In [1]:
```
import numpy as np
import pandas as pd
import nltk
nltk.download('punkt')
nltk.download("stopwords")
import re
import ru_core_news_md ### чтобы установить нужна команда python -m spacy download ru_core_news_md
from nltk.stem.snowball import SnowballStemmer
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\gleko\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\gleko\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Считаем данные для парсинга

In [2]:
```
df = pd.read_csv('test_data.csv')
```

Возьмём стоп-слова из библиотеки NLTK для предобработки текста для использования в моделях

In [3]:
```
nlp = ru_core_news_md.load()
nltk.download("stopwords")
from nltk.corpus import stopwords
russian_stopwords = stopwords.words("russian")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\gleko\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Создадим функции токенизации, лемматизации и очистки текста от лишних слов и знаков

In [4]:
```
def tokenize(text_to_tokenize):
    return nlp(text_to_tokenize)
```

In [5]:
```
def lemmatize(tokens):
    return [token.lemma_ for token in tokens]
```

Для очистки текст будет переведён в нижний регистр, удалены специальные символы, многократные пробелы, стопслова.

In [6]:
```
def text_cleaner(text_to_clean):
    text_to_clean = text_to_clean.lower()
    text_to_clean = re.sub('[!@#$1234567890.,+-:;\/<>*&?%№`~|]', '', text_to_clean)
    text_to_clean = re.sub('\s+',' ',text_to_clean)

    russian_stopwords = stopwords.words("russian")
    words = text_to_clean.split()

    words = [word for word in words if word not in russian_stopwords]
    text_to_clean = ' '.join(words)
    return text_to_clean
```

Создадим функции преобразующие список слов в bag of words и текст в bag of words.

In [7]:
```
def bag_of_words(list_to_bow, all_words_greetings):
    bow = [0]*len(all_words_greetings)

    for word in list_to_bow:
        if word in all_words_greetings:
            index_of_word = all_words_greetings.index(word)
            bow[index_of_word] = 1
    return bow
```

In [8]:
```
def text_to_bow(text_to_bow, all_words_greetings):
    text_to_bow = text_cleaner(text_to_bow)
    text_to_bow_tokens = tokenize(text_to_bow)
    text_to_bow_lemmas = lemmatize(text_to_bow_tokens)
    text_to_bow_lemmas = list(set([stemmer.stem(word) for word in text_to_bow_lemmas]))

    bow = bag_of_words(text_to_bow_lemmas, all_words_greetings)
    return bow
```

Составим вручную примеры прощаний, приветствий и реплик для создания обучающей выборки для решения задачи классификации приветствий и прощаний.

In [9]:
```
greetings_examples = ['добрый день', 'добрый вечер', 'доброе утро',
```

```
                      'приветствую', 'здравствуйте', 'доброго времени суток',
                      'привет']

### тексты с примерами ниже написаны руками. не из test-data.csv :)
# not_greetings_examples - примеры обычных реплик, которые не относятся ни к классу приветсвий, ни к классу про
not_greetings_examples = ['если уделите минуту я вам расскажу подробнее',
                          'вас интересует данное предложение?', 'зафиксировали информацию',
                          'мы учтем ваши пожелания', 'в какое время вам удобнее, чтобы мы перезвонили?',
                          'вам удобно разговаривать?', 'Отправили вам информацию', 'Да, всё верно', 'Хотим полу

byes_examples = ['До свидания', 'Всего доброго', 'Всего наилучшего', 'Хорошего вечера', "До связи"]
```

Создадим словарь всех слов из обучающей выборки и приведём их к виду списка стемматизированных лемм

In [10]:
```
stemmer = SnowballStemmer("russian")
```

In [11]:
```
def replicas_list_to_unique_lemmas_list(replicas):
    clean_text = text_cleaner(' '.join(replicas))
    lemmas = lemmatize(tokenize(clean_text))
    unique_lemmas = list(set([stemmer.stem(word) for word in lemmas]))

    return unique_lemmas
```

In [12]:
```
all_words_greetings = replicas_list_to_unique_lemmas_list(greetings_examples + not_greetings_examples)
all_words_byes = replicas_list_to_unique_lemmas_list(byes_examples + not_greetings_examples)
```

Получим реплики менеджеров

In [13]:
```
managers_replicas = df[df['role'] == 'manager'].reset_index(drop = True)
```

Создадим обучающую выборку для классификации приветствий

In [14]:
```
classes_greet_train = np.concatenate((np.ones(len(greetings_examples)), np.zeros(len(not_greetings_examples))),
```

Получим bag of words для обучающей выборки

In [15]:
```
train_greet_bows = []
for replica in greetings_examples + not_greetings_examples:
    replica_bow = text_to_bow(replica, all_words_greetings)
    train_greet_bows.append(replica_bow)
```

Создадим DataFrame для удобства при обращении к тестовой выборке

In [16]:
```
df_train_greet = pd.DataFrame(data = np.array(train_greet_bows))
```

In [17]:
```
df_train_greet['class'] = classes_greet_train
df_train_greet['text'] = greetings_examples + not_greetings_examples
```

Перемешаем строки в DataFrame, чтобы классы в обучающей выборке шли вразнобой

In [18]:
```
df_train_greet = df_train_greet.sample(frac=1).reset_index(drop=True)
```

Получим таблицу признаков и столбец целевой переменной (класса приветсвие - класс "1", не приветсвие - класс "0").

In [19]:
```
X_train_greetings = df_train_greet.drop(['class', 'text'], axis = 1)
```

In [20]:
```
y_train_greetings = df_train_greet['class']
```

Создадим обучающую выборку для прощаний. Порядок действий аналогичный.

In [21]:
```
classes_byes_train = np.concatenate((np.ones(len(byes_examples)), np.zeros(len(not_greetings_examples))), axis=
```

In [22]:
```
train_byes_bows = []
for replica in byes_examples + not_greetings_examples:
    replica_bow = text_to_bow(replica, all_words_byes)
    train_byes_bows.append(replica_bow)
```

In [23]:
```
df_train_byes = pd.DataFrame(data = np.array(train_byes_bows))
```

In [24]:
```
df_train_byes['class'] = classes_byes_train
df_train_byes['text'] = byes_examples + not_greetings_examples
```

In [25]:
```
df_train_byes = df_train_byes.sample(frac=1).reset_index(drop=True)
```

In [26]:
```
X_train_byes = df_train_byes.drop(['class', 'text'], axis = 1)
```

```
y_train_byes = df_train_byes['class']
```

Импортируем модель классификатора логистической регрессии со стандартными параметрами с L-2 регуляризацией и коэффициентом регуляризации C = 1

In [27]:
```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
```

Возьмём начальные и конечные реплики в качестве тестовой выборки для получения реплик приветствия и прощания

In [28]:
```
start_replicas_df = pd.DataFrame(columns = managers_replicas.columns)
for i in managers_replicas['dlg_id'].unique():
    start_replicas_df = pd.concat([start_replicas_df, managers_replicas.groupby('dlg_id', as_index=False).get_g
```

In [29]:
```
start_replicas_df.reset_index(drop = True, inplace = True)
```

In [30]:
```
end_replicas_df = pd.DataFrame(columns = managers_replicas.columns)
for i in managers_replicas['dlg_id'].unique():
    end_replicas_df = pd.concat([end_replicas_df, managers_replicas.groupby('dlg_id', as_index=False).get_group
```

In [31]:
```
end_replicas_df.reset_index(drop = True, inplace = True)
```

Создадим список индексов реплик приветствий и прощания, отобранные вручную, для построений метрик качества классификаторов

In [32]:
```
ind_of_greet_true = [0, 3, 6, 9, 10]
ind_of_byes_true = [2, 4, 5, 11, 14, 17]
```

Создадим тестовую выборку для прощаний и приветствий в виде таблицы признаков, где каждый столбец - это компонент в bag of words по словарю из соответствующих примеров. Также сохраним номер диалога

In [33]:
```
test_bows_greetings = []
test_dlg_ids_g = []
def test_greetings(row):
    test_bows_greetings.append(text_to_bow(row['text'], all_words_greetings))
    test_dlg_ids_g.append(row['dlg_id'])
start_replicas_df.apply(lambda row: test_greetings(row), axis=1)

test_bows_byes = []
test_dlg_ids_b = []
def test_byes(row):
    test_bows_byes.append(text_to_bow(row['text'], all_words_byes))
    test_dlg_ids_b.append(row['dlg_id'])
end_replicas_df.apply(lambda row: test_byes(row), axis=1);
```

Обучим классификатор для приветствий, сделаем прогноз для тестовой выборки и выведем прогнозируемые классификатором реплики приветствия

In [34]:
```
clf.fit(X_train_greetings, y_train_greetings)
```

Out[34]:
```
LogisticRegression
LogisticRegression()
```

In [35]:
```
greetings_predict = clf.predict(test_bows_greetings)
first_class_idxs_greetings = np.argwhere(greetings_predict==1)
```

Обучим классификатор для прощаний, сделаем прогноз для тестовой выборки и выведем прогнозируемые классификатором реплики прощаний

In [36]:
```
clf.fit(X_train_byes, y_train_byes)
```

Out[36]:
```
LogisticRegression
LogisticRegression()
```

In [37]:
```
byes_predict = clf.predict(test_bows_byes)
first_class_idxs_byes = np.argwhere(byes_predict==1)
```

Создадим DataFrame с результатами, в котором будет полное признаковое описание, текст реплики, номер диалога, прогнозируемый класс и реальный класс, размеченный вручную, для вывода реплик приветствия и прощания и расчёта метрик качества классификации.

In [38]:
```
true_classes_g = [0]*len(test_bows_greetings)
```

```
for i in ind_of_greet_true:
    true_classes_g[i] = 1

true_classes_b = [0]*len(test_bows_byes)
for i in ind_of_byes_true:
    true_classes_b[i] = 1
```

In [39]:
```
test_predict_greetings = pd.DataFrame(data = test_bows_greetings)
test_predict_greetings['predicted_class'] = greetings_predict
test_predict_greetings['true_class'] = true_classes_g
test_predict_greetings['dlg_id'] = test_dlg_ids_g
test_predict_greetings['text'] = start_replicas_df['text']
```

Создадим DataFrame для хранения результатов парсинга

In [40]:
```
columns = ['dlg_id', 'greeting', 'bye', 'greeting_and_bye_condition', 'introduce_replica' ,'name', 'company']

df_parsing_result = pd.DataFrame(columns = columns)
df_parsing_result['dlg_id'] = df['dlg_id'].unique()
df_parsing_result = df_parsing_result.set_index('dlg_id')

greetings_dict = {}
byes_dict = {}
names_dict = {}
companies_dict = {}
introduce_dict = {}
```

Создадим удобную функцию для пополнение словарей, из которых составим итоговую таблицу результатов парсинга диалогов

In [41]:
```
def update_dict(name, key, value):
    if name == "greetings":
        if key not in greetings_dict.keys():
            greetings_dict[key] = [value]
        else:
            greetings_dict[key].append(value)
    if name == "byes":
        if key not in byes_dict.keys():
            byes_dict[key] = [value]
        else:
            byes_dict[key].append(value)
    if name == "names":
        if key not in names_dict.keys():
            names_dict[key] = [value]
        else:
            names_dict[key].append(value)
    if name == "companies":
        if key not in companies_dict.keys():
            companies_dict[key] = [value]
        else:
            companies_dict[key].append(value)
    if name == "introduce":
        if key not in introduce_dict.keys():
            introduce_dict[key] = [value]
        else:
            introduce_dict[key].append(value)
```

Задание а. Извлекать реплики с приветствием – где менеджер поздоровался.

In [42]:
```
print('Задание а. Извлекать реплики с приветствием – где менеджер поздоровался.\n')
first_class_predicted_greetings_df = test_predict_greetings[test_predict_greetings['predicted_class'] == 1]
for index, row in first_class_predicted_greetings_df.iterrows():
    print(f'В диалоге {row.dlg_id} менеджер поздоровался.' + "\n" + f'Реплика "{row.text}"\n')
    update_dict("greetings", row.dlg_id, row.text)
```

```
Задание a. Извлекать реплики с приветствием – где менеджер поздоровался.

В диалоге 0 менеджер поздоровался.
Реплика "Алло здравствуйте"

В диалоге 1 менеджер поздоровался.
Реплика "Алло здравствуйте"

В диалоге 2 менеджер поздоровался.
Реплика "Алло здравствуйте"

В диалоге 3 менеджер поздоровался.
Реплика "Алло дмитрий добрый день"

В диалоге 3 менеджер поздоровался.
Реплика "Добрый меня максим зовут компания китобизнес удобно говорить"
```

In [43]:
```
test_predict_byes = pd.DataFrame(data = test_bows_byes)
test_predict_byes['predicted_class'] = byes_predict
test_predict_byes['true_class'] = true_classes_b
test_predict_byes['dlg_id'] = test_dlg_ids_b
test_predict_byes['text'] = end_replicas_df['text']
```

Задание e. Извлекать реплики, где менеджер попрощался.

In [44]:
```
print('Задание e. Извлекать реплики, где менеджер попрощался.\n')
first_class_predicted_byes_df = test_predict_byes[test_predict_byes['predicted_class'] == 1]
for index, row in first_class_predicted_byes_df.iterrows():
    print(f'В диалоге {row.dlg_id} менеджер попрощался.' + "\n" + f'Реплика "{row.text}"\n')
    update_dict("byes", row.dlg_id, row.text)
```

```
Задание e. Извлекать реплики, где менеджер попрощался.

В диалоге 0 менеджер попрощался.
Реплика "Всего хорошего до свидания"

В диалоге 1 менеджер попрощался.
Реплика "Угу да вижу я эту почту хорошо тогда исправлю на эту будем ждать ответа всего хорошего"

В диалоге 1 менеджер попрощался.
Реплика "До свидания"

В диалоге 3 менеджер попрощался.
Реплика "Угу все хорошо да понедельника тогда всего доброго"

В диалоге 4 менеджер попрощался.
Реплика "Во вторник все ну с вами да тогда до вторника до свидания"

В диалоге 5 менеджер попрощался.
Реплика "Ну до свидания хорошего вечера"
```

In [45]:
```
greetings_dlg_ids = test_predict_greetings[test_predict_greetings['predicted_class']==1]['dlg_id'].unique()
```

In [46]:
```
byes_dlg_ids = test_predict_byes[test_predict_byes['predicted_class']==1]['dlg_id'].unique()
```

In [47]:
```
greeting_and_bye_ids = [i for i in greetings_dlg_ids if i in byes_dlg_ids]
only_greeting_ids = [i for i in greetings_dlg_ids if i not in byes_dlg_ids]
only_byes_ids = [i for i in byes_dlg_ids if i not in greetings_dlg_ids]
```

Задание f. Проверять требование к менеджеру: «В каждом диалоге обязательно необходимо поздороваться и попрощаться с клиентом»

In [48]:
```
print('Задание f. Проверять требование к менеджеру: «В каждом диалоге обязательно необходимо поздороваться и по

for dlg_id in greeting_and_bye_ids:
    print(f'Менеджер в диалоге {dlg_id} поздоровался и попрощался.\n')
for dlg_id in only_greeting_ids:
    print(f'Менеджер в диалоге {dlg_id} только поздоровался.\n')
for dlg_id in only_byes_ids:
    print(f'Менеджер в диалоге {dlg_id} только попрощался.\n')
```

Задание f. Проверять требование к менеджеру: «В каждом диалоге обязательно необходимо поздороваться и попрощаться с клиентом»

Менеджер в диалоге 0 поздоровался и попрощался.

Менеджер в диалоге 1 поздоровался и попрощался.

Менеджер в диалоге 3 поздоровался и попрощался.

Менеджер в диалоге 2 только поздоровался.

Менеджер в диалоге 4 только попрощался.

Менеджер в диалоге 5 только попрощался.

Посчитаем метрики качества для получившихся классификаторов.

In [49]:
```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

In [50]:
```
recall_byes = recall_score(test_predict_byes['true_class'], test_predict_byes['predicted_class'])
precision_byes = precision_score(test_predict_byes['true_class'], test_predict_byes['predicted_class'])
f_measure_byes = f1_score(test_predict_byes['true_class'], test_predict_byes['predicted_class'])


recall_greetings = recall_score(test_predict_greetings['true_class'], test_predict_greetings['predicted_class']
precision_greetings = precision_score(test_predict_greetings['true_class'], test_predict_greetings['predicted_c
f_measure_greetings = f1_score(test_predict_greetings['true_class'], test_predict_greetings['predicted_class'])
```

Метрики качества классификаторов для задач классификации для приветсвий и прощаний.

In [51]:
```
print('Выведем метрики качества для классификации приветсвий и прощаний: \n')
print('Recall для прощаний: {0}, Precision для прощаний: {1}, F-measure для прощаний: {2}\n'.format(recall_byes
print('Recall для приветствий: {0}, Precision для приветствий: {1}, F-measure для приветствий: {2}\n'.format(re
```

Выведем метрики качества для классификации приветсвий и прощаний:

Recall для прощаний: 1.0, Precision для прощаний: 1.0, F-measure для прощаний: 1.0

Recall для приветствий: 1.0, Precision для приветствий: 1.0, F-measure для приветствий: 1.0

Создадим функцию для преобразования текста в список частей речи с помощью библиотеки Natasha

In [52]:
```
from natasha import (
    Segmenter,
    MorphVocab,

    NewsEmbedding,
    NewsMorphTagger,
    NewsSyntaxParser,
    NewsNERTagger,

    PER,
    NamesExtractor,

    Doc
)


def text_to_POS_dict(text):
    segmenter = Segmenter()
    morph_vocab = MorphVocab()

    emb = NewsEmbedding()
    morph_tagger = NewsMorphTagger(emb)
    syntax_parser = NewsSyntaxParser(emb)
    ner_tagger = NewsNERTagger(emb)

    names_extractor = NamesExtractor(morph_vocab)
    doc = Doc(text)

    doc.segment(segmenter)

    doc.tag_morph(morph_tagger)

    POS_list = []
    for i in np.arange(len(text.split())):
        POS_list.append(doc.sents[0].morph.as_json['tokens'][i].pos)
    word_list = text.split()
    return dict(zip(word_list, POS_list))
```

Создадим словарь всех имён. Для это воспользуемся списком уникальных российских имён (включает не только русские имена, но и арабские, армянские, грузинские, немецкие, греческие, еврейские, польские и тюркские на русском языке)

Источник (Портал открытых данных правительства Москвы):

https://data.mos.ru/opendata/7704111479-svedeniya-o-naibolee-populyarnyh-mujskih-imenah-sredi-novorojdennyh

https://data.mos.ru/opendata/7704111479-svedeniya-o-naibolee-populyarnyh-jenskih-imenah-sredi-novorojdennyh

In [53]:
```
female_names = pd.read_excel('./data_for_name_recognition/female_names.xlsx')
male_names = pd.read_excel('./data_for_name_recognition/male_names.xlsx')

female_names_list = female_names['Name'][1::]
male_names_list = male_names['Name'][1::]
names = pd.concat([female_names_list, male_names_list]).values
names = [name.lower() for name in names]
```

```
C:\Users\gleko\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packa
ges\Python310\site-packages\openpyxl\styles\stylesheet.py:226: UserWarning: Workbook contains no default style,
apply openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
C:\Users\gleko\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packa
ges\Python310\site-packages\openpyxl\styles\stylesheet.py:226: UserWarning: Workbook contains no default style,
apply openpyxl's default
  warn("Workbook contains no default style, apply openpyxl's default")
```

Задание b и c. Извлекать реплики, где менеджер представил себя, извлекать имя менеджера.

Выведем реплики с представлением себя менеджером по всем диалогам

In [54]:
```
print('Задание b и c. Извлекать реплики, где менеджер представил себя, извлекать имя менеджера.\n')
introduce_replicas = []
manager_names = []
introduce_dlg_ids = []
for index, row in start_replicas_df.iterrows():
    replica = row['text']
    words = replica.split()
    for word in words:
        if (lemmatize(tokenize(word))[0] in names) and ("зовут" in replica):
            manager_names.append(word)
            introduce_replicas.append(replica)
            introduce_dlg_ids.append(row['dlg_id'])
            print(f'В диалоге {row.dlg_id} менеджер {word.capitalize()} представился.' + "\n" + f'Реплика: "{re
            update_dict("names", row.dlg_id, word.capitalize())
            update_dict("introduce", row.dlg_id, replica)
```

```
Задание b и c. Извлекать реплики, где менеджер представил себя, извлекать имя менеджера.

В диалоге 0 менеджер Ангелина представился.
Реплика: "Меня зовут ангелина компания диджитал бизнес звоним вам по поводу продления лицензии а мы с серым у в
ас скоро срок заканчивается"

В диалоге 1 менеджер Ангелина представился.
Реплика: "Меня зовут ангелина компания диджитал бизнес звоню вам по поводу продления а мы сели обратила внимани
е что у вас срок заканчивается"

В диалоге 2 менеджер Ангелина представился.
Реплика: "Меня зовут ангелина компания диджитал бизнес звоню вам по поводу продления лицензии а мастера мы с ва
ми сотрудничали по видео там"

В диалоге 3 менеджер Максим представился.
Реплика: "Добрый меня максим зовут компания китобизнес удобно говорить"
```

Напишем функцию извлечения названия компании из реплики.

- **Основана на предположении, о том что, если слово "компания" встречается (в любой падежной форме) в стартовой реплике (одной из первых трёх реплик в диалоге с клиентом), то за словом "компания" следует название.**
- **Также название компании может содержать несколько слов. В этом случае предполагается, что последовательность слов будет содержать прилагательные (adjective) и существительные (noun). Проверяем это условие с помощью библиотеки Natasha**

In [55]:
```
def extract_name_of_company(replica):
    name_of_company = []
    dict_of_POS = text_to_POS_dict(replica)
    words = replica.split()
    lemmas = lemmatize(tokenize(replica))
```

```
    if 'компания' in lemmas:
        ind_of_company = lemmas.index('компания')

        if ind_of_company + 1 < len(words):
            name_of_company.append(words[ind_of_company + 1])
            ind = ind_of_company + 1

            while dict_of_POS[words[ind + 1]] in ['ADV', 'NOUN']:
                name_of_company.append(words[ind + 1])
                ind += 1

    return name_of_company
```

Задание d. Извлекать название компании.

In [56]:
```
print('Задание d. Извлекать название компании.\n')
company_names = []
dlg_ids_cmp_nms = []
for index, row in start_replicas_df.iterrows():
    replica = row['text']
    name_of_company = extract_name_of_company(replica)


    if len(name_of_company) > 1:
        name_of_company = ' '.join(name_of_company)
    if len(name_of_company) == 1:
        name_of_company = name_of_company[0]
    if len(name_of_company) > 0:
        print(f'В диалоге {row.dlg_id} менеджер из компании "{name_of_company.capitalize()}".' + "\n" f'Реплика
        company_names.append(name_of_company)
        dlg_ids_cmp_nms.append(row['dlg_id'])
        update_dict("companies", row.dlg_id, name_of_company.capitalize())
```

```
Задание d. Извлекать название компании.

В диалоге 0 менеджер из компании "Диджитал бизнес".
Реплика: "Меня зовут ангелина компания диджитал бизнес звоним вам по поводу продления лицензии а мы с серым у в
ас скоро срок заканчивается"

В диалоге 1 менеджер из компании "Диджитал бизнес".
Реплика: "Меня зовут ангелина компания диджитал бизнес звоню вам по поводу продления а мы сели обратила внимани
е что у вас срок заканчивается"

В диалоге 2 менеджер из компании "Диджитал бизнес".
Реплика: "Меня зовут ангелина компания диджитал бизнес звоню вам по поводу продления лицензии а мастера мы с ва
ми сотрудничали по видео там"

В диалоге 3 менеджер из компании "Китобизнес".
Реплика: "Добрый меня максим зовут компания китобизнес удобно говорить"
```

In [57]:
```
for dlg_id in df.sort_values(by = 'dlg_id').dlg_id.unique():
    greetings = 0 if dlg_id not in greetings_dict.keys() else greetings_dict[dlg_id]
    byes = 0 if dlg_id not in byes_dict.keys() else byes_dict[dlg_id]
    name = 0 if dlg_id not in names_dict.keys() else names_dict[dlg_id]
    introduce = 0 if dlg_id not in introduce_dict.keys() else introduce_dict[dlg_id]
    company = 0 if dlg_id not in companies_dict.keys() else companies_dict[dlg_id]

    df_parsing_result.at[dlg_id, 'greeting'] = greetings
    df_parsing_result.at[dlg_id, 'bye'] = byes
    df_parsing_result.at[dlg_id, 'name'] = name
    df_parsing_result.at[dlg_id, 'introduce_replica'] = introduce
    df_parsing_result.at[dlg_id, 'company'] = company

    if greetings == 0 or byes == 0:
        df_parsing_result.at[dlg_id, 'greeting_and_bye_condition'] = False
    else:
        df_parsing_result.at[dlg_id, 'greeting_and_bye_condition'] = True
```

In [58]:
```
pd.options.display.max_colwidth = 200
df_parsing_result
```

Out[58]:

dlg_id	greeting	bye	greeting_and_bye_condition	introduce_replica	name	company
0	[Алло здравствуйте]	[Всего хорошего до свидания]	True	[Меня зовут ангелина компания диджитал бизнес звоним вам по поводу продления лицензии а мы с серым у вас скоро срок заканчивается]	[Ангелина]	[Диджитал бизнес]
1	[Алло здравствуйте]	[Угу да вижу я эту почту хорошо тогда исправлю на эту будем ждать ответа всего хорошего, До свидания]	True	[Меня зовут ангелина компания диджитал бизнес звоню вам по поводу продления а мы сели обратила внимание что у вас срок заканчивается]	[Ангелина]	[Диджитал бизнес]
2	[Алло здравствуйте]	0	False	[Меня зовут ангелина компания диджитал бизнес звоню вам по поводу продления лицензии а мастера мы с вами сотрудничали по видео там]	[Ангелина]	[Диджитал бизнес]
3	[Алло дмитрий добрый день, Добрый меня максим зовут компания китобизнес удобно говорить]	[Угу все хорошо да понедельника тогда всего доброго]	True	[Добрый меня максим зовут компания китобизнес удобно говорить]	[Максим]	[Китобизнес]
4	0	[Во вторник все ну с вами да тогда до вторника до свидания]	False	0	0	0
5	0	[Ну до свидания хорошего вечера]	False	0	0	0

In [ ]: