GOOGLE ADS NGRAM ANALYSIS SCRIPT

```
function main() {
  // ////////////////////////////////////////////////////////////////////////////
  // Options

  var startDate = '2015-07-01';
  var endDate = '2015-07-30';
  // The start and end date of the date range for your search query data
  // Format is yyyy-mm-dd

  var currencySymbol = '£';
  // The currency symbol used for formatting. For example "£", "$" or "€".

  var campaignNameContains = '';
  // Use this if you only want to look at some campaigns
  // such as campaigns with names containing 'Brand' or 'Shopping'.
  // Leave as "" if not wanted.

  var campaignNameDoesNotContain = '';
  // Use this if you want to exclude some campaigns
  // such as campaigns with names containing 'Brand' or 'Shopping'.
  // Leave as "" if not wanted.

  var ignorePausedCampaigns = true;
  // Set this to true to only look at currently active campaigns.
  // Set to false to include campaigns that had impressions but are currently
paused.

  var ignorePausedAdGroups = true;
  // Set this to true to only look at currently active ad groups.
  // Set to false to include ad groups that had impressions but are currently
paused.

  var checkNegatives = true;
  // Set this to true to remove queries that would be excluded by your negative
keywords.

  var spreadsheetUrl = 'https://docs.google.com/YOUR-SPREADSHEET-URL-HERE';
  // The URL of the Google Doc the results will be put into.

  var minNGramLength = 1;
  var maxNGramLength = 2;
  // The word length of phrases to be checked.
  // For example if minNGramLength is 1 and maxNGramLength is 3,
  // phrases made of 1, 2 and 3 words will be checked.
  // Change both min and max to 1 to just look at single words.

  var clearSpreadsheet = true;


  // ////////////////////////////////////////////////////////////////////////////
  // Thresholds
```

```
  var queryCountThreshold = 0;
  var impressionThreshold = 10;
  var clickThreshold = 0;
  var costThreshold = 0;
  var conversionThreshold = 0;
  // Words will be ignored if their statistics are lower than any of these
thresholds


  // ////////////////////////////////////////////////////////////////////////
  // Check the spreadsheet has been entered, and that it works
  if (spreadsheetUrl.replace(/[AEIOU]/g, 'X') ==
'https://docs.google.com/YXXR-SPRXXDSHXXT-XRL-HXRX') {
    Logger.log("Problem with the spreadsheet URL: make sure you've replaces the
default with a valid spreadsheet URL.");
    return;
  }
  try {
    var spreadsheet = SpreadsheetApp.openByUrl(spreadsheetUrl);
  } catch (e) {
    Logger.log("Problem with the spreadsheet URL: '" + e + "'");
    return;
  }

  // Get the IDs of the campaigns to look at
  var dateRange = startDate.replace(/-/g, '') + ',' + endDate.replace(/-/g, '');
  var activeCampaignIds = [];
  var whereStatements = '';

  if (campaignNameDoesNotContain != '') {
    whereStatements += "AND CampaignName DOES_NOT_CONTAIN_IGNORE_CASE '" +
campaignNameDoesNotContain + "' ";
  }
  if (ignorePausedCampaigns) {
    whereStatements += 'AND CampaignStatus = ENABLED ';
  } else {
    whereStatements += "AND CampaignStatus IN ['ENABLED','PAUSED'] ";
  }

  var campaignReport = AdWordsApp.report(
    'SELECT CampaignName, CampaignId '
    + 'FROM   CAMPAIGN_PERFORMANCE_REPORT '
    + "WHERE CampaignName CONTAINS_IGNORE_CASE '" + campaignNameContains + "' "
    + 'AND Impressions > 0 ' + whereStatements
    + 'DURING ' + dateRange
  );
  var campaignRows = campaignReport.rows();
  while (campaignRows.hasNext()) {
    var campaignRow = campaignRows.next();
    activeCampaignIds.push(campaignRow.CampaignId);
  }// end while

  if (activeCampaignIds.length == 0) {
    Logger.log('Could not find any campaigns with impressions and the specified
options.');
    return;
  }

  var whereAdGroupStatus = '';
```

```
  if (ignorePausedAdGroups) {
    var whereAdGroupStatus = 'AND AdGroupStatus = ENABLED ';
  } else {
    whereAdGroupStatus += "AND AdGroupStatus IN ['ENABLED','PAUSED'] ";
  }


  // ////////////////////////////////////////////////////////////////////////
  // Find the negative keywords
  var negativesByGroup = [];
  var negativesByCampaign = [];
  var sharedSetData = [];
  var sharedSetNames = [];
  var sharedSetCampaigns = [];

  if (checkNegatives) {
    // Gather ad group level negative keywords
    var keywordReport = AdWordsApp.report(
      'SELECT CampaignId, AdGroupId, Criteria, KeywordMatchType '
      + 'FROM   KEYWORDS_PERFORMANCE_REPORT '
      + 'WHERE Status = ENABLED AND IsNegative = TRUE ' + whereAdGroupStatus
      + 'AND CampaignId IN [' + activeCampaignIds.join(',') + '] '
      + 'DURING ' + dateRange
    );

    var keywordRows = keywordReport.rows();
    while (keywordRows.hasNext()) {
      var keywordRow = keywordRows.next();

      if (negativesByGroup[keywordRow.AdGroupId] == undefined) {
        negativesByGroup[keywordRow.AdGroupId] =
[[keywordRow.Criteria.toLowerCase(), keywordRow.KeywordMatchType.toLowerCase()]];
      } else {

negativesByGroup[keywordRow.AdGroupId].push([keywordRow.Criteria.toLowerCase(),
keywordRow.KeywordMatchType.toLowerCase()]);
      }
    }

    // Gather campaign level negative keywords
    var campaignNegReport = AdWordsApp.report(
      'SELECT CampaignId, Criteria, KeywordMatchType '
      + 'FROM   CAMPAIGN_NEGATIVE_KEYWORDS_PERFORMANCE_REPORT '
      + 'WHERE  IsNegative = TRUE '
      + 'AND CampaignId IN [' + activeCampaignIds.join(',') + ']'
    );
    var campaignNegativeRows = campaignNegReport.rows();
    while (campaignNegativeRows.hasNext()) {
      var campaignNegativeRow = campaignNegativeRows.next();
      if (negativesByCampaign[campaignNegativeRow.CampaignId] == undefined) {
        negativesByCampaign[campaignNegativeRow.CampaignId] =
[[campaignNegativeRow.Criteria.toLowerCase(),
campaignNegativeRow.KeywordMatchType.toLowerCase()]];
      } else {

negativesByCampaign[campaignNegativeRow.CampaignId].push([campaignNegativeRow.Crite
ria.toLowerCase(), campaignNegativeRow.KeywordMatchType.toLowerCase()]);
      }
    }
```

```
    // Find which campaigns use shared negative keyword sets
    var campaignSharedReport = AdWordsApp.report(
      'SELECT CampaignName, CampaignId, SharedSetName, SharedSetType, Status '
      + 'FROM   CAMPAIGN_SHARED_SET_REPORT '
      + 'WHERE SharedSetType = NEGATIVE_KEYWORDS '
      + 'AND CampaignId IN [' + activeCampaignIds.join(',') + ']'
    );
    var campaignSharedRows = campaignSharedReport.rows();
    while (campaignSharedRows.hasNext()) {
      var campaignSharedRow = campaignSharedRows.next();
      if (sharedSetCampaigns[campaignSharedRow.SharedSetName] == undefined) {
        sharedSetCampaigns[campaignSharedRow.SharedSetName] =
[campaignSharedRow.CampaignId];
      } else {

sharedSetCampaigns[campaignSharedRow.SharedSetName].push(campaignSharedRow.Campaign
Id);
      }
    }

    // Map the shared sets' IDs (used in the criteria report below)
    // to their names (used in the campaign report above)
    var sharedSetReport = AdWordsApp.report(
      'SELECT Name, SharedSetId, MemberCount, ReferenceCount, Type '
      + 'FROM   SHARED_SET_REPORT '
      + 'WHERE ReferenceCount > 0 AND Type = NEGATIVE_KEYWORDS '
    );
    var sharedSetRows = sharedSetReport.rows();
    while (sharedSetRows.hasNext()) {
      var sharedSetRow = sharedSetRows.next();
      sharedSetNames[sharedSetRow.SharedSetId] = sharedSetRow.Name;
    }

    // Collect the negative keyword text from the sets,
    // and record it as a campaign level negative in the campaigns that use the set
    var sharedSetReport = AdWordsApp.report(
      'SELECT SharedSetId, KeywordMatchType, Criteria '
      + 'FROM   SHARED_SET_CRITERIA_REPORT '
    );
    var sharedSetRows = sharedSetReport.rows();
    while (sharedSetRows.hasNext()) {
      var sharedSetRow = sharedSetRows.next();
      var setName = sharedSetNames[sharedSetRow.SharedSetId];
      if (sharedSetCampaigns[setName] !== undefined) {
        for (var i = 0; i < sharedSetCampaigns[setName].length; i++) {
          var campaignId = sharedSetCampaigns[setName][i];
          if (negativesByCampaign[campaignId] == undefined) {
            negativesByCampaign[campaignId] =
[[sharedSetRow.Criteria.toLowerCase(),
sharedSetRow.KeywordMatchType.toLowerCase()]];
          } else {

negativesByCampaign[campaignId].push([sharedSetRow.Criteria.toLowerCase(),
sharedSetRow.KeywordMatchType.toLowerCase()]);
          }
        }
      }
    }
```

```
    Logger.log('Finished finding negative keywords');
  }// end if


  // ////////////////////////////////////////////////////////////////////////
  // Define the statistics to download or calculate, and their formatting

  var statColumns = ['Clicks', 'Impressions', 'Cost', 'Conversions',
'ConversionValue'];
  var calculatedStats = [['CTR', 'Clicks', 'Impressions'],
    ['CPC', 'Cost', 'Clicks'],
    ['Conv. Rate', 'Conversions', 'Clicks'],
    ['Cost / conv.', 'Cost', 'Conversions'],
    ['Conv. value/cost', 'ConversionValue', 'Cost']];
  var currencyFormat = currencySymbol + '#,##0.00';
  var formatting = ['#,##0', '#,##0', '#,##0', currencyFormat, '#,##0',
currencyFormat, '0.00%', currencyFormat, '0.00%', currencyFormat, '0.00%'];


  // ////////////////////////////////////////////////////////////////////////
  // Go through the search query report, remove searches already excluded by
negatives
  // record the performance of each word in each remaining query

  var queryReport = AdWordsApp.report(
    'SELECT CampaignName, CampaignId, AdGroupId, AdGroupName, Query, ' +
statColumns.join(', ') + ' '
    + 'FROM SEARCH_QUERY_PERFORMANCE_REPORT '
      + 'WHERE CampaignId IN [' + activeCampaignIds.join(',') + '] ' +
whereAdGroupStatus
        + 'DURING ' + dateRange
  );

  var numberOfWords = [];
  var campaignNGrams = {};
  var adGroupNGrams = {};
  var totalNGrams = [];

  for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
    totalNGrams[n] = {};
  }

  var queryRows = queryReport.rows();
  while (queryRows.hasNext()) {
    var queryRow = queryRows.next();

    if (checkNegatives) {
      var searchIsExcluded = false;

      // Checks if the query is excluded by an ad group level negative
      if (negativesByGroup[queryRow.AdGroupId] !== undefined) {
        for (var i = 0; i < negativesByGroup[queryRow.AdGroupId].length; i++) {
          if ((negativesByGroup[queryRow.AdGroupId][i][1] == 'exact'
                && queryRow.Query == negativesByGroup[queryRow.AdGroupId][i][0])
              || (negativesByGroup[queryRow.AdGroupId][i][1] != 'exact'
              && (' ' + queryRow.Query + ' ').indexOf(' ' +
negativesByGroup[queryRow.AdGroupId][i][0] + ' ') > -1)) {
            searchIsExcluded = true;
```

```
            break;
          }
        }
      }

      // Checks if the query is excluded by a campaign level negative
      if (!searchIsExcluded && negativesByCampaign[queryRow.CampaignId] !==
undefined) {
        for (var i = 0; i < negativesByCampaign[queryRow.CampaignId].length; i++) {
          if ((negativesByCampaign[queryRow.CampaignId][i][1] == 'exact'
                && queryRow.Query ==
negativesByCampaign[queryRow.CampaignId][i][0])
              || (negativesByCampaign[queryRow.CampaignId][i][1] != 'exact'
              && (' ' + queryRow.Query + ' ').indexOf(' ' +
negativesByCampaign[queryRow.CampaignId][i][0] + ' ') > -1)) {
            searchIsExcluded = true;
            break;
          }
        }
      }

      if (searchIsExcluded) { continue; }
    }

    var currentWords = queryRow.Query.split(' ');

    if (campaignNGrams[queryRow.CampaignName] == undefined) {
      campaignNGrams[queryRow.CampaignName] = [];
      adGroupNGrams[queryRow.CampaignName] = {};

      for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
        campaignNGrams[queryRow.CampaignName][n] = {};
      }
    }

    if (adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName] == undefined) {
      adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName] = [];
      for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
        adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName][n] = {};
      }
    }

    var stats = [];
    for (var i = 0; i < statColumns.length; i++) {
      stats[i] = parseFloat(queryRow[statColumns[i]].replace(/,/g, ''));
    }

    var wordLength = currentWords.length;
    if (wordLength > 6) {
      wordLength = '7+';
    }
    if (numberOfWords[wordLength] == undefined) {
      numberOfWords[wordLength] = [];
    }
    for (var i = 0; i < statColumns.length; i++) {
      if (numberOfWords[wordLength][statColumns[i]] > 0) {
        numberOfWords[wordLength][statColumns[i]] += stats[i];
      } else {
        numberOfWords[wordLength][statColumns[i]] = stats[i];
```

```
      }
    }

    // Splits the query into n-grams and records the stats for each
    for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
      if (n > currentWords.length) {
        break;
      }

      var doneNGrams = [];

      for (var w = 0; w < currentWords.length - n + 1; w++) {
        var currentNGram = '="' + currentWords.slice(w, w + n).join(' ') + '"';

        if (doneNGrams.indexOf(currentNGram) < 0) {
          if (campaignNGrams[queryRow.CampaignName][n][currentNGram] == undefined)
{
            campaignNGrams[queryRow.CampaignName][n][currentNGram] = {};
            campaignNGrams[queryRow.CampaignName][n][currentNGram]['Query Count'] =
0;
          }
          if
(adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName][n][currentNGram] ==
undefined) {

adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName][n][currentNGram] = {};

adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName][n][currentNGram]['Query
Count'] = 0;
          }
          if (totalNGrams[n][currentNGram] == undefined) {
            totalNGrams[n][currentNGram] = {};
            totalNGrams[n][currentNGram]['Query Count'] = 0;
          }

          campaignNGrams[queryRow.CampaignName][n][currentNGram]['Query Count']++;

adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName][n][currentNGram]['Query
Count']++;
          totalNGrams[n][currentNGram]['Query Count']++;

          for (var i = 0; i < statColumns.length; i++) {
            if
(campaignNGrams[queryRow.CampaignName][n][currentNGram][statColumns[i]] > 0) {

campaignNGrams[queryRow.CampaignName][n][currentNGram][statColumns[i]] += stats[i];
            } else {

campaignNGrams[queryRow.CampaignName][n][currentNGram][statColumns[i]] = stats[i];
            }

            if
(adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName][n][currentNGram][statCo
lumns[i]] > 0) {

adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName][n][currentNGram][statCol
umns[i]] += stats[i];
            } else {
```

```
adGroupNGrams[queryRow.CampaignName][queryRow.AdGroupName][n][currentNGram][statCol
umns[i]] = stats[i];
            }

            if (totalNGrams[n][currentNGram][statColumns[i]] > 0) {
              totalNGrams[n][currentNGram][statColumns[i]] += stats[i];
            } else {
              totalNGrams[n][currentNGram][statColumns[i]] = stats[i];
            }
          }

          doneNGrams.push(currentNGram);
        }
      }
    }
  }

  Logger.log('Finished analysing queries.');


  // ////////////////////////////////////////////////////////////////////////
  // Output the data into the spreadsheet

  var wordLengthOutput = [];
  var wordLengthFormat = [];
  var outputs = [];
  var formats = [];

  for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
    outputs[n] = {};
    outputs[n].campaign = [];
    outputs[n].adgroup = [];
    outputs[n].total = [];
    formats[n] = {};
    formats[n].campaign = [];
    formats[n].adgroup = [];
    formats[n].total = [];
  }

  // Create headers
  var calcStatNames = [];
  for (var s = 0; s < calculatedStats.length; s++) {
    calcStatNames.push(calculatedStats[s][0]);
  }
  var statNames = statColumns.concat(calcStatNames);
  for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
    outputs[n].campaign.push(['Campaign', 'Phrase', 'Query
Count'].concat(statNames));
    outputs[n].adgroup.push(['Campaign', 'Ad Group', 'Phrase', 'Query
Count'].concat(statNames));
    outputs[n].total.push(['Phrase', 'Query Count'].concat(statNames));
  }
  wordLengthOutput.push(['Word count'].concat(statNames));

  // Organise the ad group level stats into an array for output
  for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
    for (var campaign in adGroupNGrams) {
      for (var adGroup in adGroupNGrams[campaign]) {
```

```
        for (var nGram in adGroupNGrams[campaign][adGroup][n]) {
          // skips nGrams under the thresholds
          if (adGroupNGrams[campaign][adGroup][n][nGram]['Query Count'] <
queryCountThreshold) { continue; }
          if (adGroupNGrams[campaign][adGroup][n][nGram].Impressions <
impressionThreshold) { continue; }
          if (adGroupNGrams[campaign][adGroup][n][nGram].Clicks < clickThreshold) {
continue; }
          if (adGroupNGrams[campaign][adGroup][n][nGram].Cost < costThreshold) {
continue; }
          if (adGroupNGrams[campaign][adGroup][n][nGram].Conversions <
conversionThreshold) { continue; }

          var printline = [campaign, adGroup, nGram,
adGroupNGrams[campaign][adGroup][n][nGram]['Query Count']];

          for (var s = 0; s < statColumns.length; s++) {

printline.push(adGroupNGrams[campaign][adGroup][n][nGram][statColumns[s]]);
          }

          for (var s = 0; s < calculatedStats.length; s++) {
            var multiplier = calculatedStats[s][1];
            var divisor = calculatedStats[s][2];
            if (adGroupNGrams[campaign][adGroup][n][nGram][divisor] > 0) {
              printline.push(adGroupNGrams[campaign][adGroup][n][nGram][multiplier]
/ adGroupNGrams[campaign][adGroup][n][nGram][divisor]);
            } else {
              printline.push('-');
            }
          }
          outputs[n].adgroup.push(printline);
          formats[n].adgroup.push(['0', '0', '0'].concat(formatting));
        }
      }
    }
  }

  // Organise the campaign level stats into an array for output
  for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
    for (var campaign in campaignNGrams) {
      for (var nGram in campaignNGrams[campaign][n]) {
        // skips nGrams under the thresholds
        if (campaignNGrams[campaign][n][nGram]['Query Count'] <
queryCountThreshold) { continue; }
        if (campaignNGrams[campaign][n][nGram].Impressions < impressionThreshold) {
continue; }
        if (campaignNGrams[campaign][n][nGram].Clicks < clickThreshold) { continue;
}
        if (campaignNGrams[campaign][n][nGram].Cost < costThreshold) { continue; }
        if (campaignNGrams[campaign][n][nGram].Conversions < conversionThreshold) {
continue; }

        var printline = [campaign, nGram, campaignNGrams[campaign][n][nGram]['Query
Count']];

        for (var s = 0; s < statColumns.length; s++) {
          printline.push(campaignNGrams[campaign][n][nGram][statColumns[s]]);
        }
```

```
        for (var s = 0; s < calculatedStats.length; s++) {
          var multiplier = calculatedStats[s][1];
          var divisor = calculatedStats[s][2];
          if (campaignNGrams[campaign][n][nGram][divisor] > 0) {
            printline.push(campaignNGrams[campaign][n][nGram][multiplier] /
campaignNGrams[campaign][n][nGram][divisor]);
          } else {
            printline.push('-');
          }
        }
        outputs[n].campaign.push(printline);
        formats[n].campaign.push(['0', '0'].concat(formatting));
      }
    }
  }

  // Organise the account level stats into an array for output
  for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
    for (var nGram in totalNGrams[n]) {
      // skips n-grams under the thresholds
      if (totalNGrams[n][nGram]['Query Count'] < queryCountThreshold) { continue; }
      if (totalNGrams[n][nGram].Impressions < impressionThreshold) { continue; }
      if (totalNGrams[n][nGram].Clicks < clickThreshold) { continue; }
      if (totalNGrams[n][nGram].Cost < costThreshold) { continue; }
      if (totalNGrams[n][nGram].Conversions < conversionThreshold) { continue; }

      var printline = [nGram, totalNGrams[n][nGram]['Query Count']];

      for (var s = 0; s < statColumns.length; s++) {
        printline.push(totalNGrams[n][nGram][statColumns[s]]);
      }

      for (var s = 0; s < calculatedStats.length; s++) {
        var multiplier = calculatedStats[s][1];
        var divisor = calculatedStats[s][2];
        if (totalNGrams[n][nGram][divisor] > 0) {
          printline.push(totalNGrams[n][nGram][multiplier] /
totalNGrams[n][nGram][divisor]);
        } else {
          printline.push('-');
        }
      }
      outputs[n].total.push(printline);
      formats[n].total.push(['0'].concat(formatting));
    }
  }

  // Organise the word count analysis into an array for output
  for (var i = 1; i < 8; i++) {
    if (i < 7) {
      var wordLength = i;
    } else {
      var wordLength = '7+';
    }

    var printline = [wordLength];

    if (numberOfWords[wordLength] == undefined) {
```

```
      printline.push([0, 0, 0, 0, '-', '-', '-', '-']);
    } else {
      for (var s = 0; s < statColumns.length; s++) {
        printline.push(numberOfWords[wordLength][statColumns[s]]);
      }

      for (var s = 0; s < calculatedStats.length; s++) {
        var multiplier = calculatedStats[s][1];
        var divisor = calculatedStats[s][2];
        if (numberOfWords[wordLength][divisor] > 0) {
          printline.push(numberOfWords[wordLength][multiplier] /
numberOfWords[wordLength][divisor]);
        } else {
          printline.push('-');
        }
      }
    }
    wordLengthOutput.push(printline);
    wordLengthFormat.push(formatting);
  }

  var filterText = '';
  if (ignorePausedAdGroups) {
    filterText = 'Active ad groups';
  } else {
    filterText = 'All ad groups';
  }

  if (ignorePausedCampaigns) {
    filterText += ' in active campaigns';
  } else {
    filterText += ' in all campaigns';
  }

  if (campaignNameContains != '') {
    filterText += " containing '" + campaignNameContains + "'";
    if (campaignNameDoesNotContain != '') {
      filterText += " and not containing '" + campaignNameDoesNotContain + "'";
    }
  } else if (campaignNameDoesNotContain != '') {
    filterText += " not containing '" + campaignNameDoesNotContain + "'";
  }

  // Find or create the required sheets
  var spreadsheet = SpreadsheetApp.openByUrl(spreadsheetUrl);
  var campaignNGramName = [];
  var adGroupNGramName = [];
  var totalNGramName = [];
  var campaignNGramSheet = [];
  var adGroupNGramSheet = [];
  var totalNGramSheet = [];

  for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
    if (n == 1) {
      campaignNGramName[n] = 'Campaign Word Analysis';
      adGroupNGramName[n] = 'Ad Group Word Analysis';
      totalNGramName[n] = 'Account Word Analysis';
    } else {
      campaignNGramName[n] = 'Campaign ' + n + '-Gram Analysis';
```

```
      adGroupNGramName[n] = 'Ad Group ' + n + '-Gram Analysis';
      totalNGramName[n] = 'Account ' + n + '-Gram Analysis';
    }

    campaignNGramSheet[n] = spreadsheet.getSheetByName(campaignNGramName[n]);
    if (campaignNGramSheet[n] == null) {
      campaignNGramSheet[n] = spreadsheet.insertSheet(campaignNGramName[n]);
    }

    adGroupNGramSheet[n] = spreadsheet.getSheetByName(adGroupNGramName[n]);
    if (adGroupNGramSheet[n] == null) {
      adGroupNGramSheet[n] = spreadsheet.insertSheet(adGroupNGramName[n]);
    }

    totalNGramSheet[n] = spreadsheet.getSheetByName(totalNGramName[n]);
    if (totalNGramSheet[n] == null) {
      totalNGramSheet[n] = spreadsheet.insertSheet(totalNGramName[n]);
    }
  }

  var wordCountSheet = spreadsheet.getSheetByName('Word Count Analysis');
  if (wordCountSheet == null) {
    wordCountSheet = spreadsheet.insertSheet('Word Count Analysis');
  }

  // Write the output arrays to the spreadsheet
  for (var n = minNGramLength; n < maxNGramLength + 1; n++) {
    var nGramName = n + '-Grams';
    if (n == 1) {
      nGramName = 'Words';
    }

    writeOutput(outputs[n].campaign, formats[n].campaign, campaignNGramSheet[n],
nGramName, 'Campaign', filterText, clearSpreadsheet);
    writeOutput(outputs[n].adgroup, formats[n].adgroup, adGroupNGramSheet[n],
nGramName, 'Ad Group', filterText, clearSpreadsheet);
    writeOutput(outputs[n].total, formats[n].total, totalNGramSheet[n], nGramName,
'Account', filterText, clearSpreadsheet);
  }

  writeOutput(wordLengthOutput, wordLengthFormat, wordCountSheet, 'Word Count',
'Account', filterText, clearSpreadsheet);

  Logger.log('Finished writing to spreadsheet.');
} // end main function


function writeOutput(outputArray, formatArray, sheet, nGramName, levelName,
filterText, clearSpreadsheet) {
  for (var i = 0; i < 5; i++) {
    try {
      if (clearSpreadsheet) {
        sheet.clear();
      }

      if (nGramName == 'Word Count') {
        sheet.getRange('R1C1').setValue('Analysis of Search Query Performance by
Word Count');
      } else {
```

```
        sheet.getRange('R1C1').setValue('Analysis of ' + nGramName + ' in Search
Query Report, By ' + levelName);
      }

      sheet.getRange('R' + (sheet.getLastRow() + 2) + 'C1').setValue(filterText);

      var lastRow = sheet.getLastRow();

      if (formatArray.length == 0) {
        sheet.getRange('R' + (lastRow + 1) + 'C1').setValue('No ' +
nGramName.toLowerCase() + ' found within the thresholds.');
      } else {
        sheet.getRange('R' + (lastRow + 1) + 'C1:R' + (lastRow +
outputArray.length) + 'C' + outputArray[0].length).setValues(outputArray);
        sheet.getRange('R' + (lastRow + 2) + 'C1:R' + (lastRow +
outputArray.length) + 'C' + formatArray[0].length).setNumberFormats(formatArray);

        var sortByColumns = [];
        if (outputArray[0][0] == 'Campaign' || outputArray[0][0] == 'Word count') {
          sortByColumns.push({ column: 1, ascending: true });
        }
        if (outputArray[0][1] == 'Ad Group') {
          sortByColumns.push({ column: 2, ascending: true });
        }
        sortByColumns.push({ column: outputArray[0].indexOf('Cost') + 1, ascending:
false });
        sortByColumns.push({ column: outputArray[0].indexOf('Impressions') + 1,
ascending: false });
        sheet.getRange('R' + (lastRow + 2) + 'C1:R' + (lastRow +
outputArray.length) + 'C' + outputArray[0].length).sort(sortByColumns);
      }

      break;
    } catch (e) {
      if (e == 'Exception: This action would increase the number of cells in the
worksheet above the limit of 2000000 cells.') {
        Logger.log('Could not output ' + levelName + ' level ' +
nGramName.toLowerCase() + ": '" + e + "'");
        try {
          sheet.getRange('R' + (sheet.getLastRow() + 2) + 'C1').setValue('Not
enough space to write the data - try again in an empty spreadsheet');
        } catch (e2) {
          Logger.log("Error writing 'not enough space' message: " + e2);
        }
        break;
      }

      if (i == 4) {
        Logger.log('Could not output ' + levelName + ' level ' +
nGramName.toLowerCase() + ": '" + e + "'");
      }
    }
  }
```