# Spooky Author Identification

*Eugene Girtcius*
*Transport and Telecommunication Institute*

*03 November, 2019*

## Contents



–>

## 1 Введение

Spooky Author Identification

Целью исследования является попытка предсказать автора по короткому отрывку из написанного им текста методами ML. В данных присутствуют 3 автора: Г.Ф. Лавкрафт (HPL), Э.А. По (EAP) и М.В. Шелли (MWS)

ЗАДАЧИ:

- Проведение EDA
- Выявление значимых признаков для классификации

- Построение нескольких моделей для сопоставления результатов
- Выбор Квазиоптимальной модели (или ансамбля)
- Попадание в ТОП 20% в рейтинге Kaggle
- Освоение инструментов для дипломной работы

В качестве основных подходов в работе будет рассмотрен *XGBoost*, нейронные сети(*LSTM*, *FastText*, *BERT*) и *наивный байесовский классификатор*. Для написания кода будет использоваться язык **R**.

С исходными данными можно ознакомиться здесь. Данные представлены в виде тренировочного (`../input/train.csv`) и тестового набора (`../input/test.csv`). Каждое наблюдение содержит короткий фрагмент текста (Как правило, предложение). Также представлен образец данных в формате, необходимой для загрузки в Kaggle

# 2 Подготовка

## 2.1 Загрузка библиотек

В работе использовались классические библиотеки машинного обучения: *keras*, *caret*, *xgboost*. Для разметки и манипуляций с текстом выбраны *tidytext*, *RDRPOSTagger*, *tm*, *wordcloud*. Визуализация проводилась через *ggplot*, обработка данных средствами *tidyverse*. Полный список библиотек приведен ниже.

```
# general visualisation
library('ggplot2') # visualisation
library('scales') # visualisation
library('grid') # visualisation
library('gridExtra') # visualisation
library('RColorBrewer') # visualisation
library('corrplot') # visualisation

# general data manipulation
library('dplyr') # data manipulation
library('readr') # input/output
library('data.table') # data manipulation
library('tibble') # data wrangling
library('tidyr') # data wrangling
library('stringr') # string manipulation
library('forcats') # factor manipulation

# specific visualisation
library('alluvial') # visualisation
```

```r
library('ggrepel') # visualisation
library('ggridges') # visualisation
library('gganimate') # visualisation
library('ggExtra') # visualisation

# specific data manipulation
library('lazyeval') # data wrangling
library('broom') # data wrangling
library('purrr') # string manipulation
library('reshape2') # data wrangling

# Text / NLP
library('tidytext') # text analysis
library('tm') # text analysis
library('SnowballC') # text analysis
library('topicmodels') # text analysis
library('wordcloud') # test visualisation
library('igraph') # visualisation
library('ggraph') # visualisation
library('babynames') # names

# Models
library('Matrix')
library('xgboost')
library('caret')

library('treemapify') #visualisation

require(tidyverse)
require(tidytext)
require(textstem)
require(qdap)
require(caret)
require(widyr)
require(broom)
require(keras)
require(gridExtra)
require(plotly)
require(scales)
require(ggcorrplot)
require(RDRPOSTagger)
require(parallel)
```

```
require(gmodels)
require(knitr)
require(plotly)
require(kableExtra)
```

### 2.2 загрузка данных

```
train <- read_csv('data/train.csv')
test <- read_csv('data/test.csv')
sample <- read.csv('data/sample_submission.csv')
```

## 3 Анализ и визуализация данных

### 3.1 Структура данных

Набор данных содержит 3 колонки, id, непосредственно текст и метку автора. В тренировочных данных 19579 наблюдений. Встречаются, как короткие, так и длинные отрывки. Приведенный отрывок максимальной длины похож на ошибку в разметке. Точки между предложенриями отсутствуют и данные попали в один отрывок. Возможно, это следует считать выбросом. В тестовых данных 8392 наблюдений и отсутствуют метки классов. Также присутствуют длинные отрывки с отсутствующими точками. Пропушенных данных в наблюдениях не обнаружено.

#### 3.1.1 Тренировочные данные

Первые 5 наблюдений

```
train %>%
  slice(1:5) %>%
  kable(caption = 'Hierarchical clustering result comparison',
        row.names = T) %>%
  kable_styling(latex_options = c('striped', 'scale_down', 'HOLD_position'))
```

Table 1: Hierarchical clustering result comparison

| | id | text | author |
|---|---|---|---|
| 1 | id26305 | This process, however, afforded me no means of ascertaining the dimensions of my dungeon; as I might make its circuit, and return to the point whence I set out, without being aware of the fact; so perfectly uniform seemed the wall. | EAP |
| 2 | id17569 | It never once occurred to me that the fumbling might be a mere mistake. | HPL |
| 3 | id11008 | In his left hand was a gold snuff box, from which, as he capered down the hill, cutting all manner of fantastic steps, he took snuff incessantly with an air of the greatest possible self satisfaction. | EAP |
| 4 | id27763 | How lovely is spring As we looked from Windsor Terrace on the sixteen fertile counties spread beneath, speckled by happy cottages and wealthier towns, all looked as in former years, heart cheering and fair. | MWS |
| 5 | id12958 | Finding nothing else, not even gold, the Superintendent abandoned his attempts; but a perplexed look occasionally steals over his countenance as he sits thinking at his desk. | HPL |

```
glimpse(train)
```

```
## Observations: 19,579
## Variables: 3
## $ id     <chr> "id26305", "id17569", "id11008", "id27763", "id12958", ...
## $ text   <chr> "This process, however, afforded me no means of ascerta...
## $ author <chr> "EAP", "HPL", "EAP", "MWS", "HPL", "MWS", "EAP", "EAP",...
```

```
summary(train)
```

```
##       id                text              author
##  Length:19579       Length:19579       Length:19579
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
```

Отрывок минимальной длины

```
train %>%
    slice(which.min(nchar(text)))
```

| id | text | author |
|---|---|---|
| id20021 | I breathed no longer. | EAP |

Пропущенные значения

```
colSums(is.na(train))
```

```
##     id   text author
##      0      0      0
```

### 3.1.2 Тестовые данные

Первые 5 наблюдений

```
test %>%
    slice(1:5)
```

| id | text |
|---|---|
| id02310 | Still, as I urged our leaving Ireland with such inquietude and impatience, my father thought it best to y |
| id24541 | If a fire wanted fanning, it could readily be fanned with a newspaper, and as the government grew wea |
| id00134 | And when they had broken down the frail door they found only this: two cleanly picked human skelet |
| id27757 | While I was thinking how I should possibly manage without them, one actually tumbled out of my hea |
| id04081 | I am not sure to what limit his knowledge may extend. |

```
glimpse(test)
```

```
## Observations: 8,392
```

```
## Variables: 2
## $ id   <chr> "id02310", "id24541", "id00134", "id27757", "id04081", "i...
## $ text <chr> "Still, as I urged our leaving Ireland with such inquietu...
```

```
summary(test)
```

```
##       id                text
##  Length:8392        Length:8392
##  Class :character   Class :character
##  Mode  :character   Mode  :character
```

Отрывок минимальной длины

```
test %>%
    slice(which.min(nchar(text)))
```

| id | text |
|---|---|
| id02295 | Then he gave a start. |

Пропущенные значения

```
colSums(is.na(test))
```

```
##   id text
##    0    0
```

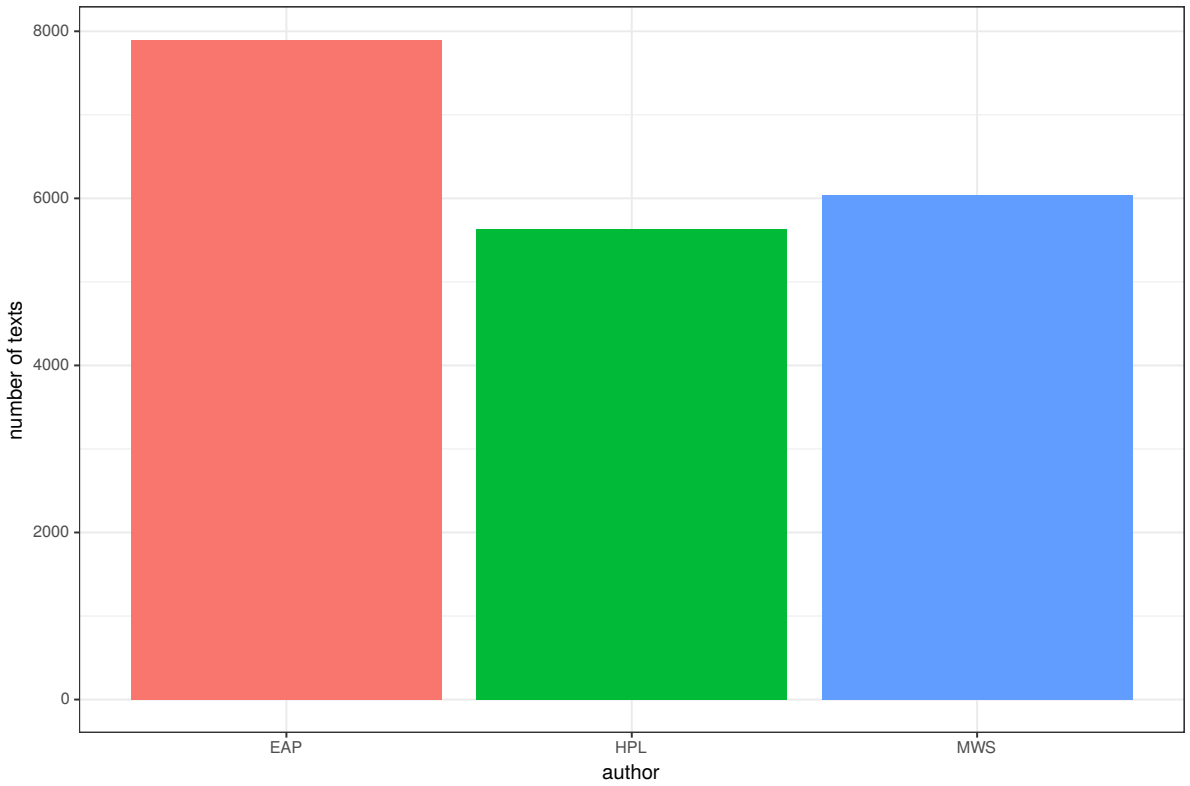
## 3.2 Пропорции

Тексты По занимают 40%, Шелли 31%, Лавкрафта 29%. Длина предложений у По немного короче, чем у остальных.

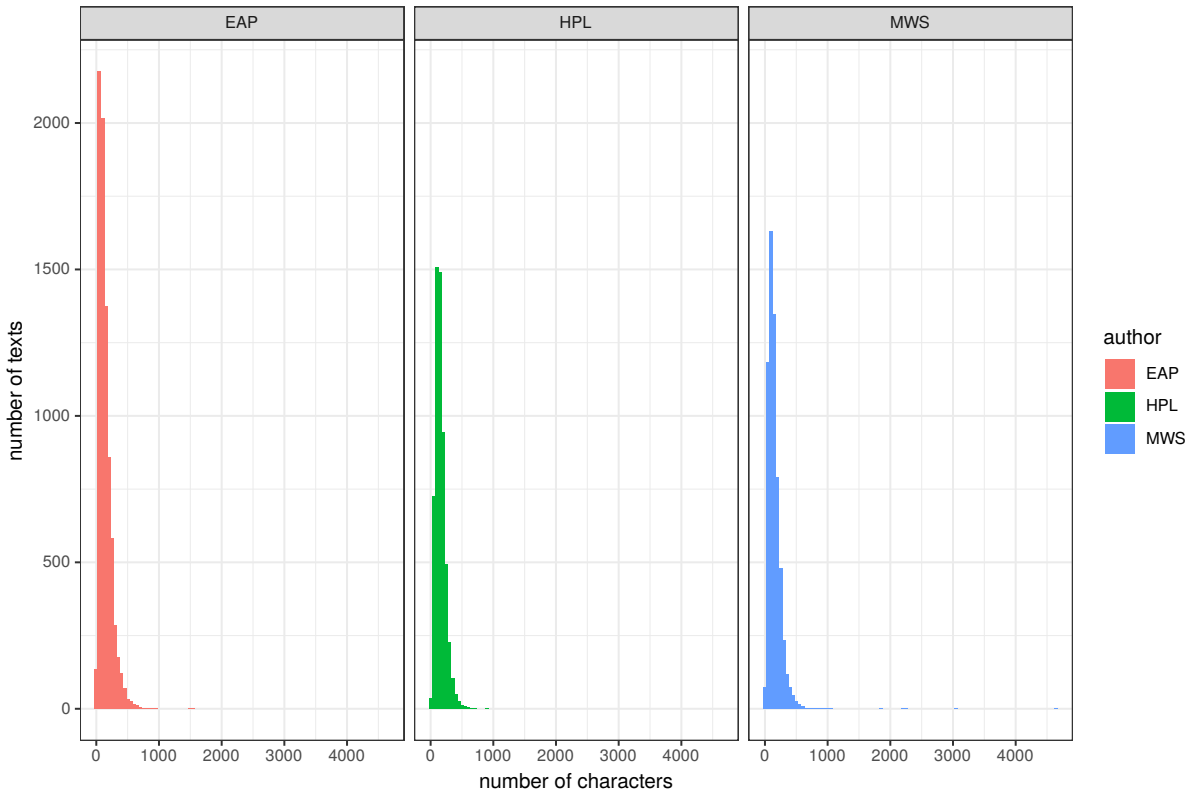
```
table(train$author) %>%
    prop.table()
```

```
##
##       EAP       HPL       MWS
## 0.4034935 0.2878084 0.3086981
```

```
train %>%
    group_by(author) %>%
    summarise(n = n()) %>%
    ggplot(data = ., aes(x = author, y = n, fill = author)) +
    geom_col(show.legend = F) +
    xlab(label = 'author') +
    ylab(label = 'number of texts') +
    theme_bw()
```




```
train %>%
    mutate(len = nchar(text)) %>%
    ggplot(data = ., aes(x = len, fill = author)) +
    geom_histogram(binwidth = 50) +
    facet_grid(. ~ author) +
    xlab(label = 'number of characters') +
    ylab(label = 'number of texts') +
    theme_bw()
```




Медиана выглядит предпочтительнее среднего, вследствие ряда длинных отрывков с пропущеннными точками.

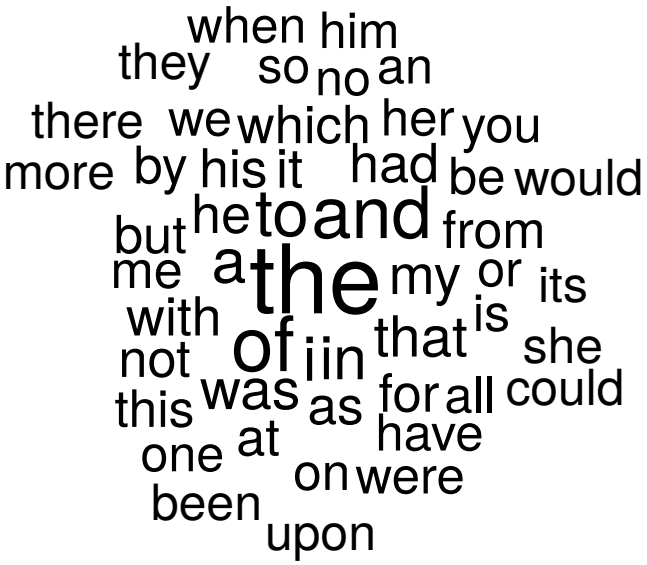
```
train %>%
    mutate(len = nchar(text)) %>%
    group_by(author) %>%
    summarise(`median text length` = median(len), `mean text length` = mean(len))
```

| author | median text length | mean text length |
|---|---|---|
| EAP | 115 | 142.2259 |
| HPL | 142 | 155.8435 |
| MWS | 130 | 151.6598 |

## 3.3 Облака слов

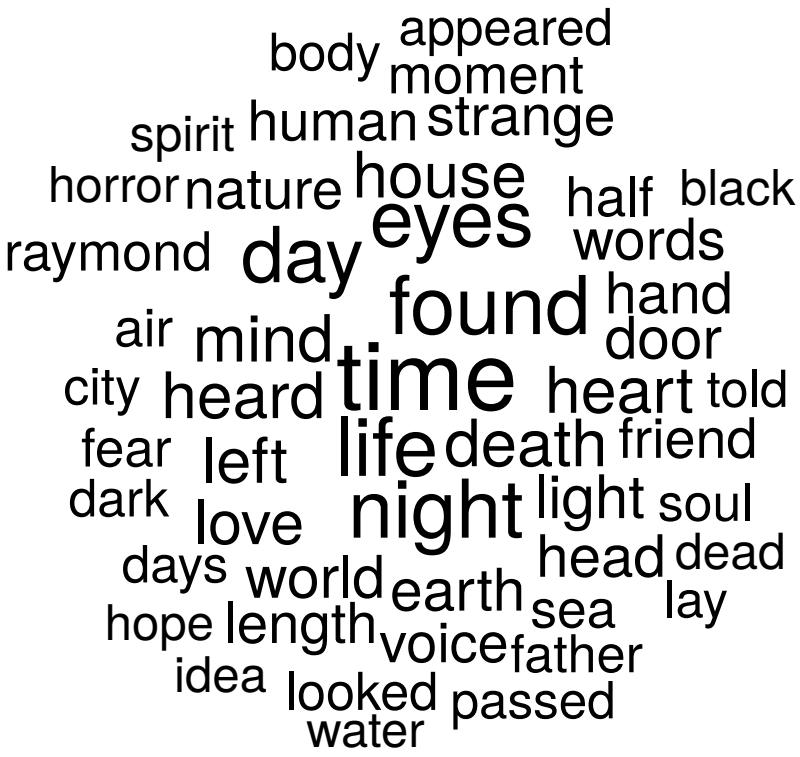
Облако слов, - удобное представление наиболее часто встречаюшихся слов в тексте. Размер отражает частоту встречаемости. Для начала применим облако к тексту в целом.

```
train %>%
    unnest_tokens(word, text) %>%
    count(word) %>%
    with(wordcloud(word, n, max.words = 50, scale = c(3, 1.5), random.order = FALSE, r
```




Ожидаемо, вспомогательные части речи, - союзы, артикли, местоимения etc в тексте встречаются чаще. Стоит ли от них избавляться при построении модели? Важность признаков будет рассмотрена позже, на этапе feature engineering. Ниже представлен вариант облака без учета вспомогательных слов.

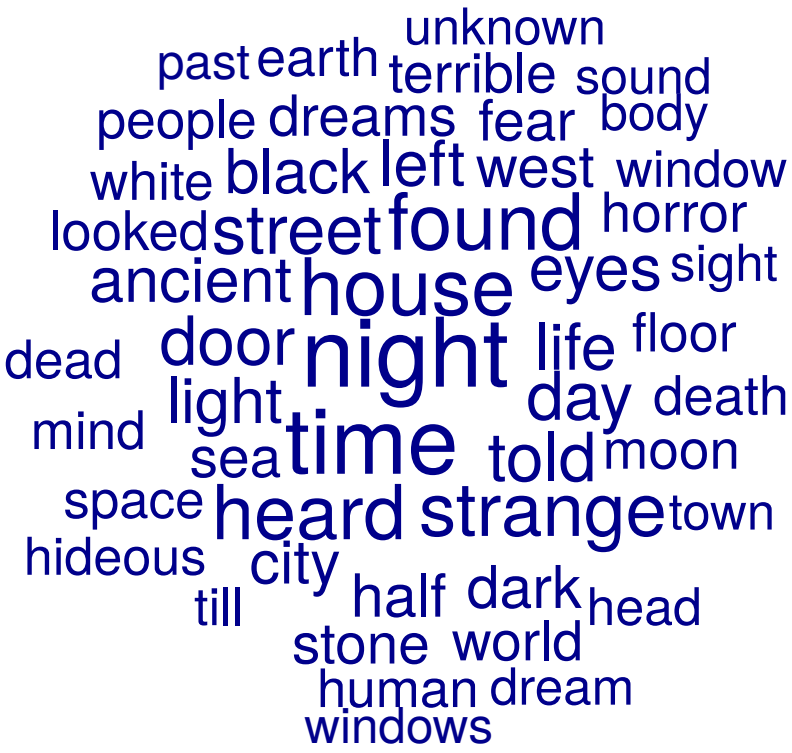
```
train_clean <- train %>%
    unnest_tokens(word, text) %>%
    anti_join(stop_words, by = 'word')

train_clean %>%
    count(word) %>%
    with(wordcloud(word, n, max.words = 50, scale = c(3, 1), random.order = FALSE, rot
```




Ключевые слова подтвеждают заявленную тематику текстов, - жизнь и смерть, время и страх. Прослеживаются детективные и даже романтические нотки. Попробуем разделить облака по авторам.

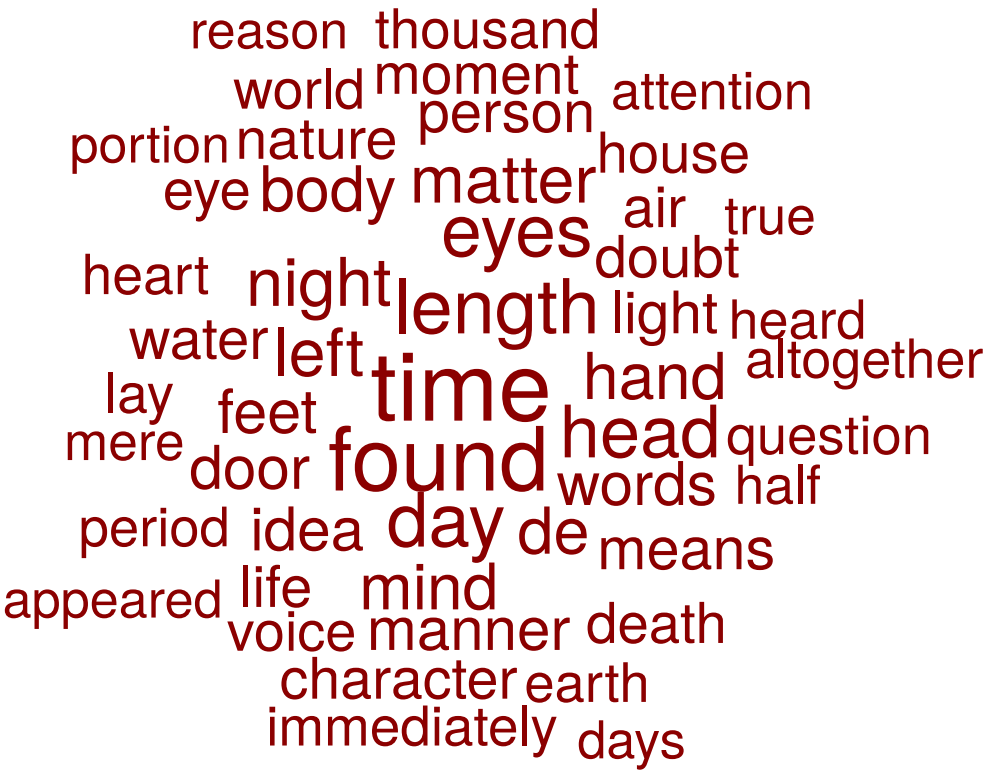
### 3.3.1 Лавкрафт

```
train_clean %>%
    filter(author == 'HPL') %>%
    count(word) %>%
    with(wordcloud(word, n, max.words = 50, scale = c(3, 1), random.order = FALSE, rot
```




#### 3.3.2 По

```
train_clean %>%
    filter(author == 'EAP') %>%
    count(word) %>%
    with(wordcloud(word, n, max.words = 50, scale = c(3, 1), random.order = FALSE, ro
```




#### 3.3.3 Шелли

```
train_clean %>%
    filter(author == 'MWS') %>%
    count(word) %>%
    with(wordcloud(word, n, max.words = 50, scale = c(3, 1), random.order = FALSE, ro
```


Есть как общие мотивы, например, тематика времени близка всем троим, так и индивидуальные особенности. Тексты Лавкрафта ближе всего к классическому пониманию ужасов. У По можно заметить склонность к детективному стилю. Шелли не чужда романтичность.

### 3.4 Ключевые слова

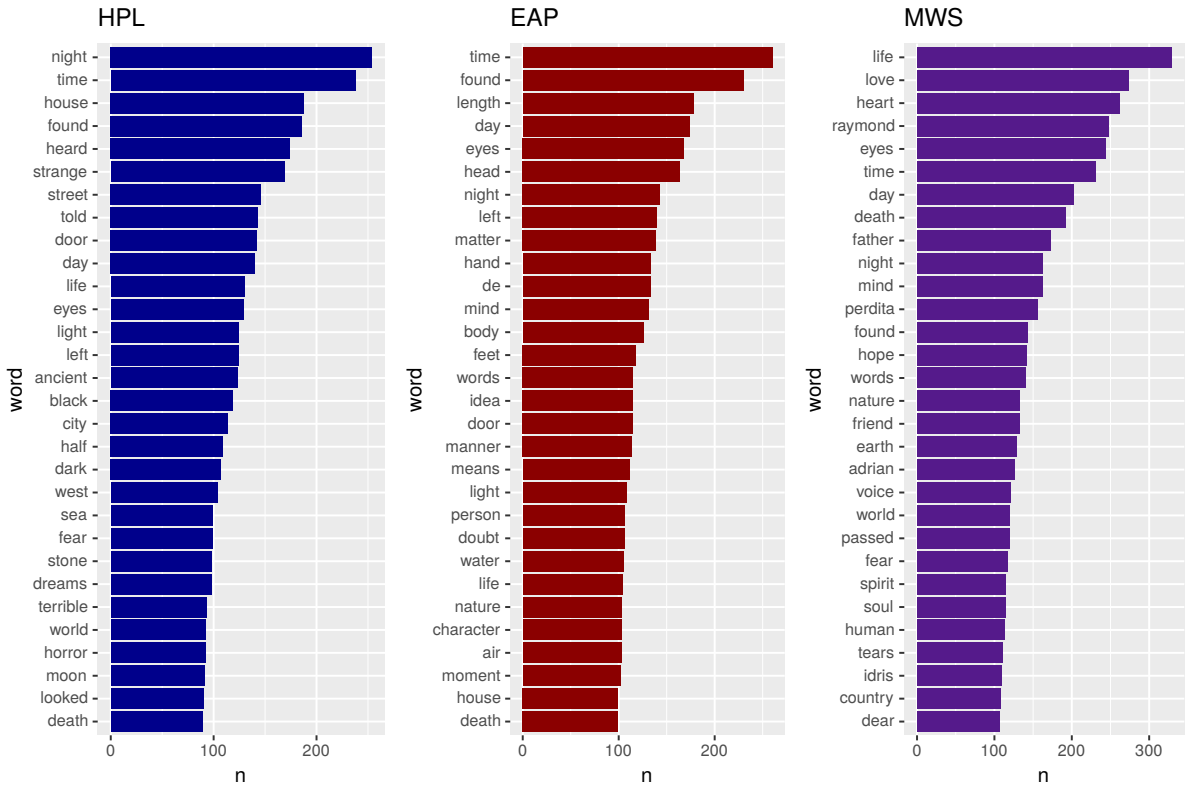
Более классическое представление о частоте встречаемости можно построить, используя столбчатые диаграммы. Ниже представлены две диаграммы, первая построена по словам, для второй предварительно использовался стемминг.

```
get_bar_plot <- function(author_id, clr, stem = F){

    if(stem){
        t <- train_clean %>%
            filter(author == author_id) %>%
            mutate(word = wordStem(word))
    }else{
        t <- train_clean %>%
            filter(author == author_id)
    }

    t %>%
```

```
        count(word) %>%
        top_n(30, n) %>%
        arrange(n) %>%
        mutate(word = factor(word, levels = word)) %>%
        ggplot() +
        geom_col(aes(word, n), fill = clr) +
        ggtitle(author_id) +
        coord_flip()
}
```

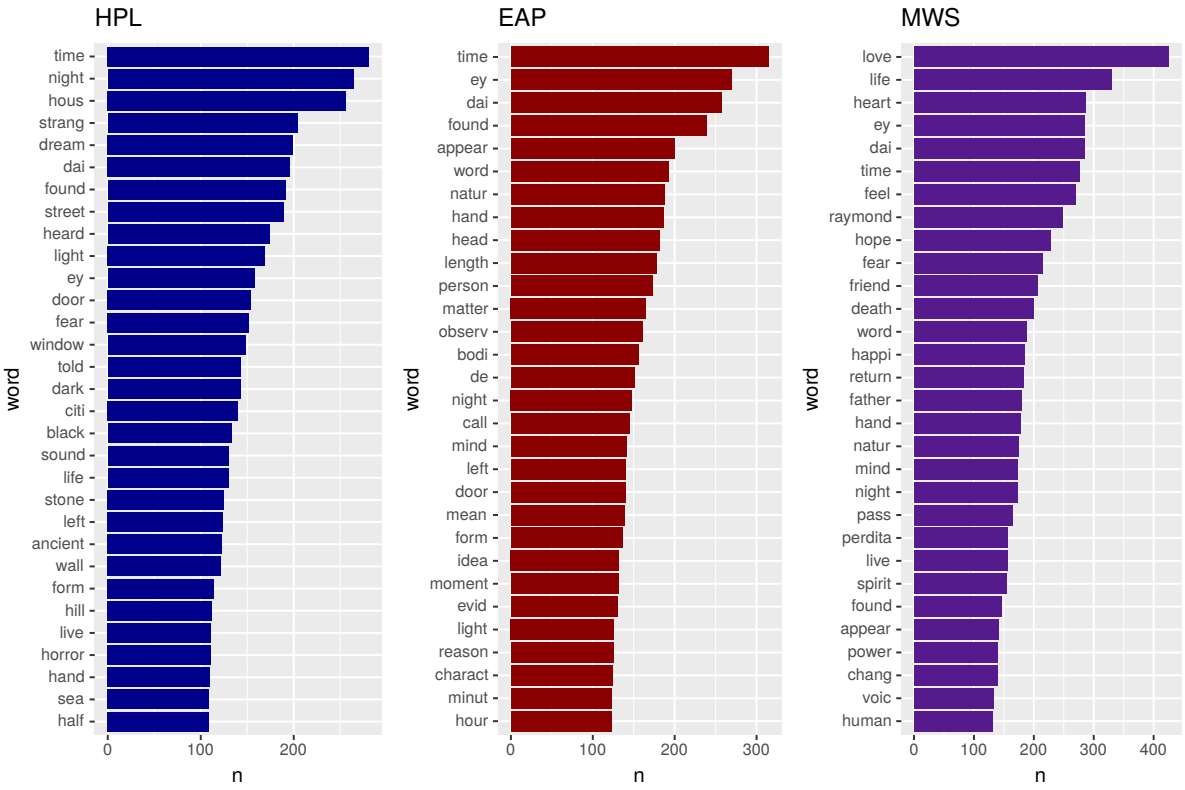
### 3.4.1 ключевые слова

```
pl1 <- get_bar_plot(author_id = 'HPL', clr = 'blue4')
pl2 <- get_bar_plot(author_id = 'EAP', clr = 'red4')
pl3 <- get_bar_plot(author_id = 'MWS', clr = 'purple4')

grid.arrange(pl1, pl2, pl3, nrow = 1)
```



### 3.4.2 ключевые слова + стемминг

```
pl1 <- get_bar_plot(author_id = 'HPL', clr = 'blue4', stem = T)
pl2 <- get_bar_plot(author_id = 'EAP', clr = 'red4', stem = T)
pl3 <- get_bar_plot(author_id = 'MWS', clr = 'purple4', stem = T)

grid.arrange(pl1, pl2, pl3, nrow = 1)
```




Суда по частоте встреч имен Raymond и Perdita у Шелли, можно предположить, что значительная часть ее отрывков взята из одного произведения. Поможет ли это в дальнейшей классификации?

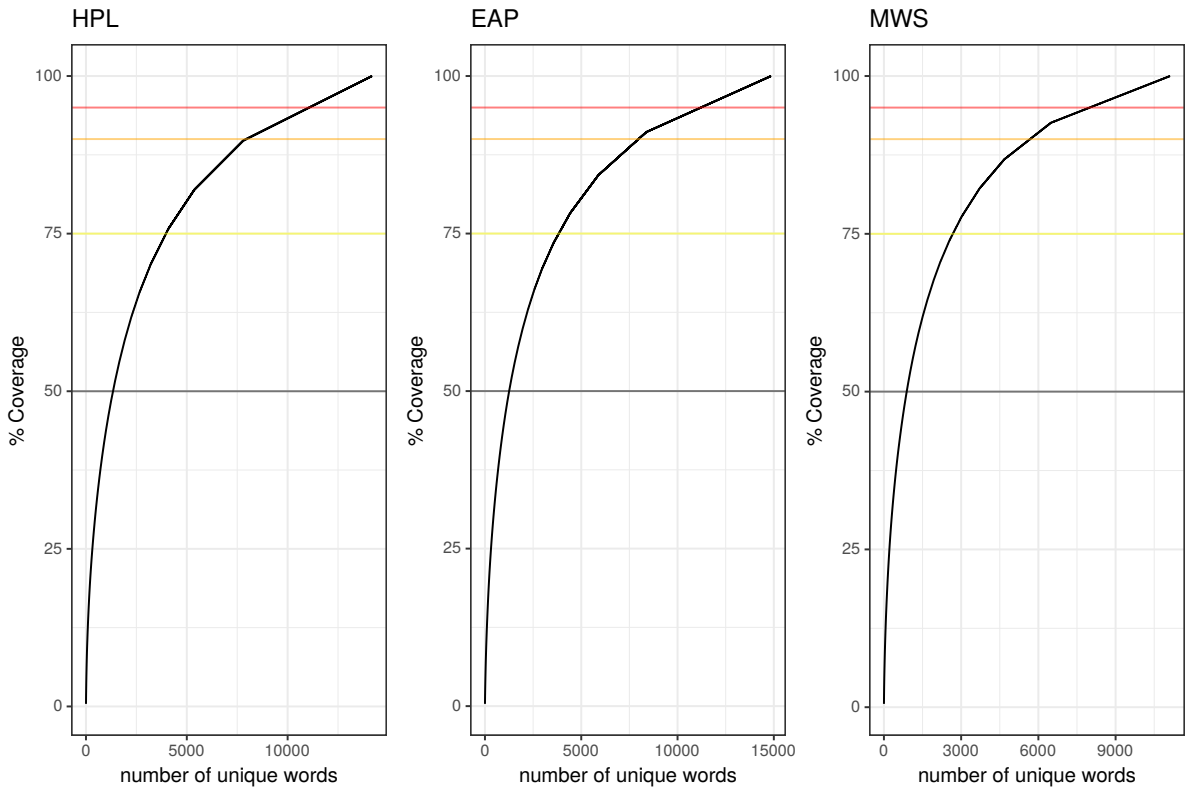
## 3.5 Уникальные слова

Лексикон Шелли, предоставленный в наборе данных заметно меньше, чем у остальных. Возможно это связано с установленным ранее фактом о больших заимствованиях из одного произведения.

```
train_clean %>%
    distinct(author, word) %>%
    count(author) %>%
    rename(`unique word count` = n)
```

| author | unique word count |
|---|---|
| EAP | 14856 |
| HPL | 14188 |
| MWS | 11115 |

```
get_uniq_words_plot <- function(author_id){

    t <- train_clean %>%
        filter(author == author_id) %>%
        count(word) %>%
        arrange(desc(n))

    t %>%
        mutate(cumsum = cumsum(n),
               cumsum_perc = round(100 * cumsum/sum(n), digits = 2)) %>%
        ggplot(aes(x = 1:nrow(t), y = cumsum_perc)) +
        geom_line() +
        geom_hline(yintercept = 50, color = 'black', alpha = 0.5) +
        geom_hline(yintercept = 75, color = 'yellow', alpha = 0.5) +
        geom_hline(yintercept = 90, color = 'orange', alpha = 0.5) +
        geom_hline(yintercept = 95, color = 'red', alpha = 0.5) +
        xlab('number of unique words') +
        ylab('% Coverage') +
        ggtitle(author_id) +
        theme_bw()

}

pl1 <- get_uniq_words_plot(author_id = 'HPL')
pl2 <- get_uniq_words_plot(author_id = 'EAP')
pl3 <- get_uniq_words_plot(author_id = 'MWS')

grid.arrange(pl1, pl2, pl3, nrow = 1)
```




Как видно из графиков, для По и Лавкрафта достаточно примерно 7500 слов, чтобы перекрыть 90% их отрывков. Для Шелли достаточно примерно 5500 слов.

## 3.6 Частота совстречаемости слов

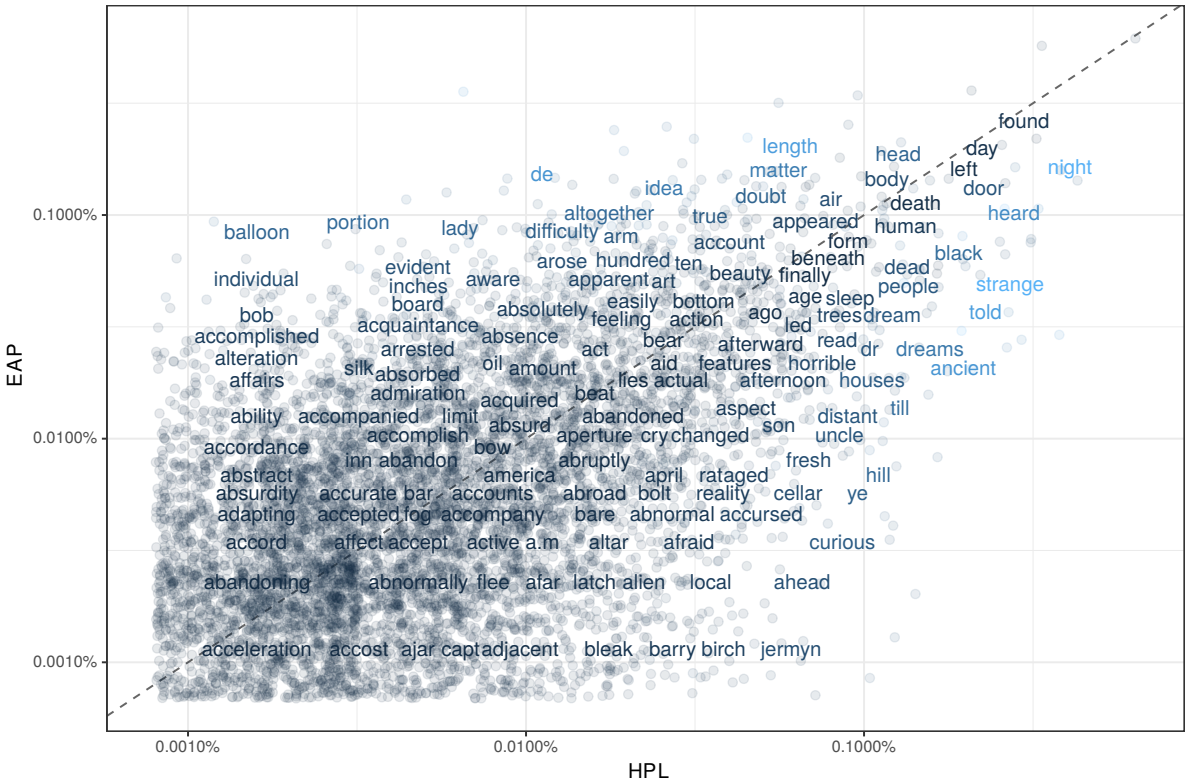
Слова, расположенные близко к пунктирной линии, встречаются с одинаковой частотой у авторов. Для данной задачи они не особенно интересны. Чем слово дальше отстоит от линии, тем больший перекос в его использовании у одного автора.

```
word_freq <- train_clean %>%
    count(author, word) %>%
    group_by(author)  %>%
    mutate(prop = n / sum(n)) %>%
    select(-n) %>%
    spread(author, prop)

get_word_freq_plot <- function(author_id1, author_id2){
    word_freq %>%
        filter(!is.na(!!author_id1) & !is.na(!!author_id2)) %>%
        mutate(clr = abs(!!author_id1 - !!author_id2)) %>%
        ggplot(aes(x = !!author_id1, y = !!author_id2, color = clr)) +
```

```
        geom_abline(color = "gray40", lty = 2) +
        geom_jitter(alpha = 0.1, size = 2, width = 0.3, height = 0.3) +
        geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
        scale_x_log10(labels = percent_format()) +
        scale_y_log10(labels = percent_format()) +
        theme_bw() +
        theme(legend.position = "none") +
        labs(x = author_id1, y = author_id2)
}
```

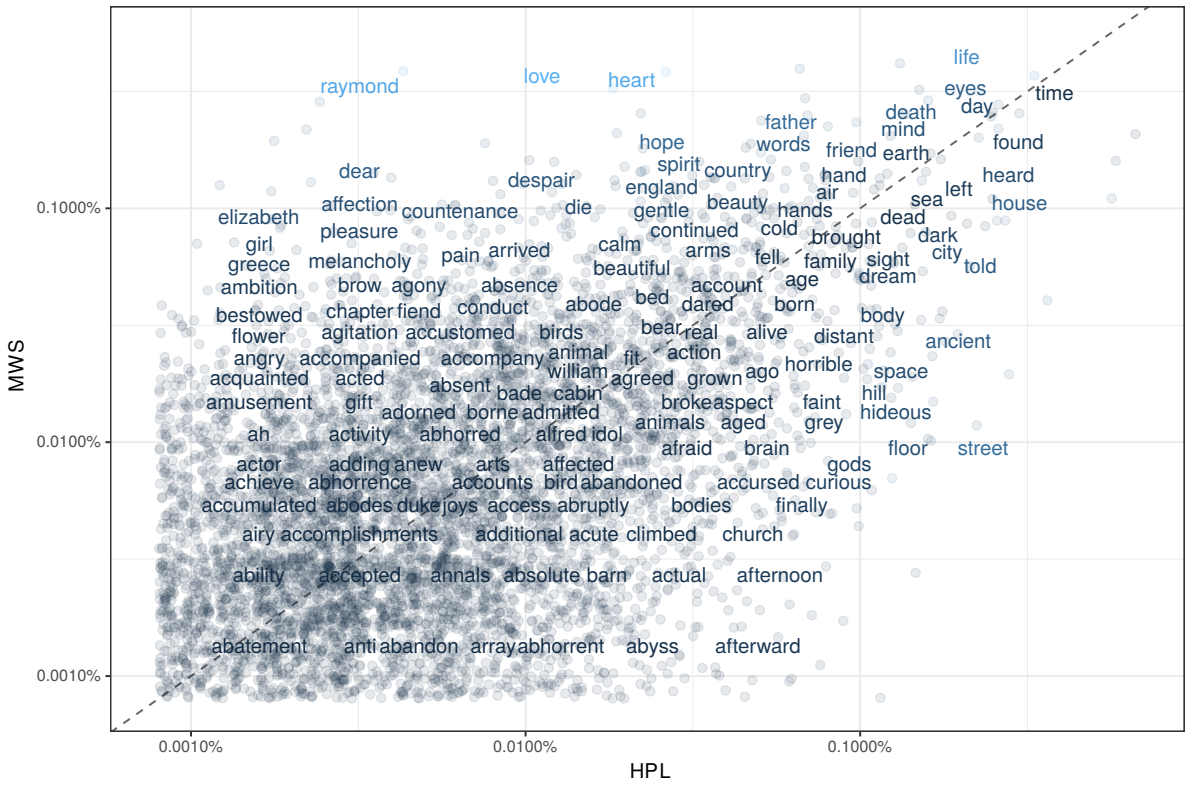
### 3.6.1 Лавкрафт и По

```
get_word_freq_plot(author_id1 = quo(HPL), author_id2 = quo(EAP))
```



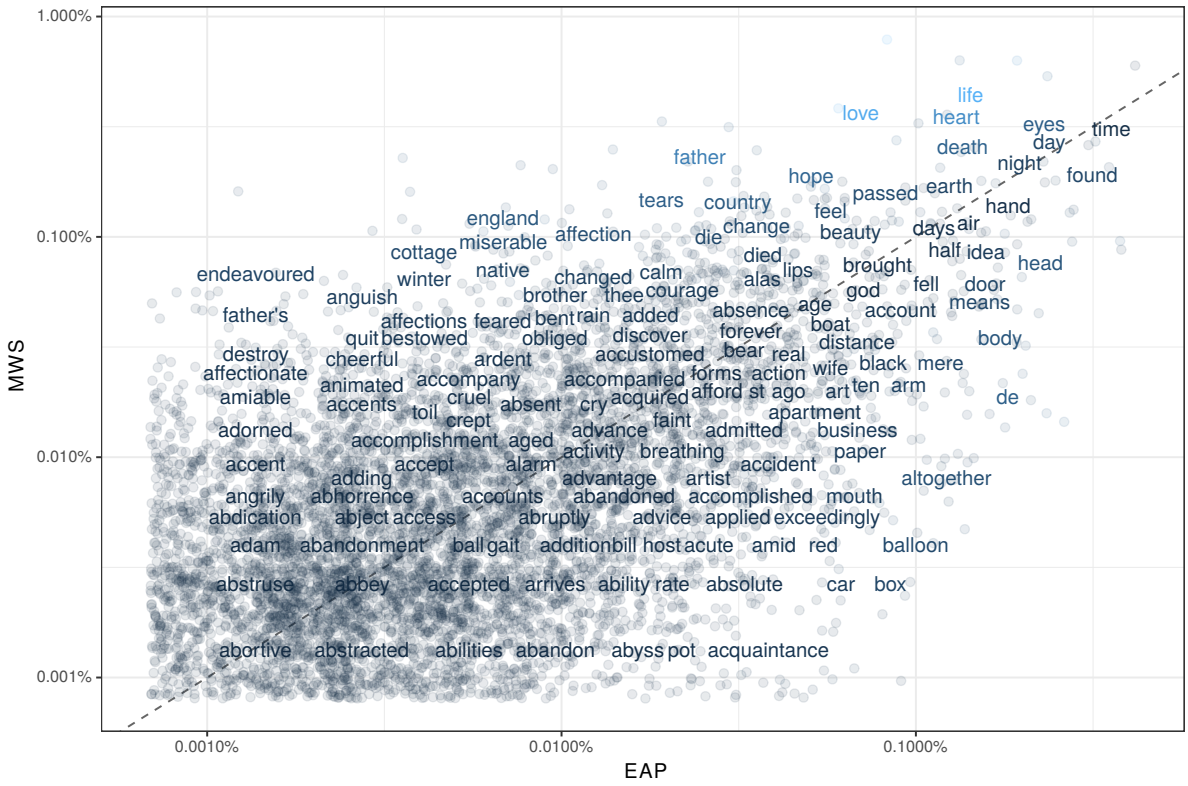

### 3.6.2 Лавкрафт и Шелли

```
get_word_freq_plot(author_id1 = quo(HPL), author_id2 = quo(MWS))
```




### 3.6.3 По и Шелли

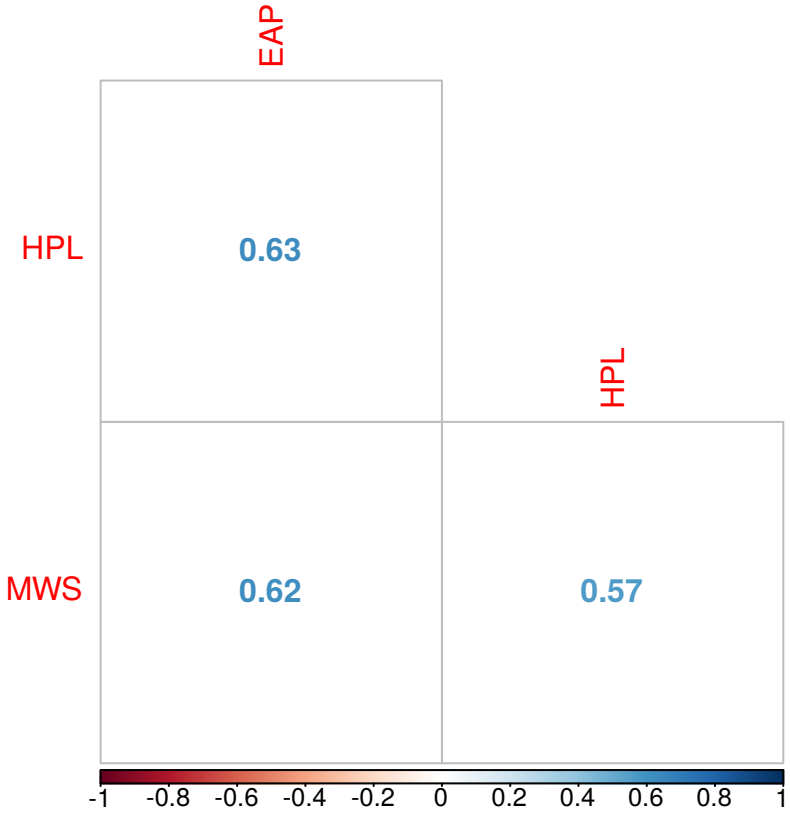
```
get_word_freq_plot(author_id1 = quo(EAP), author_id2 = quo(MWS))
```




## 3.7 Корреляция частот совстречаемости

Также стоит посмотреть на результаты корреляционного анализ частот совстречаемости. Еще раз можно убедиться, что у авторов много общего.

```
word_freq %>%
    select(-word) %>%
    cor(use = 'complete.obs', method = 'pearson') %>%
    corrplot(type = 'lower',
             method = 'number',
             diag = F)
```




### 3.8 TF-IDF

TF (term frequency — частота слова) — отношение числа вхождений некоторого слова к общему числу слов документа. Таким образом, оценивается важность слова $t_i$ в пределах отдельного документа.

$$tf(i,d) = \frac{n_t}{\sum_k n_k} \tag{1}$$

где $n_t$ есть число вхождений слова $t$ в документ, а в знаменателе — общее число слов в данном документе.

IDF (inverse document frequency — обратная частота документа) — инверсия частоты, с которой некоторое слово встречается в документах коллекции. Учёт IDF уменьшает вес широкоупотребительных слов. Для каждого уникального слова в пределах конкретной коллекции документов существует только одно значение IDF.

$$idf(t,D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|} \tag{2}$$

где

- $D$ — число документов в коллекции;
- $|\{d_i \in D \mid t \in d_i\}|$ — число документов из коллекции $D$, в которых встречается $t$ (когда $n_t \neq 0$).

Таким образом, мера TF-IDF является произведением двух сомножителей:

$$tf\text{-}idf(t, d, D) = tf(t, d) \times idf(t, D) \tag{3}$$

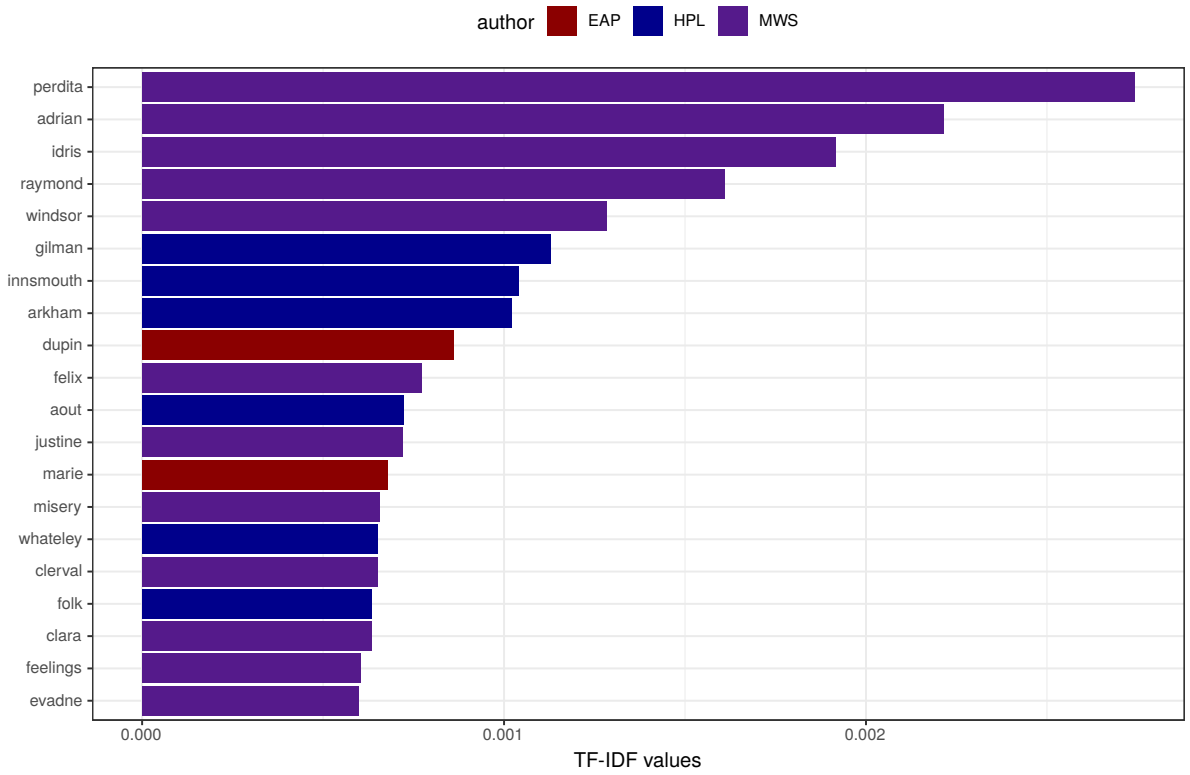
Большой вес в TF-IDF получат слова с высокой частотой в пределах конкретного документа и с низкой частотой употреблений в других документах. Наиболее высокие значения имеют имена собственные. Можно сделать вывод, что Шелли более склонна к их использованию в тексте. Также можно предположить, что эта мера окажется значимой при классификации.

```
tf_idf <- train_clean %>%
    count(author, word) %>%
    bind_tf_idf(word, author, n)

get_tf_idf_plot <- function(df, cnt){
    df %>%
        arrange(desc(tf_idf)) %>%
        mutate(word = factor(word, levels = rev(unique(word)))) %>%
        top_n(cnt, tf_idf) %>%
        ggplot(aes(word, tf_idf, fill = author)) +
        scale_fill_manual(values = c(HPL = 'blue4', EAP = 'red4', MWS = 'purple4')) +
        geom_col() +
        labs(x = NULL, y = 'TF-IDF values') +
        theme_bw() +
        theme(legend.position = 'top') +
        coord_flip()
}

get_tf_idf_facet_plot <- function(df, cnt){
    df %>%
        arrange(desc(tf_idf)) %>%
        mutate(word = factor(word, levels = rev(unique(word)))) %>%
        group_by(author) %>%
        top_n(cnt, tf_idf) %>%
        ungroup() %>%
        ggplot(aes(word, tf_idf, fill = author)) +
        scale_fill_manual(values = c(HPL = 'blue4', EAP = 'red4', MWS = 'purple4')) +
```

```
        geom_col() +
        labs(x = NULL, y = 'TF-IDF values') +
        theme_bw() +
        theme(legend.position = 'none') +
        facet_wrap(~author, ncol = 3, scales = 'free_y') +
        coord_flip()
}
```

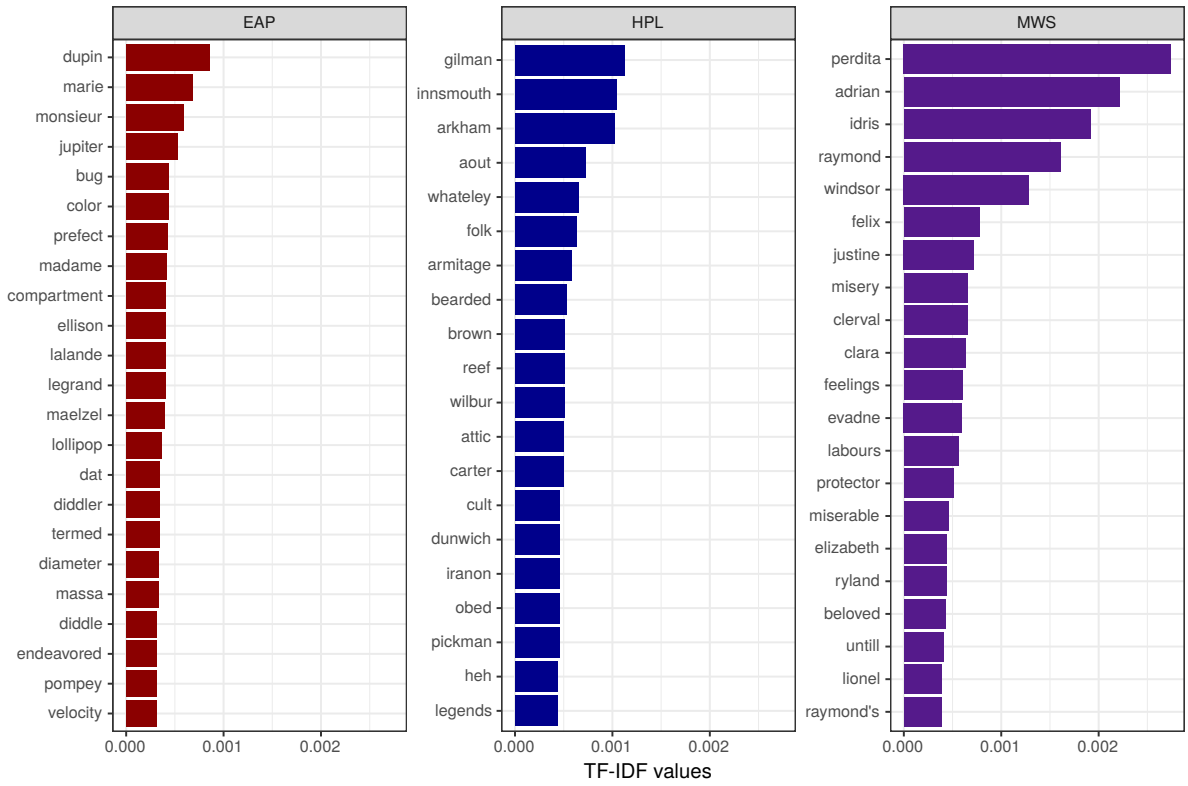
### 3.8.1 общие униграммы

```
get_tf_idf_plot(tf_idf, 20)
```




### 3.8.2 униграммы по авторам

```
get_tf_idf_facet_plot(tf_idf, 20)
```



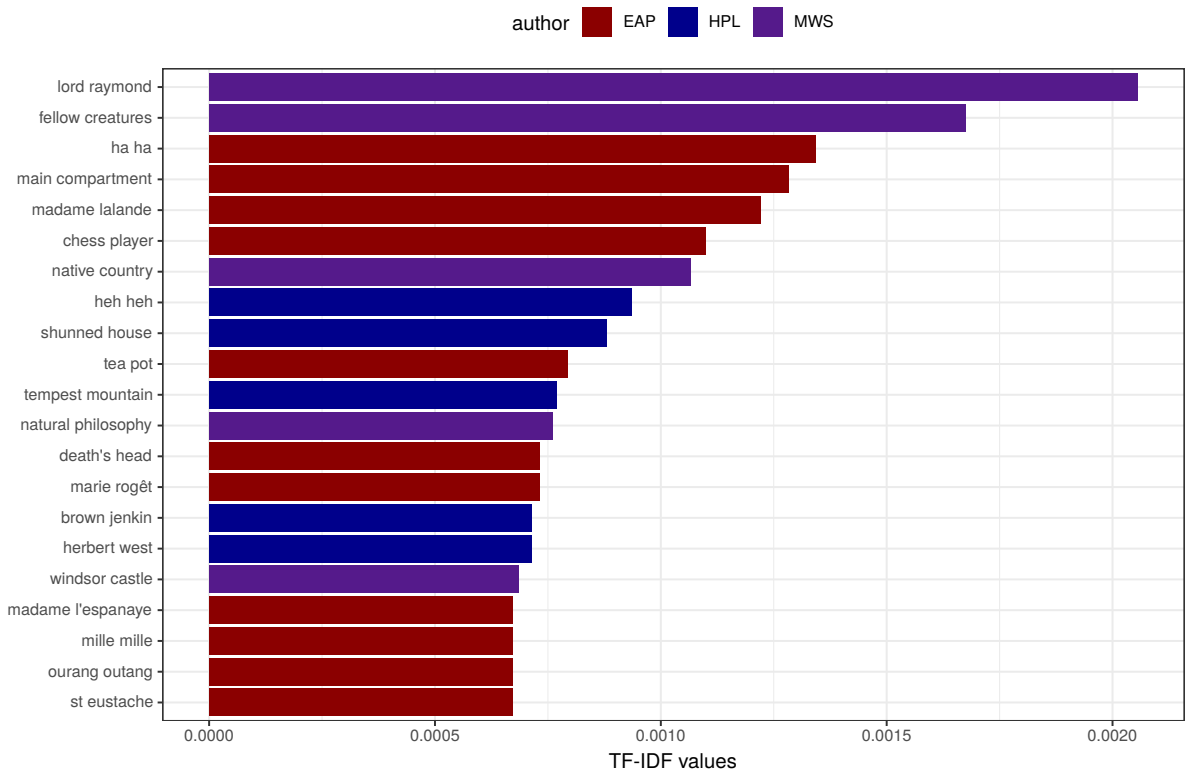

## 3.9 Биграммы

Помимо одиночных слов важное значение могут иметь словосочетания, в частности би- и триграммы. Помимо имен собственных здесь начинают появляться характерные сочетания. Как ни странно, и По, и Лавкрафт, оказывается, любили посмеяться. Шелли неравнодушна к сочетаниям, начинающимся на *dear*. По вспоминает каких-то многочисленных *madame*, а Лавкрафт нагоняет жути своими *lurking fear* и *ancient house*

```
train_clean_bigram <- train %>%
    unnest_tokens(word, text, token = 'ngrams', n = 2) %>%
    separate(word, c('word1', 'word2'), sep = ' ') %>%
    anti_join(stop_words, by = c('word1' = 'word')) %>%
    anti_join(stop_words, by = c('word2' = 'word')) %>%
    unite(word, c('word1', 'word2'), sep = ' ')

tf_idf_bigram <- train_clean_bigram %>%
    count(author, word) %>%
    bind_tf_idf(word, author, n)
```

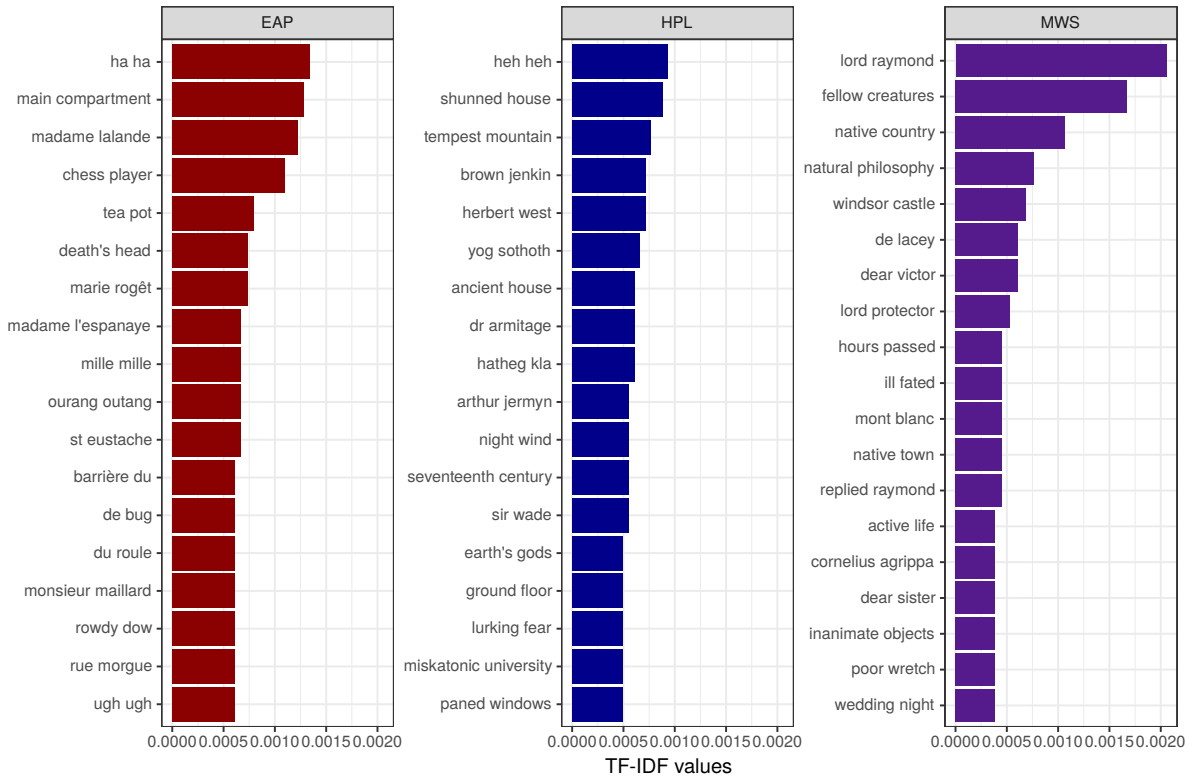
### 3.9.1 общие биграммы

```
get_tf_idf_plot(tf_idf_bigram, 20)
```




### 3.9.2 биграммы по авторам

```
get_tf_idf_facet_plot(tf_idf_bigram, 15)
```



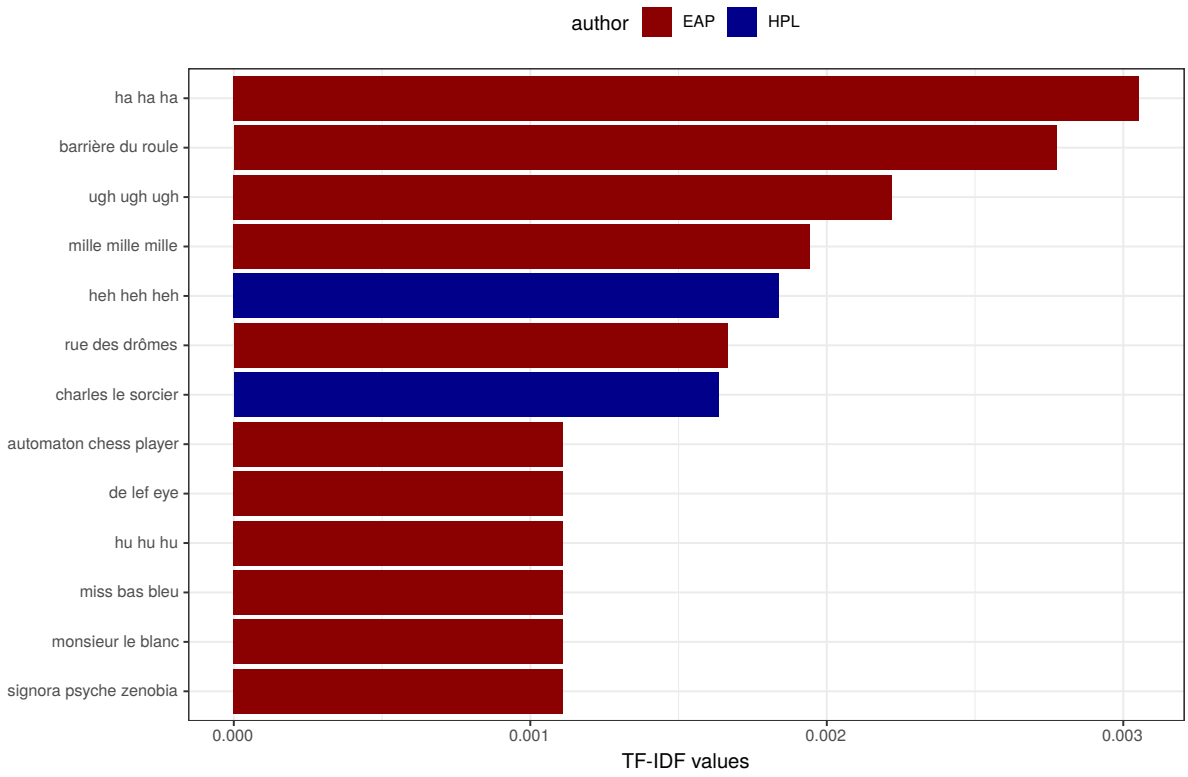

## 3.10 Триграммы

По и Лавкрафт продолжают смеяться, причем По умудряется это делать по-разному, в свободное время подкидывая иностранных словечек. Лавкрафт на все лады вспоминает безумного араба аль Хазреда, а любимый Шелли Реймонд, оказывается, связан со вселенной *доктора Кто*. Интересные открытия.

```
train_clean_trigram <- train %>%
    unnest_tokens(word, text, token = 'ngrams', n = 3) %>%
    separate(word, c('word1', 'word2', 'word3'), sep = ' ') %>%
    anti_join(stop_words, by = c('word1' = 'word')) %>%
    anti_join(stop_words, by = c('word2' = 'word')) %>%
    anti_join(stop_words, by = c('word3' = 'word')) %>%
    unite(word, c('word1', 'word2', 'word3'), sep = ' ')

tf_idf_trigram <- train_clean_trigram %>%
    count(author, word) %>%
    bind_tf_idf(word, author, n)
```

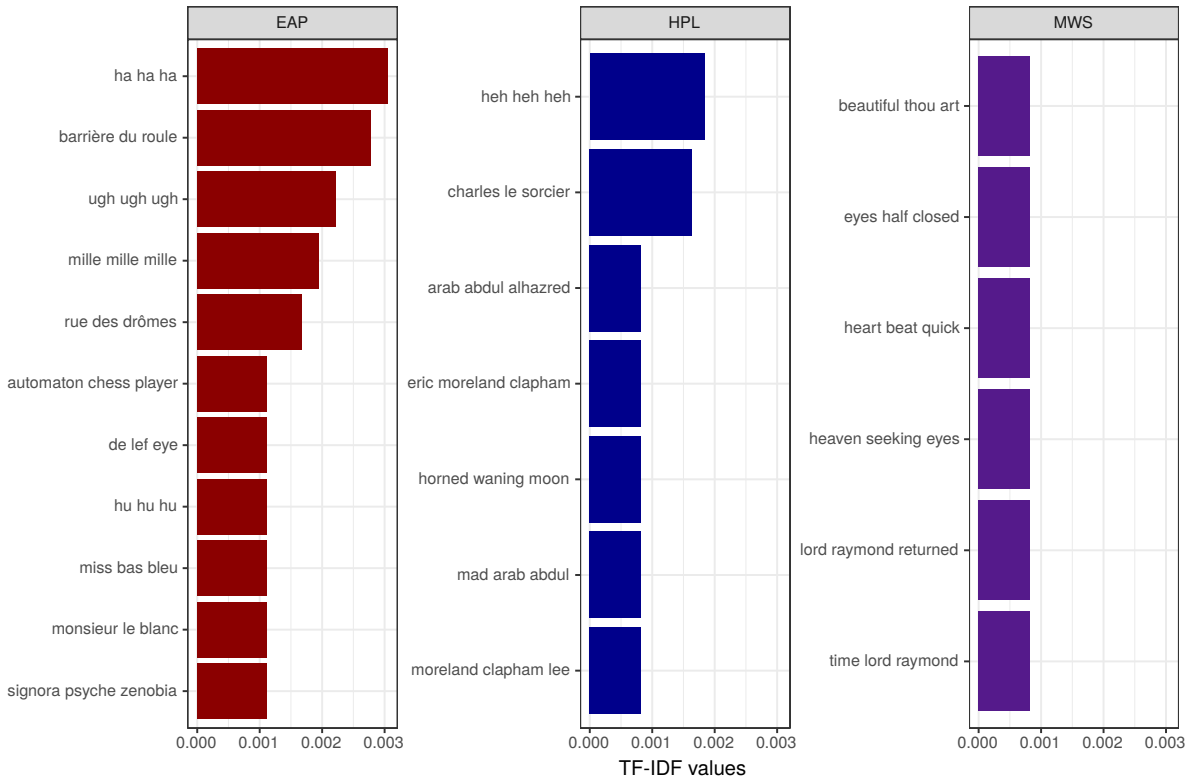
### 3.10.1 общие триграммы

```
get_tf_idf_plot(tf_idf_trigram, 10)
```




### 3.10.2 триграммы по авторам

```
get_tf_idf_facet_plot(tf_idf_trigram, 6)
```




## 3.11 Анализ тональности