# An Introduction to Watson NLP — Emotion Classification

Watson NLP is a standard embedded AI library designed to tie together the pieces of IBM's NLP. It provides a standard base NLP layer along with a single integrated roadmap, a common architecture, and a common code stack designed for widespread adoption across IBM products, and to streamline IBM Research innovations into products.
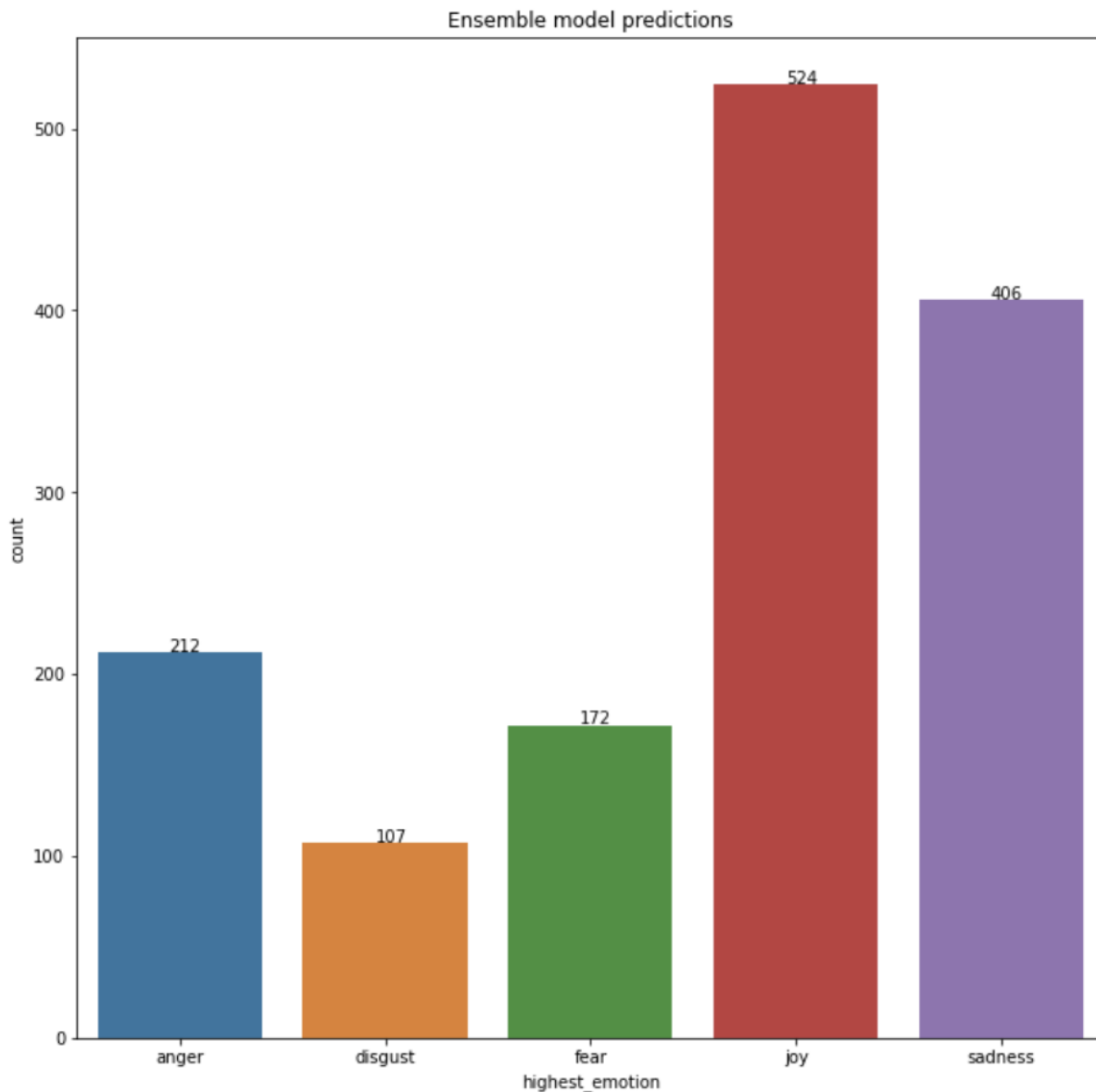
Displaying emotion classification predictions

Emotion Classification is a strong tool usually manually processed by humans in gathering groups of qualitative data. Having the ability to automatically gather and process larger datasets of text via customer feedback, comments, or an entire article written on your product

is a strong tool to gain insight into the most common emotional responses in a group of people or a block of text.

Watson NLP now provides the ability to automatically classify text into the strongest emotions we hope to track: "sadness", "joy", "anger", "fear", and "disgust". We are able to infer upon a certain emotion through syntax analysis and our emotion workflows provided by IBM's AI libraries via pre-trained models and custom trained models.

We will familiarize with the fundamentals of Watson NLP and walk through the process of running and evaluating pre-trained models to perform emotion classification.

Watson NLP's pre-trained models come in two forms: Blocks and Workflows.

Blocks

Blocks are the embodiment of NLP models, which support load(), save(), run(), and train() functions. There are two types of Blocks. Type 1 are Blocks that operate directly on the input document. The most commonly used Type 1 block is the syntax block. Syntax block performs tokenization, lemmatization, part of speech tagging, and dependency parsing on the input document. Type 2 are blocks that depend on other blocks. Type 2 are unable to directly take documents as inputs. Generally, models that require text preprocessing, such as classifiers or entity extractors, will be Type 2 blocks. In most use-cases, the output of a Type 1 syntax block will be used as the input of a Type 2 classifier block.

Workflows

Workflows are entire pipelines of the blocks. The previously mentioned flow of a Type 1 syntax block inputting to a Type 2 classifier block can be packaged as a workflow so that the process can run with one function call on the input document instead of multiple individual function calls.

This tutorial will use two Watson NLP model blocks:

Syntax Analysis
Syntax Analysis, or Watson's *Syntax block*, performs tokenization, lemmatization, parts of speech tagging, and dependency parsing on raw input documents so that custom models can properly classify documents.
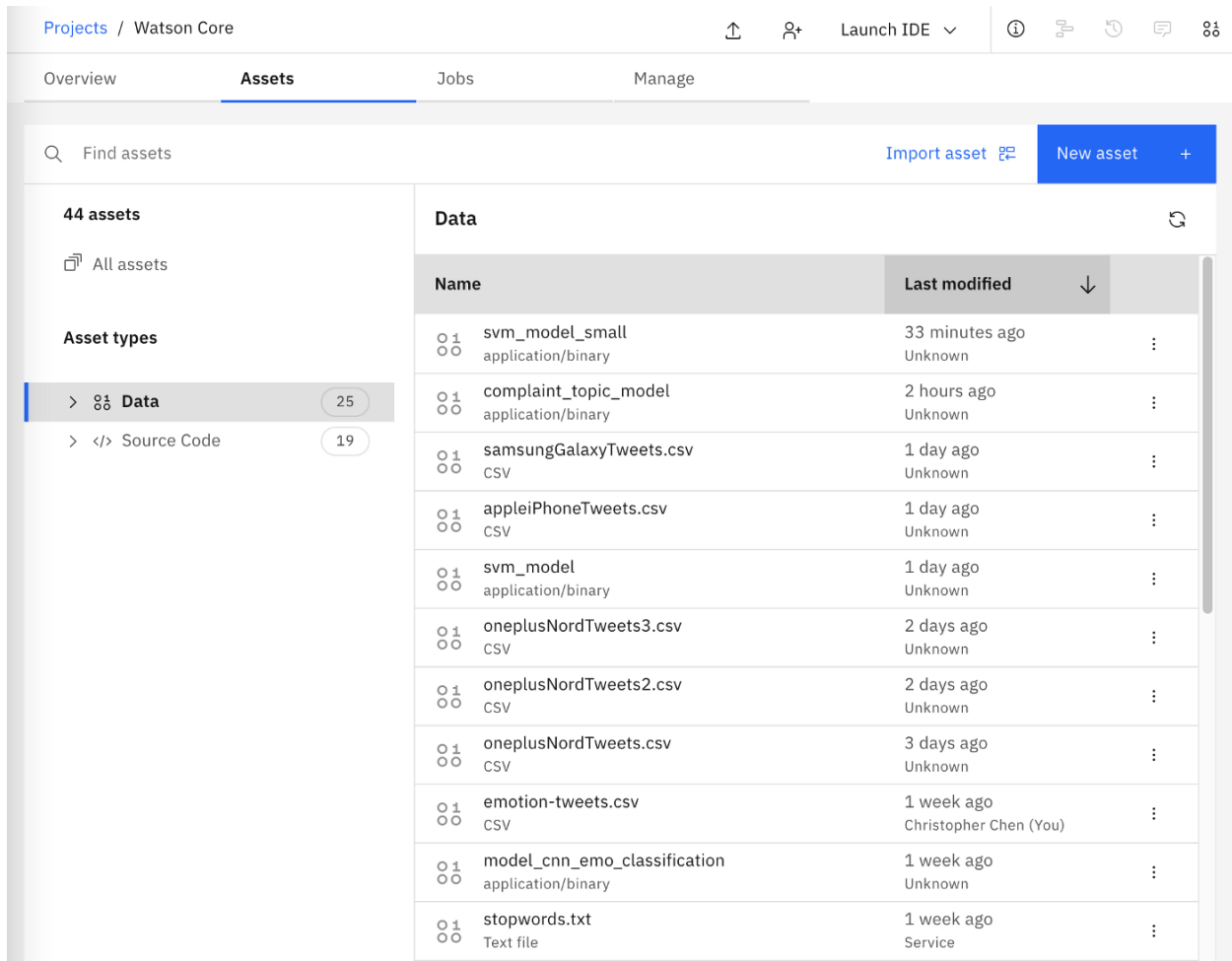
Emotion Classification
Emotion Classification, provided with the *Ensemble emotion workflow* and the *Aggregated emotion workflow*, can classify any block of text into five emotions: "sadness", "joy", "anger", "fear", and "disgust".

Although the Emotion Classification block relies on parsing through Watson's syntax block, we will focus primarily on **Emotion Classification** workflows and how they work.

1. Collecting the dataset

Datasets for emotion classification will require an input text feature column and an emotion label column with labels such as 'joy', 'anger', fear', and 'sadness'. Let's use this manually annotated tweet dataset found on Kaggle. You can download all of the .csv files and combine them or download the single test .csv file.

Watson Studio Project data assets

After downloading the dataset, you will need to upload the file to the Watson Studio project as a data asset. From there, the data will be ready to be inserted into the notebook. Notebook is set up with instructions for reading the .csv as a Pandas DataFrame.

	text	labels
6084	HartRamsey'sUPLIFT If you're still discouraged it means you're listening to the wrong voices & looking to the wrong source.Look to the LORD!	[sadness]
6830	@r0lls ppl get triggered over u smiling they're irrelevant	[joy]
270	#smile is the #respect we give everyone.	[joy]
6945	@ArcticFantasy I would have almost took offense to this if I actually snapped you	[anger]
2576	@ParrishWalton @kjmgolf Also, our best assets on offense are our RB's and our WR's (longterm). Have to have an offense to utilize both.	[anger]
...	...	...
5861	@Toucherandrich there was a US diver named steele johnson. i'd like him barred from the olympics.	[anger]
3315	Been working in Blanchardstown shopping centre for over 2 years now and I only figured out today where Marks & Spencer's is #lost	[sadness]
6675	can it stay gloomy but get cold pls	[sadness]
569	Blake lively is flawless	[joy]
5393	when you think you've got it together for a day then you randomly burst out crying	[anger]

Pandas DataFrame of the dataset

2. Data processing

Because the dataset already caters towards NLP emotion classification with the necessary feature and label columns, there will be little processing need at this step. In practice, data will often have to be cleaned and transformed to fit the model's needs. Watson NLP expects the labels in the label column to be in a list as opposed to a standalone string. To achieve this, we will apply a simple conversion function to the label column.

```python
def convertToList(x):
    return [x]
```

```python
df['label'] = df['label'].apply(convertToList)
df = df.rename(columns={'label':'labels'})
```

Function to convert labels to a list of labels

Optionally, the notebook has a step for creating a train/test split of the dataset. This can be performed on the larger, combined data of the train, test, and validation .csv files from Kaggle. Once again, the only data split that we will be using is the test split for evaluation of the models.

3. Running pretrained models

Now for the fun part! Watson NLP has two pre-trained emotion classification models using the workflow system. This means that to run the model, we'll only have to provide a text input, rather than having to set up a syntax processing block and a model block.This blog will use *"Such a sweet boy. But after much thought and careful consideration, I've decided that the ruler for the next ten thousand years is going to have to be… me."* as a single input test with the expected label to be *"joy"*.

All models will have to be downloaded from the Watson NLP library on their initial run. They will be saved to the runtime local, or local working path if the notebook is being run offline.

```
# Load the Emotion workflow model for English
ensemble_emotion_model = watson_nlp.load(watson_nlp.download('ensemble_
classification-wf_en_emotion-stock'))
```

Loading the model with watson_nlp.load(watson_nlp.download(<model_name>))

Generating a prediction with the model is as straightforward as invoking **model.run()** with the test text as the input. The return of the model prediction is a dictionary of emotions with their respective confidence scores. The highest score is the predicted emotion.

```
# Run the Emotion model on a single document
ensemble_emotion_result = ensemble_emotion_model.run("Such a sweet boy.
But after much thought and careful consideration, I've decided that the
ruler for the next ten thousand years is going to have to be... me. ")
print(ensemble_emotion_result)
```

Predicting an input text with model.run()

```
{
  "classes": [
    {
      "class_name": "joy",
      "confidence": 0.520251523364674
    },
    {
      "class_name": "sadness",
      "confidence": 0.13437655658432934
    },
    {
      "class_name": "anger",
      "confidence": 0.030973352040305283
    },
    {
      "class_name": "fear",
      "confidence": 0.023922530189972323
    },
    {
      "class_name": "disgust",
      "confidence": 0.01345159203717203
    }
  ],
  "producer_id": {
    "name": "Voting based Ensemble",
    "version": "0.0.1"
  }
}
```

Dictionary output of the emotion classifications

Certain blocks and workflows also have the ability to run and evaualate an entire dataset i.e. test dataset. This is done with the **model.evaluate_quality()** function. The outputting result is a dictionary of confusion matrices, precision, recall, and f1 score for each of the labeled emotions.

```
test_data_file = "df_test.json"

quality_report = ensemble_emotion_model.evaluate_quality(test_data_file)
print(json.dumps(quality_report, indent=4))
```

Evaluating a test file with model.evaluate_quality()

```
{
    "per_class_confusion_matrix": {
        "joy": {
            "true_positive": 246,
            "true_negative": 0,
            "false_positive": 278,
            "false_negative": 78,
            "precision": 0.46946564885496184,
            "recall": 0.7592592592592593,
            "f1": 0.5801886792452831
        },
        "fear": {
            "true_positive": 129,
            "true_negative": 0,
            "false_positive": 43,
            "false_negative": 315,
            "precision": 0.75,
            "recall": 0.2905405405405405,
            "f1": 0.4188311688311688
```

A section of the dictionary output for evaluate_quality() function

The Aggregated emotion model is the preferred emotion classification model within Watson NLP as it has the capability of detecting emotion surrounding specific targeted words.

```
 model for English
watson_nlp.load(watson_nlp.download('aggregated_classification-wf_en_emotion-stock'))
```

Loading the model with watson_nlp.load(watson_nlp.download(<model_name>))

Aggregated emotion block can take targets in the form of TargetMentions and TargetPhrases.

```
# span targets a section of the document given indices
target_mentions = dm.TargetMentionsPrediction([dm.TargetMentions([(7, 16)]), dm.TargetMentions([(28, 66)])])

aggregated_emotion_result_span = aggregated_emotion_model.run("Such a sweet boy. But after much thought and c
                              target_mentions=target_mentions)
```

```
# text targets a section of the document given phrases
target_phrases = ['sweet boy', 'careful consideration']

aggregated_emotion_result_text = aggregated_emotion_model.run("Such a sweet boy. But after much thought and c
                              target_phrases=target_phrases)
```

TargetMention and TargetPhrases as parameters for prediction

By defining a target text mention with the `target_mentions` parameter or a target text phrase with the `target_phrases` parameter, the Aggregated model can predict emotion on exact words and phrases in the input. The results are in the "target" key.

```
{
  "emotion_predictions": [
    {
      "emotion": {
        "anger": 0.014531953677202513,
        "disgust": 0.015678834170103073,
        "fear": 0.004446345653956418,
        "joy": 0.9617643175703107,
        "sadness": 0.04595572027293119
      },
      "target": "sweet boy",
      "emotion_mentions": [
        {
          "span": {
            "begin": 0,
            "end": 17,
            "text": "Such a sweet boy."
          },
          "emotion": {
            "anger": 0.014531953677202513,
            "disgust": 0.015678834170103073,
            "fear": 0.004446345653956418,
            "joy": 0.9617643175703107,
            "sadness": 0.04595572027293119
```

Results returned from Aggregated target emotion classification

Conclusion

This introduction to Watson NLP is only a brief look at how easily NLP emotion classification can be performed on ready datasets by utilizing the Watson NLP python library. As you explore the accessibility of IBM's NLP stack via Watson NLP, you will interact with many more models that come pretrained with the library such as topic modeling and sentiment analysis. The next blog will cover model deployment to show how easily you can apply these models anywhere.

You can find the referenced notebook on GitHub.