```
##################################################################################
###                Notes:
###This file consists of four Perl programs and one ado file.
###The first program divides scripts into presentation and Q&A section.
###The second one marks each speaker as either an analyst or management (CEO or CFO) using the Presentation files.
###The third one prints the presentation by CEO and CFO into txt files.
###The furth one counts the number of product-market-related words mentioned by CEO and CFO.
###The do file reads the outputs following step 4 and generte firm-year level product-market discussions.
##################################################################################




##################################################################################
###                Program 1
###This program separates the conference call scripts into presentation section and Q&A section.
###
##################################################################################
use lib "C:/Perl64/site/lib/";
# Input directory
chdir("~");
# Output directory and file name
my $OUTPUTDIRPRE = "output_step1"; #presentation section
my $OUTPUTDIRQA = "output_step1_QA"; #Q&A section

# Identifying Q&A section with orders.
# Thomson Reuters usually use =Qusetions and Answer= or =Q & A= to separate the presentation and Q&A sections ($Splitword1). However, it is not uncommon that Thomson Reuters fail to provide such
# indicators. In this case, we rely on texture analysis to identify the beginning of Q&A sections (e.g., let's start our Q&A section; let's go for today's first question, etc.). We identify these
# keywords based on our reading of 200 scripts without the indicators provided by Thomson Reuters.

my $Splitword1= '(\=.{0,20}?Questions\W+and\W+Answer|\=.{0,20}Question\W+and\W+Answer|\=.{0,20}Q\W+and\W+A)';
my $Splitword2= '(Questions\W+and\W+Answer|Question\W+Answer|q\W+and\W+a|\W+(open|ask|have|has|for|go|start|begin|opens|asks|goes|begins|starts)\W+.{0,50}question)';
my $Splitword3= '(let\'s.{1,50}$Splitword2|let\W+us.{1,50}$Splitword2|\W+now\W+.{1,50}$Splitword2|$Splitword2.{1,20}\W+now\W+|today.{1,10}first\W+question|first\W+question.{1,10}from)';

while ( defined( $file = <*.xml> ) ) {

    $file =~ /(.*?)\_(.*?)\.xml/;

    my $cid = $1;
    my $name = $2;

    local $/;
    # Open the input file and read as string
    open(INPUT,"$file");
    $text=<INPUT>;

```

```
    #replace & with space;
    $text=~s/&amp;/ and /g;
    $text=~s/&/ and /g;

    my $pre = ""; my $qa="";my $tmp="";
    #identify Q&A;
     if ($text=~ /Body\>(.*?)($Splitword1.*?)\<\/Body\>/ism) {
     $text=~ /Body\>(.*?)($Splitword1.*?)\/Body\>/ism;
     $pre = $1;
     $qa  = $2;
     } elsif ($text=~ /Body\>(.*?$Splitword3.{1,1000}?---)(.*?)\<\/Body\>/ism) {
     $pre = $1;
     $tmp=$2;
     $text=~ /Body\>.*?$tmp.{0,1000}?---(.*?)\<\/Body\>/ism;
     $qa  = $1;
     } elsif($text=~ /Body\>(.*?)(\[.{0,10}(4|5|6|7|8|9|\d{2,2}).{0,10}\].{0,300}
     $Splitword2.{1,1000}---.*?)\/Body\>/ism) {
     $pre = $1;
     $qa  = $2;
     } else {
     $text=~ /Body\>(.*?)\/Body\>/ism;
     $pre = $1;
     $qa  = "";
     }

    my $pre_file=$cid."_"."pre.txt";
    my $qa_file=$cid."_"."qa.txt";

   # Open the output file
   open(OUTPUTPRE, ">$OUTPUTDIRPRE$pre_file");
   open(OUTPUTQA, ">$OUTPUTDIRQA$qa_file");
   print OUTPUTPRE $pre;
   print OUTPUTQA $qa;
   close(OUTPUTPRE);
   close(OUTPUTQA);

    }


##########################################################################################
###                Program 2
###This program marks each speaker as either an analyst or management (CEO or CFO) using the Presentation files.
###
##########################################################################################
use lib "C:/Perl64/site/lib/";
use File::chdir;

# define subroutines
sub trim($);
sub isanalyst($);
sub isceo($);
sub iscfo($);

# input directory
my $CCDIR = "output_step1";
#output directory
my $OUTPUTDIR = "output_step2";

# change the working directory
chdir($CCDIR);

while ( defined( my $file = <*.txt> ) ) {
    print "Processing file... $file\n";

```

```
    # open the file
    my $outfile = $file;
    $outfile =~ s/\.txt/Marked.txt/;
    open(OUTPUT,">$OUTPUTDIR/$outfile");

    open (INPUT, "$file") or die &!;
    # skip the header line from processing
    my $headerline = <INPUT>;
    print OUTPUT "type\t".$headerline;

    my @txt=<INPUT>;
    my $numline=@txt;
    my $tag="";
    my @tagarray="";
    $tagarray[0]="TileIsOther";

    for($l=1;$l<$numline;$l=$l+1){
      $lagl=$l-1;
      $leadl=$l+1;
    if (($txt[$l]=~/\[\W*\d+\W*\]/) & isline($txt[$lagl]) & isline($txt[$leadl])){
        # Check title
        if(isanalyst($txt[$l])){
            $tag="TileIsAnalyst";
        }elsif(isceo($txt[$l]) || iscfo($txt[$l]) ){
            $tag="TileIsCEOCFO";
        }else{
            $tag="TileIsOther";
        }
         $tagarray[$l]=$tag;
    }
    else{
        $tagarray[$l]=$tagarray[$lagl];
    }
    print OUTPUT "$tagarray[$l]\t",$txt[$l];
    }
}


# remove empty spaces from the beginning and the end of a text
sub trim($){
    my $string = shift;
    $string =~ s/^\s+//;
    $string =~ s/\s+$//;
    return $string;
}

# check if the title is sting line
sub isline($){
    my $string = shift;
    return $string =~ /------/sxi;
}
# check if the title is a ceo
sub isceo($){
    my $string = shift;
    return $string =~ /\W+(ceo|cep\W|(chief|cheif)\sexec|senior\sexecutive|chief\sofficer|chief\syahoo|interim\scompany\sleader)\W+.{0,50}\[\W*\d+\W*\]/sxi;
}

# check if the title is a cfo
sub iscfo($){
    my $string = shift;
    return $string =~ /\W+((chief|cheif)\sfinancial|cfo|cfao|Vice\sChairman,\sFinanc)\W+.{0,50}\[\W*\d+\W*\]/sxi;
}

```

```
# check if the title is an analyst
sub isanalyst($){
    my $string = shift;
    return $string =~ /\W+(abalyst|aanalyst|analyst|analysts|aalyst|Aalyst|Aanlsyt|Aanlyst|An\salyst|Anaalyst|Anaklyst|Anakyst|Anal|Analast|Analayst|Analhst|Analhyst|Analist|Analiyst|Analkyst|Anallyst|Analoyst
    |Analst|Analsty|Analsyst|Analsyt|Analtst|Analust|Analusy|Analy|Analyast|Analyat|Analydt|Analyist|Analylst|Analys|Analys6t|Analysat|Analyse
    |Analysi|Analysis|Analysit|Analyslt|Analysrt|Analysst|Analysy|Analysyt|Analyt|Analyts|Analyuast|Analyust|Analyyst|Analyzt|Ananlyst|Anaslyst|Anaylist
    |Anaylst|Anaylt|Anaylyst|Anayst|Andalyst|Andlyst|Anlalyst|Anlayst|Anlt|Anlyst|Annalyst|Anqalyst|Anslyst|Anyalyst|Anylist|Anylst)\W+.{0,50}\[\W*\d+\W*\]/sxi;
}



################################################################################
###                Program 3
###This program prints the presentation by CEO and CFO into txt files.
###
################################################################################
use lib "C:/Perl64/site/lib/";
use File::chdir;

# define subroutines
sub trim($);
sub isanalyst($);
sub isceo($);
sub iscfo($);

##############################
# input directory: the files generated by step 2
my $CCDIR = "output_step2";

#output directory
my $OUTPUTDIR = "output_step3";
##############################

# change the working directory
chdir($CCDIR);

while ( defined( my $file = <*.txt> ) ) {
    print "Processing file... $file\n";

    # open the file
    my $outfile = $file;
    $outfile =~ s/preMarked/preCEOCFO/;
    open(OUTPUT,">$OUTPUTDIR/$outfile");

    open (INPUT, "$file") or die &!;
    my @txt=<INPUT>;
    my $numline=@txt;
    close(INPUT);

    for($l=0;$l<$numline;$l=$l+1){
    if ($txt[$l]=~/TileIsCEOCFO/gis){
    $txt[$l]=~s/TileIsCEOCFO\t//gis;
    print OUTPUT "$tagarray[$l]\t",$txt[$l];
    }
    }
```

```
    close(OUTPUT);
}


##################################################################################
###                Program 4
###This program counts the number of product-market-related words discussed by CEO and CFO.
###
##################################################################################
use lib "C:/Perl64/site/lib/";
use Data::Dumper;
use List::Util qw[min max];

# Prepare stemming
use Lingua::Stem::En;
use Lingua::StopWords qw( getStopWords );
my $stopwords = getStopWords('en');

sub trim($);
sub stemword($);

# Input directory of conference call scripts (presentation by CEO and CFO following step 3)
chdir("output_step3");
# Output directory and file name
my $OUTPUTDIR = "~";
my $OUTPUTFILENAME = "conference_product_market_counts.txt";

# Note: If you choose to use STEMWORDS = 0, changes to the code may need to be made
$STEMWORDS = 1;

#the dictionary used in our baseline model
#product,service,products,services,customer,customers,consumer,consumers,user,users,client,clients
# Open and load the dictionaries
open (DIC, "dic.txt") or die $!;

%dic_dictionary_hash = ();

while(<DIC>){
    my $line = trim($_);
    if($STEMWORDS == 1){
        # stem the dictionary word
        $line = stemword($line);
    }
    if ($line ne ''){
        # add the dictionary words to the hash
        $dic_dictionary_hash{$line} = 1;
    }
}
close(DIC);

# Open the output file
open(OUTPUT, ">$OUTPUTDIR$OUTPUTFILENAME");

# Print the output header
print OUTPUT "file\tcid\tname\ttotal_words\ttotal_words_nostop\tdic\n";

while ( defined( $file = <*.txt> ) ) {
    #print "Processing file... $OUTPUTFILENAME $file\n";

    $file =~ /(.*?)\_(.*?)\.txt/;

```

```
    my $cid = $1;
    my $name = $2;

    my $total_words = 0;
    my $total_words_nostop = 0;
    my $diccount = 0;

    # Open the input file and read by line
    open(INPUT,"$file");
    while(<INPUT>){
        # Read the line/sentence
        my $line = $_;

        # fix amp
        $line =~ s/&amp;/&/xi;

        # Remove excess spacing
        $line =~ s/\s+/ /g;
        $line =~ s/\n+//g;
        my @words = split(/ / ,$line);
        $total_words += @words;

        # Remove Stop Words
        my @wordsnostop =  grep { !$stopwords->{$_} } @words;
        $total_words_nostop += @wordsnostop;

        my $n = @words;

        # Break sentence into words
        foreach $word(@words){
            # check next five words for stock or share

            $word = trim($word);
            $word =~ s/[\[\]\>\<\?\(\)\\'!,;\/\*\/\"\.]//gi;

            # Make the word lowercase
            $word =~ tr/A-Z/a-z/;

            if($STEMWORDS == 1){
                # Stem the sentence word
                $word = stemword($word);
            }

            if ($word ne ''){
                # Check if the word exists in the dictionary hash table
                if(exists $dic_dictionary_hash{$word}){
                    ++$diccount;
                }
            }

        }
    }

    # Output
    print OUTPUT "$file\t$cid\t$name\t$total_words\t$total_words_nostop\t$diccount\n";
    close(INPUT);
}
close(OUTPUT);

# This subroutine trims the leading and trailing whitespace;
sub trim($){
    my $string = shift;
    $string =~ s/^\s+//;
    $string =~ s/\s+$//;
    return $string;
}
```

```
# This subroutine trims the leading and trailing whitespace;
sub stemword($){
    # Get the variables passed to the subroutine
    @words = shift;

    my $stemmed = Lingua::Stem::En::stem({ -words => [grep { !$stopwords->{$_} } @words
    ] });

    # if the stemmed word is null then return ''
    if(@$stemmed == 0){
        return '';
    }
    return @$stemmed[0];
}
```

```


*##########################################################################################
*###             Program 5, run by Stata
*###read the outputs and generate the depedent variable at the gvkey-year level
*###
*##########################################################################################

import delimited using "conference_product_market_counts.txt"

**obtain the header information of each call
merge 1:1 cid using conference_header,nogen keep(3)
/*conference_header.dta provides the header information of each conference call scripts, including
gvkey, date of the call (dateformatted), type of calls (eflag)*/

keep if gvkey~=""& eflag == 1
*keep only earnings calls;

keep if total_words_nostop >= 150 & total_words_nostop ~=.

g  year=year(dateformatted)

g pct0_dic = dic /total_words_nostop*1000

bys gvkey year:egen  pre_pct_dic  = mean(pct0_dic )

keep gvkey year pre_pct_dic

duplicates drop

save words_conference,replace

```