

## ✓ PPLM Colab Demo (Self-contained)

This notebook is prepared to run on Google Colab and demonstrates the PPLM technique (BoW and discriminator steering) using the code from this repository.

How to use:

1. Open this notebook in Colab.
2. Run cells top-to-bottom. The notebook will clone the repository, install dependencies, and run short demos.

Notes:

- For speed this demo uses `gpt2` (small) and short generation lengths.
- If a GPU is available in Colab, the notebook will use it automatically.
- Outputs are captured and printed at the end of each demo cell for easy copying into slides.

```
# Detect runtime and print environment info
import sys, os
print('Python', sys.version)
try:
    import torch
    print('Torch', torch.__version__, 'CUDA available:', torch.cuda.is_available())
except Exception as e:
    print('Torch not installed yet')
```

```
Python 3.12.11 (main, Jun  4 2025, 08:56:18) [GCC 11.4.0]
Torch 2.8.0+cu126 CUDA available: False
```

◆ Gemini

```
# Install minimal dependencies (transformers pinned to 3.4.0, requests for the shim)
# Using pip install -q to keep output tidy
!pip install -q transformers==3.4.0 colorama==0.4.4 nltk==3.4.5 requests
```

```
# Install tokenizers separately to avoid the build error
!pip install -q tokenizers==0.8.2
!pip install -q tokenizers==0.11.6
```

```
# Download or upgrade torch if necessary (Colab usually has a suitable torch)
import importlib
try:
    import torch
    print('Torch found:', torch.__version__)
except Exception as e:
    print('Installing torch')
    !pip install -q torch
```

```
Installing build dependencies ... done
Getting requirements to build wheel ... done
Preparing metadata (pyproject.toml) ... done
error: subprocess-exited-with-error
```

```
× Building wheel for tokenizers (pyproject.toml) did not run successfully.
| exit code: 1
└─> See above for output.
```

**note:** This error originates from a subprocess, and is likely not a problem with pip.

```
Building wheel for tokenizers (pyproject.toml) ... error
ERROR: Failed building wheel for tokenizers
ERROR: ERROR: Failed to build installable wheels for some pyproject.toml based projects (tokenizers)
ERROR: Ignored the following yanked versions: 0.12.0, 0.20.4
ERROR: Could not find a version that satisfies the requirement tokenizers==0.8.2 (from versions: 0.0.2, 0.0.3, 0.0.4, 0.0.5, 0.0.6, 0.0.7, 0.0.8, 0.0.9, 0.0.10, 0.0.11, 0.0.12, 0.0.13, 0.0.14, 0.0.15, 0.0.16, 0.0.17, 0.0.18, 0.0.19, 0.0.20, 0.0.21, 0.0.22, 0.0.23, 0.0.24, 0.0.25, 0.0.26, 0.0.27, 0.0.28, 0.0.29, 0.0.30, 0.0.31, 0.0.32, 0.0.33, 0.0.34, 0.0.35, 0.0.36, 0.0.37, 0.0.38, 0.0.39, 0.0.40, 0.0.41, 0.0.42, 0.0.43, 0.0.44, 0.0.45, 0.0.46, 0.0.47, 0.0.48, 0.0.49, 0.0.50, 0.0.51, 0.0.52, 0.0.53, 0.0.54, 0.0.55, 0.0.56, 0.0.57, 0.0.58, 0.0.59, 0.0.60, 0.0.61, 0.0.62, 0.0.63, 0.0.64, 0.0.65, 0.0.66, 0.0.67, 0.0.68, 0.0.69, 0.0.70, 0.0.71, 0.0.72, 0.0.73, 0.0.74, 0.0.75, 0.0.76, 0.0.77, 0.0.78, 0.0.79, 0.0.80, 0.0.81, 0.0.82, 0.0.83, 0.0.84, 0.0.85, 0.0.86, 0.0.87, 0.0.88, 0.0.89, 0.0.90, 0.0.91, 0.0.92, 0.0.93, 0.0.94, 0.0.95, 0.0.96, 0.0.97, 0.0.98, 0.0.99, 0.0.100, 0.0.101, 0.0.102, 0.0.103, 0.0.104, 0.0.105, 0.0.106, 0.0.107, 0.0.108, 0.0.109, 0.0.110, 0.0.111, 0.0.112, 0.0.113, 0.0.114, 0.0.115, 0.0.116, 0.0.117, 0.0.118, 0.0.119, 0.0.120, 0.0.121, 0.0.122, 0.0.123, 0.0.124, 0.0.125, 0.0.126, 0.0.127, 0.0.128, 0.0.129, 0.0.130, 0.0.131, 0.0.132, 0.0.133, 0.0.134, 0.0.135, 0.0.136, 0.0.137, 0.0.138, 0.0.139, 0.0.140, 0.0.141, 0.0.142, 0.0.143, 0.0.144, 0.0.145, 0.0.146, 0.0.147, 0.0.148, 0.0.149, 0.0.150, 0.0.151, 0.0.152, 0.0.153, 0.0.154, 0.0.155, 0.0.156, 0.0.157, 0.0.158, 0.0.159, 0.0.160, 0.0.161, 0.0.162, 0.0.163, 0.0.164, 0.0.165, 0.0.166, 0.0.167, 0.0.168, 0.0.169, 0.0.170, 0.0.171, 0.0.172, 0.0.173, 0.0.174, 0.0.175, 0.0.176, 0.0.177, 0.0.178, 0.0.179, 0.0.180, 0.0.181, 0.0.182, 0.0.183, 0.0.184, 0.0.185, 0.0.186, 0.0.187, 0.0.188, 0.0.189, 0.0.190, 0.0.191, 0.0.192, 0.0.193, 0.0.194, 0.0.195, 0.0.196, 0.0.197, 0.0.198, 0.0.199, 0.0.200, 0.0.201, 0.0.202, 0.0.203, 0.0.204, 0.0.205, 0.0.206, 0.0.207, 0.0.208, 0.0.209, 0.0.210, 0.0.211, 0.0.212, 0.0.213, 0.0.214, 0.0.215, 0.0.216, 0.0.217, 0.0.218, 0.0.219, 0.0.220, 0.0.221, 0.0.222, 0.0.223, 0.0.224, 0.0.225, 0.0.226, 0.0.227, 0.0.228, 0.0.229, 0.0.230, 0.0.231, 0.0.232, 0.0.233, 0.0.234, 0.0.235, 0.0.236, 0.0.237, 0.0.238, 0.0.239, 0.0.240, 0.0.241, 0.0.242, 0.0.243, 0.0.244, 0.0.245, 0.0.246, 0.0.247, 0.0.248, 0.0.249, 0.0.250, 0.0.251, 0.0.252, 0.0.253, 0.0.254, 0.0.255, 0.0.256, 0.0.257, 0.0.258, 0.0.259, 0.0.260, 0.0.261, 0.0.262, 0.0.263, 0.0.264, 0.0.265, 0.0.266, 0.0.267, 0.0.268, 0.0.269, 0.0.270, 0.0.271, 0.0.272, 0.0.273, 0.0.274, 0.0.275, 0.0.276, 0.0.277, 0.0.278, 0.0.279, 0.0.280, 0.0.281, 0.0.282, 0.0.283, 0.0.284, 0.0.285, 0.0.286, 0.0.287, 0.0.288, 0.0.289, 0.0.290, 0.0.291, 0.0.292, 0.0.293, 0.0.294, 0.0.295, 0.0.296, 0.0.297, 0.0.298, 0.0.299, 0.0.300, 0.0.301, 0.0.302, 0.0.303, 0.0.304, 0.0.305, 0.0.306, 0.0.307, 0.0.308, 0.0.309, 0.0.310, 0.0.311, 0.0.312, 0.0.313, 0.0.314, 0.0.315, 0.0.316, 0.0.317, 0.0.318, 0.0.319, 0.0.320, 0.0.321, 0.0.322, 0.0.323, 0.0.324, 0.0.325, 0.0.326, 0.0.327, 0.0.328, 0.0.329, 0.0.330, 0.0.331, 0.0.332, 0.0.333, 0.0.334, 0.0.335, 0.0.336, 0.0.337, 0.0.338, 0.0.339, 0.0.340, 0.0.341, 0.0.342, 0.0.343, 0.0.344, 0.0.345, 0.0.346, 0.0.347, 0.0.348, 0.0.349, 0.0.350, 0.0.351, 0.0.352, 0.0.353, 0.0.354, 0.0.355, 0.0.356, 0.0.357, 0.0.358, 0.0.359, 0.0.360, 0.0.361, 0.0.362, 0.0.363, 0.0.364, 0.0.365, 0.0.366, 0.0.367, 0.0.368, 0.0.369, 0.0.370, 0.0.371, 0.0.372, 0.0.373, 0.0.374, 0.0.375, 0.0.376, 0.0.377, 0.0.378, 0.0.379, 0.0.380, 0.0.381, 0.0.382, 0.0.383, 0.0.384, 0.0.385, 0.0.386, 0.0.387, 0.0.388, 0.0.389, 0.0.390, 0.0.391, 0.0.392, 0.0.393, 0.0.394, 0.0.395, 0.0.396, 0.0.397, 0.0.398, 0.0.399, 0.0.400, 0.0.401, 0.0.402, 0.0.403, 0.0.404, 0.0.405, 0.0.406, 0.0.407, 0.0.408, 0.0.409, 0.0.410, 0.0.411, 0.0.412, 0.0.413, 0.0.414, 0.0.415, 0.0.416, 0.0.417, 0.0.418, 0.0.419, 0.0.420, 0.0.421, 0.0.422, 0.0.423, 0.0.424, 0.0.425, 0.0.426, 0.0.427, 0.0.428, 0.0.429, 0.0.430, 0.0.431, 0.0.432, 0.0.433, 0.0.434, 0.0.435, 0.0.436, 0.0.437, 0.0.438, 0.0.439, 0.0.440, 0.0.441, 0.0.442, 0.0.443, 0.0.444, 0.0.445, 0.0.446, 0.0.447, 0.0.448, 0.0.449, 0.0.450, 0.0.451, 0.0.452, 0.0.453, 0.0.454, 0.0.455, 0.0.456, 0.0.457, 0.0.458, 0.0.459, 0.0.460, 0.0.461, 0.0.462, 0.0.463, 0.0.464, 0.0.465, 0.0.466, 0.0.467, 0.0.468, 0.0.469, 0.0.470, 0.0.471, 0.0.472, 0.0.473, 0.0.474, 0.0.475, 0.0.476, 0.0.477, 0.0.478, 0.0.479, 0.0.480, 0.0.481, 0.0.482, 0.0.483, 0.0.484, 0.0.485, 0.0.486, 0.0.487, 0.0.488, 0.0.489, 0.0.490, 0.0.491, 0.0.492, 0.0.493, 0.0.494, 0.0.495, 0.0.496, 0.0.497, 0.0.498, 0.0.499, 0.0.500, 0.0.501, 0.0.502, 0.0.503, 0.0.504, 0.0.505, 0.0.506, 0.0.507, 0.0.508, 0.0.509, 0.0.510, 0.0.511, 0.0.512, 0.0.513, 0.0.514, 0.0.515, 0.0.516, 0.0.517, 0.0.518, 0.0.519, 0.0.520, 0.0.521, 0.0.522, 0.0.523, 0.0.524, 0.0.525, 0.0.526, 0.0.527, 0.0.528, 0.0.529, 0.0.530, 0.0.531, 0.0.532, 0.0.533, 0.0.534, 0.0.535, 0.0.536, 0.0.537, 0.0.538, 0.0.539, 0.0.540, 0.0.541, 0.0.542, 0.0.543, 0.0.544, 0.0.545, 0.0.546, 0.0.547, 0.0.548, 0.0.549, 0.0.550, 0.0.551, 0.0.552, 0.0.553, 0.0.554, 0.0.555, 0.0.556, 0.0.557, 0.0.558, 0.0.559, 0.0.560, 0.0.561, 0.0.562, 0.0.563, 0.0.564, 0.0.565, 0.0.566, 0.0.567, 0.0.568, 0.0.569, 0.0.570, 0.0.571, 0.0.572, 0.0.573, 0.0.574, 0.0.575, 0.0.576, 0.0.577, 0.0.578, 0.0.579, 0.0.580, 0.0.581, 0.0.582, 0.0.583, 0.0.584, 0.0.585, 0.0.586, 0.0.587, 0.0.588, 0.0.589, 0.0.590, 0.0.591, 0.0.592, 0.0.593, 0.0.594, 0.0.595, 0.0.596, 0.0.597, 0.0.598, 0.0.599, 0.0.600, 0.0.601, 0.0.602, 0.0.603, 0.0.604, 0.0.605, 0.0.606, 0.0.607, 0.0.608, 0.0.609, 0.0.610, 0.0.611, 0.0.612, 0.0.613, 0.0.614, 0.0.615, 0.0.616, 0.0.617, 0.0.618, 0.0.619, 0.0.620, 0.0.621, 0.0.622, 0.0.623, 0.0.624, 0.0.625, 0.0.626, 0.0.627, 0.0.628, 0.0.629, 0.0.630, 0.0.631, 0.0.632, 0.0.633, 0.0.634, 0.0.635, 0.0.636, 0.0.637, 0.0.638, 0.0.639, 0.0.640, 0.0.641, 0.0.642, 0.0.643, 0.0.644, 0.0.645, 0.0.646, 0.0.647, 0.0.648, 0.0.649, 0.0.650, 0.0.651, 0.0.652, 0.0.653, 0.0.654, 0.0.655, 0.0.656, 0.0.657, 0.0.658, 0.0.659, 0.0.660, 0.0.661, 0.0.662, 0.0.663, 0.0.664, 0.0.665, 0.0.666, 0.0.667, 0.0.668, 0.0.669, 0.0.670, 0.0.671, 0.0.672, 0.0.673, 0.0.674, 0.0.675, 0.0.676, 0.0.677, 0.0.678, 0.0.679, 0.0.680, 0.0.681, 0.0.682, 0.0.683, 0.0.684, 0.0.685, 0.0.686, 0.0.687, 0.0.688, 0.0.689, 0.0.690, 0.0.691, 0.0.692, 0.0.693, 0.0.694, 0.0.695, 0.0.696, 0.0.697, 0.0.698, 0.0.699, 0.0.700, 0.0.701, 0.0.702, 0.0.703, 0.0.704, 0.0.705, 0.0.706, 0.0.707, 0.0.708, 0.0.709, 0.0.710, 0.0.711, 0.0.712, 0.0.713, 0.0.714, 0.0.715, 0.0.716, 0.0.717, 0.0.718, 0.0.719, 0.0.720, 0.0.721, 0.0.722, 0.0.723, 0.0.724, 0.0.725, 0.0.726, 0.0.727, 0.0.728, 0.0.729, 0.0.730, 0.0.731, 0.0.732, 0.0.733, 0.0.734, 0.0.735, 0.0.736, 0.0.737, 0.0.738, 0.0.739, 0.0.740, 0.0.741, 0.0.742, 0.0.743, 0.0.744, 0.0.745, 0.0.746, 0.0.747, 0.0.748, 0.0.749, 0.0.750, 0.0.751, 0.0.752, 0.0.753, 0.0.754, 0.0.755, 0.0.756, 0.0.757, 0.0.758, 0.0.759, 0.0.760, 0.0.761, 0.0.762, 0.0.763, 0.0.764, 0.0.765, 0.0.766, 0.0.767, 0.0.768, 0.0.769, 0.0.770, 0.0.771, 0.0.772, 0.0.773, 0.0.774, 0.0.775, 0.0.776, 0.0.777, 0.0.778, 0.0.779, 0.0.780, 0.0.781, 0.0.782, 0.0.783, 0.0.784, 0.0.785, 0.0.786, 0.0.787, 0.0.788, 0.0.789, 0.0.790, 0.0.791, 0.0.792, 0.0.793, 0.0.794, 0.0.795, 0.0.796, 0.0.797, 0.0.798, 0.0.799, 0.0.800, 0.0.801, 0.0.802, 0.0.803, 0.0.804, 0.0.805, 0.0.806, 0.0.807, 0.0.808, 0.0.809, 0.0.810, 0.0.811, 0.0.812, 0.0.813, 0.0.814, 0.0.815, 0.0.816, 0.0.817, 0.0.818, 0.0.819, 0.0.820, 0.0.821, 0.0.822, 0.0.823, 0.0.824, 0.0.825, 0.0.826, 0.0.827, 0.0.828, 0.0.829, 0.0.830, 0.0.831, 0.0.832, 0.0.833, 0.0.834, 0.0.835, 0.0.836, 0.0.837, 0.0.838, 0.0.839, 0.0.840, 0.0.841, 0.0.842, 0.0.843, 0.0.844, 0.0.845, 0.0.846, 0.0.847, 0.0.848, 0.0.849, 0.0.850, 0.0.851, 0.0.852, 0.0.853, 0.0.854, 0.0.855, 0.0.856, 0.0.857, 0.0.858, 0.0.859, 0.0.860, 0.0.861, 0.0.862, 0.0.863, 0.0.864, 0.0.865, 0.0.866, 0.0.867, 0.0.868, 0.0.869, 0.0.870, 0.0.871, 0.0.872, 0.0.873, 0.0.874, 0.0.875, 0.0.876, 0.0.877, 0.0.878, 0.0.879, 0.0.880, 0.0.881, 0.0.882, 0.0.883, 0.0.884, 0.0.885, 0.0.886, 0.0.887, 0.0.888, 0.0.889, 0.0.890, 0.0.891, 0.0.892, 0.0.893, 0.0.894, 0.0.895, 0.0.896, 0.0.897, 0.0.898, 0.0.899, 0.0.900, 0.0.901, 0.0.902, 0.0.903, 0.0.904, 0.0.905, 0.0.906, 0.0.907, 0.0.908, 0.0.909, 0.0.910, 0.0.911, 0.0.912, 0.0.913, 0.0.914, 0.0.915, 0.0.916, 0.0.917, 0.0.918, 0.0.919, 0.0.920, 0.0.921, 0.0.922, 0.0.923, 0.0.924, 0.0.925, 0.0.926, 0.0.927, 0.0.928, 0.0.929, 0.0.930, 0.0.931, 0.0.932, 0.0.933, 0.0.934, 0.0.935, 0.0.936, 0.0.937, 0.0.938, 0.0.939, 0.0.940, 0.0.941, 0.0.942, 0.0.943, 0.0.944, 0.0.945, 0.0.946, 0.0.947, 0.0.948, 0.0.949, 0.0.950, 0.0.951, 0.0.952, 0.0.953, 0.0.954, 0.0.955, 0.0.956, 0.0.957, 0.0.958, 0.0.959, 0.0.960, 0.0.961, 0.0.962, 0.0.963, 0.0.964, 0.0.965, 0.0.966, 0.0.967, 0.0.968, 0.0.969, 0.0.970, 0.0.971, 0.0.972, 0.0.973, 0.0.974, 0.0.975, 0.0.976, 0.0.977, 0.0.978, 0.0.979, 0.0.980, 0.0.981, 0.0.982, 0.0.983, 0.0.984, 0.0.985, 0.0.986, 0.0.987, 0.0.988, 0.0.989, 0.0.990, 0.0.991, 0.0.992, 0.0.993, 0.0.994, 0.0.995, 0.0.996, 0.0.997, 0.0.998, 0.0.999, 1.0.0, 1.0.1, 1.0.2, 1.0.3, 1.0.4, 1.0.5, 1.0.6, 1.0.7, 1.0.8, 1.0.9, 1.0.10, 1.0.11, 1.0.12, 1.0.13, 1.0.14, 1.0.15, 1.0.16, 1.0.17, 1.0.18, 1.0.19, 1.0.20, 1.0.21, 1.0.22, 1.0.23, 1.0.24, 1.0.25, 1.0.26, 1.0.27, 1.0.28, 1.0.29, 1.0.30, 1.0.31, 1.0.32, 1.0.33, 1.0.34, 1.0.35, 1.0.36, 1.0.37, 1.0.38, 1.0.39, 1.0.40, 1.0.41, 1.0.42, 1.0.43, 1.0.44, 1.0.45, 1.0.46, 1.0.47, 1.0.48, 1.0.49, 1.0.50, 1.0.51, 1.0.52, 1.0.53, 1.0.54, 1.0.55, 1.0.56, 1.0.57, 1.0.58, 1.0.59, 1.0.60, 1.0.61, 1.0.62, 1.0.63, 1.0.64, 1.0.65, 1.0.66, 1.0.67, 1.0.68, 1.0.69, 1.0.70, 1.0.71, 1.0.72, 1.0.73, 1.0.74, 1.0.75, 1.0.76, 1.0.77, 1.0.78, 1.0.79, 1.0.80, 1.0.81, 1.0.82, 1.0.83, 1.0.84, 1.0.85, 1.0.86, 1.0.87, 1.0.88, 1.0.89, 1.0.90, 1.0.91, 1.0.92, 1.0.93, 1.0.94, 1.0.95, 1.0.96, 1.0.97, 1.0.98, 1.0.99, 2.0.0, 2.0.1, 2.0.2, 2.0.3, 2.0.4, 2.0.5, 2.0.6, 2.0.7, 2.0.8, 2.0.9, 2.0.10, 2.0.11, 2.0.12, 2.0.13, 2.0.14, 2.0.15, 2.0.16, 2.0.17, 2.0.18, 2.0.19, 2.0.20, 2.0.21, 2.0.22, 2.0.23, 2.0.24, 2.0.25, 2.0.26, 2.0.27, 2.0.28, 2.0.29, 2.0.30, 2.0.31, 2.0.32, 2.0.33, 2.0.34, 2.0.35, 2.0.36, 2.0.37, 2.0.38, 2.0.39, 2.0.40, 2.0.41, 2.0.42, 2.0.43, 2.0.44, 2.0.45, 2.0.46, 2.0.47, 2.0.48, 2.0.49, 2.0.50, 2.0.51, 2.0.52, 2.0.53, 2.0.54, 2.0.55, 2.0.56, 2.0.57, 2.0.58, 2.0.59, 2.0.60, 2.0.61, 2.0.62, 2.0.63, 2.0.64, 2.0.65, 2.0.66, 2.0.67, 2.0.68, 2.0.69, 2.0.70, 2.0.71, 2.0.72, 2.0.73, 2.0.74, 2.0.75, 2.0.76, 2.0.77, 2.0.78, 2.0.79, 2.0.80, 2.0.81, 2.0.82, 2.0.83, 2.0.84, 2.0.85, 2.0.86, 2.0.87, 2.0.88, 2.0.89, 2.0.90, 2.0.91, 2.0.92, 2.0.93, 2.0.94, 2.0.95, 2.0.96, 2.0.97, 2.0.98, 2.0.99, 3.0.0, 3.0.1, 3.0.2, 3.0.3, 3.0.4, 3.0.5, 3.0.6, 3.0.7, 3.0.8, 3.0.9, 3.0.10, 3.0.11, 3.0.12, 3.0.13, 3.0.14, 3.0.15, 3.0.16, 3.0.17, 3.0.18, 3.0.19, 3.0.20, 3.0.21, 3.0.22, 3.0.23, 3.0.24, 3.0.25, 3.0.26, 3.0.27, 3.0.28, 3.0.29, 3.0.30, 3.0.31, 3.0.32, 3.0.33, 3.0.34, 3.0.35, 3.0.36, 3.0.37, 3.0.38, 3.0.39, 3.0.40, 3.0.41, 3.0.42, 3.0.43, 3.0.44, 3.0.45, 3.0.46, 3.0.47, 3.0.48, 3.0.49, 3.0.50, 3.0.51, 3.0.52, 3.0.53, 3.0.54, 3.0.55, 3.0.56, 3.0.57, 3.0.58, 3.0.59, 3.0.60, 3.0.61, 3.0.62, 3.0.63, 3.0.64, 3.0.65, 3.0.66, 3.0.67, 3.0.68, 3.0.69, 3.0.70, 3.0.71, 3.0.72, 3.0.73, 3.0.74, 3.0.75, 3.0.76, 3.0.
```

```
remote: Compressing objects: 100% (82/82), done.
remote: Total 93 (delta 11), reused 78 (delta 10), pack-reused 0 (from 0)
Receiving objects: 100% (93/93), 2.35 MiB | 11.71 MiB/s, done.
Resolving deltas: 100% (11/11), done.
Current dir: /content/PPLM/PPLM
Files: ['.git', 'human_annotation', 'run_pplm_discrim_train.py', '.gitignore', '.travis.yml', 'run_pplm.py', 'LICENS
```

```
# Find the folder that contains run_pplm.py and switch to it so relative paths work.
```

```
import os, glob
candidates = glob.glob('*/run_pplm.py', recursive=True)
if candidates:
    run_pplm_path = candidates[0]
    repo_dir = os.path.dirname(run_pplm_path)
    # If dirname is empty string, the file is in the current working directory
    if not repo_dir:
        repo_dir = os.getcwd()
        print('run_pplm.py found in current working directory:', run_pplm_path)
    else:
        repo_dir = os.path.abspath(repo_dir)
    try:
        os.chdir(repo_dir)
        print('Found run_pplm.py at', run_pplm_path)
        print('Changed working directory to', os.getcwd())
    except Exception as e:
        print('Could not chdir to', repo_dir, 'error:', e)
else:
    print('Could not find run_pplm.py in subfolders. Current dir:', os.getcwd())
```

```
# Show whether a local wordlists directory exists from this cwd
```

```
wl = os.path.join(os.getcwd(), 'wordlists')
print('wordlists dir exists:', os.path.isdir(wl))
if os.path.isdir(wl):
    print('wordlists files:', os.listdir(wl)[:40])
else:
    # show some nearby files to help debugging
    print('Top-level files:', os.listdir('.')[40])
```

```
run_pplm.py found in current working directory: run_pplm.py
```

```
Found run_pplm.py at run_pplm.py
```

```
Changed working directory to /content/PPLM/PPLM
```

```
wordlists dir exists: False
```

```
Top-level files: ['.git', 'human_annotation', '__pycache__', 'run_pplm_discrim_train.py', '.gitignore', '.travis.yml', 'LICENS
```

```
# Compatibility shim for older/newer `transformers` APIs: provide `cached_path` if missing
```

```
import os
import hashlib
import requests
try:
    # try to import the name normally so we don't overwrite a working implementation
    from transformers.file_utils import cached_path # type: ignore
    print('cached_path found in transformers.file_utils')
except Exception:
    print('cached_path not found. Installing a lightweight shim into transformers.file_utils')
    def cached_path(url_or_filename, *args, **kwargs):
        """
        Lightweight fallback for `cached_path` used by PPLM examples.
        - If argument is a local path, return it.
        - If argument is a url, download it to a small cache and return local path.
        """
        if os.path.exists(url_or_filename):
            return url_or_filename
        cache_dir = os.path.expanduser('~/.cache/huggingface/transformers')
        os.makedirs(cache_dir, exist_ok=True)
        key = hashlib.md5(url_or_filename.encode('utf-8')).hexdigest()
        filename = os.path.join(cache_dir, key)
        if os.path.exists(filename):
            return filename
        # download
        print(f'Downloading {url_or_filename} ...')
        resp = requests.get(url_or_filename, stream=True)
        resp.raise_for_status()
        with open(filename, 'wb') as fh:
            for chunk in resp.iter_content(chunk_size=8192):
                if chunk:
                    fh.write(chunk)
        return filename
    # attach shim to module
    try:
        import transformers.file_utils as _tfu
        _tfu.cached_path = cached_path
```

```
except Exception:
    # best-effort: ensure attribute exists on transformers module
    import transformers as _t
    setattr(_t, 'cached_path', cached_path)
print('Shim installed')
```

cached\_path found in transformers.file\_utils

```
# Compatibility shim: ensure `transformers.modeling_gpt2` exists and that GPT2LMHeadModel returns (logits, past, h:
import sys, types
import transformers
from transformers import GPT2LMHeadModel as _RealGPT2
import torch

class CompatGPT2(_RealGPT2):
    # Override forward to return a tuple like older transformers versions
    def forward(self, *args, **kwargs):
        outputs = super().forward(*args, **kwargs)
        logits = getattr(outputs, 'logits', None)
        # Modern HF returns past_key_values as tuple of (k,v)
        pkv = getattr(outputs, 'past_key_values', None) or getattr(outputs, 'past', None)
        # Convert past_key_values (tuple of (k,v)) into the older past tensor format expected by PPLM:
        # a tuple of tensors each shaped (batch, 2, n_head, seq_len, head_dim)
        past = None
        if pkv is not None:
            past_list = []
            for layer in pkv:
                # layer may be a tuple (k, v)
                if isinstance(layer, tuple) or isinstance(layer, list):
                    k, v = layer
                else:
                    # if already tensor, assume it's the combined form
                    tensor = layer
                    # ensure tensor has expected dims
                    past_list.append(tensor)
                    continue
                # ensure tensors have shape (batch, n_head, seq_len, head_dim)
                # stack into (batch, 2, n_head, seq_len, head_dim)
                stacked = torch.stack([k, v], dim=1)
                past_list.append(stacked)
            past = tuple(past_list)
        else:
            past = None
        hidden = getattr(outputs, 'hidden_states', None)
        return logits, past, hidden

# Install the compatibility class into the legacy module path and transformers namespace
mod_name = 'transformers.modeling_gpt2'
if mod_name not in sys.modules:
    mod = types.ModuleType(mod_name)
    mod.GPT2LMHeadModel = CompatGPT2
    sys.modules[mod_name] = mod
else:
    sys.modules[mod_name].GPT2LMHeadModel = CompatGPT2

# Ensure transformers.modeling_gpt2 attribute exists and points to our module
import importlib
transformers.modeling_gpt2 = sys.modules[mod_name]
transformers.GPT2LMHeadModel = CompatGPT2
print('Compatibility GPT2LMHeadModel installed')
```

Compatibility GPT2LMHeadModel installed

```
# Small helper: run PPLM example using `full_text_generation` so outputs are returned and can be displayed neatly
import importlib
import run_pplm
import torch
from transformers import GPT2Tokenizer, GPT2LMHeadModel
import glob, os

# Ensure run_pplm uses the compatibility GPT2 class defined earlier (if present)
try:
    from transformers.modeling_gpt2 import GPT2LMHeadModel as CompatFromShim
    run_pplm.GPT2LMHeadModel = CompatFromShim
    print('Patched run_pplm.GPT2LMHeadModel to compatibility class')
except Exception:
    # if shim didn't provide that, try to set to transformers.GPT2LMHeadModel if present
    import transformers
    if hasattr(transformers, 'GPT2LMHeadModel'):
```

```

run_pplm.GPT2LMHeadModel = transformers.GPT2LMHeadModel
print('Patched run_pplm.GPT2LMHeadModel from transformers')
else:
    print('Could not patch run_pplm.GPT2LMHeadModel; proceeding anyway')

# reload to be safe
importlib.reload(run_pplm)

from run_pplm import full_text_generation
from run_pplm import generate_text_pplm # Import the generate_text_pplm function

def resolve_bow_path(path_like):
    if path_like is None:
        return None
    if os.path.exists(path_like):
        return path_like
    base = os.path.basename(path_like)
    matches = glob.glob('*/' + base, recursive=True)
    if matches:
        print(f'Resolved {path_like} -> {matches[0]}')
        return matches[0]
    print(f'Could not resolve bag-of-words path: {path_like}')
    return path_like

_model_cache = {}

def get_model_and_tokenizer(pretrained_model='gpt2', device=None):
    device = device or ('cuda' if torch.cuda.is_available() else 'cpu')
    key = (pretrained_model, device)
    if key in _model_cache:
        return _model_cache[key]
    # Ensure output_hidden_states is True when loading the model
    model = GPT2LMHeadModel.from_pretrained(pretrained_model, output_hidden_states=True)
    model.to(device)
    model.train() # Set model to training mode to enable gradients
    # for p in model.parameters(): # Removed this loop to allow gradients
    #     p.requires_grad = False
    tokenizer = GPT2Tokenizer.from_pretrained(pretrained_model)
    _model_cache[key] = (model, tokenizer, device)
    return model, tokenizer, device

def run_short_demo(cond_text, mode='bow', mode_value=None, device=None, pretrained_model='gpt2'):
    model, tokenizer, device = get_model_and_tokenizer(pretrained_model=pretrained_model, device=device)
    print('Using device:', device)

    tokenized_cond = tokenizer.encode(tokenizer.bos_token + cond_text, add_special_tokens=False)

    if mode == 'bow':
        bow_arg = resolve_bow_path(mode_value)
        discrim = None
        class_label = -1
    elif mode == 'discrim':
        bow_arg = None
        discrim = mode_value
        class_label = -1
    else:
        raise ValueError('mode must be "bow" or "discrim"')

    # Call full_text_generation which returns tokenized outputs
    unpert_tok, pert_tok_texts, discrim_losses, losses_in_time = full_text_generation(
        model=model,
        tokenizer=tokenizer,
        context=tokenized_cond,
        device=device,
        num_samples=1,
        bag_of_words=bow_arg,
        discrim=discrim,
        class_label=class_label,
        length=30,
        stepsize=0.02,
        temperature=1.0,
        top_k=10,
        sample=True,
        num_iterations=2,
        grad_length=10000,
        horizon_length=1,
        window_length=0,
        decay=False,
        gamma=1.5,
        gm_scale=0.9,

```

```

        kl_scale=0.01,
        verbosity_level=0
    )

    # Decode and print neatly
    unpert_text = tokenizer.decode(unpert_tok.tolist()[0]) if unpert_tok is not None else ''
    print('\n=== Unperturbed ===\n')
    print(unpert_text)

    for i, pert in enumerate(pert_tok_texts):
        try:
            pert_text = tokenizer.decode(pert.tolist()[0])
            print(f'\n=== Perturbed [{i}] ===\n')
            print(pert_text)
        except Exception as e:
            print('Error decoding perturbed text:', e)

    return unpert_text, [tokenizer.decode(p.tolist()[0]) for p in pert_tok_texts]

print('Notebook helper ready')

```

Patched run\_pplm.GPT2LMHeadModel to compatibility class  
Notebook helper ready

## ✓ BoW demo (topic steering)

We use a local wordlist shipped in the repo for determinism.

```

bow_path = 'wordlists/space.txt'
prefix = 'The expedition set out'
print('Running BoW demo - this prints generated text below:')
unpert_text, pert_texts = run_short_demo(prefix, mode='bow', mode_value=bow_path)

```

Running BoW demo - this prints generated text below:  
Using device: cpu  
Resolved wordlists/space.txt -> paper\_code/wordlists/space.txt

```

-----
TypeError                                Traceback (most recent call last)
/tmp/ipython-input-4070235611.py in <cell line: 0>()
      2 prefix = 'The expedition set out'
      3 print('Running BoW demo - this prints generated text below:')
----> 4 unpert_text, pert_texts = run_short_demo(prefix, mode='bow', mode_value=bow_path)

```

```

-----
2 frames
/content/PPLM/PPLM/run_pplm.py in generate_text_pplm(model, tokenizer, context, past, device, perturb, bow_indices,
classifier, class_label, loss_type, length, stepsize, temperature, top_k, sample, num_iterations, grad_length,
horizon_length, window_length, decay, gamma, gm_scale, kl_scale, verbosity_level)
    565
    566 unpert_logits, unpert_past, unpert_all_hidden = model(output_so_far)
--> 567 unpert_last_hidden = unpert_all_hidden[-1]
    568
    569 # check if we are above grad max length

TypeError: 'NoneType' object is not subscriptable

```

Next steps: [Explain error](#)

## ✓ Discriminator demo (sentiment steering)

We use the `sentiment` discriminator included with the repo.

```

prefix = 'I felt that the evening was'
print('Running discriminator demo - this prints generated text below:')
run_short_demo(prefix, mode='discrim', mode_value='sentiment')

```

## Troubleshooting notes

- If any downloads fail (discriminator weights or tokenizer files), re-run the cell or check internet access in Colab.
- To speed up: switch `pretrained_model` to a smaller model (already set to `gpt2`) or enable GPU under Colab Runtime > Change runtime type > GPU.
- If you want the notebook to save generated outputs to a file for later inclusion in slides, I can wire that up as well.

