Question 1: Feature Extraction

The feature extraction uses a bag-of-words representation that converts each tweet into a feature dictionary mapping tokens to frequency counts. The `to_feature_vector()` function iterates through preprocessed tokens and counts their occurrences, while a global dictionary tracks unique tokens across the corpus.

Question 2: Cross-Validation

10-fold cross-validation was used to train the Linear SVM on 9 folds and testing on the remaining fold. Using `precision_recall_fscore_support` with weighted averaging, I computed metrics for each fold and calculated means. With `random.seed(42)` for reproducibility, baseline results were: Precision = 0.8290, Recall = 0.8310, F1 = 0.8294, Accuracy = 0.8310.

Question 3: Error Analysis

The confusion matrix revealed 255 false positives and 212 false negatives. Numerous patterns were noticed.

The bag-of-words model treats all words with equal importance allowing common neutral words (e.g "we", "the") to dilute words that carry signal. For example, In "ruined my Sunday night," features "the," "we," "time" held as much importance as "ruined".

Negation handling failed critically. For example, "didn't have," "not good" were misclassified because negation tokens were treated independently from modified words.

Mixed-sentiment tweets proved challenging. For example, "Slept 11 hours... odd mood... play offs tonight and Jack White Sunday!!!" (labeled negative) was predicted positive because the bag-of-words model weighted positive tokens ("tonight," "Sunday," "!!!") more heavily than the negative mood indicator ("odd mood").

False positives often involved sarcasm ("#ThankYouObama" used ironically) while false negatives contained context-dependent slang ("bad" meaning good, "sick" meaning awesome).

Question 4: Optimization

Improved pre-proccessing and feature extraction

For optimising pre-processing and feature extraction, techniques were tested systematically in isolation. First, I evaluated individual improvements (lowercasing, bigrams, lemmatization, etc.) separately against the baseline to see what each contributed on its own. Next, the most effective techniques were combined into a single preproccessing function. Twitter noise removal (the removal of hashtags, @mentions, etc) was implemented on top of this combined system for further experimentation. Table 1 shows results from these stages

Method	**F1**	**Δ F1**	**Acc**	**Δ Acc**
Baseline (simple split)	0.8294	–	0.8310	–
Lowercase + word_tokenize	0.8519	+0.0225	0.8526	+0.0216
Bigrams	0.8436	+0.0142	0.8460	+0.0150
Lemmatization	0.8418	+0.0124	0.8430	+0.0120
Stopword removal	0.8220	-0.0074	0.8240	-0.0070
Negation tagging	0.8324	+0.0030	0.8339	+0.0029
Combined (best techniques)	0.8698	+0.0380	0.8709	+0.0376
Twitter noise removal	0.8685	-0.0013	0.8696	-0.0013

Table 1: Cross-validation results showing isolated and combined technique performance

Lowercasing and word_tokenize (+2.0% F1): Lowercasing reduces feature sparsity by treating case variants identically. NLTK's `word_tokenize()` separates punctuation and splits contractions linguistically ("don't" → "do", "n't"), capturing negation and emotional signals.

Bigrams (+1.42% F1): Bigrams create features from adjacent word pairs, directly addressing negation failures. While unigrams treat "not" and "good" independently, "not_good" becomes a distinct negative feature, correctly classifying previously misclassified negated phrases.

Lemmatization (+1.24% F1): WordNetLemmatizer reduces words to base forms ("movies" → "movie"), consolidating morphological variants. This enables recognition that "loving," "loved," and "loves" express identical sentiment.

Stopword removal (-0.74% F1): Performance decreased. This is likely because stopwords like "but," "very," and "too" carry sentiment-relevant information.

Negation tagging (+0.30% F1): Minimal improvement occurred because bigrams already capture negation effectively through features like "n't_like."

Combined techniques (+4.04% F1): Combining lowercasing, word_tokenize, bigrams, and lemmatization achieved near-additive improvements, demonstrating complementary benefits.

Twitter noise removal (-0.13% F1): Removing @mentions, URLs, and hashtags decreased performance. Twitter-specific elements carry sentiment signals—hashtags express sentiment directly (e.g #love, #terrible), and retweets indicate endorsement. Removal of these is counterproductive for social media sentiment analysis.

Improving Model Performance

Building on the optimized preprocessing (F1 = 0.8698), different model architectures and feature engineering was used to further improve model performance.

Each enhancement was added subsequently on top of the enhancement prior. For example, LogisicRegression and TF-IDF (with C=2.0) were implemented simultaneously, and then hyperparameter tuning (chaniging C to 5.0) was added on top of this. i.e, the Δ F1 and Δ Acc values are increments upon the F1 and and Acc values in the row above. As previously, these metrics and improvements come from using model on the training data at the cross-validation stage. The final F1 = 0.8861 and Accuracy = 0.8859 can be observed in the submitted NLP_Q4.ipynb when notebook is executed from top cell down to the cell with the cross-validation procedure.

Method	F1	Δ F1	Acc	Δ Acc
Optimized preprocessing baseline	0.8698	–	0.8709	–
+ LogisticRegression + TF-IDF (C=2.0)	0.8784	+0.0086	0.8780	+0.0071
+ Hyperparameter tuning (C=5.0)	0.8792	+0.0008	0.8791	+0.0011
+ Character n-grams (3-5 chars)	0.8792	+0.0000	0.8791	+0.0000
+ Opinion lexicon features	0.8852	+0.0060	0.8850	+0.0059
+ Trigrams (1-3 word n-grams)	0.8861	+0.0009	0.8859	+0.0009

Table 2: Sequential model improvements from optimized preprocessing baseline

LogisticRegression + TF-IDF (+0.86% F1): TF-IDF weighting downweights common words while emphasizing distinctive terms, addressing function word dominance identified in error analysis.

Hyperparameter Tuning (+0.08% F1): Increasing regularization parameter to C=5.0 with `class_weight='balanced'` optimized the decision boundary and handled class imbalance.

Character N-grams: Character n-grams (3-5) were tested but provided no observable improvement. This suggests that bigrams already captured sufficient patterns.

Opinion Lexicon (+0.6% F1): Incorporating Hu and Liu's sentiment lexicon provided explicit sentiment signals, complementing learned TF-IDF features with domain knowledge.

Trigrams Trigrams (+0.09% F1): Extending to 3-word n-grams (1,3) captured longer negation patterns like that bigrams cannot represent as single features.

Final Performance

This final logistic regression model achieved F1 = 0.8861 and Accuracy = 0.8859, representing substantial improvements of 5.67% and 5.49% over the baseline system (F1 = 0.8294, Accuracy = 0.8310).

After cross-validation, the optimized model was trained on all 26,832 training samples and evaluated on the 6,708 held-out test samples.

Final test performance: F1 = 0.8791, Accuracy = 0.8788. This is a difference of just Δ F1 = 0.70% and Δ Acc = 0.71% compared to cross-validation results. Thus, the model shows reliable generalization with minimal overfitting .