Comparative Analysis of Human and ChatGPT Empathetic Dialogues

Andrew Maho William Jan François-Xavier Bonnefont Zaher Hamadeh

November 1, 2025

Abstract

This report consolidates our investigations on the EmpatheticDialogues (human) and 2GPTEmpathicDialogues (ChatGPT) corpora. We describe the preprocessing workflow, linguistic feature engineering with a focus on language style matching (LSM), and modelling experiments that separate human from synthetic conversations. Each methodological step is supported by code excerpts and quantitative outputs accompanied by short analytical comments.

1 Introduction

Large language models (LLMs) are increasingly deployed in emotionally charged conversations. EmpatheticDialogues and its synthetic mirror, 2GPTEmpathicDialogues, offer a controlled setting to study linguistic divergence between human pairs and two ChatGPT agents. The reference study "Linguistic Comparison between Human and ChatGPT-Generated Conversations" addresses the same task through descriptive statistics, affective features, and supervised detection, reporting that transformer embeddings deliver accuracy above 95% while handcrafted affect indicators expose stylistic polishing. Our objective is to reproduce and critically extend these findings through a unified workflow.

2 Data Preparation and Prompt Analysis

2.1 Corpus alignment

Notebook Extract 1 shows the alignment routine used to harmonise conversation identifiers and reshape both corpora at the conversation level.

Notebook Extract 1: *Conversation alignment and aggregation.*

```
human_byutt["utterance"] = human_byutt["utterance"].str.replace("
    _comma_", ",", regex=False)
human_byconv = human_byutt.groupby("conv_id").agg({
  "context": "first",
  "prompt": "first",
  "speaker_idx": "nunique",
  "utterance": lambda x: "␣".join(x)
})

gpt_byutt = (gpt_byconv.assign(utterance=gpt_byconv["processed"].str.
    split("\n"))
       .explode("utterance").query("utterance␣!=␣''"))
gpt_byutt["utterance_idx"] = gpt_byutt.groupby("conv_id").cumcount() +
    1
gpt_byutt["speaker_idx"] = gpt_byutt["utterance_idx"] % 2
```

Analysis. After filtering conversations with a single speaker, the workflow retains 19 531 aligned pairs. ChatGPT dialogues average 5.27 turns and 188.8 words per conversation, whereas human dyads average 4.31 turns and 57.7 words. The synthetic corpus therefore delivers longer, highly structured exchanges that must be normalised before feature comparison.

2.2 Context curation

To avoid unstable statistics on rare scenarios, we enforce a minimum frequency threshold, as illustrated in Notebook Extract 2.

Notebook Extract 2: *Context filtering heuristic.*

```
MIN_CONTEXT_SIZE = 20
context_summary = (human_byconv['context'].value_counts()
        .rename('pairable_conversations').to_frame()
        .assign(keep=lambda df: df['pairable_conversations'] >=
           MIN_CONTEXT_SIZE))
rare_context_summary = context_summary.query('~keep')
selected_contexts = context_summary.query('keep').index.tolist()
```

Analysis. Contexts with fewer than 20 aligned conversations (e.g., prompts labelled "Other") are documented in `rare_context_summary` and excluded from downstream analysis. The retained set still covers more than 95% of the data, ensuring reliable stratified evaluation.

2.3 Prompt design

Sampling representative prompts reveals the templated structure used to instruct ChatGPT agents.

Notebook Extract 3: *Prompt inspection.*

```
prompt_examples = (gpt_byconv[["context", "prompt"]].reset_index()
        .sample(n=3, random_state=42))
```

Analysis. Prompts specify the emotional context, role allocation, and explicit empathy instructions. This templating enforces balanced turn-taking but injects repeated lexical patterns that later translate into higher LSM scores for the synthetic corpus.

3 Linguistic Feature Engineering

3.1 LSM computation

We implement LSM by tagging each speaker's concatenated utterances with spaCy and comparing part-of-speech distributions, as shown in Notebook Extract 4.

Notebook Extract 4: *LSM computation pipeline.*

```
CATEGORIES = {
  "pronouns": ["PRP", "PRP$", "WP", "WP$"],
  "articles": ["DT"],
  "prepositions": ["IN"],
  "aux_verbs": ["MD", "AUX"],
  "adverbs": ["RB", "RBR", "RBS"],
  "conjunctions": ["CC"],
  "negations": ["RB"],
}

def conv_lsm_score(doc_A, doc_B):
  def pos_freqs(doc):
```

```
    pos_counts = defaultdict(int)
    total = 0
    for token in doc:
      if token.is_alpha:
        total += 1
        cat = TAG2CAT.get(token.tag_)
        if cat:
          pos_counts[cat] += 1
    return {cat: pos_counts[cat] / total if total > 0 else 0 for cat in
        CATEGORIES}
  freq_A, freq_B = pos_freqs(doc_A), pos_freqs(doc_B)
  lsm_scores = {cat: 1 - abs(freq_A[cat] - freq_B[cat]) / (freq_A[cat]
     + freq_B[cat] + 1e-5)
        for cat in CATEGORIES}
  return sum(lsm_scores.values()) / len(lsm_scores)
```

Analysis. Mean LSM equals 0.621 for humans and 0.733 for ChatGPT (+18%), confirming that synthetic agents mirror each other's style more closely. These metrics become key discriminators in later models.

3.2 Scenario diagnostics

The per-context LSM aggregation highlights where stylistic gaps are largest.

Notebook Extract 5: *Context-level LSM aggregation.*

```
lsm_by_context = (lsm_by_conversation.groupby(['context', 'source'])['
    LSM']
        .mean().unstack('source')
        .rename(columns={'chatgpt': 'LSM␣GPT', 'human': 'LSM␣human'}))
lsm_by_context['delta'] = lsm_by_context['LSM␣GPT'] - lsm_by_context['
    LSM␣human']
focus_contexts = (lsm_by_context.loc[selected_contexts]
        .assign(abs_delta=lambda df: df['delta'].abs())
        .sort_values('abs_delta', ascending=False))
```

Analysis. High-intensity emotions such as *furious*, *jealous*, and *anxious* show the largest LSM deltas, signalling scenarios where ChatGPT's stylistic mirroring diverges most from human behaviour. These contexts guide targeted evaluation.

4 Modelling Experiments

4.1 Embedding baseline

We encode each conversation with Sentence-BERT and fit a logistic regression classifier (Notebook Extract 6).

Notebook Extract 6: *Embedding baseline training.*

```
X_train_emb, X_test_emb, y_train_emb, y_test_emb, train_idx, test_idx =
    train_test_split(
  embeddings_mini,
  labels,
  np.arange(len(labels)),
  test_size=0.2,
  stratify=labels,
  random_state=0
)
```

```
baseline_clf = Pipeline([
  ('logistic', LogisticRegression(max_iter=1000, solver='liblinear'))
])
baseline_clf.fit(X_train_emb, y_train_emb)

baseline_proba = baseline_clf.predict_proba(X_test_emb)[:, 1]
baseline_metrics = {
  'accuracy': accuracy_score(y_test_emb, baseline_clf.predict(
     X_test_emb)),
  'roc_auc': roc_auc_score(y_test_emb, baseline_proba)
}
```

Analysis. The embedding-only detector achieves 0.971 accuracy and a ROC-AUC close to 0.99, aligning with the reference study and confirming that semantic embeddings capture major stylistic cues.

4.2 LSM-only classifier

To quantify the standalone predictive power of LSM, we train linear models on the LSM feature alone or combined with dialogue length (Notebook Extract 7).

Notebook Extract 7: *Comparison of LSM-only and LSM+length classifiers.*

```
lsm_length_results = pd.DataFrame([
  {'model': 'SVM', 'features': 'LSM', 'accuracy': accuracy_score(
     y_test_cls, svm_lsm.predict(X_test_lsm))},
  {'model': 'SVM', 'features': 'LSM␣+␣length', 'accuracy':
     accuracy_score(y_test_cls, svm_both.predict(X_test_both))},
  {'model': 'LogisticRegression', 'features': 'LSM', 'accuracy':
     accuracy_score(y_test_cls, logreg_lsm.predict(X_test_lsm))},
  {'model': 'LogisticRegression', 'features': 'LSM␣+␣length', 'accuracy
     ': accuracy_score(y_test_cls, logreg_both.predict(X_test_both))}
])
```

Analysis. LSM alone yields roughly 0.67 accuracy, while adding dialogue length improves scores by several points, indicating that structural pacing complements stylistic similarity when discriminating human and ChatGPT exchanges.

4.3 Augmented feature stack

Finally, we extend the embedding baseline with affective features (Notebook Extract 8).

Notebook Extract 8: *Embedding + affect feature fusion.*

```
feature_frame = (extraction.set_index(['source', 'conv_id'])[
   emotion_features]
       .reindex(tot_byconv.index).fillna(0))
scaled_features = StandardScaler().fit_transform(feature_frame)

augmented_embeddings = np.hstack([embeddings_mini, scaled_features])
X_train_aug = augmented_embeddings[baseline_split['train_idx']]
X_test_aug = augmented_embeddings[baseline_split['test_idx']]

augmented_clf = Pipeline([
  ('logistic', LogisticRegression(max_iter=1000, solver='liblinear'))
])
augmented_clf.fit(X_train_aug, baseline_split['y_train'])

y_proba_aug = augmented_clf.predict_proba(X_test_aug)[:, 1]
augmented_metrics = {
```

```
  'accuracy': accuracy_score(baseline_split['y_test'], augmented_clf.
     predict(X_test_aug)),
  'roc_auc': roc_auc_score(baseline_split['y_test'], y_proba_aug)
}
performance_delta = {
  'accuracy_gain': augmented_metrics['accuracy'] - baseline_metrics['
     accuracy'],
  'roc_auc_gain': augmented_metrics['roc_auc'] - baseline_metrics['
     roc_auc']
}
```

Analysis. Accuracy increases to 0.972 with a modest ROC-AUC gain of approximately 0.01. Scenario-wise deltas highlight the largest improvements for the previously identified high-intensity contexts, validating the inclusion of affective features.

5 Results Summary

Model	Feature set	Accuracy	ROC-AUC	Interpretation
Logistic regression	LSM only	≈ 0.675	–	Stylistic alignment alone partially separates corpora but leaves substantial overlap.
Logistic regression	Sentence-BERT embeddings	0.971	≈ 0.99	Semantic embeddings encode most discriminative cues between human and ChatGPT dialogues.
Logistic regression	Embeddings + LSM + affect features	0.972	≈ 0.99	Affect-aware features offer consistent yet marginal gains, especially on high-intensity contexts.

6 Discussion and Conclusion

Our unified workflow demonstrates that (i) preprocessing decisions, particularly context filtering, stabilise downstream comparisons; (ii) ChatGPT prompts introduce strong stylistic alignment, reflected in elevated LSM scores; and (iii) transformer embeddings already yield near-perfect separation, with affective extensions providing incremental benefits. Future work should diversify prompts to reduce templating artefacts, enrich affect descriptors beyond LIWC-inspired categories, and evaluate robustness on mixed human–LLM interactions.