Unsupervised training data reweighting for natural language understanding with local distribution approximation

Jose Garrido Ramas
Amazon Alexa AI
jrramas@amazon.com

Thu Le
Amazon Alexa AI
deule@amazon.com

Bei Chen
Amazon Alexa AI
chenbe@amazon.com

Manoj Kumar
Amazon Alexa AI
abithm@amazon.com

Kay Rottmann
Amazon Alexa AI
krrottm@amazon.com

Abstract

One of the major challenges of training Natural Language Understanding (NLU) production models lies in the discrepancy between the distributions of the offline training data and of the online live data, due to, e.g., biased sampling scheme, cyclic seasonality shifts, annotated training data coming from a variety of different sources, and a changing pool of users. Consequently, the model trained by the offline data is biased. We often observe this problem especially in task-oriented conversational systems, where topics of interest and the characteristics of users using the system change over time. In this paper we propose an unsupervised approach to mitigate the offline training data sampling bias in multiple NLU tasks. We show that a local distribution approximation in the pre-trained embedding space enables the estimation of importance weights for training samples guiding resampling for an effective bias mitigation. We illustrate our novel approach using multiple NLU datasets and show improvements obtained without additional annotation, making this a general approach for mitigating effects of sampling bias.

1 Introduction

Production Natural Language Understanding (NLU) models are typically trained on the offline annotated data. Models learn from the offline data to perform classification on the online live data in production after the model being deployed.

The core of voice-controlled assistants, such as *Google Home*, *Amazon Alexa*, or *Siri*, apply NLU models to perform both *intent classification* and *slot labelling* (Weld et al., 2021). For example, the input utterance *"set alarm at 9 am"*, would be classified as "*SetAlarmIntent*" intent, and the slots *"9"* and *"am"* would be labelled as *Time*.

In the deployed NLU systems, a distribution mismatch between training and live data is common. Some factors contributing to such a mismatch are changes of the live data distribution over time (due to, for example, new users or to seasonal changes), and usage of data from other more or less unrelated tasks to enrich the training data, so called out-of-domain data.

The issue with this mismatch in distribution between training and inference time is that models learn a bias towards specific classifications that is not existing at inference time. Even if the label distributions are matched, it is still possible that the model will have biased performance, since demographic and speech differences need not perfectly correlate with label distribution, resulting in degraded accuracy and possibly unequal performance across populations (Subramanian et al., 2021). Thus, mitigation of this distribution mismatch is an important step in the development of models.

While a common approach of dealing with this kind of bias is manual upsampling of classes in the training data (Estabrooks et al., 2004), this approach is not always optimal, due to the complexity and variation of natural language. Data of the same class in a classification task often come from very different forms of language, for example slang vs. formal language. A simple upsampling based on the classes does not mitigate differences in usage of slang during inference time compared with the training data.

Another difficulty in class based resampling is to get the correct label distribution from the live data in the case that it is missing ground-truth. For example, when a model is deployed, its training data will ideally match the distribution of the current live data, since this is the data the model will be applied to. This current live data distribution will be more similar to recent live data, compared to historical data. However, manually annotating data to obtain the ground-truth labels takes time. Thus, during deployment, the training data should match the unannotated live data, and in this case it is only

feasible to use bias reduction methods which don't rely on manual annotation.

In this work we build on top of importance weighting, which is an approach that has gained traction in other machine learning fields, but until now has found little attention on natural language understanding. We propose a method to assign weights to every individual utterance in a training corpus based on observed live usage of the system, by using utterances' *neighbourhood* in the embedding space. We choose to find the *neighbourhood* of utterances with KNN and KMeans (in the case of KMeans, each cluster is considered a neighbourhood). We choose these methods due to their easy interpretation: For example, in the KMeans case, one can observe a cluster of utterances, its frequency online and offline, and easily understand why this specific pattern has a high or small weight.

The two unsupervised re-weighting based on KNN and KMeans are compared with two baselines: keeping the training data as it is (with the distribution mismatch), and, on the other hand, a semi-supervised intent-based approach, in effect class up/down-sampling. We evaluate our methods on both public data and in a deployed commercial NLU system. In the public datasets, we simulate a distribution mismatch by both introducing a label mismatch and also combining different sources of data with different distributions.

We show that the unsupervised approaches can better mitigate certain kinds of sampling bias compared to the intent-based approach, while also having the advantage that we can perform re-weighting of the training data without need of annotation: thus our method is suitable for test data with fast-changing distribution. Without the need for any labeled data, our unsupervised approaches are generic enough to be applicable to multiple different natural language processing tasks.

2 Related Work

The problem of dealing with training data sampling bias in machine learning is well studied. The idea of adjusting training data distribution to meet the distribution at inference time is discussed in (Zadrozny, 2004), (Shimodaira, 2000) and (Dudík et al., 2005). These methods however require estimation of biased densities or selection probabilities, which pose a challenge in the real world.

Similarly in (Grover et al., 2019), to deal with bias in generative models, a classifier is learned to distinguish the data distribution from the generative model. This allows guidance of the generation of additional data to better mimic the existing data. In this work we extend on the work above towards natural language understanding, and focus on the real world problem in which the training data is biased with respect to the unannotated real world application data (live data).

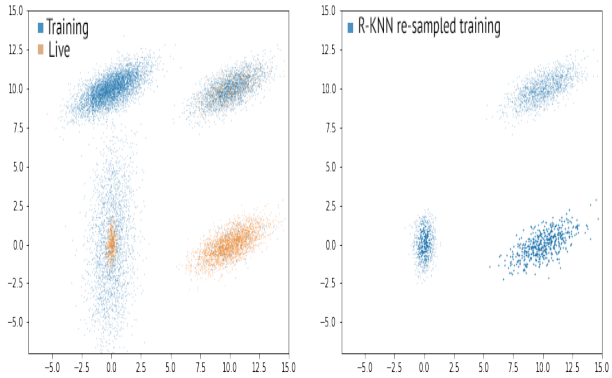

Figure 1: Example of *training* (blue, left) and live data (orange, left) with different distributions, as well as the output of R-KNN resampling the training data (blue, right). Darker points indicate higher weight.

In (Huang et al., 2006) unsupervised model-agnostic importance weights for every training sample are computed. Our unsupervised approach differs from theirs in that we calculate the weights based on the neighbourhood, which makes interpretation of the individual weights easier in the case of natural language data. A closer investigation of importance weighting can also be found in (Cortes et al., 2010) providing theoretical bounds, as well as in the recent work of (Fang et al., 2020) that looks specifically at the application of importance weighting and weight estimation for deep learning tasks. An important difference to these approaches is that they focus on including importance weighting directly into the learning of the models. In our work we focus however solely on the underlying data distribution of utterances, while keeping the estimation model the same.

In contrast to importance weighting, another common approach in real world applications is the use of pure upsampling of training utterances for certain classes, based on automatic labelling of the live data. In (Estabrooks et al., 2004) the effect of upsampling for certain underrepresented classes

is investigated, showing its effectiveness. On the other hand looking at the class distribution alone will also not reduce data bias as described in Sec 1, making the requirement of an automatic way of handling different kinds of distribution mismatch more pronounced.

3 Utterance Weight Estimation

In this section, we describe our approach on how to estimate the weight of each individual utterance in offline training data based on a random sample from online live data.

Let X represent the random variable of an utterance from online live data, where X follows some distribution P_X, denoted as $X \sim P_X$. Let Y be the random variable of true labels of X, where Y follows some distribution P_Y, denoted as $Y \sim P_Y$. Also, let X' and Y' be the corresponding random variables of X and Y in offline data, where $X' \sim P_{X'}$ and $Y' \sim P_{Y'}$. The issue we aim to resolve is that typically $P_{X'} \neq P_X$ and $P_{Y'} \neq P_Y$.

Analysing the difference in distributions of utterances P_X and $P_{X'}$ is particularly challenging in NLP because the different surface forms of utterances do not necessarily imply the semantic difference in classification tasks. However, due to the advance of natural language embeddings, we are now able to efficiently approximate the local distributions over the semantic meanings of text which allows the estimation of P_X and $P_{X'}$. Specifically, we propose to approximate the difference of local distributions in offline and online utterances summarized as follows:

1. Map all utterances of offline training and online live data into the embedding space.

2. For every offline training utterance x_i, estimate the local approximations of P_X and $P_{X'}$, denoted as $\hat{P}_X$ and $\hat{P}_{X'}$ and compute the weight using its neighbourhood utterances

$$w_i = \frac{\hat{P}_X(X = x_i)}{\hat{P}_{X'}(X' = x_i)}.$$

3. Resample the utterance x_i in offline training data according to the weight w_i.

3.1 Mapping utterances into embedding space

Pre-trained BERT-based models sentence-level representations do not guarantee that semantically similar utterances will be close in the embedding space. Thus, for the mapping of the text into the embedding space, we use Sentence-BERT (Reimers and Gurevych, 2019a), which modifies the original BERT architecture via siamese and triplet network structures to compute semantically meaningful embeddings which can be compared using several functions such as cosine similarity or euclidean distance.

3.2 Local Distribution Approximation

To mitigate the distribution mismatch we aim to debias the local distribution of each training utterance to match the live data distribution. Having embedded utterances into the embedding space in the first step, it is now possible to estimate the local neighbourhood of text utterances by using the distances in the embedding space. Then, we are able to determine an approximation of the local distributions P_X and $P_{X'}$ at some given utterance by looking at the number of samples in this neighbourhood that belong to either X or X'. In the following we propose three different reweighting methods, which differ on how the neighbourhood of each utterance is defined: Reweighting K-Nearest-Neighbour (R-KNN), Reweighting KMeans (R-KMeans) and, as an additional method, an intent-based approach (B-intent; effectively class up/down sampling).

R-KNN (Reweighting via KNN): The first local approximation we discuss is based on k-nearest-neighbours. We follow the standard procedure to use $K = \sqrt{N}$ with N being the total number of utterances in training and the live sample combined.

We aim to determine the weight the individual training utterance x_i, by using a sample of embedded training samples T and of live samples L. Let $\text{KNN}(x)$ be the set of K nearest neighbours to a point x in the embedding space. We determine:

$$D^{(i)}_{train} = \bigcup_{e \in \text{KNN}(x_i),\ e \in T} e$$

the set of all utterances that are part of both the neighbourhood of a training utterance x_i and the training data. In a similar way we determine the set of all utterances from the live traffic sample L that fall into the neighbourhood of x_i:

$$D^{(i)}_{live} = \bigcup_{e \in \text{KNN}(u_i), e \in L} e$$

With these two sets, we approximate the probability of having a training sample in this region of the

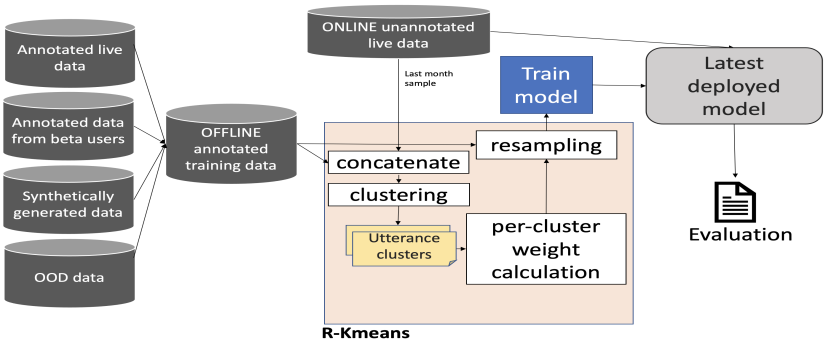

Figure 2: Pipeline for utterance reweighting. We combine many different sources of training data, and then assign a high/low weight to each utterance depending on the recent, unannotated *live* data, which follows the most similar distribution the data the model will be applied to (compared to, for example, historical annotated live data)

embedding space:

$$p(x \in T|\text{neighbourhood}(x_i)) = \frac{|D^{(i)}_{train}|}{|T|}$$

And similarly we approximate the probability of a live utterance x being seen in this region of the embedding space as

$$p(x \in L|\text{neighbourhood}(x_i)) = \frac{|D^{(i)}_{live}|}{|L|}$$

. The ratio of these two probability approximations is the weight we assign to the utterance x_i:

$$w_i = \frac{p(x \in L|\text{neighbourhood}(t_i))}{p(x \in T|\text{neighbourhood}(t_i))}$$

w_i therefore indicates therefore how much more likely it is that an utterance in a certain region is part of the live traffic in comparison to being part of the training data.

R-Kmeans (Reweighting via KMeans) Another way of approximating neighbourhoods is with unsupervised clustering. In this case the training and live data are combined and then clusters are computed in the embedding space. Then, the neighbourhood is all utterances within the same cluster. Thus, all utterances within a cluster obtain the same weight. After having found the neighbourhoods $D^{(i)}_{train}$ and $D^{(i)}_{live}$ through clustering, we follow exactly the same equations as above to compute the weights. For simplicity we chose K-Means clustering (MacQueen et al., 1967) and chose K as $K = \sqrt{N}$. If the live data and the training data came from the same distribution, it would be expected to find that, in each cluster i, $\frac{D^{(i)}_{train}}{D^{(i)}_{live}} \approx \frac{|T|}{|L|}$. After reweighting each utterance with a weight calculated with R-KMeans, the above equality is true on every cluster.

B-intent (Baseline via intent) As a baseline, we reweight the data based on the label distribution. The problem with this approach is it can't address latent distribution mismatches not directly related to the labels, as for example formal and informal language (see Sec 1). We train a classifier on the biased training data to infer $\hat{P}_Y$, an approximation of P_Y, and we use P'_Y as is known from the annotated training data. We give each intent a weight as: $w_{intent} = \frac{\hat{P}_Y(intent)}{P'_Y(intent)}$ which is in line with the description above for R-KNN, considering the neighbourhood of an utterance to be made of all utterances with the same label. As a result after reweighting the utterances of every intent with the weight of their intent w_{intent}, the labels of the resampled data will follow $\hat{P}_Y(intent)$.

3.3 Resampling the Training Data

With the computed weights for every training example, we are now able to resample the training data according to the live data distribution.

A weight < 1.0 means, that this training utterance is less reflective of the live distribution, while a weight > 1.0 reflects utterances more important for matching the live distribution.

While there are different ways in the literature of using this reweighting information, like (Fang et al., 2020) and (Huang et al., 2006) using it directly as part of the optimisation in the learning of the machine learning model, we chose the most straight forward of up- and down-sampling the utterances directly in the training data. A toy example of R-KNN resampling can be seen in Fig. 1.

4 Experiments

In our experiments we evaluated our methods on multiple different NLU datasets to verify the feasi-

	B-Bias	B-intent	R-KNN	R-KMeans	R-intent (rel)	R-KNN(rel)	R-KMeans (rel)
snips int	0.0164	0.0161	0.0162	0.0157	-1.82927	-1.21951	-4.26829
slurp int	0.1542	0.15	0.148	0.1462	-2.72374	-4.02075	-5.18807
snips utt	0.1329	0.13	0.1214	0.1258	-2.18209	-8.65312	-5.34236
slurp utt	0.34	0.3372	0.3324	0.3322	-0.82353	-2.23529	-2.29412

Table 1: Intent ("int") and utterance ("utt") error rates of the different methods in SLURP/ SNIPS datasets. Best result in bold. Each experiment is run ten times, and the average is reported. Both absolute value and relative change with respect to the first baseline is also reported.

bility of the approach.

4.1 Datasets

We tested our methods on a large commercial voice assistant dataset, as well as in two public ones: SLURP (Bastianelli et al., 2020) and SNIPS (Coucke et al., 2018). In all these datasets, the NLU task is intent classification and slot labelling. In the commercial dataset case, data is de-identified.

The training and test data are manually annotated, whereas the live data isn't. In the commercial voice assistant scenario, we take a sample of last month's unannotated live data as representative of current usage of the system. The size of the sample is the same as the offline training data. The annotated live data (test data), is not available during model deployment, but can be obtained afterwards to estimate the performance of the method.

4.2 Bias simulation strategies

Most available natural language understanding datasets are very well curated, with the test sets closely resembling the distribution of the training data. Thus, in the public datasets we simulate bias that could occur in real world applications via two different strategies on the training data:

Intent-based sampling bias: We introduce bias in the label distribution in the following way: each intent is assigned to either a low-sampling bucket (with probaility 20%) or to a high-sampling bucket (with probability 80%). The two intents that are in common between SNIPS and SLURP tasks (related to weather and to music) are both assigned to the low-sampling bucket. Finally, intents in the low-sampling buckets are down-sized to 20% of their original size, by randomly removing 80% of utterances which are annotated as belonging to this intent. The high-sampling intents are left as is.

Add OOD data: To introduce bias not directly related to the labels, as well as mimic the real-life scenario in which the training set is composed of different data sources with different amounts of noise, we also add, to each task, the training data of the other task. That is, we add the SNIPS data to the SLURP training set, and we add the SLURP data to the SNIPS training set. Prior to adding the data, we first produce machine-annotated labels for the SNIPS utterances in the SLURP label space, as well as labels to the SLURP utterances in the SNIPS label space.

4.3 Experimental Setup

The embeddings were generated with *paraphrase-MiniLM-L6-v2* model part of (Reimers and Gurevych, 2019a) sentence transformer model family. This model is fine-tuned so that semantically similar sentences are close in the embedding space with respect to distance functions, including euclidean distance (Reimers and Gurevych, 2019b).

To not leak information of the unseen test data into the reweighting, we used the development data for the distribution approximation.

For the resampling, we upsampled utterances with a weight w_i to frequency: $n_i = \lfloor w_i \rfloor + \theta$, where θ is random variable that is 1 with $p = w_i - \lfloor w_i \rfloor$, and 0 otherwise. The expected value is $E[n_i] = w_i$

We train a BERT model (Chen et al., 2019). For hyperparameter tuning, we follow (Chen et al., 2019), and use adam optimizer (Kingma and Ba, 2014) over 4 epochs, with a learning rate of 5e-5 and batch size 32. We use the implementation from (Wolf et al., 2019), with *bert-base-uncased* pretrained model. We report f1-score on the test data.

We compare our unsupervised approaches (**R-KNN** and **R-KMeans** from Sec 3.2) with two baselines: **B-Bias** (baseline model trained on the biased data) and **B-Intent**, baseline model in which the biased data is up/down-sampled so that the label distribution matches the live data (see Sec 3.2). The BERT model described above is used to obtain the hypothesised intent on the live data.

4.4 Results

Public datasets: The results of our experiments can be seen in table 1. We report intent classification error rate, as well as utterance error rate. We define utterance error rate as the fraction of utterances in which there is an error either in the slot labelling or intent classification task.

Each experiment is run ten times, and the average error rate is reported. The difference between both R-KMeans and the two baselines (B-intent and B-bias) passes a two-sided paired t-test for statistical significance at 95% confidence level.

The difference Between the R-KMeans and R-KNN approaches is, however, not statistically significant. R-KMeans has the advantage over R-KNN of easier interpretation of the weights: one weight is produced per cluster, instead of per utterance. The clusters can manually be inspected, and, comparing the in-cluster frequency of the *live* and *training* data, understand why this cluster got a high/ low weight.

For example, we observe in our SNIPS run two distinct clusters related to weather queries that get different weights: the first one, related to questions about specific weather events (such as snow or rain: includes, for example, the utterance *"is it snowing in California"*. Using **R-KMeans** reweighting, this cluster receives a weight of 1.04 (which can be interpreted roughly as: this pattern of utterances is equally frequent in the live data (development SNIPS data in this case) as in the training data. Thus, it does not need to be upsampled or downsampled.

However, a different cluster of weather queries containing more general questions *"what is the weather forecast for Akers New Hampshire"* receives a weight of 12: This cluster is 12 times less frequent in the training data than in the live data. Thus, this cluster is upsampled by 12.

Overall, in our experiments the two unsupervised methods perform better than both intent-based resampling and the baseline. A limitation of our work, however, is that it requires live annotated data to use as test data, to estimate the performance post model deployment. Obtaining this data can be a challenge in real-life applications.

Commercial dataset: On the commercial dataset, we show that, in the case that the training data has a different distribution to the *live* and test data, applying reweighting techniques with local distribution approximation can improve performance. We compare the results of applying reweighting on the training data vs. without reweighting strategy and report the relative differences. We use R-KMeans reweighting, due to the easier manual inspection of the assigned weights (see Sec 3.2).

	Intent	Utterance
Overall	-4.63	-1.99
Global	-2.02	-1.29
HomeAutomation	-13.77	-9.49
Knowledge	-2.1	-2.17
ToDos	-12.34	-3.99
Notifications	-9.09	-6.29

Table 2: Relative reduction in error rates (both intent and utterance) in the commercial dataset.

We report the relative difference in both intent error rate and utterance-error rate. As shown in Table 2, we see improvements in both utterance and intent error metric, with the biggest coming from Home Automation domain (13.77%), and an overall improvement of 4.63% accross all domains. The results with respect to the baseline passes a two-sided paired t-test for statistical significance at 95% confidence level.

5 Conclusion and future work

In this work, we showed how the reweighting of training data using local distribution approximation helps in mitigating sampling bias in natural language understanding production models. We simulated the bias in public training datasets to mimic real world application scenarios in which different data sources are used, and they each come from different distributions. We reweighted utterances based on the approximation of local distribution to minimise the mismatch between the training and online live traffic data. The simplicity of our approach, and the fact that it does not require manual or machine annotation, means that it can be used to quickly adapt the training data to the ever-changing live data in deployed models. Experiments in both a commercial dataset and two public datasets have shown that our approach can mitigate the mismatch and bias in training data without additional manual tuning. In the future, we want to experiment the combined impact of our method with different data augmentation techniques, study the impact on fairness across populations, as well as bias detection methods to trigger the reweighting model.

6 Ethical considerations

In this work we apply a reweighting method before model deployment to mitigate the problem of bias in the training data compared to the live data. We target overall accuracy as the metric we aim to improve, and we achieve so by tailoring the model to the latest live data at model deployment. However, the impact of reweighting on per-population accuracy has not been studied. There is a risk that, due to focusing on current live data, populations which at the time of a model deployment are not extensively using the model are not well-served by the reweighting, even though overall accuracy improves.

References

Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020. SLURP: A spoken language understanding resource package. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7252–7262, Online. Association for Computational Linguistics.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Corinna Cortes, Yishay Mansour, and Mehryar Mohri. 2010. Learning bounds for importance weighting. In *Nips*, volume 10, pages 442–450. Citeseer.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Miroslav Dudík, Steven Phillips, and Robert E Schapire. 2005. Correcting sample selection bias in maximum entropy density estimation. *Advances in neural information processing systems*, 18:323–330.

Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. 2004. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36.

Tongtong Fang, Nan Lu, Gang Niu, and Masashi Sugiyama. 2020. Rethinking importance weighting for deep learning under distribution shift. *arXiv preprint arXiv:2006.04662*.

Aditya Grover, Jiaming Song, Alekh Agarwal, Kenneth Tran, Ashish Kapoor, Eric Horvitz, and Stefano Ermon. 2019. Bias correction of learned generative models using likelihood-free importance weighting. *arXiv preprint arXiv:1906.09531*.

Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. 2006. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

Nils Reimers and Iryna Gurevych. 2019a. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019b. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.

Shivashankar Subramanian, Xudong Han, Timothy Baldwin, Trevor Cohn, and Lea Frermann. 2021. Evaluating debiasing techniques for intersectional biases. *CoRR*, abs/2109.10441.

Henry Weld, Xiaoqi Huang, Siqi Long, Josiah Poon, and Soyeon Caren Han. 2021. A survey of joint intent detection and slot-filling models in natural language understanding. *CoRR*, abs/2101.08091.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Bianca Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114.