SVD Plots

Importing Libraries

In [46]:
```
from keras import backend as K
from keras import regularizers
import numpy as np
from gensim_download import pickle_rw
from keras.layers import Dense
from keras.models import Sequential
import matplotlib.pyplot as plt
import plotly.plotly as py
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
import numpy.linalg



from numpy.linalg import det

from numpy.linalg import inv

import sympy as sympy

from sympy import Matrix
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
```

Defining Functions

In [15]:
```
def make_dict(vocab, vectors):
    """Make dictionary of vocab and vectors"""
    return {vocab[i]: vectors[i] for i in range(len(vocab))}


def vocab_train_test(embedding, lg1, lg2, lg1_vocab):
    """Create training and test vocabularies"""
    if embedding == 'zeroshot':
        with open('../data/zeroshot/transmat/data/' +
                  'OPUS_en_it_europarl_train_5K.txt') as f:
            vocab_train = [(_.split(' ')[0], _.split(' ')[1])
                           for _ in f.read().split('\n')[:-1]]
        with open('../data/zeroshot/transmat/data/' +
                  'OPUS_en_it_europarl_test.txt') as f:
            vocab_test = [(_.split(' ')[0], _.split(' ')[1])
                          for _ in f.read().split('\n')[:-1]]

    elif embedding in ['fasttext_random', 'fasttext_top']:
        embedding, split = embedding.split('_')
        lg1_lg2, lg2_lg1 = pickle_rw((lg1 + '_' + lg2, 0),
                                     (lg2 + '_' + lg1, 0), write=False)
        # T = Translation, R = Reverse (translated and then translated back)
        # Create vocab from 2D translations
        vocab_2D = []
        for lg1_word in lg1_vocab:
            # Translate lg1_word
            if lg1_word in lg1_lg2:
                lg1_word_T = lg1_lg2[lg1_word]

                # Check if translated word (or lowercase) is in lg2_lg1
                if lg1_word_T in lg2_lg1.keys():
                    lg1_word_R = lg2_lg1[lg1_word_T]
                elif lg1_word_T.lower() in lg2_lg1.keys():
                    lg1_word_T = lg1_word_T.lower()
                    lg1_word_R = lg2_lg1[lg1_word_T]
                else:
                    lg1_word_R = None

                # Check if lg1_word and lg1_word_R are equal (lowercase)
                if lg1_word_R:
                    if lg1_word.lower() == lg1_word_R.lower():
                        vocab_2D.append((lg1_word, lg1_word_T))
```

```
        print('length of '+ lg1+'-'+ lg2+ ' vocab: '+str(len(vocab_2D)))

        #Create Train/Test vocab

        if split == 'random':
            sample = np.random.choice(len(vocab_2D), 6500, replace=False)
            vocab_train = np.asarray(vocab_2D)[sample[:5000]].tolist()
            vocab_test = np.asarray(vocab_2D)[sample[5000:]].tolist()
        elif split == 'top':
            sample = np.random.choice(range(6500), 6500, replace=False)
            vocab_train = np.asarray(vocab_2D)[:5000, :].tolist()
            vocab_test = np.asarray(vocab_2D)[:1500, :].tolist()
        else:
            pass

        # if split == 'random':
        #     sample = np.random.choice(len(vocab_2D), 900, replace=False)
        #     vocab_train = np.asarray(vocab_2D)[sample[:700]].tolist()
        #     vocab_test = np.asarray(vocab_2D)[sample[700:]].tolist()
        # elif split == 'top':
        #     sample = np.random.choice(range(900), 900, replace=False)
        #     vocab_train = np.asarray(vocab_2D)[:700, :].tolist()
        #     vocab_test = np.asarray(vocab_2D)[:200, :].tolist()
        # else:
        #     pass

    return vocab_train, vocab_test


def vectors_train_test(vocab_train, vocab_test):
    """Create training and test vectors"""
    X_train, y_train = zip(*[(lg1_dict[lg1_word], lg2_dict[lg2_word])
                             for lg1_word, lg2_word in vocab_train])
    X_test, y_test = zip(*[(lg1_dict[lg1_word], lg2_dict[lg2_word])
                           for lg1_word, lg2_word in vocab_test])
    return map(np.asarray, (X_train, X_test, y_train, y_test))


def translation_matrix(X_train, y_train):
    """Fit translation matrix T"""
    model = Sequential()
    model.add(Dense(300, use_bias=False, input_shape=(X_train.shape[1],),kernel_regularizer=regularizers
    model.compile(loss='mse', optimizer='adam')
```

```
    history = model.fit(X_train, y_train, batch_size=128, epochs=20,
                        verbose=False)
    T = model.get_weights()[0]

    T = np.matrix(T)

    M = np.multiply(np.matrix(T),100)

    T_norm, T_normed = normalize(M)

    D = np.linalg.det(M)

    I = inv(T)

    #Fr_norm = np.linalg.det(np.subtract(np.matmul(M,M.getH()),np.matmul(M.getH(),M)))

    Fr_norm = np.linalg.det(np.matrix(np.subtract(np.matmul(T,T.getH()),np.matmul(T.getH(),T))))

    #print(T_normed)
    #print(T_normed.getH())

    #print(np.around(np.matmul(T_normed,T_normed.getH())))

    #print(np.around(np.matmul(T_normed.getH(),T_normed)))

    print ("Determinant:"+str(D))

    #print ("Fr_norm:"+str(Fr_norm))

    if np.array_equal(np.around(np.matmul(T_normed,T_normed.getH())), np.around(np.matmul(T_normed.getH(
        tf = "True"
    else:
        tf = "False"

    return model, history, T, D, tf, I, M

def translation_accuracy(X_test, y_test):
    """Get predicted matrix 'yhat' using 'T' and find translation accuracy"""
    # yhat
    yhat = X.dot(T)
    count = 0
    for i in range(len(y_test)):
        if yhat[i,:].all() == y_test[i,:].all():
```

```
            count = count + 1
    accuracy = count/len(y_test)*100
    return accuracy

def svd(T):
    """Perform SVD on the translation matrix 'T' """
    U, s, Vh = numpy.linalg.svd(T, full_matrices=False )
    return U, s, Vh

def T_svd_EDA(s):
    """Perform SVD on the translation matrix 'T' """
    plt.hist(s, bins='auto', range = (0,1),normed = 1)
    plt.show()




def normalize(matrix):
    """Normalize the rows of a matrix"""
    matrix_norm = np.linalg.norm(matrix, axis=1)
    matrix_normed = matrix / np.repeat(matrix_norm, matrix.shape[1]). \
        reshape(matrix.shape)
    return matrix_norm, matrix_normed


def translation_results(X, y, vocab, M, lg2_vectors, lg2_vocab):
    """X, y, vocab - The training or test data that you want results for
    T - The translation matrix
    lg2_vectors, lg2_vocab - Foreign language used to find the nearest neighbor
    """

    # Data Prep on Inputs
    X_word, y_word = zip(*vocab)
    X_norm, X_normed = normalize(X)
    y_norm, y_normed = normalize(y)
    lg2_vectors_norm, lg2_vectors_normed = normalize(lg2_vectors)

    # yhat
    yhat = X.dot(M)
    yhat_norm, yhat_normed = normalize(yhat)

    #X_norm = normalize(X)
```

```
    # Nearest Neighbors
#     neg_cosine = -yhat_normed.dot(lg2_vectors_normed.T)
#     ranked_neighbor_indices = np.argsort(neg_cosine, axis=1)

#     # Nearest Neighbor
#     nearest_neighbor_indices = ranked_neighbor_indices[:, 0]
#     yhat_neighbor = lg2_vectors[nearest_neighbor_indices, :]
#     yhat_neighbor_norm, yhat_neighbor_normed = normalize(yhat_neighbor)
#     yhat_neighbor_word = np.asarray(lg2_vocab)[nearest_neighbor_indices]

#     # Results DF
#     cols = ['X_norm', 'y_norm', 'yhat_norm', 'yhat_neighbor_norm',
#             'X_word', 'y_word', 'yhat_neighbor_word']
#     results_df = pd.DataFrame({'X_norm': X_norm,
#                                'y_norm': y_norm,
#                                'yhat_norm': yhat_norm,
#                                'yhat_neighbor_norm': yhat_neighbor_norm,
#                                'X_word': X_word,
#                                'y_word': y_word,
#                                'yhat_neighbor_word': yhat_neighbor_word,})
#     results_df = results_df[cols]
#     results_df['neighbor_correct'] = results_df.y_word == \
#         results_df.yhat_neighbor_word

    return yhat_norm


def T_norm_EDA(results_df):
    """Plot result norms side-by-side"""
    test_size = results_df.shape[0]
    test_accuracy = round(results_df.neighbor_correct.mean(), 2)

    print('Test Accuracy: '+str(test_accuracy)+'\n')

    plot_data = ['X_norm', 'y_norm', 'yhat_norm', 'yhat_neighbor_norm']
    # f, ax = plt.subplots(len(plot_data), sharex=True, sharey=True,
    #                      figsize=(10, 10))
    # for i, d in enumerate(plot_data):
    #     ax[i].hist(results_df[d], bins=100)
    #     ax[i].axis('off')
    #     title = '{}: mean={}, std={}'.format(d, round(results_df[d].mean(), 2), round(results_df[d].st
    #     ax[i].set_title(title)
```

```
    # f.subplots_adjust(hspace=0.7)
    # plt.savefig('../images/' + lg1 + '_' + lg2 + '_' + embedding +
    #             '_T_norm.png')
    # plt.close('all')
    return


def T_pca_EDA(T):
    """PCA on matrix T"""
    T_ss = StandardScaler().fit_transform(T)
    pca = PCA().fit(T_ss)
    n = pca.n_components_

    # plt.figure(figsize=(10, 6))
    # plt.xlim((0, n))
    # plt.ylim((0, 1))
    # plt.plot(range(n + 1), [0] + np.cumsum(pca.explained_variance_ratio_).
    #          tolist())
    # plt.plot(range(n + 1), np.asarray(range(n + 1)) / n)
    # plt.xlabel('Number of Eigenvectors')
    # plt.ylabel('Explained Variance')
    # plt.savefig('../images/' + lg1 + '_' + lg2 + '_' + embedding +
    #             '_T_isotropy.png')
    # plt.close('all')

    isotropy = (1 - sum(np.cumsum(pca.explained_variance_ratio_) * 1 / n)) / .5
    return isotropy


def T_report_results(embedding, lg1, lg2, lg1_vectors, lg2_vectors,
                     X_train, X_test, D, results_df, isotropy):
    md = '## ' + lg1.title() + ' to ' + lg2.title() + ' ' + \
        embedding.title() + '  \n'
    md += '- ' + lg1.title() + ' Vocabulary Size = ' + \
        '{:,.0f}'.format(lg1_vectors.shape[0]) + '  \n'
    md += '- ' + lg1.title() + ' Embedding Length = ' + \
        '{:,.0f}'.format(lg1_vectors.shape[1]) + '  \n'
    md += '- ' + lg2.title() + ' Vocabulary Size = ' + \
        '{:,.0f}'.format(lg2_vectors.shape[0]) + '  \n'
    md += '- ' + lg2.title() + ' Embedding Length = ' + \
        '{:,.0f}'.format(lg2_vectors.shape[1]) + '  \n'
    md += '- Train Size = ' + '{:,.0f}'.format(X_train.shape[0]) + '  \n'
    md += '- Test Size = ' + '{:,.0f}'.format(X_test.shape[0]) + '  \n'
```

```
    md += '- Determinant = ' + '{:,.0f}'.format(D) + '  \n'

    md += '- <b>Test Accuracy = ' + \
        '{:,.1%}'.format(results_df.neighbor_correct.mean()) + '</b>  \n\n'



    md += '#### Test L2 Norms  \n'
    md += '- X_norm: L2 norms for ' + lg1.title() + ' test vectors  \n'
    md += '- y_norm: L2 norms for ' + lg2.title() + ' test vectors  \n'
    md += '- yhat_norm: L2 norms for X.dot(T) test vectors ' + \
        '(T = translation matrix)  \n'
    md += '- yhat_neighbor norm: L2 norms for nearest neighbor' + \
        'to X.dot(T) in y test vectors  \n'
    md += '![](../images/' + lg1 + '_' + lg2 + '_' + embedding + \
        '_T_norm.png)  \n\n'

    md += '#### Translation Matrix Isotropy  \n'
    md += '- Isotropy = ' + '{:,.1%}'.format(isotropy) + '  \n'
    md += '![](../images/' + lg1 + '_' + lg2 + '_' + embedding + \
        '_T_isotropy.png)  \n\n'

    return md
```

Main Function (with SVD stats)

In [36]:
```
if __name__ == '__main__':
    # Manually set list of translations (embedding, lg1, lg2)
    translations = [#('fasttext_random', 'en', 'ru'),
                    ('fasttext_top', 'en', 'en'),
                    ('fasttext_top', 'en', 'ru'),
                    #('fasttext_random', 'en', 'de'),
                    ('fasttext_top', 'en', 'de'),
                    #('fasttext_random', 'en', 'es'),
                    ('fasttext_top', 'en', 'es'),
                    #('fasttext_random', 'en', 'zh-CN'),
                    ('fasttext_top', 'en', 'zh-CN'),



                     #('fasttext_random', 'ru', 'en'),
                    ('fasttext_top', 'ru', 'en'),
                    #('fasttext_random', 'ru', 'es'),
                    ('fasttext_top', 'ru', 'ru'),
                    ('fasttext_top', 'ru', 'de'),
                    ('fasttext_top', 'ru', 'es'),
                    #('fasttext_random', 'ru', 'zh-CN'),
                    ('fasttext_top', 'ru', 'zh-CN'),
                    #('fasttext_random', 'ru', 'de'),



                    #('fasttext_random', 'de', 'en'),
                    ('fasttext_top', 'de', 'en'),
                    #('fasttext_random', 'de', 'es'),
                    ('fasttext_top', 'de', 'ru'),
                    #('fasttext_random', 'de', 'ru'),
                    ('fasttext_top', 'de', 'de'),
                    ('fasttext_top', 'de', 'es'),
                    #('fasttext_random', 'de', 'zh-CN'),
                    ('fasttext_top', 'de', 'zh-CN'),



                    #('fasttext_random', 'es', 'en'),
                    ('fasttext_top', 'es', 'en'),
                    #('fasttext_random', 'es', 'de'),
                    ('fasttext_top', 'es', 'ru'),
```

```
                    ('fasttext_top', 'es', 'de'),
                    #('fasttext_random', 'es', 'ru'),
                    ('fasttext_top', 'es', 'es'),
                    #('fasttext_random', 'es', 'zh-CN'),
                    ('fasttext_top', 'es', 'zh-CN'),



                    #('fasttext_random', 'zh-CN', 'en'),
                    ('fasttext_top', 'zh-CN', 'en'),
                    #('fasttext_random', 'zh-CN', 'es'),
                    ('fasttext_top', 'zh-CN', 'ru'),
                    ('fasttext_top', 'zh-CN', 'de'),
                    ('fasttext_top', 'zh-CN', 'es'),
                    #('fasttext_random', 'zh-CN', 'ru'),
                    ('fasttext_top', 'zh-CN', 'zh-CN'),
                    #('fasttext_random', 'zh-CN', 'de'),




                    ]

s_min_en=[]
s_max_en=[]
s_mean_en=[]
s_median_en=[]
s_std_en=[]

s1_min_en=[]
s1_max_en=[]
s1_mean_en=[]
s1_median_en=[]
s1_std_en=[]



s_min_es=[]
s_max_es=[]
s_mean_es=[]
s_median_es=[]
s_std_es=[]
```

```
s1_min_es=[]
s1_max_es=[]
s1_mean_es=[]
s1_median_es=[]
s1_std_es=[]


s_min_ru=[]
s_max_ru=[]
s_mean_ru=[]
s_median_ru=[]
s_std_ru=[]

s1_min_ru=[]
s1_max_ru=[]
s1_mean_ru=[]
s1_median_ru=[]
s1_std_ru=[]



s_min_de=[]
s_max_de=[]
s_mean_de=[]
s_median_de=[]
s_std_de=[]

s1_min_de=[]
s1_max_de=[]
s1_mean_de=[]
s1_median_de=[]
s1_std_de=[]



s_min_zh=[]
s_max_zh=[]
s_mean_zh=[]
s_median_zh=[]
s_std_zh=[]

s1_min_zh=[]
s1_max_zh=[]
```

```
s1_mean_zh=[]
s1_median_zh=[]
s1_std_zh=[]

md = ''
for translation in translations:
    embedding, lg1, lg2 = translation
    # Vocab/Vectors/Dicts
    lg1_vocab, lg1_vectors, lg2_vocab, lg2_vectors = \
        pickle_rw((lg1 + '_' + embedding.split('_')[0] + '_vocab', 0),
                  (lg1 + '_' + embedding.split('_')[0] + '_vectors', 0),
                  (lg2 + '_' + embedding.split('_')[0] + '_vocab', 0),
                  (lg2 + '_' + embedding.split('_')[0] + '_vectors', 0),
                  write=False)
    lg1_dict = make_dict(lg1_vocab, lg1_vectors)
    lg2_dict = make_dict(lg2_vocab, lg2_vectors)

    print('Translation: '+lg1+'->'+lg2+'\n')

    # Train/Test Vocab/Vectors
    vocab_train, vocab_test = vocab_train_test(embedding, lg1, lg2, lg1_vocab)
    X_train, X_test, y_train, y_test = vectors_train_test(vocab_train,
                                                          vocab_test)

    # Fit tranlation matrix to training data
    model, history, T, D, tf,I, M = translation_matrix(X_train, y_train)

    print('biggest element:'+str(np.max(T))+'\n')

    U,s,Vh = svd(T)

    #scaler = MinMaxScaler()

    #scaler.fit(s)

    #s = scaler.transform(s)

    s1 = np.log10(s)

    print("min: "+str(min(s))+"\n")

    print("max: "+str(max(s))+"\n")
```

```
        print("mean: "+str(np.mean(s))+"\n")

        print("median: "+str(np.median(s))+"\n")

        print("std: "+str(np.std(s))+"\n")



        print("min: "+str(min(s1))+"\n")

        print("max: "+str(max(s1))+"\n")

        print("mean: "+str(np.mean(s1))+"\n")

        print("median: "+str(np.median(s1))+"\n")

        print("std: "+str(np.std(s1))+"\n")


        if lg1 == 'en':
            s_min_en.append(min(s))
            s_max_en.append(max(s))
            s_mean_en.append(np.mean(s))
            s_median_en.append(np.median(s))
            s_std_en.append(np.std(s))

            s1_min_en.append(min(s1))
            s1_max_en.append(max(s1))
            s1_mean_en.append(np.mean(s1))
            s1_median_en.append(np.median(s1))
            s1_std_en.append(np.std(s1))

        if lg1 == 'es':

            s_min_es.append(min(s))
            s_max_es.append(max(s))
            s_mean_es.append(np.mean(s))
            s_median_es.append(np.median(s))
            s_std_es.append(np.std(s))

            s1_min_es.append(min(s1))
            s1_max_es.append(max(s1))
            s1_mean_es.append(np.mean(s1))
```

```
            s1_median_es.append(np.median(s1))
            s1_std_es.append(np.std(s1))

        if lg1 == 'ru':

            s_min_ru.append(min(s))
            s_max_ru.append(max(s))
            s_mean_ru.append(np.mean(s))
            s_median_ru.append(np.median(s))
            s_std_ru.append(np.std(s))

            s1_min_ru.append(min(s1))
            s1_max_ru.append(max(s1))
            s1_mean_ru.append(np.mean(s1))
            s1_median_ru.append(np.median(s1))
            s1_std_ru.append(np.std(s1))

        if lg1 == 'de':

            s_min_de.append(min(s))
            s_max_de.append(max(s))
            s_mean_de.append(np.mean(s))
            s_median_de.append(np.median(s))
            s_std_de.append(np.std(s))

            s1_min_de.append(min(s1))
            s1_max_de.append(max(s1))
            s1_mean_de.append(np.mean(s1))
            s1_median_de.append(np.median(s1))
            s1_std_de.append(np.std(s1))

        if lg1 == 'zh-CN':

            s_min_zh.append(min(s))
            s_max_zh.append(max(s))
            s_mean_zh.append(np.mean(s))
            s_median_zh.append(np.median(s))
            s_std_zh.append(np.std(s))

            s1_min_zh.append(min(s1))
            s1_max_zh.append(max(s1))
            s1_mean_zh.append(np.mean(s1))
            s1_median_zh.append(np.median(s1))
```

```
            s1_std_zh.append(np.std(s1))




#         #print(s)

#         plt.hist(s,bins='auto')#bins=50,normed='True',range = (0.0,0.2))
#         #plt.plot(s)
#         plt.show()
```

```
max: 0.625777

mean: 0.0208729

median: 0.0114701

std: 0.0446892

min: -3.99049

max: -0.203581

mean: -1.9647

median: -1.94044

std: 0.491782
```

```
--------------------------------------------------------------------------------
```

Heatmaps of SVD Statistics

In [53]:
```
s_min = []
s_min.append(s_min_en)
s_min.append(s_min_ru)
s_min.append(s_min_de)
s_min.append(s_min_es)
s_min.append(s_min_zh)
s_max = []
s_max.append(s_max_en)
s_max.append(s_max_ru)
s_max.append(s_max_de)
s_max.append(s_max_es)
s_max.append(s_max_zh)
s_mean = []
s_mean.append(s_mean_en)
s_mean.append(s_mean_ru)
s_mean.append(s_mean_de)
s_mean.append(s_mean_es)
s_mean.append(s_mean_zh)
s_median = []
s_median.append(s_median_en)
s_median.append(s_median_ru)
s_median.append(s_median_de)
s_median.append(s_median_es)
s_median.append(s_median_zh)
s_std = []
s_std.append(s_std_en)
s_std.append(s_std_ru)
s_std.append(s_std_de)
s_std.append(s_std_es)
s_std.append(s_std_zh)

s1_min = []
s1_min.append(s1_min_en)
s1_min.append(s1_min_ru)
s1_min.append(s1_min_de)
s1_min.append(s1_min_es)
s1_min.append(s1_min_zh)
s1_max = []
s1_max.append(s1_max_en)
s1_max.append(s1_max_ru)
s1_max.append(s1_max_de)
s1_max.append(s1_max_es)
```

```
s1_max.append(s1_max_zh)
s1_mean = []
s1_mean.append(s1_mean_en)
s1_mean.append(s1_mean_ru)
s1_mean.append(s1_mean_de)
s1_mean.append(s1_mean_es)
s1_mean.append(s1_mean_zh)
s1_median = []
s1_median.append(s1_median_en)
s1_median.append(s1_median_ru)
s1_median.append(s1_median_de)
s1_median.append(s1_median_es)
s1_median.append(s1_median_zh)
s1_std = []
s1_std.append(s1_std_en)
s1_std.append(s1_std_ru)
s1_std.append(s1_std_de)
s1_std.append(s1_std_es)
s1_std.append(s1_std_zh)

init_notebook_mode(connected=True)
```

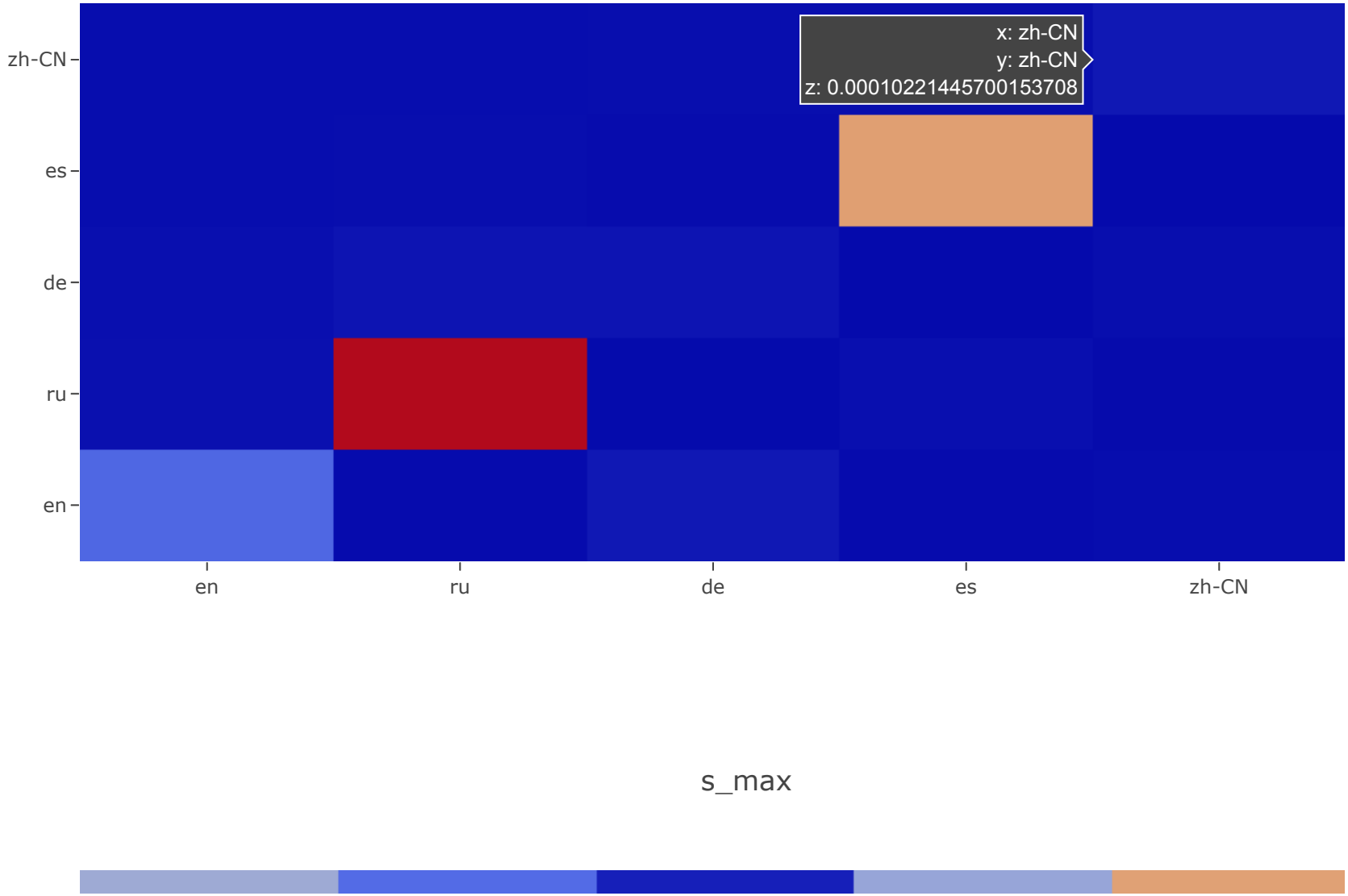
```
trace = go.Heatmap(z=s_min,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
layout = go.Layout(
title='s_min',
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s_min')

trace = go.Heatmap(z=s_max,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
layout = go.Layout(
title='s_max',
)
```

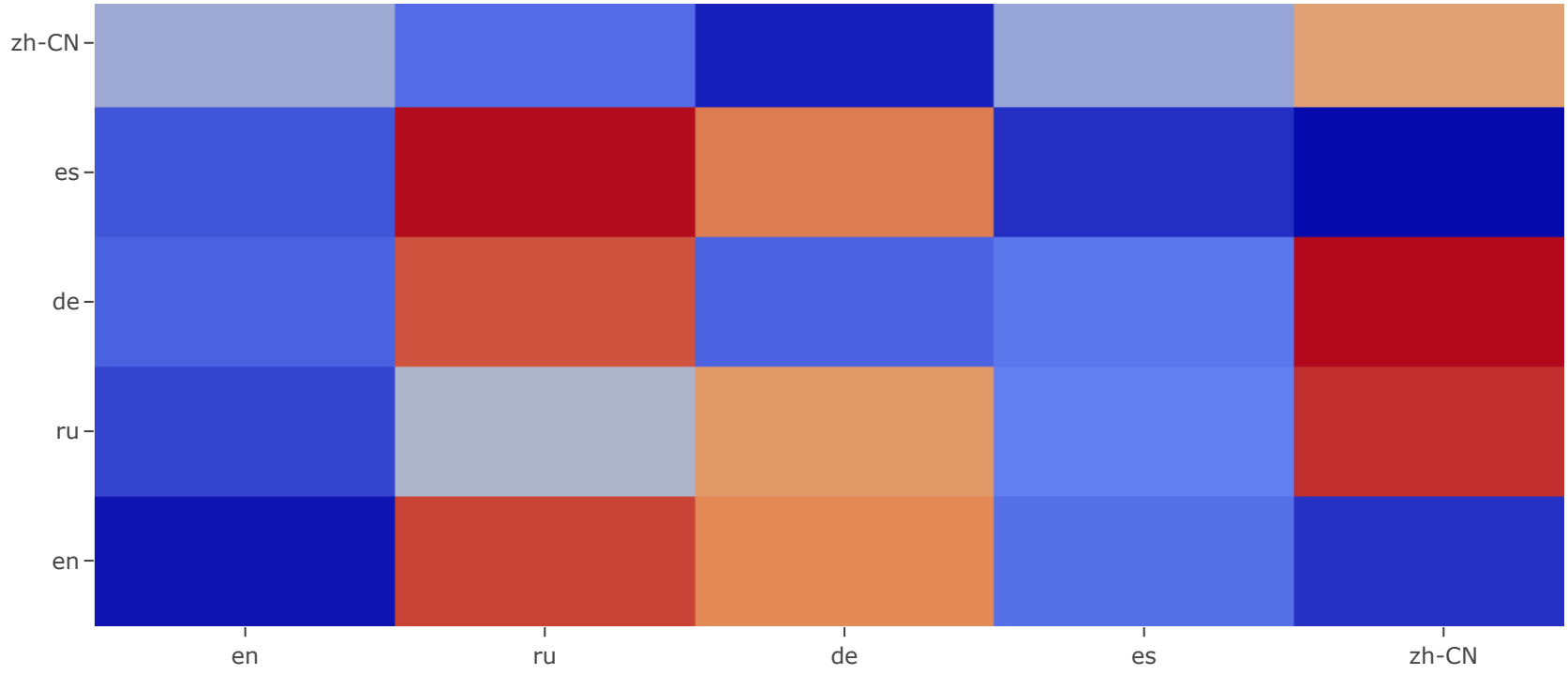
```
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s_max')

trace = go.Heatmap(z=s_mean,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
layout = go.Layout(
title='s_mean',
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s_mean')

trace = go.Heatmap(z=s_median,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
layout = go.Layout(
title='s_median',
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s_median')

trace = go.Heatmap(z=s_std,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
fig = go.Figure(data=data, layout=layout)
layout = go.Layout(
title='s_std',
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s_std')
```

```
trace = go.Heatmap(z=s1_min,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
```

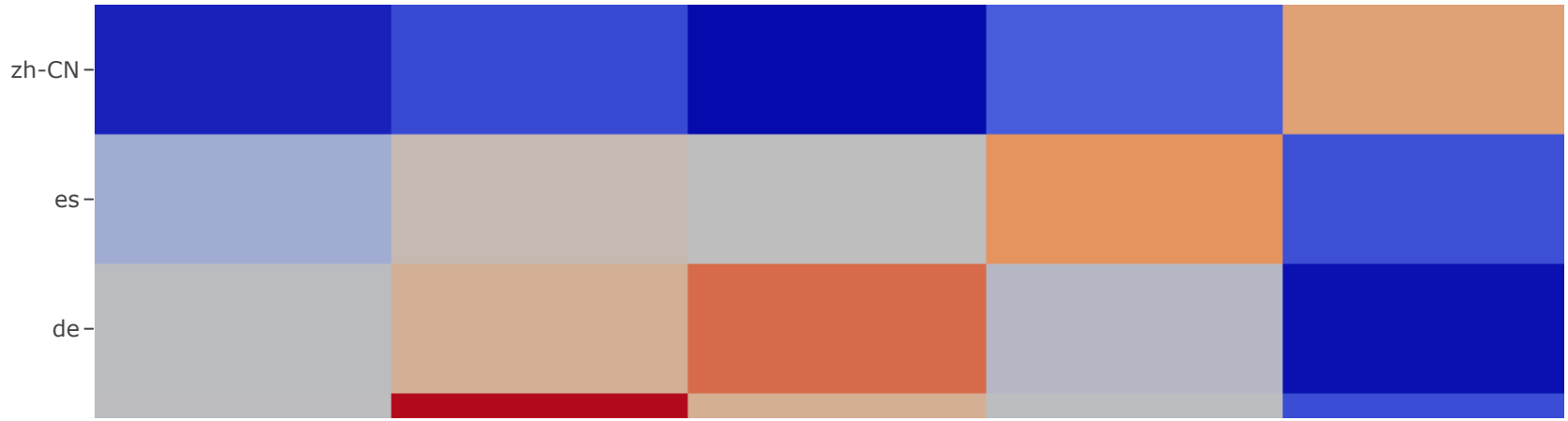
```
layout = go.Layout(
title='s1_min',
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s1_min')

trace = go.Heatmap(z=s1_max,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
layout = go.Layout(
title='s_max',
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s_max')

trace = go.Heatmap(z=s1_mean,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
layout = go.Layout(
title='s1_mean',
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s1_mean')

trace = go.Heatmap(z=s1_median,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
layout = go.Layout(
title='s1_median',
)
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s1_median')

trace = go.Heatmap(z=s1_std,
                   x=['en', 'ru', 'de', 'es', 'zh-CN'],
                   y=['en', 'ru', 'de', 'es', 'zh-CN'])
data=[trace]
layout = go.Layout(
title='s1_std',
)
```

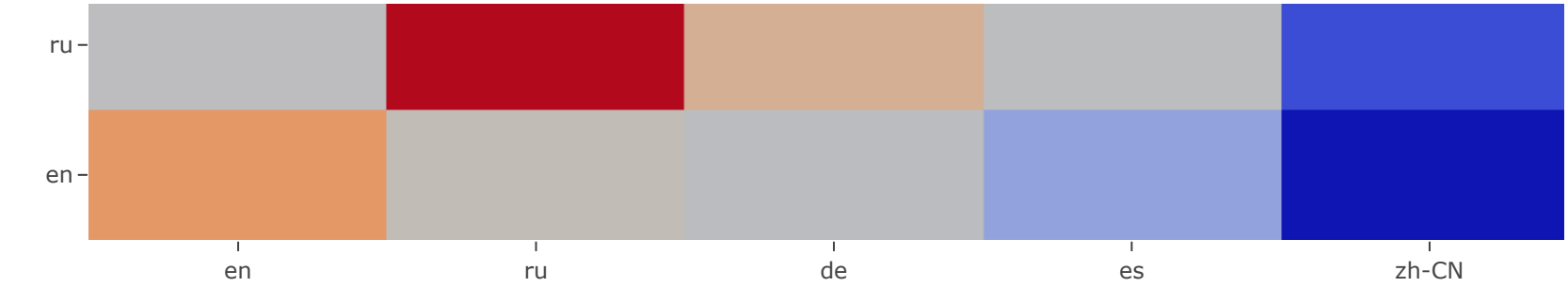
```
fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='s1_std')
```



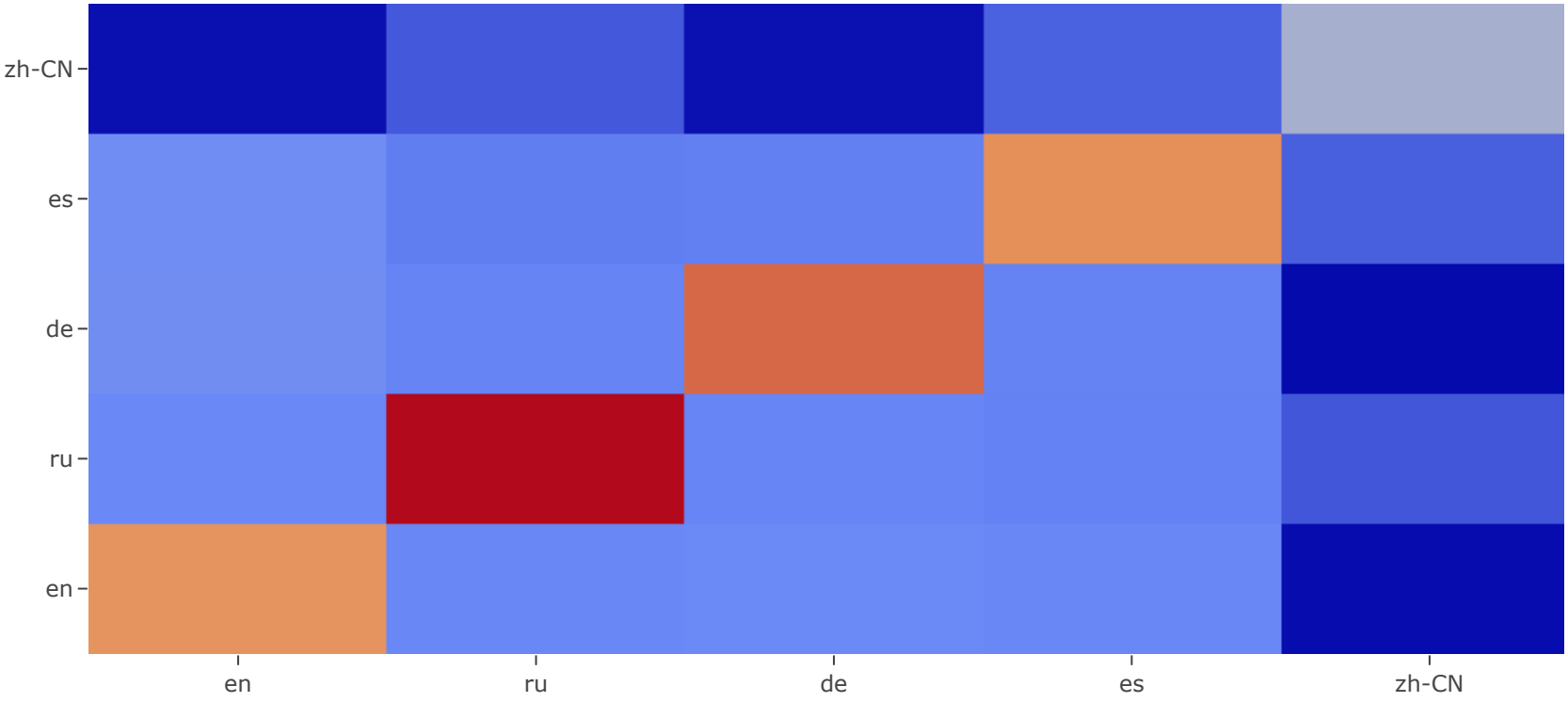


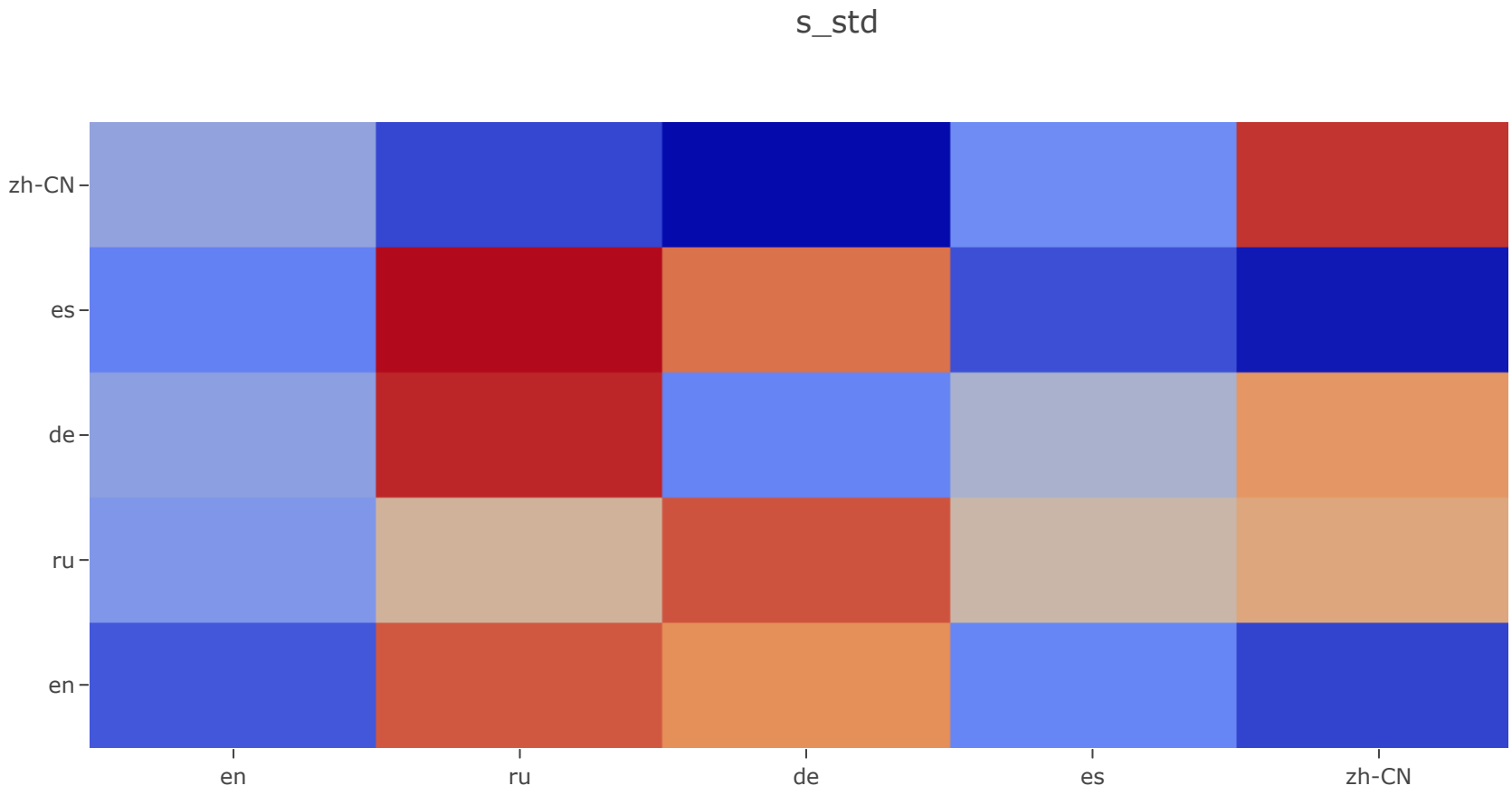
s_mean

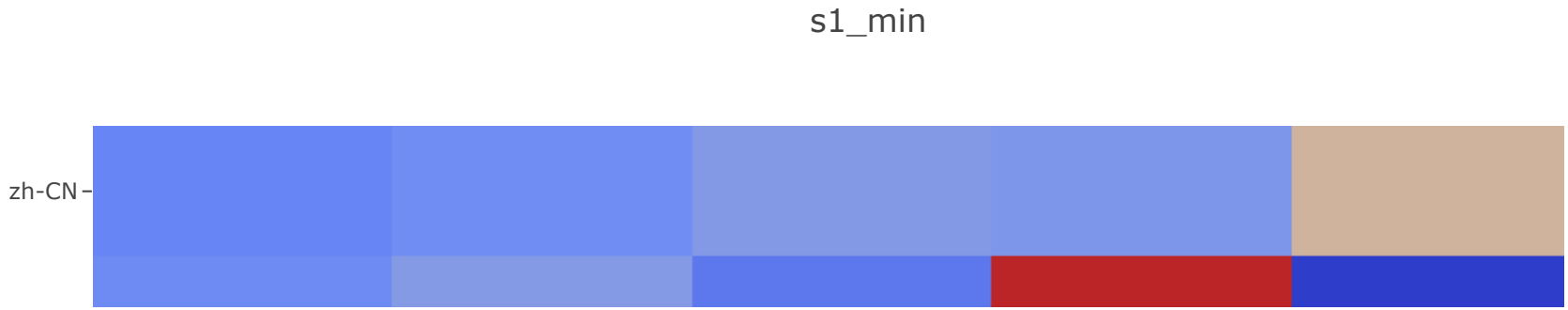


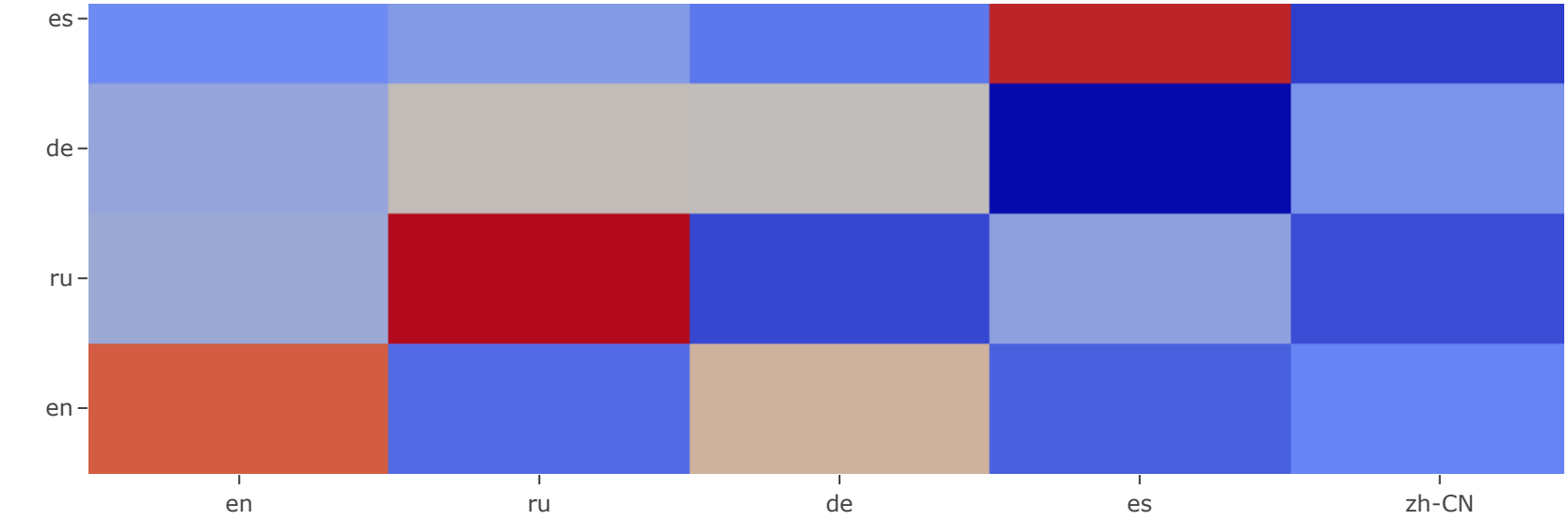


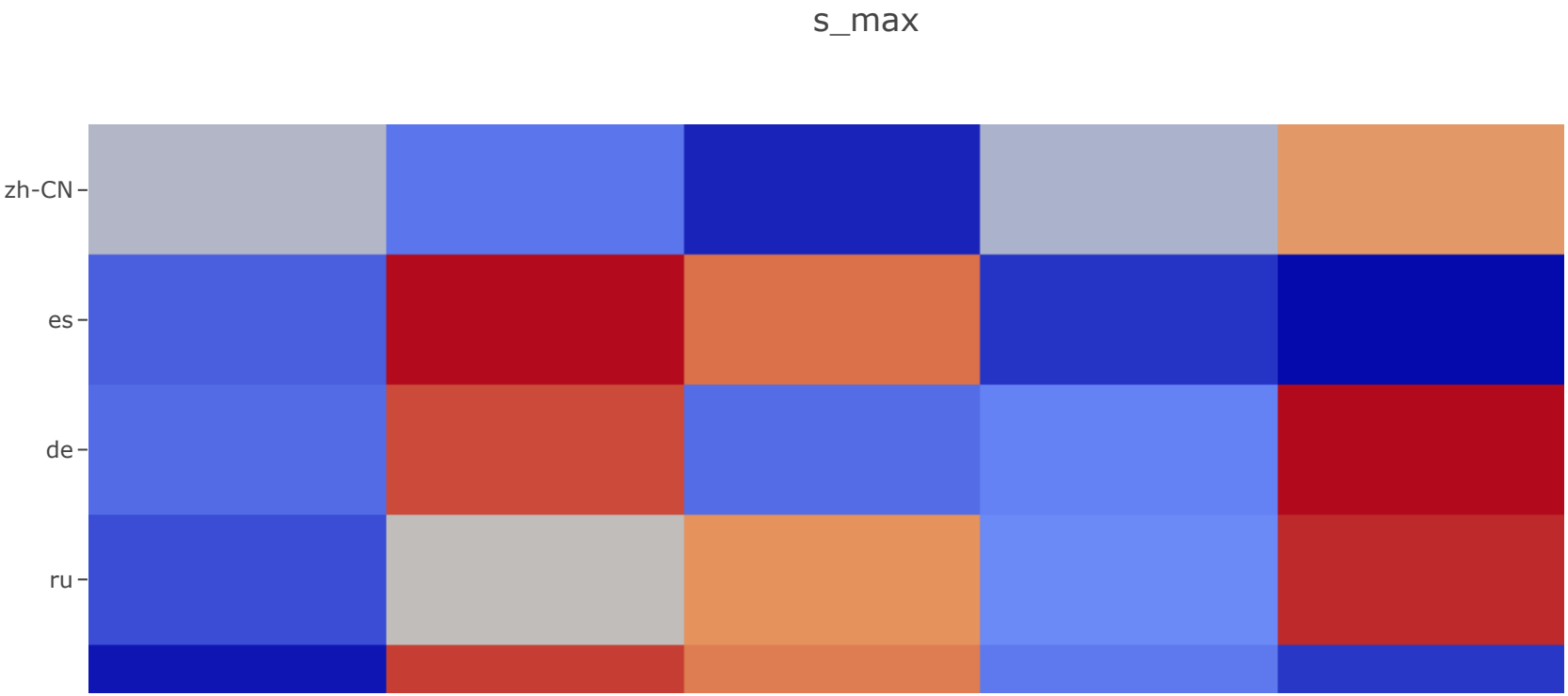
s_median



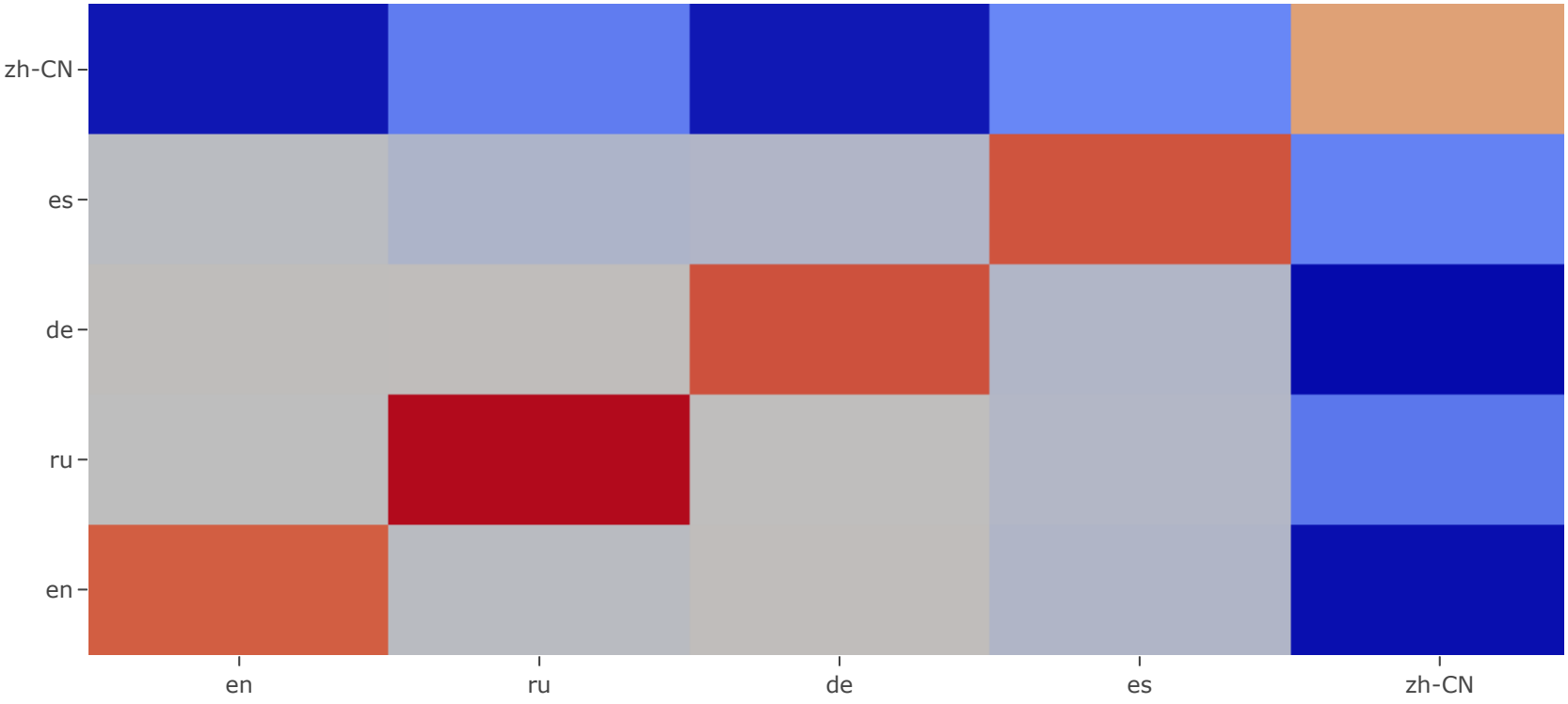

s_std
zh-CN
es
de
ru
en
en
ru
de
es
zh-CN


s1_min
zh-CN

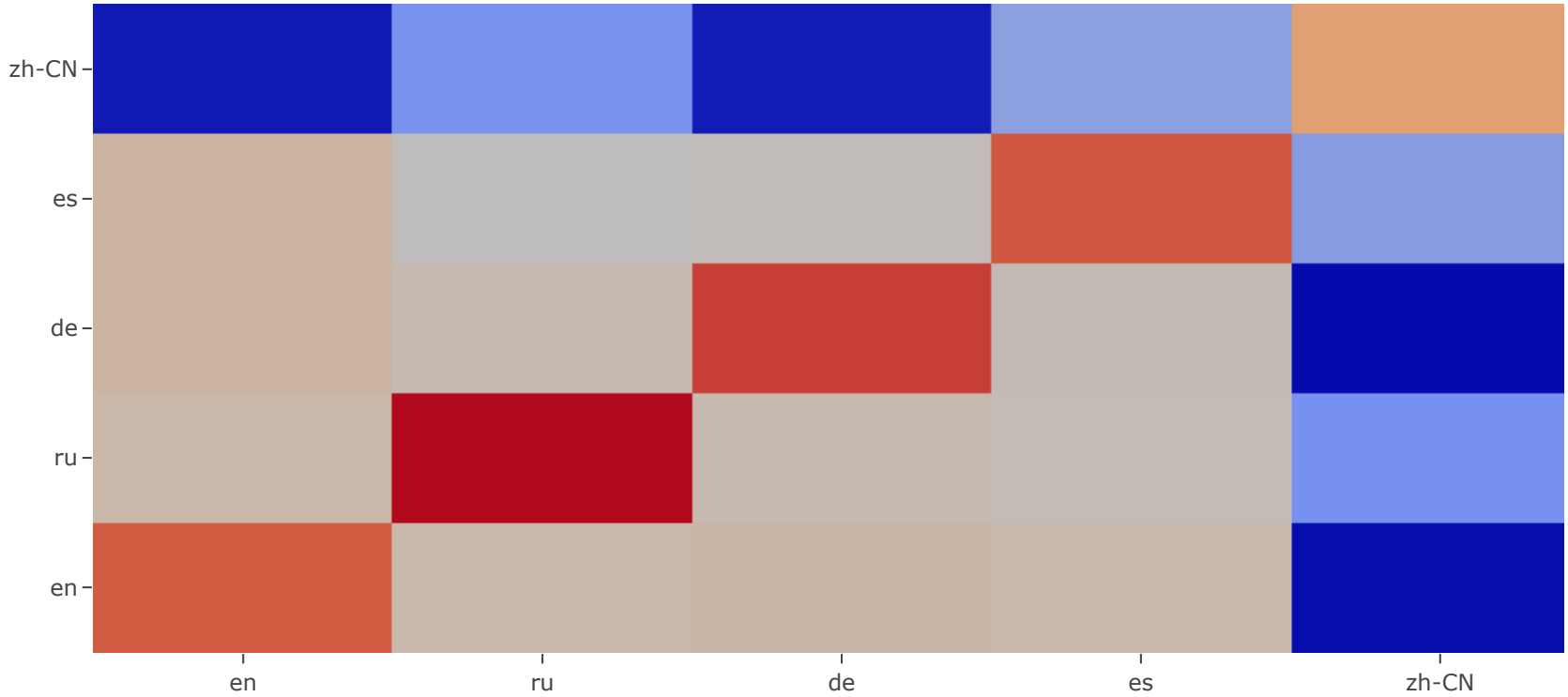

es
de
ru
en
en
ru
de
es
zh-CN

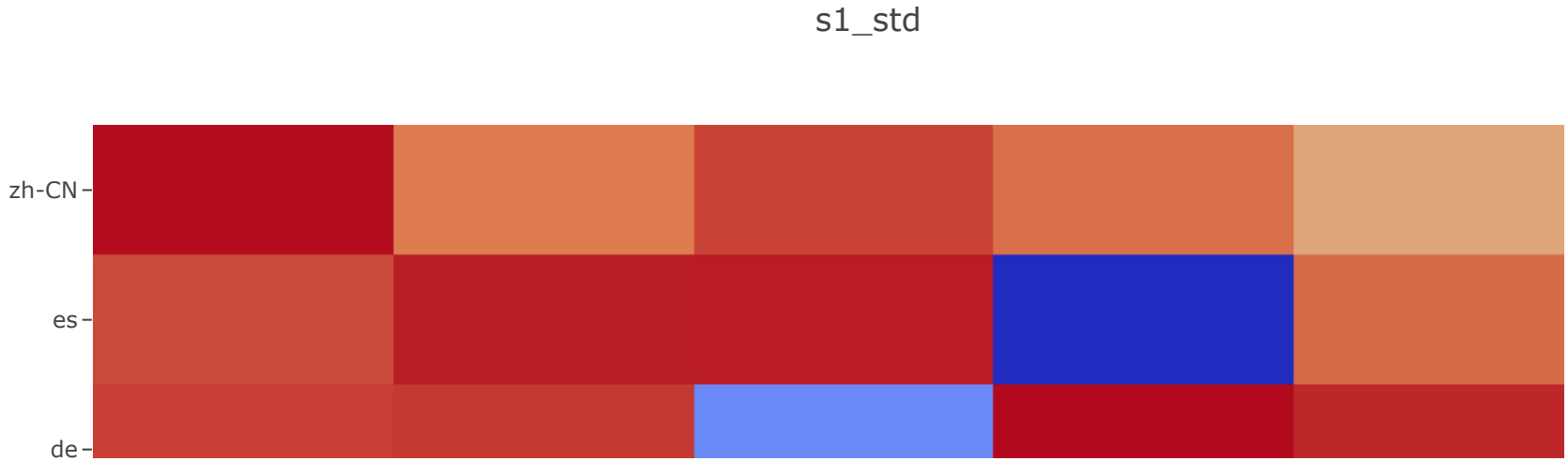

s_max
zh-CN
es
de
ru

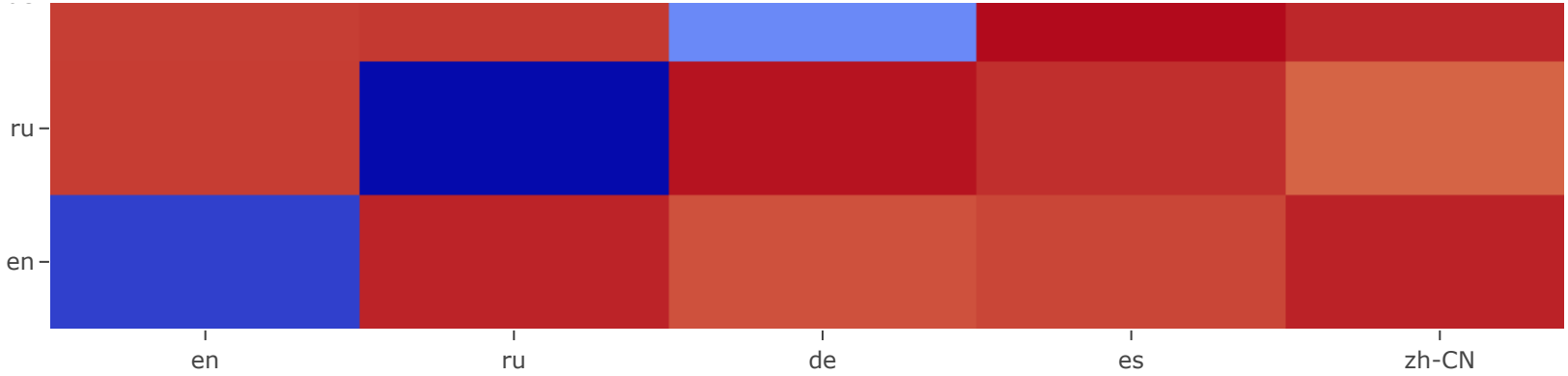
s1_mean



s1_median


zh-CN
es
de
ru
en
en
ru
de
es
zh-CN


s1_std
zh-CN
es
de



Inverted Matrices

In [11]:
```
if __name__ == '__main__':
    # Manually set list of translations (embedding, lg1, lg2)
    translations = [#('fasttext_random', 'en', 'ru'),
                    ('fasttext_top', 'en', 'ru'),
                    #('fasttext_random', 'en', 'de'),
                    ('fasttext_top', 'en', 'de'),
                    #('fasttext_random', 'en', 'es'),
                    ('fasttext_top', 'en', 'es'),
                    #('fasttext_random', 'en', 'zh-CN'),
                    ('fasttext_top', 'en', 'zh-CN'),
                    #('fasttext_random', 'de', 'en'),
                    ('fasttext_top', 'de', 'en'),
                    #('fasttext_random', 'de', 'es'),
                    ('fasttext_top', 'de', 'es'),
                    #('fasttext_random', 'de', 'ru'),
                    ('fasttext_top', 'de', 'ru'),
                    #('fasttext_random', 'de', 'zh-CN'),
                    ('fasttext_top', 'de', 'zh-CN'),
                    #('fasttext_random', 'ru', 'en'),
                    ('fasttext_top', 'ru', 'en'),
                    #('fasttext_random', 'ru', 'es'),
                    ('fasttext_top', 'ru', 'es'),
                    #('fasttext_random', 'ru', 'zh-CN'),
                    ('fasttext_top', 'ru', 'zh-CN'),
                    #('fasttext_random', 'ru', 'de'),
                    ('fasttext_top', 'ru', 'de'),
                    #('fasttext_random', 'zh-CN', 'en'),
                    ('fasttext_top', 'zh-CN', 'en'),
                    #('fasttext_random', 'zh-CN', 'es'),
                    ('fasttext_top', 'zh-CN', 'es'),
                    #('fasttext_random', 'zh-CN', 'ru'),
                    ('fasttext_top', 'zh-CN', 'ru'),
                    #('fasttext_random', 'zh-CN', 'de'),
                    ('fasttext_top', 'zh-CN', 'de'),
                    #('fasttext_random', 'es', 'en'),
                    ('fasttext_top', 'es', 'en'),
                    #('fasttext_random', 'es', 'de'),
                    ('fasttext_top', 'es', 'de'),
                    #('fasttext_random', 'es', 'ru'),
                    ('fasttext_top', 'es', 'ru'),
                    #('fasttext_random', 'es', 'zh-CN'),
                    ('fasttext_top', 'es', 'zh-CN')
```

```
                ]

    md = ''
    for translation in translations:
        embedding, lg1, lg2 = translation
        # Vocab/Vectors/Dicts
        lg1_vocab, lg1_vectors, lg2_vocab, lg2_vectors = \
            pickle_rw((lg1 + '_' + embedding.split('_')[0] + '_vocab', 0),
                      (lg1 + '_' + embedding.split('_')[0] + '_vectors', 0),
                      (lg2 + '_' + embedding.split('_')[0] + '_vocab', 0),
                      (lg2 + '_' + embedding.split('_')[0] + '_vectors', 0),
                      write=False)
        lg1_dict = make_dict(lg1_vocab, lg1_vectors)
        lg2_dict = make_dict(lg2_vocab, lg2_vectors)

        print('Translation: '+lg1+'->'+lg2+'\n')

        # Train/Test Vocab/Vectors
        vocab_train, vocab_test = vocab_train_test(embedding, lg1, lg2, lg1_vocab)
        X_train, X_test, y_train, y_test = vectors_train_test(vocab_train,
                                                              vocab_test)

        # Fit tranlation matrix to training data
        model, history, T, D, tf, I, M = translation_matrix(X_train, y_train)

        print('Inverse:'+str(I)+'\n')
```

```
Translation: en->ru

length of en-ru vocab: 6839
Determinant:6.68409e-20
Fr_norm:-0.0
Inverse:[[  11.04030704  -58.05263901  -70.07519531 ...,   25.41628647
    30.97108078  -40.67876816]
 [  91.81015015  155.81185913   63.31090164 ...,  -51.2548027   -34.41306305
    47.90806961]
 [  23.46608734   12.1979866   -52.79821777 ...,   23.93603897
    -7.66915035    5.51542711]
```

```
 ...,
 [ -95.40709686  -29.07827187   87.29953003 ...,  100.88738251  -16.7738266
   153.37922668]
 [-147.87860107  -81.68314362  -44.3213768  ...,  173.213974    -59.09873962
    59.42921829]
 [  16.56924438 -103.65172577 -142.10444641 ...,  -78.63152313
   117.04570007 -168.28567505]]

Translation: en->de

length of en-de vocab: 16693
Determinant:2.40427e-21
Fr_norm:0.0
Inverse:[[  755.14819336  -133.340271     206.62522888 ...,  1882.01904297
   -709.5010376  -1083.3581543 ]
 [  242.50422668   -23.07047462     5.77775288 ...,   960.47021484
   -322.10214233  -420.7321167 ]
 [ -431.28991699    97.79253387   -95.18783569 ..., -1025.21264648
    330.7355957    584.13464355]
 ...,
 [ -422.92025757    46.88298035    96.03134155 ..., -1551.60412598
    493.1998291    651.64038086]
 [  611.00964355  -180.95664978   202.88879395 ...,  1523.80834961
   -622.27716064  -950.3012085 ]
 [ -259.61535645    10.06036949   -96.83132935 ...,  -761.62390137
    190.62011719   434.51565552]]

Translation: en->es

length of en-es vocab: 10422
Determinant:-1.16279e-23
Fr_norm:0.0
Inverse:[[ -98.68668365 -260.83862305 -587.78167725 ...,  379.01580811
    57.62456131  647.65167236]
 [ -93.85202026   31.28058624  144.4175415  ...,  -42.15445328
     3.08430982 -151.31071472]
 [ 284.58007812 -404.34399414 -712.90319824 ...,  552.19470215
   -39.45856476  884.69006348]
 ...,
 [ 200.77197266 -179.4989624  -525.70465088 ...,  256.63024902
   -38.14390945  414.31124878]
 [-100.97593689 -223.18373108 -465.73358154 ...,  310.7088623    63.51074219
   427.6505127 ]
```

```
 [-123.85021973 -202.60536194 -432.22723389 ...,  316.81027222
    15.20920467  519.00701904]]

Translation: en->zh-CN

length of en-zh-CN vocab: 1460
Determinant:-0.0
Fr_norm:-0.0
Inverse:[[ 279.08325195  -50.90645218  174.09918213 ...,   90.67852783
  -125.23526001  138.2858429 ]
 [ 282.04492188  -24.62430573  276.77432251 ...,  248.19404602
   -95.89566803  -59.2610321 ]
 [  -3.75852799 -152.42700195   65.89362335 ...,  302.29708862
  -105.84304047 -232.65254211]
 ...,
 [ 211.2144165  -100.81359863  124.05513    ...,  331.4234314  -124.95433044
  -216.73834229]
 [-125.23327637  128.56265259   89.98539734 ...,  -53.60145569
    78.57273102   57.74006271]
 [ 179.14727783  -62.25080872  120.82265472 ...,  -73.63300323
   103.99213409   30.01247787]]

Translation: de->en

length of de-en vocab: 16693
Determinant:-2.15217e-17
Fr_norm:-0.0
Inverse:[[   1.84647417  -55.90779495  -43.97912598 ...,   39.81010818
   -39.84086227 -105.50863647]
 [  91.50335693  135.40769958  272.12155151 ...,  170.50970459
   -79.33867645 -129.79850769]
 [  -1.02397203   67.86239624   -5.41641092 ...,   94.55872345
  -108.43566895  -34.19537735]
 ...,
 [ -55.88882065  -73.20679474 -167.59153748 ...,  -99.45224762
    50.38871765   40.59305191]
 [  71.21019745   31.15022278  143.98403931 ...,   29.07103348
   -51.79080582  -67.52401733]
 [   1.01958323   -3.17144513   -0.39861795 ...,  -27.27108765   64.1984024
    79.6170578 ]]

Translation: de->es
```

```
length of de-es vocab: 11456
Determinant:4.61531e-23
Fr_norm:0.0
Inverse:[[-208.12774658  129.30685425  286.38415527 ...,  161.75546265
   -65.80812836 -153.16976929]
 [  47.54762268  -11.62705421   95.54808807 ...,  122.13251495
    95.22483063 -182.45480347]
 [ -68.93095398   29.17188644   10.01784801 ...,   80.02574921
   -23.01441765  -80.31826782]
 ...,
 [ -60.41200256   46.29036331   83.87097168 ...,  -34.00827026
  -115.82743073  -35.87575531]
 [-335.98999023   66.93195343  189.75874329 ...,   24.15080833
  -275.91699219   87.48773193]
 [   1.96587789  -25.36977577  105.56586456 ...,  206.55760193
    43.54348755 -296.44512939]]

Translation: de->ru

length of de-ru vocab: 5910
Determinant:-1.09467e-19
Fr_norm:0.0
Inverse:[[-151.40771484  -30.88466072   77.30550385 ...,   38.58583069
   128.15287781  -20.5597744 ]
 [ -38.43872452  -37.58361816  -96.61126709 ...,   37.91732788
   100.82440948   20.59609604]
 [-176.77554321   23.42058182   27.32201767 ...,  -75.23674011
   106.20679474 -273.70327759]
 ...,
 [-194.94673157   -2.72268939 -151.26535034 ..., -133.86204529
   145.88725281 -384.5640564 ]
 [  70.48829651  -24.49956322  -85.98475647 ...,  -49.60044098
   -56.12587738  -85.30728912]
 [-378.18856812    7.60398245  -20.77565765 ..., -214.16625977
    72.85224152 -715.15490723]]

Translation: de->zh-CN

length of de-zh-CN vocab: 1820
Determinant:0.0
Fr_norm:-0.0
Inverse:[[  182.77256775   -12.98318291     8.14419556 ...,   -44.43376541
     78.37240601   -77.56101227]
```

```
 [   80.42079163  -243.8031311   -268.67929077 ...,   112.42276764
     37.31299591  -114.50111389]
 [  780.19976807  -752.38586426  -606.61517334 ...,  -107.82647705
    144.76489258  -893.77630615]
 ...,
 [ -113.08462524   269.40768433   195.55578613 ...,   -34.58014297
   -143.97172546   110.33963776]
 [  214.31100464  -313.65765381   -16.46645927 ...,   -42.66078186
    139.57568359  -233.36196899]
 [  653.07806396 -1090.84179688  -776.70593262 ...,    27.33714294
    350.93087769  -820.984375  ]]

Translation: ru->en

length of ru-en vocab: 6839
Determinant:-1.14019e-20
Fr_norm:0.0
Inverse:[[-234.50318909   22.7103157   468.10519409 ..., -142.00621033
   224.19677734 -438.63296509]
 [  44.44205093 -110.10383606 -214.72457886 ...,   56.50108719
  -131.72332764   84.38804626]
 [ 288.88677979 -277.25476074 -490.5708313  ...,  175.72525024
  -196.50019836  467.42556763]
 ...,
 [-241.14767456  226.14082336  325.89718628 ...,  -45.59486771
   226.55795288 -221.37988281]
 [  -8.15202427  145.6502533    33.30801773 ..., -147.27935791
   -15.43331718 -206.29437256]
 [-225.9641571   250.87538147  396.88607788 ..., -133.33071899
   170.97853088 -433.68011475]]

Translation: ru->es

length of ru-es vocab: 5870
Determinant:1.06374e-22
Fr_norm:-0.0
Inverse:[[-138.00456238   -6.45686722   71.60140991 ...,  -36.70297241
    58.82127762  154.65583801]
 [  49.72098923  -52.75326538  104.88195801 ...,   20.12860298
   -74.70955658   40.26996613]
 [  -2.66938472 -147.09407043   98.62489319 ...,   85.64072418
    23.54691696  125.16918945]
 ...,
```

```
 [-118.95442963   13.25885868   91.0585022  ...,  -67.21045685
    62.11876678  115.23573303]
 [ -36.5912056    65.4972229    -3.35349679 ...,  -29.58674812   53.3549614
   113.96194458]
 [ -71.00119781 -125.49362183  112.67459106 ...,   49.41052246
    34.73974228  -78.90220642]]

Translation: ru->zh-CN

length of ru-zh-CN vocab: 1266
Determinant:0.0
Fr_norm:-0.0
Inverse:[[ 123.87176514  101.6381073   194.61276245 ...,  -42.12559891
    96.63561249  138.75669861]
 [-744.82489014 -704.24475098 -446.41647339 ..., -491.5687561   -63.25275421
   222.48565674]
 [  20.97368813   34.40427399  192.9274292  ..., -135.43104553
   119.09024811  102.5777359 ]
 ...,
 [ 195.69140625   27.52499962   45.47441101 ...,  102.32037354
     3.24064541  -41.37684631]
 [ 245.32792664  146.08032227  202.41160583 ...,   92.10801697
    32.16794586    6.01663494]
 [ 108.89143372  119.61774445  169.33660889 ...,  119.6232605    44.22268295
    93.17635345]]

Translation: ru->de

length of ru-de vocab: 5910
Determinant:-1.0791e-19
Fr_norm:0.0
Inverse:[[ 231.7795105   -53.77041626   35.14373016 ...,   41.9601593   -18.7463932
   138.1625824 ]
 [-185.92315674 -348.37637329 -128.53903198 ...,   -3.67792106
   113.00837708  -41.82150269]
 [ 274.20281982   89.54084778   52.89003372 ...,  130.93800354
  -106.85772705  159.0308075 ]
 ...,
 [ 195.22572327 -163.60261536   73.5228653  ..., -102.59391785
   -55.86291504   83.46475983]
 [  13.79369545  -42.29308701   -1.58617067 ...,   57.35632324
   -16.52289391   40.29460144]
 [  70.41092682  278.4899292   115.18925476 ...,  110.06312561
```

```
   -41.55050659   37.66139984]]

Translation: zh-CN->en

length of zh-CN-en vocab: 1460
Determinant:-0.0
Fr_norm:0.0
Inverse:[[  750.95532227 -2677.73120117 -2790.56835938 ...,  -500.21502686
   6477.76611328  -906.92370605]
 [ -856.35272217  2233.87646484  2544.17041016 ...,   560.26184082
  -5371.02587891   496.52096558]
 [-1010.84552002  2835.54589844  2717.16064453 ...,   713.1763916
  -6876.35498047   718.14050293]
 ...,
 [ -357.97131348   629.50219727   654.39550781 ...,   254.75791931
  -1516.64099121   115.86322784]
 [  610.44421387 -1544.09875488 -1381.81567383 ...,  -359.84481812
   3450.50390625  -447.62716675]
 [-1167.98925781  2801.60986328  2767.17675781 ...,   756.30444336
  -6591.30371094   868.70123291]]


Translation: zh-CN->es

length of zh-CN-es vocab: 1164
Determinant:-0.0
Fr_norm:0.0
Inverse:[[  17.54487991   23.9071846   -69.86971283 ...,   89.22011566
    67.72798157  108.96387482]
 [ -19.26591682  -58.13566971  -63.12947464 ...,   12.89937305  102.1413269
    64.83384705]
 [  53.17620087 -105.45384216  -46.82474899 ...,   29.39961052
   285.44174194   51.69458771]
 ...,
 [ -97.70675659 -172.0501709    11.54387856 ...,  -69.68674469
   -87.34493256  -99.78017426]
 [ 132.74069214  -67.03653717 -117.35790253 ...,   83.51582336
   427.69055176  193.97187805]
 [ -33.97097397  -28.83506012   35.82284927 ...,   70.83348083
   -68.67223358   97.35033417]]

Translation: zh-CN->ru
```

```
length of zh-CN-ru vocab: 1266
Determinant:-0.0
Fr_norm:-0.0
Inverse:[[ -37.29304123   18.05200195  120.03998566 ..., -134.6721344    97.88696289
    -2.2412765 ]
 [  81.02592468   48.94137955 -144.85075378 ...,   -4.63170481
   -69.93479919   10.25763702]
 [  61.70194244 -108.61361694  -82.55732727 ...,  -18.13778687
    68.86730194   25.89509964]
 ...,
 [  39.08507919  -40.19970322  -63.80712128 ...,   24.37394714
    10.56565571  -40.56039429]
 [ -98.20683289   39.0988884   -53.12154007 ...,   48.00292587
   -98.26872253   -1.27546012]
 [  12.94785118  -89.91964722   41.29096603 ...,   84.5228653    60.32333374
   -83.64056396]]

Translation: zh-CN->de

length of zh-CN-de vocab: 1820
Determinant:-0.0
Fr_norm:-0.0
Inverse:[[-167.89735413 -169.32441711  262.71853638 ...,  -49.06270218
  -218.04176331 -502.0098877 ]
 [-211.6706543   -78.65090179  313.97622681 ...,   24.95234871
  -260.29180908 -415.22360229]
 [-234.07113647  170.86058044  -28.66379738 ...,   49.25692368
   -20.01120186  285.25259399]
 ...,
 [ 909.95092773  -99.72676086 -110.36309052 ..., -330.12548828
   526.26983643  551.90856934]
 [ 923.55871582 -237.44372559  -63.26832199 ..., -316.19677734
   187.92474365   53.57709885]
 [  54.23075104 -414.93478394  291.21899414 ..., -158.23045349
  -240.38543701 -549.38092041]]

Translation: es->en

length of es-en vocab: 10422
Determinant:-8.86887e-23
Fr_norm:0.0
Inverse:[[  3.66905022e+01   5.63451538e+01  -4.43629608e+01 ...,   1.47553272e+01
   -2.23721695e+01   1.92639114e+02]
```

```
 [ -2.38301963e-01  -8.82871704e+01   1.55858719e+02 ...,   6.18005409e+01
    6.82027817e+01  -8.13018112e+01]
 [ -9.10500641e+01   5.23132744e+01  -4.51392250e+01 ...,   3.51174103e+02
   -7.27205811e+01  -9.86932297e+01]
 ...,
 [  2.95722256e+01  -2.66521339e+01   1.52041245e+02 ...,  -9.97995734e-01
    5.86561394e+01   1.26562370e+02]
 [ -2.66694126e+01   1.24087509e+02  -1.58618713e+02 ...,   1.02118347e+02
    1.51513977e+02  -4.62004089e+02]
 [  3.96808014e+01   2.40333366e+01   3.03186607e+01 ...,  -7.37969131e+01
    1.03059483e+01   2.31150391e+02]]

Translation: es->de

length of es-de vocab: 11456
Determinant:-1.67781e-22
Fr_norm:-0.0
Inverse:[[-211.0153656  -109.92762756 -235.15696716 ..., -462.22366333
  -298.97665405  -98.86308289]
 [  50.72947311   31.43056297    7.85100508 ...,   32.22510147
    61.28936005  -22.80888176]
 [ 153.84313965  146.11160278  214.22206116 ...,  406.88845825
   189.72543335   96.86721039]
 ...,
 [-216.62428284  -85.879776   -237.69984436 ..., -293.89746094 -273.0133667
  -253.99163818]
 [-192.18464661  -32.1150589  -150.17906189 ..., -291.48083496
  -243.19403076 -141.9500885 ]
 [  84.87358093  -22.03468704  134.11616516 ...,  305.0322876   173.19493103
    64.18505859]]

Translation: es->ru

length of es-ru vocab: 5870
Determinant:3.42516e-22
Fr_norm:-0.0
Inverse:[[ -36.88964081  -38.66049194  171.79954529 ...,  -23.12669563
   -56.12367249    6.87218428]
 [  53.82025909  212.07670593  -42.72406769 ..., -787.01470947
   112.25466919   61.76227188]
 [  -6.68660975  177.28474426   32.79327393 ..., -769.54364014
   125.00292969   -8.67141724]
 ...,
```

```
 [  38.5022583   150.08508301  -30.97211075 ..., -298.6656189   152.74972534
   -61.62663651]
 [  51.41888809  211.19586182  -60.89271164 ..., -822.25720215
   239.45413208  -25.49222755]
 [  19.69354248  128.50161743  -26.33551598 ..., -596.45941162
    98.39897156   11.74662304]]

Translation: es->zh-CN

length of es-zh-CN vocab: 1164
Determinant:-0.0
Fr_norm:-0.0
Inverse:[[  1.54408991e-01  -7.29956970e+01  -5.20634460e+01 ...,  -1.46622116e+02
   -1.52809097e+02  -3.85142250e+01]
 [  7.74740076e+00   3.60843201e+01  -7.10157700e+01 ...,  -4.23915291e+01
   -9.08426361e+01   3.14684830e+01]
 [  1.35625973e+01  -5.47755051e+01  -1.08091764e+01 ...,  -7.55519513e-03
   -1.25363716e+02   2.30807434e+02]
 ...,
 [  1.06153564e+02  -2.54409866e+02   1.49036392e+02 ...,  -4.00590363e+01
   -2.20805008e+02   1.89460587e+02]
 [  2.15465317e+02  -2.01847336e+02   9.14659595e+00 ...,  -8.39445953e+01
   -2.12597778e+02   1.59251205e+02]
 [ -4.10690651e+01   3.38989067e+01   4.17062912e+01 ...,   8.21285019e+01
   -1.81661606e+02   2.83516327e+02]]
```

Sanity Check on Determinants

In [19]:
```
if __name__ == '__main__':
    # Manually set list of translations (embedding, lg1, lg2)
    translations = [#('fasttext_random', 'en', 'ru'),
                    ('fasttext_top', 'en', 'ru'),
                    #('fasttext_random', 'en', 'de'),
                    ('fasttext_top', 'en', 'de'),
                    #('fasttext_random', 'en', 'es'),
                    ('fasttext_top', 'en', 'es'),
                    #('fasttext_random', 'en', 'zh-CN'),
                    ('fasttext_top', 'en', 'zh-CN'),
                    #('fasttext_random', 'de', 'en'),
                    ('fasttext_top', 'de', 'en'),
                    #('fasttext_random', 'de', 'es'),
                    ('fasttext_top', 'de', 'es'),
                    #('fasttext_random', 'de', 'ru'),
                    ('fasttext_top', 'de', 'ru'),
                    #('fasttext_random', 'de', 'zh-CN'),
                    ('fasttext_top', 'de', 'zh-CN'),
                    #('fasttext_random', 'ru', 'en'),
                    ('fasttext_top', 'ru', 'en'),
                    #('fasttext_random', 'ru', 'es'),
                    ('fasttext_top', 'ru', 'es'),
                    #('fasttext_random', 'ru', 'zh-CN'),
                    ('fasttext_top', 'ru', 'zh-CN'),
                    #('fasttext_random', 'ru', 'de'),
                    ('fasttext_top', 'ru', 'de'),
                    #('fasttext_random', 'zh-CN', 'en'),
                    ('fasttext_top', 'zh-CN', 'en'),
                    #('fasttext_random', 'zh-CN', 'es'),
                    ('fasttext_top', 'zh-CN', 'es'),
                    #('fasttext_random', 'zh-CN', 'ru'),
                    ('fasttext_top', 'zh-CN', 'ru'),
                    #('fasttext_random', 'zh-CN', 'de'),
                    ('fasttext_top', 'zh-CN', 'de'),
                    #('fasttext_random', 'es', 'en'),
                    ('fasttext_top', 'es', 'en'),
                    #('fasttext_random', 'es', 'de'),
                    ('fasttext_top', 'es', 'de'),
                    #('fasttext_random', 'es', 'ru'),
                    ('fasttext_top', 'es', 'ru'),
                    #('fasttext_random', 'es', 'zh-CN'),
                    ('fasttext_top', 'es', 'zh-CN')
```

```
                ]

    md = ''
    for translation in translations:
        embedding, lg1, lg2 = translation
        # Vocab/Vectors/Dicts
        lg1_vocab, lg1_vectors, lg2_vocab, lg2_vectors = \
            pickle_rw((lg1 + '_' + embedding.split('_')[0] + '_vocab', 0),
                      (lg1 + '_' + embedding.split('_')[0] + '_vectors', 0),
                      (lg2 + '_' + embedding.split('_')[0] + '_vocab', 0),
                      (lg2 + '_' + embedding.split('_')[0] + '_vectors', 0),
                      write=False)
        lg1_dict = make_dict(lg1_vocab, lg1_vectors)
        lg2_dict = make_dict(lg2_vocab, lg2_vectors)

        print('Translation: '+lg1+'->'+lg2+'\n')

        # Train/Test Vocab/Vectors
        vocab_train, vocab_test = vocab_train_test(embedding, lg1, lg2, lg1_vocab)
        X_train, X_test, y_train, y_test = vectors_train_test(vocab_train,
                                                              vocab_test)



        # Fit tranlation matrix to training data
        model, history, T, D, tf, I, M = translation_matrix(X_train, y_train)


        y_norm = translation_results(X_test, y_test, vocab_test, T,
                                     lg2_vectors, lg2_vocab)


        #print('Volume of X matrix: '+ np.prod(X_norm)+'\n')
        print(y_norm)
        print(T)
        print('Volume of yhat matrix: '+ str(np.prod(y_norm))+'\n')
```

```
Translation: en->ru

length of en-ru vocab: 6839
```

```
Determinant:2.03775e-19
[ 0.67061174  0.6873163   0.85272115 ...,  1.20578921  1.40582168
  0.93844253]
[[  4.34536341e-04  -1.01042946e-03   1.29217806e-04 ...,   2.51814048e-03
    2.25269631e-03  -3.78022756e-04]
 [ -2.70378281e-04   1.11204310e-04   3.61257058e-04 ...,   1.47847389e-03
    2.56023044e-03   3.06686998e-04]
 [ -7.82030053e-04  -2.08674697e-03  -3.18972440e-03 ...,   1.43284816e-03
    8.56383296e-04  -5.04361931e-04]
 ...,
 [  3.68732936e-03   1.84817414e-03   2.45620660e-03 ...,   4.04990744e-03
   -1.71988842e-03  -2.58150208e-03]
 [  9.60162492e-04   5.78932850e-05   2.89971940e-03 ...,   8.86483584e-04
   -2.09363061e-03  -8.01478338e-04]
 [ -3.22616356e-03  -2.79117911e-03  -1.80032675e-03 ...,   1.22218125e-03
   1.76331308e-03   7.65946519e-04]]
```

In [ ]: