David vs Goliath: Can PLStream surprise?

Comparative study of language capabilities between `PLStream` and supervised state-of-the-art models using `CheckList`

Iben Huse ibhu@itu.dk

Ludek Cizinsky luci@itu.dk

Lukas Rasocha lukr@itu.dk

Jonas-Mika Senghaas jsen@itu.dk

1 Abstract

`PLStream` is a novel framework for fast polarity labelling of massive data streams. In this paper we put the framework to a test by analysing its linguistic capabilities through `CheckList`, a general-purpose framework facilitating comprehensive testing of NLP models. The results are compared to two state-of-the art supervised models. Our analysis reveals shortcomings in `PLStream`'s overall ability to understand and learn from language - especially regarding not understanding contextual information and learning biases from training data.

2 Introduction

Today, data is generated at a higher speed than ever before. This is especially true for natural language data produced on online services such as Twitter or Amazon's review section.

In various business contexts it is critical for companies to understand the sentiment of this language data. For instance, to figure out the general opinion about a company's new product.

It is a standard practice to use supervised machine learning models to obtain high performance classification results. However, in high-velocity data systems, data freshness is often critical for high model performance, meaning that it is desirable to use the most recent data to update the machine learning model. Supervised methods therefore require to be retrained frequently to achieve outstanding performance. Yet, this recurring offline fine-tuning process takes significant time, making it incapable of keeping up with the high speed at which new data is generated (Wu et al., 2022).

Since one of the core bottlenecks of supervised models is the need for human annotation, eliminating human intervention in the classification pipeline is necessary for the model to scale. Unsupervised classification methods do not require these manual annotations and are therefore well-suited for this new type of challenge (Wu et al., 2022).

Although unsupervised classification of natural language data is not a new research topic (Turney, 2002; Baccianella et al., 2010; Taboada et al., 2011), it got recent attention in the research field, ultimately leading to the framework `PLStream` which is designed for classifying fast ongoing unlabelled data streams at scale on modern parallel machines. It is capable of producing consistent, high quality predictions, while maintaining high system throughput and low latency (Wu et al., 2022).

The goal of this paper is to analyse `PLStream`'s performance beyond simple accuracy using the `CheckList` approach (Ribeiro et al., 2020) and compare it to the language capabilities as measured by Ribeiro et al. (2020) of selected pre-trained state-of-the-art (SOTA) supervised learning methods. This led to the research question:

How do the language capabilities of the latest unsupervised polarity detection framework `PLStream` compare to the current SOTA-supervised classification models?

We hope our work creates starting points for further research into the development of unsupervised models to continue on increasing the performance of fully automatic classification systems.

3 Related work

3.1 PLStream

In the absence of ground truth labels, unsupervised learning approaches have to take a fundamentally different perspective to learn and infer polarity of natural language data. Older approaches such as lexicon-based systems define a manually crafted mapping of a fixed vocabulary of sentiment-carrying words to polarity measures (Baccianella et al., 2010). Just as other approaches, such as probabilistic models (Lin and He, 2009), it suffers from being too rigid and not scale beyond their original domains.

PLStream takes a novel approach, combining the research advances in the NLP (Mikolov et al., 2013) and large-scale stream processing (Carbone et al., 2015; Zhang et al., 2020) communities. For a massive online stream of data $S = D_1, D_2, ..., D_n$ with n approaching ∞, each incoming document D_i is used for both training and inference. Before tokenising D_i, user handles and any non-alphabetical characters are removed. The cleaned input sequence is then split on white-space characters. Finally, stop words are removed. (Wu et al., 2022).

Each token in D_i is represented through the word embeddings stored in a continuously, online trained word2vec model. The entire document D_i is summarised through the arithmetic mean of these.

Each document's embeddings are used to retrain the word2vec model, promising incrementally more accurate word embeddings that adapt dynamically to the streamed data. The updated model then predicts the sentiment of each document. The polarity inference step is done by computing the cosine similarity between the average document vector and the embeddings of a manually selected list of positive and negative reference words such as 'good' and 'bad' (the full list can be seen in the *Appendix*). The reference words' vectors are being dynamically learned together with all the other vectors. The cumulative similarities are stored in SUM_{POS} and SUM_{NEG} for the positive and negative reference words respectively and the sign of their difference is the final prediction of the model (Wu et al., 2022).

All processes in `PLStream` are designed to be run in parallel for the framework to scale to truly massive data streams. The distributed data system used was Apache Flink (Carbone et al., 2015), but the framework adapts easily to any stream-processing framework.

In the original paper, high quality predictions were synonymous with a convergence of the accuracy score to $\sim 80\%$. To provide a better picture of the frameworks' language capabilities, we use the `Checklist` framework.

3.2 CheckList

`CheckList` is a general-purpose framework facilitating comprehensive testing of linguistic capabilities of NLP models (Ribeiro et al., 2020). Being based on principles of behavioral testing in software engineering, it offers a testing environment allowing to create, structure and evaluate tests in different linguistic domains and test types. The linguistic domains tested usually depend on the type of NLP model tested. However, any test can be classified into one of three types:

1. **MFT**: Minimum Functionality Test
   Defines a single test case and an expected label. Test fails if model fails to predict expected label.

2. **INV**: Invariance Test
   Defines two test cases with slight differences. The minor change is not expected to have an influence on the prediction. Test fails if model's prediction confidence changes above some threshold.

3. **DIR**: Directional Expectation Test
   Defines two test cases with slight differences. The minor change is expected to have a directional influence on the prediction confidence. Test fails if model's prediction confidence does not change in the correct direction above some threshold.

4 Data

The `CheckList` framework is easily adjustable to incorporate custom tests. However, with `PLStream` being a sentiment analysis model, it was sensible to evaluate the model's performance on the default set of tests for sentiment analysis as proposed in the original paper (Ribeiro et al., 2020). This not only simplified the implementation process, but also allowed for easy comparison of `PLStream`'s checklist performance against SOTA supervised models already tested in the original paper.

The default sentiment test suite contained a total number of 38 tests in seven language capability dimensions. For each dimension a series of different tests were created. Each test consisted of synthetically generated variations allowing to compute a robust performance on the test. An overview of all performed tests, including thorough descriptions, number of tests run, expected model behaviors and example cases can be found in the *Appendix*.

To get reasonable results of `PLStream`'s performance on the `CheckList` tests, the tests needed to be appended to a training stream of data, so that the model had already learned accurate word embeddings before its performance was evaluated. Within this paper, `PLStream` was trained on the same data source as suggested by Wu et al. (2022) in the original paper. The Yelp review data set (Xiang, 2015) is a commonly used NLP dataset that was initially released as part of the *2015 Yelp Data Challenge*. It contains $560,000$ labelled reviews from the online review platform Yelp. Each review can either be positive or negative, the label distribution is balanced. Different amounts ($20,000$, $50,000$ and $100,000$) of reviews were used for training before evaluating on the checklist tests, leading to three different variants of the `PLStream` model, referred to as Ⓟ **20**, Ⓟ **50** and Ⓟ **100** from now.

5 Methodology

The primary goal was to get a trained `PLStream` model to predict on the set of linguistic tests proposed for sentiment analysis tasks within the `CheckList` paper.

The original `PLStream` source code (Wu et al., 2022) served as a starting point for all experiments. The model was trained and evaluated in Apache Flink's Stream Execution Environment through the Python API (Pyflink) on a single core. To simulate fast-pace incoming data streams, the Yelp review training data was concatenated with the `CheckList` test cases, which were then continuously fed to the model.

Evaluating supervised learning approaches requires a strict separation of data into training, (validation) and test splits to get unbiased results for the generalisation capabilities of the model. This is not true for `PLStream`, since training and inference happen simultaneously online. The model can't overfit to training data, because it has no information about true labels at any point in time.

In order to be able to evaluate the model's performance through `CheckList`'s default test suite a series of adjustments to the source code were necessary. `CheckList` requires probabilities for three sentiment labels ("negative", "neutral" or "positive"). However, in the current implementation of `PLStream` neither the three-way classification nor outputting probabilities is supported.

Adjusting `PLStream` to be a multiclass-classifier is non-trivial and beyond the scope of this project. In the absence of neutral labels, tests whose evaluation depended on this metric (marked ✖ in the *Appendix* Table 3, 4, 5) were unreasonable and therefore not considered for further analysis. This simplification had no effect on any results of this paper.

Since a majority of the `CheckList` tests (every **INV** and **DIR** test) assesses the change of probabilities, it was critical to alter `PLStream` to output prediction confidence scores. To make predictions, `PLStream` is computing the cumulative cosine similarity of the average sentence vector and all positive and negative reference words. Intuitively, high differences in the cosine similarities should lead to higher confidence in the results, while small differences can be interpreted as uncertainty. The cosine similarities were mapped to probability measures through the SOFTMAX function:

$$P(similarity)_i = \frac{e^{similarity_i}}{\sum_j^K e^{similarity_j}}$$

Our experiments show that this approach produced sensible results. These adjustments allowed to use the standard evaluation system of the Python Checklist API, which generated formatted summaries of the test performances.

6 Results and Discussion

Table 1 shows a comprehensive summary of the failure rates of all tested models for each test. The analysis focuses on the results of the family of `PLStream` models (Ⓟ **20**, Ⓟ **50**, Ⓟ **100**), in contrast to the results of a SOTA-pretrained BERT model (**Bert**) and the commercial Google model (**G**) reported by Ribeiro et al. (2020).

6.1 MFT

The `PLStream` family is outperformed by supervised classifiers in the majority of MFTs. This behavior is generally expected, since SOTA-supervised models learn deep contextual information and are therefore in general superior in making correct predictions. Nonetheless, some results are notable.

`PLStream` is expected to do well on single word tests, as for example the embedding of a positive word is presumed to be more similar to the embeddings of the set of positive reference words. While the `PLStream` family scores reasonably good results (5.9% and 28.6%), proving that the system has learned and is not guessing at random,

	Test Type and Capability	Failure Rates (%)					
		Bert	G	Ⓟ 20	Ⓟ 50	Ⓟ 100	Ⓟ Trend
MFT	Single Positive Word	**0**	14.7	**17.6**	5.9	5.9	↓
	Single Negative Word	**0**	**48.6**	22.9	25.7	28.6	↑
	Simple Negations 1	**8.4**	46.2	**64.6**	58.8	20	↓
	Simple Negations 2	**13.2**	54.2	57.4	86.6	**94.8**	↑
	Simple Negations 4	2.2	8.2	**19.4**	7.2	**0**	↓
	Simple Negations 5	98.6	90.4	97.6	100	**100**	↑
	Hard Negations 1	74	**100**	19.4	17	**8.4**	↓
	Hard Negations 2	**99.8**	**70.8**	94.2	94.2	99	↑
INV	Change Neutral Words (BERT)	10.2	**16.2**	3.0	6.0	**2.6**	↓
	Add Random URLs and Handles	11.4	**13.4**	**6.0**	8.0	13.2	↑
	Typos	5.2	**10.2**	2.8	2.4	**1.2**	↓
	Contractions	2.8	**3.0**	**0.4**	0.6	**0.4**	-
	Punctuation	4.2	**7.6**	1.0	0.6	**0.4**	↓
	Protected: Race	**88.6**	**0.6**	66.0	60.0	66.6	-
	Protected: Sex	90	**1.2**	**97.8**	73.8	71.4	↓
DIR	Intensifiers	**0.8**	1.7	**19.4**	16.6	23.6	-
	Reducers	**0**	1.5	6.2	**20.6**	8.7	-
	”Used to” should reduce	**100**	**2.0**	10.9	27.4	30.4	↑

Table 1: *(Excerpt of Table 2)* Checklist Failure Rates (*Lowest Failure Rate* and *Highest Failure Rate are highlighted; trend arrow indicates if performance change for larger stream sizes of PLStream*)

the failure rates are still higher than expected. The difference in the performance of single positive and negative words is not inherent from how the system works. We therefore believe it to be reasonable that the divergence can be attributed to the training data set not including some of the test words (enough).

Negations are generally challenging for any kind of NLP model. This is especially true for the `PLStream` family, as the representation of documents through mean sentence vectors removes any kind of contextual information that is necessary to detect the influence of negating words. Furthermore, eliminating stop words during tokenisation (including ”not”) makes the model blind towards some types of negation in a first place. The overall performance on negation tests is therefore expected to be poor. Interestingly, a pattern evolved leading to surprisingly good results on half of the negation tests. Looking closer, good performance was achieved on those tests that expected a negative label (SN1, SN4 and HN1), while the model struggled on all tests expecting a positive label (SN2, SN3, HN2). Reviewing some of the failed tests closely, a systematic bias towards negative sentiment prediction was apparent. No matter the expected sentiment, the model naively predicted a negative polarity. This might be again caused by a bias in the training set towards negative words appearing more frequently. This hypothesis is further supported by the fact that larger training sizes led to decreasing performances.

6.2 INV

Invariance tests expect NLP models to not change their confidence in the sentiment label, despite slight changes in the input document. The `PLStream` family proved to be robust against the vast majority of invariance tests, mostly outperforming both supervised methods. All variants of `PLStream` make predictions independent of additions or removals of neutral words, URLS, user handles, contractions, punctuation and typos. The robustness of `PLStream` in these cases can be explained by the inference procedure. Adding or removing punctuation, contractions or user handles has little to zero effect on the model’s polarity inference due to the tokenisation process. Moreover, representing each document through an average leads to individual tokens having only a small effect on the final prediction, especially in long documents. Furthermore, a typo might lead to a token not being present in the word2vec model. This token then does not contribute to the average sentence vector. All these combined explain `PLStream`'s robustness on the above tests.

On the other hand, the `PLStream` family performs poorly on all fairness tests. While changing races and sexes should not influence predictions, `PLStream` showed systematic biases in protected classes.

Destiny is a **black** women. (40% N, 60% P)
Destiny is an **asian** women. (60% N, 40% P)

Example of a failed test case of Ⓟ **100**

The unfairness might be caused by biases in the input data.

6.3 DIR

Intensifiers are supposed to increase a model's confidence in its prediction (in both the positive and negative direction depending on the context), while reducers should decrease the model's confidence. Such a contextual change of confidence can never be achieved by `PLStream`, since every word is encoded through a single word vector and will therefore always effect the sentiment in the same direction independently of the context. In the sentence *”I (definitely) like this”*, the intensifier ”definitely” is expected to result in more confidence in a positive sentiment, while in *”I (definitely) do not like this”*, it is expected to result in more confidence in a negative sentiment. Since ”definitely” can only be encoded through a single vector, the model is doomed to fail on at least one of the above cases.

Indeed, the intensifier and reducer tests proved to be problematic for `PLStream`, especially compared to supervised classifiers. This can be seen in the below failed test, in which the reducer ”reasonably” had a positive polarity and therefore functioned as an intensifier, rather than a reducer.

The plane is adorable. (80% P)
The plane is **reasonably** adorable. (90% P)

Example of a failed test case of Ⓟ **100**

In missing capabilities of encoding contextual and temporal information, the `PLStream` family also failed on temporal test cases like the *Used To, But Now* tests. Ⓟ **100** achieved the worst failure rate of 30.4%.

This was a lousy crew (70% N)
I used to think this was a lousy crew (83% N)

Example of a failed test case of Ⓟ **100**

7 Findings

`PLStream` intentionally sacrifices complexity in the way it learns and infers labels in order to scale to massive data streams. It is therefore not expected to be able to compete in its language understanding capabilities with SOTA-supervised models. Most of the conducted tests confirm this hypothesis.

The analysis showed that `PLStream` lacks general predictive performance measured in accuracy compared to SOTA-supervised models (MFT tests). Furthermore, its simplistic approach makes it incapable of using contextual or temporal information in its decision-making (DIR tests and negation tasks). Furthermore, systematic biases connected to protected classes were found. These, however, cannot be attributed to flaws of `PLStream` directly, but are likely to be caused by our choice of training data. However, one evident advantage originating from the simplicity of the model was the model's robustness as shown in the invariance tests. In these tests `PLStream` even outperformed SOTA supervised models.

8 Concluding Remarks

To have an unsupervised model that continuously trains itself on online data streams of potentially huge data sizes, which can give similar performance guarantees as supervised SOTA classifiers would be a break-through technology in the NLP domain. In this study we attempted to test the language capabilities and limitations of `PLStream` within the `CheckList` framework.

The original paper proves `PLStream`'s good predictive accuracy while maintaining consistently high throughput and low latency (Wu et al., 2022). Our analysis, however, revealed shortcomings in the overall ability to understand language. We hope that future work can build upon our findings to enhance unsupervised polarity detection systems by improving `PLStream`'s understanding of contextual information. For current and future users of `PLStream`, we strongly recommend to get familiar with the flaws of the framework in detail before using them in real-world contexts.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).

Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache flink™: Stream and batch processing in a single engine. *IEEE Data Eng. Bull.*, 38:28–38.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, page 375–384, New York, NY, USA. Association for Computing Machinery.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-based methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307.

Peter D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews.

Huilin Wu, Mian Lu, Zhao Zheng, and Shuhao Zhang. 2022. A framework for fast polarity labelling of massive data streams.

Zhang Xiang. 2015. Yelp reviews - full.

Shuhao Zhang, Yingjun Wu, Feng Zhang, and Bingsheng He. 2020. Towards concurrent stateful stream processing on multicore processors. pages 1537–1548.

9 Appendix

9.1 Group Contributions

Contribution was evenly spread amongst group members.

9.2 Github

The source code of this project and instructions for reproducibility can be found both on ITU's and public GitHub.

1. ITU GitHub
2. Public GitHub

The project was developed on Public GitHub. the commit history is therefore only visible there.

9.3 Pre-Project

The results and source code of Phase 1 and 2 of this project (including getting baseline performance scores on music reviews and creating and testing difficult cases) are documented in detail on GitHub.

9.4 Positive and Negative Reference Words

'love', 'best', 'beautiful', 'great', 'cool', 'awesome', 'wonderful', 'brilliant', 'excellent', 'fantastic'

Positive Reference Words

'bad', 'worst', 'stupid', 'disappointing', 'terrible', 'rubbish', 'boring', 'awful', 'unwatchable', 'awkward'

Negative Reference Words

	Test Type and Capability	Failure Rates (%)					
		Bert	G	Ⓟ 20	Ⓟ 50	Ⓟ 100	Ⓟ Trend
MFT	Single Positive Word.	**0**	14.7	**17.6**	5.9	5.9	↓
	Single Negative Word	**0**	**48.6**	22.9	25.7	28.6	↑
	Sentiment-Laden Words in Context	**0**	15.0	23.8	23.2	**30.2**	↑
	Used To, But Now	**18.8**	36.6	**46.8**	46.6	46.8	-
	Simple Negations 1	**8.4**	46.2	**64.6**	58.8	20	↓
	Simple Negations 2	**13.2**	54.2	57.4	86.6	**94.8**	↑
	Simple Negations 4	2.2	8.2	**19.4**	7.2	**0**	↓
	Simple Negations 5	98.6	90.4	97.6	100	**100**	↓
	Hard Negations 1	74	**100**	19.4	17	**8.4**	↓
	Hard Negations 2	**99.8**	**70.8**	94.2	94.2	99	↑
	My Opinion is what Matters	**38.8**	**62.4**	48.4	48.8	46.8	-
	Q&A: Yes	**3.6**	**57.6**	35.6	32	44.2	↑
	Q&A: NO	**55.4**	**90.8**	57	64	55.8	↓
INV	Change Neutral Words (BERT)	10.2	**16.2**	3.0	6.0	**2.6**	↓
	Add Random URLs and Handles	11.4	**13.4**	**6.0**	8.0	13.2	↑
	Punctuation	4.2	**7.6**	1.0	0.6	**0.4**	↓
	Typos	5.2	**10.2**	2.8	2.4	**1.2**	↓
	2 Typos	11.2	**13.6**	**4.8**	5.0	6.8	↑
	Contractions	2.8	**3.0**	**0.4**	0.6	**0.4**	-
	Change Names	6.6	**15.1**	**1.2**	2.1	5.4	↑
	Change Locations	7.6	**20.8**	**4.0**	4.2	4.6	↑
	Change Numbers	2.2	**7.6**	**0**	**0**	**0**	-
	Protected: Race	**88.6**	**0.6**	66.0	60.0	66.6	-
	Protected: Sex	90	**1.2**	**97.8**	73.8	71.4	↓
	Protected: Religion	**96.4**	**1.6**	38.6	26.8	8.0	↓
	Protected: Nationality	14.4	**0.4**	**93.6**	64.0	91.4	-
DIR	Intensifiers	**0.8**	1.7	**19.4**	16.6	23.6	-
	Reducers	**0**	1.5	6.2	**20.6**	8.7	-
	Add Positive Phrases	**0.2**	**12.4**	12.2	10.2	10.2	↓
	Add Negative Phrases	**0**	**34.6**	9.4	12.8	6.0	↓
	”Used to” should reduce	**100**	**2.0**	10.9	27.4	30.4	↑

Table 2: Checklist Failure Rates on SOTA supervised models compared to different PLStream configurations (*Lowest Failure Rate* and *Highest Failure Rate* are highlighted; trend arrow indicates if performance change for larger stream sizes of PLStream)

Test Type and Capability	Test size		Description	Example
SPW: Single Positive Word	✓	34	single positive words	great
SNW: Single Negative Word	✓	35	single negative words	bad
SNW: Single Neutral Word	✗	13	single neutral words	bad
SLW: Sentiment-Laden word in context	✓	500	Use positive and negative verbs and adjectives with airline nouns such as seats, pilot, flight, etc.	We dreaded that cabin crew.
NWC: Neutral Words in Context	✗	500	Use neutral verbs and adjectives with airline nouns such as seats, pilot, flight, etc.	That seat is private.
UBN: Used to, but now	✓	500	Have two conflicting statements, one about the past and one about the present. Expect the present to carry the sentiment.	I think this airline is adorable, I used to think it was horrible.
SN1: Simple Negations 1	✓	500	Very simple negations of positive statements	That wasn't an awesome airline.
SN2: Simple Negations 2	✓	500	Very simple negations of negative statements. Expectation requires prediction to NOT be negative (i.e. neutral or positive)	I don't abhor this airline.
SN3: Simple Negations 3	✗	500	Negating neutral statements should still result in neutral predictions	It is not a British service.
SN4: Simple Negations 4	✓	500	simple negations: I thought x was positive, but it was not (should be negative)	I thought this food would be wonderful, but it was not.
SN5: Simple Negations 5	✓	500	simple negations: I thought x was negative, but it was not (should be neutral or positive)	I thought this service would be terrible, but it was not.
SN6: Simple Negations 6	✗	500	simple negations: but it was not (neutral) should still be neutral	I thought that seat would be Italian, but it wasn't.
HN1: Hard Negations 1	✓	500	Hard: Negation of positive with neutral stuff in the middle (should be negative)	I can't say, given the time that I've been flying, that the seat was exciting.
HN2: Hard Negations 2	✓	500	Hard: Negation of negative with neutral stuff in the middle (should be positive or neutral)	i can't say, given that I am from Brazil, that the is a frustrating seat.
NNM: negation of neutral with neutral in the middle	✗	500	negation of neutral with neutral in the middle, should still be neutral	I can't say, given my history with airplanes, that that company is Australian.
MOM: My Opinion is what Matters	✓	500	Have conflicting statements where the author has an opinion and a third party has a contrary opinion. Expect sentiment to be the authors'.	I think you are lousy, but some people think you are sweet.
QAY: Q&A: Yes	✓	500	Question, that when replied with yes carries a positive sentiment	Did I recommend that pilot? Yes
QAYN: Q&A: Yes (Neutral)	✗	500	Question, that when replied with yes carries a neutral sentiment	Do I think that service was commercial? Yes
QAN: Q&A: NO	✓	500	Question, that when replied with no carries a negative sentiment	Do I think that is a fun plane? No
QANN: Q&A: NO (Neutral)	✗	500	Question, that when replied with no carries a neutral sentiment	Do I think that was an Australian customer service? No

Table 3: **MFT**: Capabilities, description and examples. *(symbols ✓ and ✗ are used to indicate whether the test was used in our checklist testing)*

Test Type and Capability	Test size		Description	Example
CNW: Change Neutral Words (BERT)	✓	500	Change a set of neutral words with other context-appropriate neutral words (using BERT).	@SouthwestAir no flights to HRL :(is this a limited route? @SouthwestAir no flights over HRL :(is this a limited route?
AR: Add Random URLs and Handles	✓	500	add randomly generated urls and handles to the start or end of sentence	@AmericanAir thanks for reFlight Booking Problems me on a flight and then leaving my luggage in NYC. Unbelievable. @T3vlJ8 @AmericanAir thanks for reFlight Booking Problems me on a flight and then leaving my luggage in NYC. Unbelievable.
PUN: Punctuation	✓	500	strip punctuation and / or add "."	@virginamerica spruce moose. @virginamerica spruce moose !
TYP: Typos	✓	500	Add one typo to input by swapping two adjacent characters	@united Great, thank you!! I'll send it now. @united Great, thank yuo!! I'll send it now.
2TYP: 2 Typos	✓	500	Add two typos to input by swapping two adjacent characters twice	@USAirways thanks! @USAiwrays thanks!
CON: Contractions	✓	500	Contract or expand contractions, e.g. What is → What's	@JetBlue I did get the email. Thought i wasn't supposed to reply to those @JetBlue I did get the email. Thought i was not supposed to reply to those
CN: Change Names	✓	331	Replace names with other common names	@AmericanAir Nah, just horrible dining options outside of club. Luckily Manuel in the Admirals can make a Bloody Mary. @AmericanAir Nah, just horrible dining options outside of club. Luckily Christian in the Admirals can make a Bloody Mary.
CL: Change Locations	✓	500	Replace city or country names with other cities or countries	@USAirways 5223 out of Ocoee, FL @USAirways 5223 out of Coachella, FL
CNUM: Change Numbers	✓	500	Replace integers with random integers within a 20% radius of the original	no seat assignment 5 mins before a flight...line 20 people deep. Guess I'm sitting with the baggage? @MarinaDomine @JetBlue no seat assignment 3 mins before a flight...line 20 people deep. Guess I'm sitting with the baggage? @MarinaDomine
PRA: Protected: Race	✓	500	Prediction should be the same for various adjectives within a protected class	Destiny is a black teacher. Destiny is a white teacher.
PS: Protected: Sex	✓	500	Prediction should be the same for various adjectives within a protected class	Rachel is a heterosexual twin. Rachel is a queer twin.
PRE: Protected: Religion	✓	500	Prediction should be the same for various adjectives within a protected class	Aaron is a Christian individual. Aaron is an Ahmadiyya individual.
PN: Protected: Nationality	✓	500	Prediction should be the same for various adjectives within a protected class	Tracy is a Chinese name. Tracy is an Ethiopian name.

Table 4: **INV**: Capabilities, description and examples. *(symbols ✓ and ✗ are used to indicate whether the test was used in our checklist testing). The green text illustrates the changes between the examples.*

Test Type and Capability	Test size		Description	Example
IN: Intensifiers	✓	500	Test is composed of pairs of sentences (x1, x2), where we add an intensifier such as ”really”,or ”very” to x2 and expect the confidence to NOT go down (with tolerance=0.1). We disregard cases where the prediction of x1 is neutral.	We like the airline We genuinely like the airline.
RE: Reducers	✓	500	Test is composed of pairs of sentences (x1, x2), where we add a reducer such as ”somewhat”, or ”kinda” to x2 and expect the confidence to NOT go up (with tolerance=0.1). We disregard cases where the prediction of x1 is neutral.	This aircraft was ugly. This aircraft was a little ugly.
APP: Add Positive Phrases	✓	500	Add very positive phrases (e.g. I love you) to the end of sentences, expect probability of positive to NOT go down (tolerance=0.1)	@JetBlue you want me to talk to the wall? @JetBlue you want me to talk to the wall. You are nice.
ANP: Add Negative Phrases	✓	500	Add very negative phrases (e.g. I hate you) to the end of sentences, expect probability of positive to NOT go up (tolerance=0.1)	NO“@JetBlue: Our fleet’s on fleek. http://t.co/UgFCKErmrW” NO“@JetBlue: Our fleet’s on fleek. http://t.co/UgFCKErmrW. Never flying with you again.
USR: ”Used to” should reduce	✓	500	A model should not be more confident on ”I used to think X” when compared to ”X”	this was an amazing staff. I used to think this was an amazing staff.

Table 5: **DIR**: Capabilities, description and examples. *(symbols ✓ and ✗ are used to indicate whether the test was used in our checklist testing). The green text illustrates the changes between the examples.*