Approach

1. **Data Understanding**
   a. **Data Structure**

      Train data set contains 20800 rows, and 5 columns

      Id: unique identifier

      Title: title of the article

      Author: author of the article

      Text: article

      Label: 1 for fake news, 0 for real news

   b. **Missing Values**
      i. There were missing values in both test and train dataset in text columns

```
In [44]: ## Check for missing values
         train.isnull().sum()

Out[44]: id           0
         title      558
         author    1957
         text        39
         label        0
         dtype: int64
```

Author column has more than 5% missing values - cannot drop all rows. Imputing with empty text

```
In [45]: test.isnull().sum()

Out[45]: id          0
         title     122
         author    503
         text        7
         dtype: int64
```

      Imputed empty string " " to treat missing values.

   c. **Data Cleaning:**
      i. Removing punctuations and digits and stopwords (from nltk) from text
      ii. Word tokenization and Lemmatizing nouns, to capture action words for feature engineering

   d. **Feature Engineering**
      i. Quantifying the following hypothesis about fake news
         1. Polarizing words
         2. Vague and opinionated - less citations
         3. Shorter due to lack of citations and supporting facts
      ii. Creating below variables
         1. Title_polarity : polarity sentiment of title
         2. Text_polarity: polarity sentiment of text
         3. Ttl_words: count of total meaningful words in text
         4. Text_digit_cnt: count of numerical words in text eg: "1984", "67" etc

2. Exploratory Data Analysis

a. Text Columns

   i. Word Cloud words in text, author and title column for test and train

   ii. Word Cloud of words in text, author and title column for fake and real news in train

      1. Fake news articles have more anonymous and vague named bloggers as authors and use more trending names

Fake news authors:

Real new authors:

b. Numeric Data Columns

   i. Title polarity was dropped, as it did not capture anything

   ii. The distribution and features of these values align with the hypothesis

In [96]:
```
## Analyzing new variables distribution w.r.t. to the label varaible
print(pd.pivot_table(train, index = 'label', values = cols_num[1:],aggfunc = 'sum'))
print(pd.pivot_table(train, index = 'label', values = cols_num[1:],aggfunc = 'min'))
print(pd.pivot_table(train, index = 'label', values = cols_num[1:],aggfunc = 'max'))
```

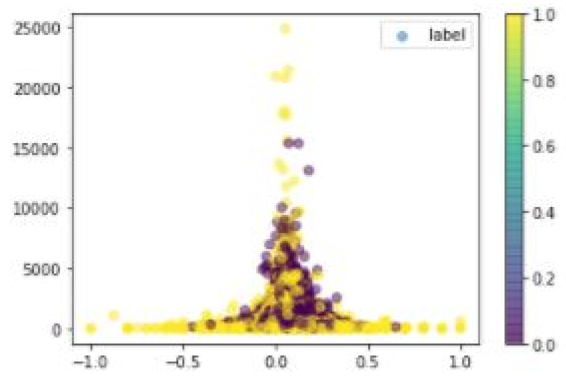
```
       text_digit_cnt  text_polarity  ttl_wrds
label
0              108444     631.065967   9522464
1               79922     589.091514   6807367
       text_digit_cnt  text_polarity  ttl_wrds
label
0                   0          -0.45         1
1                   0          -1.00         0
       text_digit_cnt  text_polarity  ttl_wrds
label
0                 235           0.65     15349
1                 470           1.00     24870
       text_digit_cnt  text_polarity    ttl_wrds
label
0           10.440358       0.060755  916.767498
1            7.675214       0.056573  653.737348
```

On average fake news articles have less data citations, range to extreme polarity, and are shorter than real news

Visualization of text sentiments with count of words and label

In [57]:
```
## text polarity and ttl_wrds have small positive correlation, check their distribution
plt.scatter(train.text_polarity,train.ttl_wrds ,c = train.label,label = 'label',alpha = 0.5)
plt.legend()
```

Out[57]: <matplotlib.colorbar.Colorbar at 0x201d3299c40>



3. **Modelling**

   a. **Data Preparation**

      i. Text Columns

         1. Used TF-IDF vectorizer to extract features from text columns
         2. For title and text ignored the words appearing in less 5% of data
         3. For author, vectorized all data

      ii. Numerical Columns

         1. Min Max Scaler to scale the data

      iii. Fitted the tfidf vectorizer on entire (train + test) dataset

   b. **Model Selection**

      i. Binary Classification problem, picked 4 models:

         1. Logistic Regression
         2. Random forest Classification

3. Stochastic Gradient Descent Classifier
4. Support Vector Classifier

c. Model Testing

i. 5 fold cross validation score

ii. Train and test split : 75:25

1. Metrics: Precision, Recall, F1 Score
2. Analysed ROC Curve

Model	Cross Validation	Metrics on 25% test
Stochastic Gradient Descent Classifier	[0.9875, 0.98846154 , 0.9900641, 0.98173077, 0.98910256]	Confusion Matrix: [[2605 37] [30 2528]] Precision score: 0.9856 Recall score: 0.9883 F1 score: 0.9869
Logistic Regression	[0.98044872, 0.97980769, 0.98397436, 0.975, 0.98044872]	Confusion Matrix: [[2598 44] [49 2509]] Precision score: 0.9828 Recall score: 0.9808 F1 score: 0.9818
Random Forest Classifier	[0.94807692, 0.94871795, 0.94230769, 0.94775641, 0.94391026]	Confusion Matrix: [[2528 114] [164 2394]] Precision score: 0.9545 Recall score: 0.9359 F1 score: 0.9451
SVC	[0.98878205, 0.99166667, 0.99134615, 0.98589744, 0.99038462]	Confusion Matrix: [[2614 28] [24 2534]] Precision score: 0.9891 Recall score: 0.9906 F1 score: 0.9898

d. Conclusion

i. ROC curve: SVC and SGD perform better

ii. F1 Score: SVC and SGD perform better

e. Final Predictions (tested on Kaggle, SVC and SGD performed better than other two models)

Submission and Description	Private Score	Public Score	Use for Final Score
SVC(random_state42).csv 3 minutes ago by Apurva S svc clf	0.99203	0.98589	☐
SGDClassifier(random_state42).csv 3 minutes ago by Apurva S sgd clf	0.98956	0.98141	☐
RandomForestClassifier(random_state42).csv 4 minutes ago by Apurva S random forest clf	0.95714	0.95000	☐
LogisticRegression(random_state42).csv 4 minutes ago by Apurva S Log CLf	0.98543	0.97628	☐