A Method for Hebrew Stylometry, Based on Short Messages

Lev, Asi
ID 039833900
asi.lev@gmail.com

Jashek, Ronen
ID 025340993
ronen.jashek@gmail.com

Abstract

Given a set of short text messages in Hebrew, we attempt to classify each message[1] to its respective author, according to its style of writing, using supervised machine learning technique. We use Tweeter as our data set, and we've checked 10 users with approximately several thousands of tweets per user to train our model. We then use a different set of several hundreds of tweets per user in order to evaluate our model's accuracy. We've noticed that Support Vector Machine provides a slightly better accuracy for author identification than Logistic Regression, and using 17 style marker features was proved worse than a simple "Bag of Words" model. Trying to use more sophisticated models such as stemming and 2-grams, did not improve the accuracy either. We've also tested what single feature would best improve bag of words accuracy, and found that the amount of characters was slightly better than any other style marker.

1 Introduction

Stylometry is the practice being used to identify an author based on his style of writing. In today's vastly popular social media, establishing identity of writers can be useful in several ways - identifying anonymous talkback writers, identifying anonymous internet trolls, locating a person using collection of SMSs [2] in given area, identify frauds of chatbots and more. While stylometry has been used historically to identity authors of long texts such as novels and plays, legacy practices were proven inefficient when applied to short text messages such as SMS and social media posts such as Facebook, Twitter, etc. [Green and Sheppard, 2013] Recently, more novel approaches were suggested for stylometry that utilize algorithms in machine learning and deep learning, to improve accuracy of stylometry on short texts. These works rely on several learning models, such as Support Vector Machine [Green and Sheppard, 2013], [Schwartz et al., 2013]. These works also suggest different approaches to features selection. The study of feature selection may not be suitable to any language, as different languages may hold different features that may be suitable to a more accurate system. In this work we started an initial research and benchmarking on the influence of different learning models and feature-sets, in order to learn more about which of these models and features influence on the accuracy of a possible stylometry system for short messages *in Hebrew*. Source code and resources used for this work can be found on our github repository: https://github.com/asilev/NLP_FinalProject and on libsvm's site: https://www.csie.ntu.edu.tw/~cjlin/libsvm/ [Chang and Lin, 2011]

This document is arranged according to the following order: We begin by describing related work this article relies on in section 2. We then proceed by describing the architecture of our utility, that implements a framework for model benchmarking in section 3. After that, we describe our approach of research and benchmarking in section 4. We then describe our results in section 5. Further work is suggested in section 6. Finally, we conclude our work in section 7.

2 Related Work

There has been a significant amount of work and progress achieved in the past 10-15 years in the field of author classification and identification based on their texts, initially ranging from a few paragraphs to complete books. As the usage of emails in general, and social media in particular, became increasingly popular, significant amounts of work were invested in shorter texts. On longer texts, usually satisfactory-high results were achieved. [Qian et al., 2017] showed that deep learning models on authorship identification and authorship verification can yield an accuracy of 69.1% on Reuters_50_50 (Reuters data-set) and 89.2% on Gutenberg data-set (a data-set established by the authors), while Siamese network achieved a verification accuracy of 99.8% on both Reuters_50_50 and Gutenberg data-sets.

However, despite the vast amount of work invested, it

[1] We use the term "Message" as a single datum in our data set, as opposed to "Sentence", since in many cases, a single datum contains more than one syntactic "sentence".

[2] Short Message Service

is still challenging to achieve reasonably-high detection results on messages derived from the aforementioned social media (Twitter, Facebook, etc.), emails and such, especially as the number of authors increases; for a small set of authors (sometimes as low as 2) good results can be achieved. [Green and Sheppard, 2013] experimented with both BOW and Style Markers, and showed that SM is superior (average classification accuracy of 92.3% vs. 76.7%). However, this high classification rate was achieved on 2 authors, whereas increasing from 2 to 5 authors dropped the accuracy quickly (around 55%) up to the lowest accuracy of 40% reached on 12 authors. [Schwartz et al., 2013] experimented (among other criteria) with varying training set sizes and number of authors. They demonstrated an improvement in accuracy from 49.5% (50 tweets per 50 authors) to 69.7% (larger amount of tweets per 50 authors), but with a large number of authors (1000) showed 30.3% accuracy (using 200 tweets/author), which correlates to the trend shown in other works as well. Albeit, 30.3% is still distinguishably better than the random baseline of 0.1%. This trend is further validated in the work of [Brocardo et al., 2013], albeit their work was founded on emails from the Enron data-set (200K emails from 150 users, average number of words per email is 200). Their techniques were based on a combination of supervised learning and n-gram analysis. Even after applying a few pre-processing operations on the text (e.g. e-mails were combined to produce a single long message per individual, and then divided into smaller blocks used for authorship verification), their experiments yielded an EER (an Equal Error Rate metric used by the authors which corresponds to the operating point where False Acceptance and False Rejection rates have the same value) of 14.35% for 87 users for relatively small block sizes.

It is important to notice that thus far the related works described herein are mostly involved with messages in the English language. In an overall perspective, little work was done on Semitic languages such as Hebrew and Arabic. The work done by [Rabab'ah et al., 2016] on tweets in Arabic showed the best accuracy when employing SVM technique on a unified collection of features (more than 900) using tweets from 12 authors, usually around 3000 tweets per author. The best accuracy achieved was 68.67%.

An excellent and comprehensive overview of modern authorship attribution methods can be found in [Stamatatos].

3 Architecture

Figure 1 illustrates the architecture of the benchmarking platform we've built for this work.

We use "Data Source" as the abstract utility to pull text from short messages source, and save the messages with the correct tagged author for each short message in a well formed structure. In here we've implemented the "Data Source" with a "Tweeter Data Source", that pulls information according to parameters such as time period and a list of users from tweeter. This data is then being processed by "Data Normalization" utility, that reads this data, filters unwanted cases (as will be described in section 4), and transforms the data to a structure that can be used by YAP[3] [More and Tsarfaty, 2016]. We then use YAP to provide initial parsing for our data, and provide more information about it, such as morphological segmentation, stemming, POS [4] tagging, etc. Then, for each test, we choose our features for the test. The "Feature Model" takes a list of messages, and builds for each message a vector of features, where each feature is assigned with a value that describes that feature for the particular message. We create two lists of messages: one for the training (gold list) and one for the evaluation (test list). We've split our data set to two disjoint sets of messages, with 70% for training and 30% for evaluation. Then, for each run, we use a choice of "Learning Model" to train on the gold list, and produce the parameters that will be used for the encoder. Lastly, we use the encoder on the test list, and compare its results with the actual authors from the test list using the evaluation component, to test the system's accuracy. We use the fraction of correctly identified authors as the evaluation metric.

4 Benchmarking Approach

We now describe the benchmarking approach step by step, and describe how each step is implemented in our work.

4.1 Data Acquisition

We used the Twitter4j library [5] in order to extract the tweets from twitter. Twitter4j is an unofficial yet a very popular library for using the Twitter API. As we needed to get thousands of tweets per user, we used the "getUserTimeline()" API using paging (as Twitter API limits the number of tweets one can get in a single API

[3] Yet Another (Natural Language) Parser

[4] Part of Speech

[5] Twitter4J is released under Apache License 2.0, Copyright 2007 Yusuke Yamamoto. For additional information: http://twitter4j.org/en/index.html

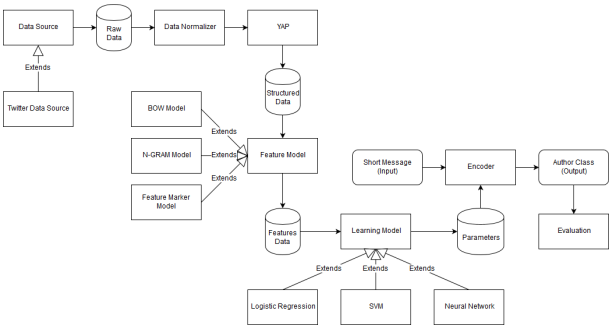

Figure 1: Platform Architecture

call). Overall we executed 50 user timeline page API calls, with 200 tweets for each page. For most users this ended up with about 3000-4500 tweets, but obviously some had fewer and one had significantly more (around 10.5K). See table 1 in appendix A for more details about the data-set.

4.2 Data Exploration

For this research, we've downloaded initially 85,869 tweets, from 20 different authors. Since we've seen that working with 20 authors is extremely slow to train and results are quite poor (also in other related works), we've downsized our data set to a more manageable size of 10 authors with 37,627 tweets. The average number of tweets per user is 3,763, and the standard deviation is 616. Going through parts of the data set, we've encountered some criteria that we consider not suitable to our research. The criteria is usage of languages other than Hebrew (at least one word is not in Hebrew), and retweets. After eliminating those cases, we've remained with 29,043 tweets. The average number of tweets per user is 2,904 and the standard deviation is 270. Table 1 in appendix A summarizes the statistics of the data we've used in our experiments after which being run through the steps depicted above, covering the authors (name and user ID), their area of work, the number of tweets we got using the Twitter4j API, and the number of tweets we ended up with after pre-processing (preparing the tweets in the format that YAP requires) and applying the YAP tool.

4.3 Data Preparation

In order to conduct our experiments we could not use the data we extracted via the Twitter API "as-is". The tweets we extracted directly from the Twitter API were in JSON format containing an abundance of information (content and meta-data) such as creation date, links, re-tweets, mentions, etc. This format not only includes more information than needed for the task at hand, but is also impossible to be processed by a morphological analyzer (such as YAP). As such, we needed to apply a series of normalization operations on the raw data, in order to be able to run YAP on it, and get the suitable output we needed to continue with the other steps in the model (see figure 1). We performed the following operations on the raw tweet data in order to prepare it for YAP morphological analysis:

1. Strip any meta-data information
2. Ignore/omit any re-tweets: As we're interested in the style of only the particular author, text from other authors can only impede the analysis
3. Remove English words and characters
4. Leave punctuation marks and numbers
5. Formatting: For each message, format it such that each word (or punctuation mark) is in its own line, and tweets are separated from one to another by an empty line.

The above steps generated a "normalized" input file, per author, ready to be processed by YAP. We then executed two YAP commands - morphological analysis and dis-ambiguity. The output from these operations was a file (per author) containing all the tweets after analysis, resulting in the following information for each tweet:

- **Original Word**
- **Stemmed Word**
- **Original Word POS**
- **Stemmed Word POS**
- **Gender**
- **Single or Plural**
- **Person Attribute (first, second, etc.)**

The process itself is not particularly language specific, but since writing style is a language specific feature, results may very compared to work on different languages. Also, in our work we are not checking specific words for identification, and instead we use morphemes as base units of a message, as Hebrew is rich with word internal morphemes.

4.4 Feature Selection and Benchmarking

There are several approaches to features selection. For example, [Green and Sheppard, 2013] compared two approaches, namely bag of words vs. style markers. While testing each of these approaches ourselves, we suggest a more generalized approach that also tests combinations of these approaches, by using some features as words from bag of words model, and some *additional* features as style markers. We also tested other approaches, such as N-Grams and POS analysis. The list features we used is:

- ***Simple Word (Bag of Words)***: Each feature is a word, and the value is the number of times the word appears in the sentence.
- ***Stemmed Word***: Each feature is the stem analysis of a word, and the value is the number of times the stemmed word appears in the sentence.
- ***Bigram***: Each feature is a sequence of two words, and the value is the number of times this pair of words appears in the sentence. We can further generalize this feature to any N-gram, but for a small corpus, this is proven as ineffective even for a bigram, since specific pair are very unlikely to reappear. On a larger training data-set, it is suggested to try testing with several N-gram models.

- ***POS Analysis***: Each feature is a word, combined with its proper POS tag. This helps separate different words that may be written the same way, but used for different meanings.
- ***Number of Morphemes***: A single feature, describing the number of morphemes the message holds.
- ***Average Word Size***: A single feature, describing the average size of a word in the message.
- ***Number of Characters***: A single feature, describing the number of characters in the message.
- ***Number of Punctuation Marks***: a single feature, describing the number of punctuation marks used in the message.
- ***Average Number of Punctuation Marks***: A single feature, describing the number of morphemes that represent punctuation marks, relative to the total number of morphemes in the message.
- ***POS Usage***: Each feature is a distinct POS, and the value is the number of times the POS appears in the sentence. We've only used the most common POS's, namely VB, NN, JJ, PRP, INTJ and CD.
- ***Average Sentence Size***: If the given message is divided to more than one actual sentence (using a dot, identified as full stop by YAP), this feature holds the average number of morphemes in each actual sentence in the complete given message.
- ***Number of Long Words***: Each feature represents the number of words in the sentences, that are longer than a given threshold. We've used the following thresholds: 3, 5, 7, 9.

4.5 Model Selection and Benchmarking

In this work we've compared two machine learning models, namely "Logistic Regression" and "Support Vector Machine". In our platform, Logistic Regression model was implemented from scratch, as it is a very basic model, and for SVM we've used libSVM[Chang and Lin, 2011]. We've compared Logistic Regression and SVM for a feature model that contains all style marker features combined. Initially, features were tested without "feature scaling". We tested each learning model for 2-10 authors. In logistic regression, we've used the following hyper parameters selection method: α was chosen by searching for the highest α that does not increase the value of the model's cost function. The value we've found ranges from 0.003 to 0.005, depends on the number of authors. The number of iterations was dynamically chosen, as the number of calculation that stabilizes the cost function on a change of $<$0.0001 per iteration. For SVM, we've chosen a polynomial kernel with 1 degree, as these hyper parameters were proven ideal for small cases we ran. For the C and γ hyper parameters, we've performed a grid search and ran the test on all combinations of values from the following list: We've used the following values for γ: (0.00003, 0.0001, 0.0005, 0.002, 0.008, 0.03125, 0.125, 0.5, 2, 8, 32, 128), and the following values for C: (0.03125, 0.125, 0.5, 2, 32, 8, 128, 512, 2048, 8192). For this test, we've seen that SVM provides slightly better results than Logistic Regression, and both models provide better results from a naïve guessing baseline. We can see the results in the following graph: Figure 2 illustrates accuracy evaluations of logistic regression, SVM, and naïve baseline. Since we've seen that logistic regression provides results that were less accurate comparing to SVM, we chose to use SVM as our model for further research. We've now checked if scaling the features can improve our results for SVM, for Style Marker and for Bag of Words. This check was conducted for a data set of 10 authors. For Style marker, scaling features provided an accuracy of 21.81%, while not scaling features provided a slightly better results of 22.42%. We also tested SVM feature scaling with simple Bag of Words feature set. Non scaled simple Bag of Words provided accuracy of 43.89%, and scaled simple Bag of Words provided 43.59%. We conclude that feature scaling is useless for both style markers and bag of words. Next, we turn to test other feature sets. These models were all tested with 10 authors data set, SVM model, unscaled, and the hyper parameters that were found to provide best accuracy in the regular Bag of Words model: C=8 and γ=0.03125. We've checked the following feature sets:

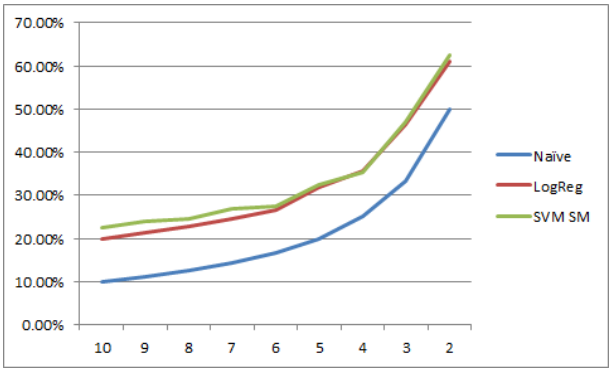

Figure 2: Logistic Regression, SVM, and baseline

- ***Bag of Stemmed Words (BOSW)***: Instead of using the original morphemes, we've used the stemmed version.
- ***Bag of Bigrams (BOBW)***: Each feature is a pair of subsequent words. Start of sentence and end of sentence symbols added to each sentence.
- ***Bag of POS-Words (BOPW)***: Each feature is a morpheme, along with its Part of Speech.

This comparison showed that the most simple bag of words provides the most accurate results, as can be seen in figure 3. Lastly, we consider the question what

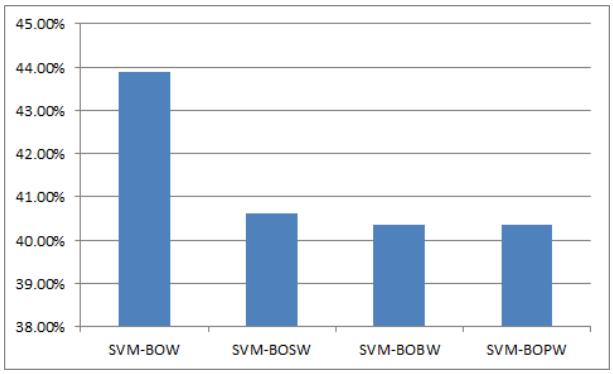

Figure 3: Feature Comparison for 10 authors

would be the best (single) style marker set that most contribute as an addition to bag of words in our model. We've tested each style marker feature set combined with the bag of words feature set on our 10 authors data set, and concluded that the number of characters is most likely to contribute to the accuracy of bag of words, as illustrated in figure 4.

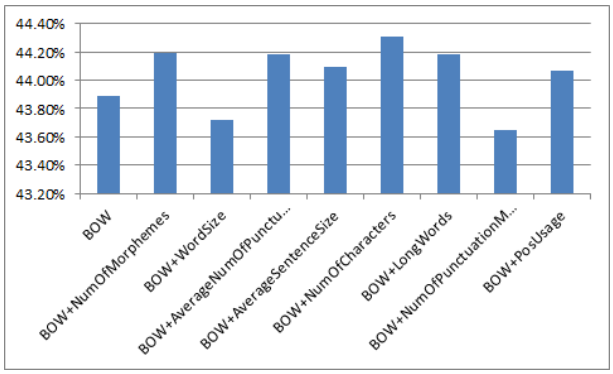

Figure 4: Single Feature-set Contribution with Bag of Words for 10 authors

5 Results and Discussion

Comparing Logistic Regression with Support Vector Machine learning models with style marker feature sets yields a slightly better results for SVM. This may show that style is hard to identify, and it is better to use a model that separates the classes better. Scaling the features before training and evaluating was proven useless. When it comes to type of feature sets, a simple bag of words model was proven better than more sophisticated models. We assume that the low number of tested features in the feature style model is the reason this approach was not accurate enough. A combination of bag of words with a single style marker feature showed only limited improvement, and the full case of style marker features was not tested. Trying to figure the most helpful single style feature that can contribute to the Bag of Words model again resulted with a non sophisticated number of characters result. This shows us that the number of characters is a good distinguisher of identity, and that it is loosely dependent on the choice of words. Table 2 in Appendix B shows the confusion matrix of our best model (Bag of Words with Number of Characters). We can see that according to the matrix, all authors had more hits for themselves than for any other author. We also see that some authors in our data set are much more distinguishable than others. For example, Guy Rolnik was correctly identified as himself 71.48% of the time, While Tal Schneider was identified was correctly identified as herself only 28.38% of the time.

6 Further Work

Due to time constraints and low CPU/GPU power, and the long time it takes to train our models, we've left many interesting cases still unresolved for benchmarking. We've noticed that the number of authors, and the number of features used, slowed down the training exponentially, and running training could take from few minutes to days and weeks on our platforms. We haven't checked the Logistic Regression model with Bag of Words feature set, or any other words based feature sets. We tried finding best kernel for SVM by working on small data sets, and projecting our results to larger sets. This may not be true, and checking other kernels directly on larger data sets may improve results. We haven't checked more sophisticated learning models, such as neural networks and deep neural networks models. It would be also interesting to check more feature rich combinations, and check the contribution of specific features to the overall accuracy. This requires training on CPU/GPU intensive machines, with perhaps more concurrency capabilities within the models. Also, this research used small data sets, where it is easy to gain much more data from publicly available social networks publications. Specifically, it can be interesting to check on more areas of work of the tested users. Looking at the relation between the number of messages a user has and the accuracy of the user's predictions, we see a sharp possible curve that may tell us that even by adding not too much data, we may improve accuracy greatly. This relation is illustrated in figure 5. Hebrew has some unique style based features that can be used as basis for identification, such as letter repetitions, deliberated typos and slang usage. Since our corpus was taken from the tweets of journalists, these unique style based features were not seen. We could use for journalism a sentiment analysis approach, and we suppose that this approach can help achieve good improvement, and can also be good idea for further research. Lastly, Using the confusion matrix, we can further investigate what makes our model better, and what makes it fail.

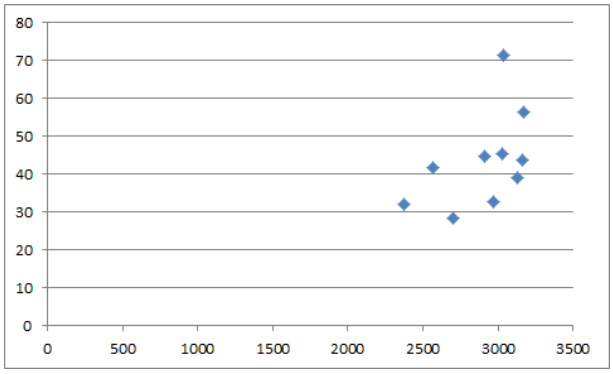

Figure 5: Number of Tweets / Accuracy relation

7 Conclusion

Our attempt to research the capabilities of machine learning with NLP methods for Hebrew stylometry with Hebrew driven methods such as morphological analysis and disambiguation POS analysis provided results similar to those of works on the English language, without those Hebrew driven methods. Still, our model's accuracy was quite the same as those of work performed on English based data-sets, and we assume that with better models and more training our model can provide even better accuracy.

References

M. L. Brocardo, I. Traore, S. Saad, and I. Woungang. Authorship verification for short messages using stylometry. In *2013 International Conference on Computer, Information and Telecommunication Systems (CITS)*, pages 1–6, May 2013. doi: 10.1109/CITS.2013.6705711.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

R.M. Green and John Sheppard. Comparing frequency- and style-based features for twitter author identification. pages 64–69, 01 2013.

Amir More and Reut Tsarfaty. Data-driven morphological analysis and disambiguation for morphologically rich languages and universal dependencies. In *Proceedings of COLING 2016*, december 2016.

Chen Qian, Ting He, and Rao Zhang. Deep learning based authorship identification. 2017.

A. Rabab'ah, M. Al-Ayyoub, Y. Jararweh, and M. Aldwairi. Authorship attribution of arabic tweets. In *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pages 1–6, Nov 2016. doi: 10.1109/AICCSA.2016.7945818.

Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891. Association for Computational Linguistics, 2013. URL http://www.aclweb.org/anthology/D13-1193.

Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556. doi: 10.1002/asi.21001. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21001.

Appendices

A Data Set

User Name	User ID	Area of Work	Number of Tweets	Number of Tweets after Pre-Processing and YAP
Ben Caspit	bencaspit	Political journalist	4136	2912
Nadav Eyal	NadavEyalDesk	Political journalist	4391	2971
Amit Segal	amit_segal	Political journalist	3552	3160
Ayala Hasson	AyallaHasson	Political journalist	3143	2569
Guy Rolnik	grolnik	Financial Journalist	4639	3032
Tal Schneider	talschneider	Political Commentator	4141	2700
Alon Ben-David	alonbd	Military Journalist	3568	3024
Rino Zror	RinoZror	Political journalist	3824	3173
Or Heller	OrHeller	Political journalist	3693	3128
Baruch Kra	baruchikra		2540	2374
Total	**10**	**N/A**	**37,627**	**29,043**

Table 1: Data Statistics

B Confusion Matrix

	Ben Caspit	**Nadav Eyal**	**Amit Segal**	**Ayala Hasson**	**Guy Rolnik**	**Tal Schneider**	**Alon Ben-David**	**Rino Zror**	**Or Heller**	**Baruch Kra**
Ben Caspit	44.82%	5.18%	8.08%	2.78%	3.66%	5.30%	3.41%	20.58%	1.89%	4.29%
Nadav Eyal	15.49%	32.75%	10.92%	4.93%	4.58%	5.75%	6.10%	11.74%	3.29%	4.46%
Amit Segal	9.56%	7.22%	43.78%	4.89%	10.22%	4.89%	2.78%	8.44%	2.00%	6.22%
Ayala Hasson	15.15%	4.97%	10.18%	41.70%	4.97%	3.15%	1.45%	13.58%	0.73%	4.12%
Guy Rolnik	5.77%	3.22%	5.88%	1.66%	71.48%	4.11%	1.55%	3.88%	0.89%	1.55%
Tal Schneider	12.64%	11.90%	12.02%	4.46%	4.96%	28.38%	3.72%	15.37%	2.35%	4.21%
Alon Ben-David	12.31%	4.71%	6.21%	5.46%	1.82%	2.03%	45.40%	7.82%	11.35%	2.89%
Rino Zror	19.20%	2.84%	4.80%	4.41%	3.72%	3.43%	0.88%	56.51%	1.96%	2.25%
Or Heller	10.49%	2.80%	4.90%	4.00%	5.09%	1.90%	16.58%	12.09%	39.16%	3.00%
Baruch Kra	13.91%	4.95%	12.67%	6.96%	9.27%	4.79%	2.01%	10.66%	2.63%	32.15%

Table 2: Confusion Matrix