is inspired by Kiperwasser and Goldberg (2016), who pass the arcs of dependency parses through a multi-layer perceptron and optimize a structured margin loss. Differently to them, we elaborate on the case of the margin rescaled with the structural loss, which includes max-violating inference. The objective E in Eq. 2 is sub-differentiable as a summation of *edge networks*; Δ in E does not depend on θ. We propagate the error from the margin loss E in Eq. 2 back to input layer of net_θ.

One iteration of the algorithm operates on one sample of training data $(\mathbf{x}, \mathbf{y})$, where, in the context of intent task, $\mathbf{x} = \{x_i\}$ is a set of questions, $\mathbf{y}$ are gold clusters of the questions in $\mathbf{x}$. We pass all the pairs of questions, i.e., edges $\mathbf{e} = (x_i, x_j), i < j$ of a fully connected graph G, through net_θ, and compute the global error E. The error computation includes finding (i) the max-violating graph, $\hat{\mathbf{h}}$, among all possible spanning graphs $\mathbf{h}$ of G; and (ii) the max-scoring correct spanning graph, $\mathbf{h}^*$, over the set of graphs that comply with the gold label $\mathbf{y}$. If $E > 0$, the backward pass of the model computes gradients for an update of the model. The partial derivatives with respect to the parameters θ_j in the last layer of the network are

$$\frac{\partial E}{\partial \theta_j} = \sum_{\mathbf{e}\in\hat{\mathbf{h}}} \frac{\partial\, net_\theta(\mathbf{e})}{\partial \theta_j} - \sum_{\mathbf{e}\in\mathbf{h}^*} \frac{\partial\, net_\theta(\mathbf{e})}{\partial \theta_j},$$

[2] The scoring function follows the standard formulation of structured prediction tasks, where the score of a structure is computed by aggregating the scores of its constituent parts. In our case, it is a summation of edge scores. The reader may refer to the work on dependency parsing by Kiperwasser and Goldberg (2016).

and so on down to the input nodes of net_θ following the chain rule.

5 Our Baseline Network

We intentionally do not specify the architecture of net_θ in Sec. 4.2 as it could be of any form once encoding a pair of questions (x_i, x_j). However, we further describe the architecture with which we experiment in this work.

We use a simple feedforward neural network, which consists of (i) an input layer encoding a pair (x_i, x_j), (ii) one fully connected hidden layer with ReLU activation functions, and (iii) an output layer, which is a linear operation over the outputs of the hidden layer. This way, for edge $\mathbf{e}$, $net_\theta(\mathbf{e})$ is a real number without any restriction on its range.

The pairwise encoder is practically trained to score good edges higher than bad edges. However, doing it jointly for all the edges over a sample, in a structured way, has the goal of producing a more consistent decision in terms of clustering.

5.1 Input layer

We consider two ways for representing question pairs $\phi(\mathbf{e}) = \phi(x_i, x_j) \in \mathbb{R}^d$, using embeddings:

(1) We use a sentence encoder (Severyn and Moschitti, 2016) to map each question x_i into a fixed-size intermediate vector representation $\psi(x_i)$. The encoder operates on a sentence matrix S, in which the k-th column corresponds to the k-th word in x_i and is a concatenation of the word embedding and overlap embedding: $S_k = [word\_emb(w_k), ov\_emb(w_k)]$. The ov_emb part for x_i, in each pair, is formed in association with the other question of the pair, x_j. S is given as input to a series of convolution operations with ReLU activations followed by a max-pooling layer. From the obtained question representations $\psi(x_i)$ and $\psi(x_j)$, we compose a symmetrical pairwise representation as $\phi(x_i, x_j) = \left[\max\big(\psi(x_i), \psi(x_j)\big), \min\big(\psi(x_i), \psi(x_j)\big)\right]$, where $\max$ and $\min$ are component-wise vector operations, i.e., max and min are applied to pair of components, so that two final vectors are obtained.

(2) We exploit BERT (Devlin et al., 2019) embeddings: $\phi(x_i, x_j) = \frac{1}{2}(bert\_emb(x_i, x_j) + bert\_emb(x_j, x_i))$, where $bert\_emb$ for a pair of questions comes from the final hidden layer representations, i.e., [CLS] token from the BERT model.

6 Question Clustering Data

In this section, we describe two datasets: (i) IC&OOS for intent classification; and (ii) Quora for intent clustering. We illustrate our procedure to transform the former into a dataset for clustering.

6.1 Intent clustering with IC&OOS

The dataset for Intent Classification and Out-Of-Scope prediction by Larson et al. (2019), which we denote IC&OOS, is a classification dataset, composed of user's queries distributed into 150 different intent classes over 10 domains, plus out-of-scope (OOS) queries falling outside the pre-defined classes. The data[3] contains 50, 20 and 30 user's queries per intent class in training, dev. and test parts, respectively. Plus 100 OOS queries for training, 100 – for development, and 1000 for test.

For example, we may see class categories such as MEAL SUGGESTION, with queries such as, *'suggestions for thai food'* or *'help me find some new dinner recipes'*, which may be challenging to separate from the items of RESTAURANT REVIEWS, e.g., *'at yakamoto how is their sushi'*, and even more difficult to discern from *'what are some good sushi restaurants in reno'*, belonging to RESTAURANT SUGGESTION class.

The data from all of the pre-defined classes are present in training, dev. and test parts. The main steps for transforming this dataset in one for clustering are (i) merging the items of all the categories together; and (ii) using the original class labels as indication of belonging to different clusters.

However, a real-world application scenario of automatic clustering would entail that new incoming data can contain items which constitute new clusters (class categories). Thus, in order to demonstrate the capability of the supervised clustering models to group together the items of unseen clusters, we use one set of intent classes for training and another set for evaluation, which is constituted by a completely different set of intent classes and questions. This way, we retain the queries from one third of intent classes, i.e., 50, from the training part, the dev. queries from another third of intent classes, and the test queries from the remaining one third[4] of classes, and use them as new training, dev. and test parts, respectively.

Additionally, it should be noted that the original dataset contains OOS queries, which we keep all.

[3] We use the **Small** variant of the dataset.

[4] The split of the classes into three sets is done randomly without reference to the 10 original topic domains.

Thus, our new split can be also used to analyze OOS queries, which might not be put in any semantically meaningful cluster, as well as unseen intent items, for which, we know that their natural clusters (original categories) exist.

Data sampling and instance creation Training and test examples in a clustering problem are sets of items. In order to be able to effectively update the structural clustering objective in Eq. 2 with NNs, we need to limit the size of training examples $(\mathbf{x}, \mathbf{y})$. Precisely we need to limit the size of input query sets $\mathbf{x}$, as the number of edges $\mathbf{e}$ (Eq. 3) to be passed trough the network grows exponentially with it. Thus, we split the data into samples: (i) from test set, we just extract random disjoint samples of equal size M; and (ii) from training and dev. sets, we form samples according to a more elaborated procedure to avoid having too many singletons in an instance.

More specifically, for each class C: (i) we shuffle its items in a random order; and (ii) we split them into a set P of m disjoint parts (mini-clusters) of random sizes (different sizes, necessarily ≥ 2) each, s.t., $\cup_{p \in P} = C$. Then, to build the training clustering examples, $\mathbf{x}$, we iterate for several times over the entire list of classes $\mathcal{C}$, in a random order, and, $\forall C \in \mathcal{C}$, we select a mini-cluster $p \in P_C$, which we append to a current sample S (initialized as empty), if $|S \cup p| \leq M$. Otherwise, we start a new sample with $S = \{p\}$, and go to the next category, until all P_C are exhausted (this happens simultaneously $\forall C$, as P_C have the same size). Now, our $\mathbf{x}$ sets consist of items contained in S's.

This procedure makes the presence of each category uniform (binary presence, yes/no): after seeing each N number of samples S, we encounter elements of all the classes, however, without preserving the original relative proportions of the class distribution. This way, by setting the sample size limit $M = 100$, we obtain around 28 and 12 clustering examples from training and dev. sets, respectively, and 35 examples from the test set.

6.2 Quora question clustering

The Quora dataset, made available by Lee et al. (2017), was designed to learn and test question duplication classifiers. That is, for automatically detecting if two questions are semantically duplicate or not. We use the Quora Intent Corpus by Haponchyk et al. (2018) based on a sample of questions from Quora.

One main difference with IC&OOS is the fact that the negative examples selected by the organizers of the Quora challenge refer to pairs of questions that have always some degree of lexical overlap. For example, the following pair *How much water on earth is consumable?* and *How much water is on earth?* is not duplicate. On the other hand, the two questions could be surely put in the same cluster WATER OF EARTH. This means that a similarity function learned from Quora labels may not be enough accurate for clustering. Also a simple scalar product between two embeddings would not be enough as it can only capture lexical overlap. The latter would surely fail on the pairs of the following questions *What is a recursion tree?*, *What does your family tree look like?*, *How does your Christmas tree look like?* since their specific semantics is different but the overlap is large. These examples suggest that a clustering algorithm must learn a similarity that looks to the entire set of items to be clustered, not just to single pairs. This requirement is inline with the characteristics of the methods we presented in Sec 4.

7 Experiments

We present the results of our empirical evaluation of the neural clustering models using IC&OOS data, followed by that using Quora Intent Corpus. We summarize afterwards the highlights of a deeper investigation, we conducted on the performance of our approach and of its errors.

7.1 Setup

Data: We created data from IC&OOS as discussed in Sec 6.1, which contains, 2650, 1100, and 2470 queries and 28, 12, 35 clustering examples, in training, dev., and test sets, respectively.

We also evaluate our approaches on Quora Intent Corpus[5] (Haponchyk et al., 2018) based on 1,334 questions from Quora duplicate detection competition[6]. This corpus contains 270, 146 and 212 question clusters respectively in the training, dev. and test parts. The clusters in each part are split into samples: training – in 10 samples, both dev. and test – in 5. A part of the test set is also provided with an expert annotation. We refer to the whole test set with labels automatically derived from Quora annotation as *automatic* test set, and to its part with expert annotation – as *manual*

[5] https://ikernels-portal.disi.unitn.it/repository/intent-qa

[6] https://www.kaggle.com/c/quora-question-pairs

Type of Supervision	**Model**		**Clustering measure**			**CEAF$_e$**
			Precision	Recall	F1	
None	Spectral clustering	tfidf	78.44	78.38	78.41±0.24	71.55±0.57
Clustering Function (instance similarity)		CNN	75.75	75.75	75.75±0.29	69.09±0.54
		BERT	75.74	75.74	75.74±0.21	69.02±0.36
	Kruskal	CNN	97.72	79.00	87.28±0.82	80.23±1.15
		BERT	79.95	91.30	85.25±0.46	78.59±0.61
Our Supervised Clustering	NSC-CNN		88.83	89.61	89.19±0.99	82.66±1.07
	NSC-BERT		**94.98**	**96.36**	**95.65**±0.82	**91.76**±1.51

Table 1: Comparison of clustering models: completely unsupervised, using supervised instance similarity function, and our supervised clustering on the test set of IC&OOS by Larson et al. (2019); disjoint scenario.

Models: We experiment with two variants of our Neural model for Supervised Clustering (NSC), based on two ways to encode question pairs outlined in Sec. 5.1: (i) NSC-CNN, using word and word overlap embeddings, and (ii) NSC-BERT, using BERT embeddings of question pairs.

For NSC-CNN, we employ fastText[7] word embeddings, in dimension 300, pre-trained for English language on Wikipedia (Bojanowski et al., 2017). We set the max length of questions to 50 and pad the shorter questions on the right. The size of the hidden layer is set to $\frac{1}{3}$ of the size of the input layer. The convolution filter width varies from 1 to 3.

For NSC-BERT, we use BERT$_{\text{BASE}}$ model, which we train for 3 epochs for fine-tuning on the question pair classification task.

Since the training samples vary in size, we clip the gradients to have their L_∞ norm less than or equal to 1. This is to prevent the updates being dominated by the samples of bigger size.

Evaluation: We follow the evaluation setting of Haponchyk et al. (2018). Thus, we compute (i) clustering F1 measure, based on assigning each cluster to the most frequent (gold/output) cluster, (ii) coreference resolution CEAF$_e$ score (Luo, 2005; Cai and Strube, 2010).

Parameterization: We use dev. set for tuning the loss parameter r, which takes values from $\{0.1, 0.5, 1.0\}$, and selecting the best epoch with respect to clustering F1.

Baselines We consider a number of baselines based on the pairwise query similarities. We experiment with the following sources of pairwise signals: (i) tf-idf scores, (ii) outputs of the binary question pair classifier, which we train in two modalities, CNN and BERT.

We group the pairwise signals into a clustering output using spectral clustering algorithm (Ng et al., 2001) (implementation from *smile*[8] library), which we run on a matrix of pairwise similarities between data points. Spectral clustering is unsupervised, and requires the indication of the number of clusters k. For each sample, we set the parameter k to the gold number of clusters. This means that we are computing an upper bound and unrealistic performance, which can be used to provide a meaningful comparison (especially if our approach outperforms it). As an alternative to spectral clustering, we run Kruskal's algorithm on the graph of pairwise edges, using 0.5 as a threshold for the question pair classifier scores on them; pairs having the scores lower 0.5 are neglected.

7.2 Experiments on IC&OOS task

In Tab. 1, we present the results on IC&OOS dataset averaged over 10 different sample splits, obtained with 10 different random seeds. First, we note that NSC consistently improves over all the baselines in terms of both F1 and CEAF. It also shows a good precision/recall balance.

The significantly lower results of the unsupervised baseline, spectral clustering+tf-idf, suggest that, in IC&OOS, we are supposedly dealing with a rather non-trivial task, where queries expressing the same intent do not necessarily have surface closeness. Even if the model is aware of the true number of present intents in a sample (gold k).

The other four baselines capitalize on training a supervised scoring function for query pairs, to be further used as a clustering criterion. From our experiments with IC&OOS data, we conclude that it is also not trivial to train a pairwise classifier to convey a notion of semantic similarity to the pairs of queries from unseen classes. This is reflected in the results of using a supervised similarity function. The performance of spectral clustering on the output of the pairwise classifier, equally low for CNN and BERT, and lower than model using tf-idf

[7] https://fasttext.cc/docs/en/pretrained-vectors.html

[8] http://haifengl.github.io/smile/

Model	Clustering measure			CEAF_e
	Precision	Recall	F1	
LSSVM	84.92	51.76	64.32	49.72
LSP	71.36	89.45	79.38	59.99
LSP_{py}	70.80	**90.00**	79.22 ± 0.33	59.82
NSC-CNN	80.25	82.16	**81.12** ± 1.76	62.80
NSC-BERT	**86.93**	72.96	79.19 ± 1.41	**63.89**

Table 2: Comparison of our neural models to the structural baselines on the **manual** test set of Quora Intent Corpus.

Model	Clustering measure			CEAF_e
	Precision	Recall	F1	
LSSVM	80.16	77.81	78.96	63.68
LSP	66.06	91.64	76.78	51.50
LSP_{py}	65.32	91.47	76.18 ± 0.63	50.62
NSC-CNN	71.44	91.10	80.07 ± 0.28	59.08
NSC-BERT	**88.01**	**96.96**	**92.23** ± 0.98	**86.07**

Table 3: Comparison of our neural models to the structural baselines on the **automatic** test set of Quora Intent corpus.

scores, is clear evidence for this. When we run Kruskal's on top of the output of the pairwise classifier, we observe a huge bias, towards precision in case of CNN and recall in case of BERT. The use of a threshold does not seem robust, when training examples (query pairs) are treated as independent.

In NSC, we assume, this problem is mitigated by "collaborative" updates of the structural loss. Overall, we note the impressive performance of NSC, especially when fed by BERT. A clustering F1 of 95.65 suggests that NSC can replicate the clusters of questions that the human annotator/knowledge engineer devised.

7.3 Experiments on Quora Intent Corpus

We run each model with 10 different random seeds for shuffling training examples and report the averaged results on the manual, Tab. 2, and automatic test sets, Tab. 3.

7.3.1 Baselines

We compare NSC with two state-of-the-art structural approaches, LSSVM and LSP proposed in (Haponchyk et al., 2018), reporting their numbers on the same data. LSP_{py} is our LSP reimplementation in python using text similarity. We trained LSP_{py} for 100 epochs.

7.3.2 Results

NSC-CNN improves over the state of the art on both the test sets for both measures. On the manual test set, NSC-BERT achieves, as expected, higher CEAF. One possible explanation for its lower F1 on the manual test set is its small size, which probably does not enable an accurate evaluation.

	BiMPM	Our fine-tuned BERT
Accuracy	88.17	90.88

Table 4: Accuracy comparison on question duplicate detection task on Quora split by Wang et al. (2017).

Model	Inclass Acc.	OOS Recall
Larson et al. (2019)		
NN + avg. FastText emb.	84.50	23.20
CNN + Glove emb.	88.90	22.20
BERT	96.40	40.90
Our Models		
CNN	80.20	28.88
BERT	94.87	38.80

Table 5: Comparison of our intent classification baselines to the intent classification models from Larson et al. (2019) on IC&OOS test set.

In contrast, on the evaluation over the automatic test set, NSC-BERT largely outperforms any model, according to any measure. This is mainly due to the fact that the automatic clusters are more consistent with the information present in the training data, used for fine-tuning BERT. Indeed, we fine-tuned $\text{BERT}_{\text{BASE}}$ on the full Quora dataset of question pairs[9] (except for the pairs containing questions from development and test parts of Quora Intent Corpus). For the sake of transparency, in Tab. 4, we report the accuracy of the fine-tuned BERT on Quora split by Wang et al. (2017) compared to the official results of their BiMPM model.

We believe the result of NSC-BERT is promising, and, in the scope of intent detection, by not being bounded to a particular set of intents, it contributes to the existing neural solutions (Xia et al., 2018; Lin and Xu, 2019; Lin et al., 2020).

7.4 Deeper Analysis

In this section, we investigate the general clustering ability of NSC, and in this way, enable the comparison to the upper bound of intent detection, i.e., the intent classifier, and list its most common mistakes.

7.4.1 How good is the clustering per se?

Here, we address the standard IC&OOS scenario with the original class distribution of dataset, where all the 150 intent classes are equally presented in the data. Moreover, we explore the upper bound to any clustering algorithm, i.e., the use of a supervised classifier in an unrealistic (useless for clustering) scenario, that is, having in the training data, all the clusters (classes) to be discovered. To carry out this comparison, we trained two intent classifier

[9] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

Model	Clustering measure			CEAF$_e$	OOS Recall
	Precision	Recall	F1		
CNN intent classifier	88.70	93.14	90.76±0.93	86.12±0.75	28.88
BERT intent classifier	90.53	98.36	94.29±0.24	89.10±0.40	38.80
NSC-CNN	89.32	91.25	90.24±1.09	84.86±1.34	85.49±3.94
NSC-BERT	93.58	97.27	95.38±0.34	92.05±0.58	71.45±4.35

Table 6: Comparison of the neural clusterings models to the classification baselines on the test set of IC&OOS dataset by Larson et al. (2019); full scenario.

models, CNN and BERT, with 150+1(OOS) target classes. In Tab. 5, we report their performance in terms of in-class accuracy and OOS recall. We also report the performance of the classification models from Larson et al. (2019) for reference. As it can be observed, our models perform comparably, e.g., our BERT model is just 1.5 points behind.

We trained NSC on the same data, split into samples, so that we could compare to the above classifiers. For this purpose, we follow our sampling procedure in Sec. 6.1, this time keeping all the classes, which gives us around 79 and 32 samples for training and dev., respectively, and 66 samples for test. To keep the two types of systems aligned, we evaluate the classifiers also in terms of clustering F1 on the same test samples (of size M), which are then averaged. Namely, we consider queries, within a sample, predicted by a classifier model, as the same class to form a distinct cluster, while those predicted as OOS – singletons.

We note that (i) as expected, the results of NSC in Tab. 6 improve with respect to the completely disjoint setting (Tab.1). (ii) NSC-CNN is able to almost replicate the result of the CNN classifier in terms of F1, yielding only 1.3 of a point in terms of CEAF. (iii) Interestingly, the OOS Recall is more than 85% (2-3 times the one of the classifiers), which means that 85% of all OOS queries were detected by NSC-CNN (predicted singletons in their corresponding samples). Although, we recognize that it can be easier to detect OOS queries in a small sample than in a big set. (iv) NSC-BERT improves over the classifier model on the test samples by 1.5 in terms of clustering F1 and by more than 3 CEAF points, also achieving a better precision/recall balance (same as for CNN modality). We hypothesize here an advantage of the supervised clustering model might lie over the classification models, which are generally not as well adaptive to class imbalance in data. (v) Again, the NSC-BERT highly improves (at least 2 times) the recall of the classifier for the OOS task.

7.4.2 Error analysis

Analysing the output of NSC (here, we limit the discussion to NSC-BERT in the disjoint scenario of Sec. 7.2), we discovered that the majority of the mistakes made by the clustering algorithm can be traced back to several interpretable causes.

A trivial case of word overlap or generally string matching in Ex. 4 made NSC put the examples of seemingly distinct classes together. Actual ground truth intent classes are denoted in parentheses.

(4) **cluster**:
(1) *what is the reason **humans** even exist* (meaning_of_life)
(2) *let me know if you are a **human** or are a computer* (are_you_a_bot)

Next, we find the presence of the word-indicators of the same semantic category, i.e., SPEED, in Ex. 5, that misled NSC.

(5) **cluster**:
(1) *speak more **quickly*** (change_speed)
(2) *i'm in the mood for **slow** songs and nothing else* (play_music)
(3) *talk **faster*** (change_speed)

A frequent type of NSC's mistakes is merging together instances of different intent classes which belong to the same topic domain, especially in case of rather close subtopics as in cluster Ex.-s 6–7.

(6) **cluster**:
(1) *put on my 90s playlist* (play_music)
(2) *put on some metallica music* (play_music)
(3) *what kind of music on the speaker now* (what_song)

(7) **cluster**:
(1) *how do i **freeze** my bank account* (freeze_account)
(2) *why is there a **stop** on my deposit account* (account_blocked)

In addition, Ex. 7 has another complicating factor of using semantically very close expressions for distinct intent concepts. Right the opposite situation of erroneously splitting the instances of the same intent class is also common, as in Ex.-s 8–9.

(8) **cluster**:
(1) *bye bye then* (goodbye)

(9) **cluster**:
(1) *good speaking to you* (goodbye)
(2) *it was great talking to you* (goodbye)

In general, we assume, that the last two types of mistakes can be reduced if the model sees on training the data from the corresponding intent classes.

NSC also drew some (not absolutely meaningless) connections between OOS queries (Ex. 10).

(10) **cluster**:
(1) *what is the highest quality* ***carpet*** *available* (oos)
(2) *find schematics for ikea* ***desk assembly*** (oos)
(3) *i have a super* ***runny nose*** *and want to find a doctor* (oos)
(4) *what was the latest* ***tremor*** *on the richter* ***scale*** (oos)

And finally, the clustering decision in Ex. 11 potentially highlights an annotation error of query (2) being a false positive OOS.

(11) **cluster**:
(1) ***where is the closest*** *mcdonald's to foxwoods casino* (directions)
(2) ***where is the closest*** *driving range* (oos)

7.5 Discussion

In this section, we discuss some of the important findings of our paper: First, our experiments suggest that the transformer model boosts the performance of our clustering approach. This is justified by the mainstream research: with respect to the standard embeddings (word2vec, glove,..), transformer models provide contextual representation of words, i.e., the embedding of a word is defined with respect to the others that are in the same piece of text. They provide a very powerful representation of pieces of text. Thus, we can obtain a precise similarity between pairs of questions. Thanks to our structural loss function, we can back-propagate structure properties of the entire cluster back to the transformer models so that we enrich even more its contextual similarity.

Second, in the field of dialog systems, our approach can be extended to jointly predict intent and slot attributes. NSC can use information about slots and the background knowledge given by attributes and values, to cluster questions into intents. The latter will be then more related to the specific task defined by the available slot information. Conversely, if we suppose the developer has already the intents, our clustering algorithm could be used to cluster values into attributes. Then, since NSC can reach performance similar to supervised classification methods, it would be interesting to see if it can be more accurate than them, considering the critical problems of transfer learning (i.e., when the data for training is different from the one the deployed system receives).

Third, we showed the performance on NSC exactly on unseen clusters. Our approach only uses some clusters of the data for training (each cluster is a training example). Then, it can predict unseen clusters in the test set. In other words, our models generalize what they learn from some clusters to unseen clusters.

Finally, given one of our models trained on a set of clusters, we can easily continue its training with new examples, i.e., new training clusters, as our neural architecture is an online framework. One main scalability question could be: *Given one domain for which we have clusters to train our approach, how can we scale to other domains?*

We will need new clusters for the new domains, i.e., target domain data, which is typically used for effective transfer learning. This does not mean that we need a large number of clusters, we just need some of them to transfer our clustering model from one domain to another. The transferred models will be able to predict many more new clusters from the new target domain.

8 Conclusions

In this work, we firstly proposed supervised neural clustering based on traditional LSSVM and LSP models, which hinge on optimizing the structural margin loss. This extends the structured prediction methods for supervised clustering to a neural setting. Our experiments on IC&OOS and Quora Intent Corpora show an impressive improvement over the state of the art, 17.24% absolute over unsupervised models, and 8% points more than our proposed semi-supervised approaches. This suggests that our neural structured prediction can (i) effectively optimize a structural clustering objective function on structured examples, such as sets of questions for intent detection, and (ii) uncover clusters of questions of unseen classes, i.e., potential intents not seen in training.

Acknowledgements

We would like to thank the anonymous reviewers as well as the entire PC for their valuable work.

References

Steven M Beitzel, Eric C Jensen, David D Lewis, Abdur Chowdhury, and Ophir Frieder. 2007. Automatic classification of web queries using very large unlabeled query logs. *ACM Transactions on Information Systems (TOIS)*, 25(2):9.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jie Cai and Michael Strube. 2010. Evaluation metrics for end-to-end coreference resolution systems. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '10, pages 28–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

P. Deepak. 2016. Mixkmeans: Clustering question-answer archives. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1585. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312. Association for Computational Linguistics.

Eraldo Rezende Fernandes, Cícero Nogueira dos Santos, and Ruy Luiz Milidiú. 2014. Latent trees for coreference resolution. *Computational Linguistics*, 40(4):801–835.

Thomas Finley and Thorsten Joachims. 2005. Supervised clustering with support vector machines. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 217–224, New York, NY, USA. ACM.

Iryna Haponchyk, Antonio Uva, Seunghak Yu, Olga Uryupina, and Alessandro Moschitti. 2018. Supervised clustering of questions into intents for dialog system applications. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2310–2321. Association for Computational Linguistics.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional lstm feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Joseph Bernard Kruskal. 1956. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In *Proceedings of the American Mathematical Society, 7*.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1311–1316, Hong Kong, China. Association for Computational Linguistics.

Yann LeCun, Sumit Chopra, Raia Hadsell, Fu Jie Huang, and et al. 2006. A tutorial on energy-based learning. In *PREDICTING STRUCTURED DATA*. MIT Press.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197. Association for Computational Linguistics.

Xiao Li, Ye-Yi Wang, and Alex Acero. 2008. Learning query intent from regularized click graphs. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 339–346. ACM.

Ting-En Lin and Hua Xu. 2019. Deep unknown intent detection with margin loss. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5491–5496, Florence, Italy. Association for Computational Linguistics.

Ting-En Lin, Hua Xu, and Hanlei Zhang. 2020. Discovering new intents via constrained deep adaptive clustering with cluster refinement. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8360–8367. AAAI Press.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 25–32, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ruy Luiz Milidiú and Rafael Rocha. 2018. Structured prediction networks through latent cost learning. *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 645–649.

Atsushi Miyauchi, Tomohiro Sonobe, and Noriyoshi Sukegawa. 2018. Exact clustering via integer programming and maximum satisfiability. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, pages 1387–1394. AAAI press. 32nd AAAI Conference on Artificial Intelligence, AAAI 2018 ; Conference date: 02-02-2018 Through 07-02-2018.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 849–856, Cambridge, MA, USA. MIT Press.

Hao Peng, Sam Thomson, and Noah A. Smith. 2018. Backpropagating through structured argmax using a spigot. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1863–1873. Association for Computational Linguistics.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004*, pages 1–8, Boston, Massachusetts, USA. Association for Computational Linguistics.

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *CoRR*, abs/1604.01178.

Noah A. Smith. 2011. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool.

Po-Wei Wang, Priya L. Donti, Bryan Wilder, and J. Zico Kolter. 2019. Satnet: Bridging deep learning and logical reasoning using a differentiable satisfiability solver. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6545–6554. PMLR.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 4144–4150. AAAI Press.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333. Association for Computational Linguistics.

Ji-Rong Wen, Jian-Yun Nie, and Hong-Jiang Zhang. 2001. Clustering user queries of a search engine. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 162–168, New York, NY, USA. ACM.

Congying Xia, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S. Yu. 2018. Zero-shot user intent detection via capsule neural networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3090–3099. Association for Computational Linguistics.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. 2018. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5502–5511, Stockholmsmässan, Stockholm Sweden. PMLR.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1169–1176, New York, NY, USA. ACM.

Hanlei Zhang, Hua Xu, Ting-En Lin, and Rui Lyu. 2021. Discovering new intents with deep aligned clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*.