TextCritical.net - Bug #480

Feature # 466 (Closed): Lemma lookup

Task # 472 (Closed): Lemma lookup REST API

Make morphology lookups insenstive to browser differences

12/22/2012 08:05 AM - Luke Murphey

Status:	Closed	**Start date:**	12/22/2012
Priority:	Normal	**Due date:**	
Assignee:	Luke Murphey	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	0.2		

Description

For some reason different browsers seem to send different requests for looking words. For an example, go to Romans 16 and looking ἐκκλησιας in verse one on both FireFox on a desktop and and Safari on iOS. The query on the desktop succeeds while the iOS device fails. This does not apply to all words though.

History

#1 - 12/22/2012 08:07 AM - Luke Murphey

- Subject changed from Fix browser lookup bug to Make morphology lookups insenstive to browser differences

#2 - 12/22/2012 08:42 AM - Luke Murphey

On iOS the request for Ἀσπάσασθε looks like:

[22/Dec/2012 02:38:47] "GET /api/word_parse/%E1%BC%88%CF%83%CF%80%CE%AC%CF%83%CE%B1%CF%83%CE%B8%CE%B5 HTTP/1.1" 200 2

On the desktop:

[22/Dec/2012 02:38:21] "GET /api/word_parse/%E1%BC%88%CF%83%CF%80%E1%BD%B1%CF%83%CE%B1%CF%83%CE%B8%CE%B5 HTTP/1.1" 200 460

From what I can tell, the diacritics are represented differently.

#3 - 12/22/2012 08:42 AM - Luke Murphey

I tried normalizing the Unicode using all four forms of unistr (http://docs.python.org/2/library/unicodedata.html; none worked.

#4 - 12/22/2012 09:16 AM - Luke Murphey

The issue is that the data in the database must be normalized

αβαις in the database is equivalent to u'\u1f71\u03b2\u03b1\u03b9\u03c2'

Normalizing in Python results in the following, none of which are equivalent:

- NFC: u'\u03ac\u03b2\u03b1\u03b9\u03c2'
- NFKC: u'\u03ac\u03b2\u03b1\u03b9\u03c2'
- NFD: u'\u03b1\u0301\u03b2\u03b1\u03b9\u03c2'
- NFKD: u'\u03b1\u0301\u03b2\u03b1\u03b9\u03c2'

However, normalizing the value manually works:

```
s = u'\u1f71\u03b2\u03b1\u03b9\u03c2'
wf = WordForm.objects.filter(form=s)[0]
wf.form = unicodedata.normalize("NFKC", s)
wf.save()

# This returns successfully
WordForm.objects.filter(form=unicodedata.normalize("NFKC", s))
```

#5 - 12/22/2012 09:53 AM - Luke Murphey

May want to store versions without diacritic marks in case a search fails because the diacritics are different. See http://stackoverflow.com/questions/517923/what-is-the-best-way-to-remove-accents-in-a-python-unicode-string

#6 - 12/22/2012 09:55 AM - Luke Murphey

The following script should convert an existing database to use normalized Unicode:

```
import unicodedata
import reader.models
from django.db import transaction

@transaction.commit_on_success
def normalize_database_unicode():
    for wf in reader.models.WordForm.objects.all():
        wf.form = unicodedata.normalize("NFKC", wf.form)
        wf.save()

normalize_database_unicode()
```

#7 - 12/22/2012 10:19 AM - Luke Murphey

- Status changed from New to In Progress

#8 - 12/22/2012 06:52 PM - Luke Murphey

- Description updated

- Status changed from In Progress to Closed

- % Done changed from 0 to 100