№1

а) Рассмотрим анализатор ***Mystem***. Я пробовал анализировать при помощи терминала грамматическую информацию о каждом из следующих слов: *"кот", "стул", "компьютер", "лол", "чиллить"*. Как можно увидеть из результата ниже, простые слова отрабатываются качественно, в отличие от слов из молодежного сленга, которые данный инструмент воспринимает как незнакомые:

```
Кот{кот=S,муж,од=им,ед}
Стул{стул=S,муж,неод=вин,ед|стул=S,муж,неод=им,ед}
Компьютер{компьютер=S,муж,неод=вин,ед|компьютер=S,муж,неод=им,ед}
Лол{лола=S,имя,жен,од=вин,мн|лола=S,имя,жен,од=род,мн}
Чиллить{чиллить?=S,жен,од=вин,ед|чиллить?=S,жен,од=им,ед|чиллить?=S,
жен,неод=вин,ед|чиллить?=S,жен,неод=им,ед|чиллить?=S,имя,жен,од=пр,м
н|чиллить?=S,имя,жен,од=пр,ед|чиллить?=S,имя,жен,од=вин,мн|чиллить?=
S,имя,жен,од=вин,ед|чиллить?=S,имя,жен,од=дат,мн|чиллить?=S,имя,жен,
од=дат,ед|чиллить?=S,имя,жен,од=род,мн|чиллить?=S,имя,жен,од=род,ед|
чиллить?=S,имя,жен,од=твор,мн|чиллить?=S,имя,жен,од=твор,ед|чиллить?
=S,имя,жен,од=им,мн|чиллить?=S,имя,жен,од=им,ед|чиллитя?=S,имя,жен,о
д=вин,мн|чиллитя?=S,имя,жен,од=род,мн|чиллитя?=S,имя,муж,од=вин,мн|ч
иллитя?=S,имя,муж,од=род,мн}
```

Так, Mystem неплохо справляется с легкими словами, но хуже с молодежным сленгом.

б) Рассмотрим анализатор ***snowball***. Будем использовать предложенные на лекции программы `stemmer.py` и `snow_word.py`. Проанализируем слова *"окно", "пылесос", "помощь", "вода", "кринж", "рофлить", "вайбик"* и получим следующий результат:

```
['окн', 'пылесос', 'помощ', 'вод', 'кринж', 'рофл', 'вайбик']
```

Как можно видеть, данный инструмент хорошо справляется с реализацией стем из набора слов, в том числе из молодежного сленга, но все равно могут возникать трудности (например, для слова "вайбик" правильным стемом будет 'вайб').

в) Рассмотрим лемматизатор ***WordNet*** для английских слов. Вновь будем использовать предложенную на лекцию программу `snow_word.py` и проанализируем следующие слова: *"world", "computers", "science", "live", "cringe", "diss"*. Получим:

```
['world', 'computer', 'science', 'live', 'cringe', 'dis']
```

Так, этот инструмент хорошо справляется с лемматизацией слов, в т.ч. с жаргонными словами, но в редких случаях возникают неточности (например, со словом "diss").

г) Теперь рассмотрим такой мощный инструмент, как ***pymorphy2***. Будем использовать предложенную на лекции программу `pymorph.py` и слова *“кот”, “музыка”, “кек”*. Получим:

```
Parse(word='кот', tag=OpencorporaTag('NOUN,anim,masc sing,nomn'),
normal_form='кот', score=1.0, methods_stack=((DictionaryAnalyzer(),
'кот', 52, 0),))
```

```
Parse(word='музыка', tag=OpencorporaTag('NOUN,inan,femn sing,nomn'),
normal_form='музыка',                                    score=1.0,
methods_stack=((DictionaryAnalyzer(), 'музыка', 44, 0),))
Parse(word='кек',  tag=OpencorporaTag('NOUN,inan,masc  sing,nomn'),
normal_form='кек',                                  score=0.833333,
methods_stack=((DictionaryAnalyzer(), 'кек', 19, 0),))
Parse(word='кек',  tag=OpencorporaTag('NOUN,inan,masc  sing,accs'),
normal_form='кек',                                  score=0.166666,
methods_stack=((DictionaryAnalyzer(), 'кек', 19, 3),))
```

Как можно видеть, данный инструмент является достаточно мощным, поскольку отображает больше информации о каждом слове и анализирует его несколькими методами с оценкой вероятности их корректности.

№2

Рассмотрим следующие слова: *“кот”, “компьютер”, “лол”* для русского языка и *“Phone”, “Running”, “Lol”* для английского.

Для русского языка:

Результат при помощи mystem:

```
Кот{кот=S,муж,од=им,ед}
Компьютер{компьютер=S,муж,неод=вин,ед|компьютер=S,муж,неод=им,ед}
Лол{лола=S,имя,жен,од=вин,мн|лола=S,имя,жен,од=род,мн}
```

Результат при помощи snowball:

```
Stems: ['кот', 'компьютер', 'лол']
```

Результат при помощи pymorphy2:

```
Parse(word='кот',  tag=OpencorporaTag('NOUN,anim,masc  sing,nomn'),
normal_form='кот', score=1.0, methods_stack=((DictionaryAnalyzer(),
'кот', 52, 0),))
Parse(word='компьютер',          tag=OpencorporaTag('NOUN,inan,masc
sing,nomn'),        normal_form='компьютер',        score=0.580645,
methods_stack=((DictionaryAnalyzer(), 'компьютер', 1844, 0),))
Parse(word='компьютер',          tag=OpencorporaTag('NOUN,inan,masc
sing,accs'),        normal_form='компьютер',        score=0.419354,
methods_stack=((DictionaryAnalyzer(), 'компьютер', 1844, 3),))
Parse(word='лол',           tag=OpencorporaTag('NOUN,anim,femn,Name
plur,gent'),             normal_form='лола',             score=0.5,
methods_stack=((DictionaryAnalyzer(), 'лол', 69, 8),))
```

```
Parse(word='лол',              tag=OpencorporaTag('NOUN,anim,femn,Name
plur,accs'),              normal_form='лола',              score=0.5,
methods_stack=((DictionaryAnalyzer(), 'лол', 69, 10),))
```

Для английского языка:

Результат при помощи snowball:

```
Stems: ['phone', 'running', 'lol']
```

Результат при помощи WordNet:

```
Lemmas: ['phone', 'running', 'lol']
```

Как можно видеть, в целом анализаторы одинаково хорошо обрабатывают легкие и привычные нам слова, но с разным качеством отрабатывают жаргонные слова, что, конечно, связано со стремительным развитием языка как в разговорной речи, так и в целом. Также можно заметить, что библиотека pymorphy2 дает больше информации о словах.

№3

Составим программу `task3.py` (прикреплена отдельна) для печати всех словоформ для выбранного нами набора слов и их кол-ва при помощи инструмента **pymorphy2**. Так, проанализировав слова *“читать”, “слушать”, “мир”, “красивый”*, получим, что наибольшее среди них кол-во словоформ имеет слово “читать” – 132 словоформы.

№4

В ходе выполнения этой практической работы инструмент **Pymorphy2** мне понравился больше, так как он предоставляет большое количество информации о слове и хорошо справляется с обычными словами и частично с молодежным сленгом. Также он поддерживает синтез словоформ, что делает его полезным инструментом для морфологического анализа. Однако стоит отметить, что данная библиотека ограничена в выборе языков.