Continuous User Identification

Mohammad Reza Mahmoudian Motlagh

Master's Thesis
Master of Science in Information Security
30 ECTS
Department of Computer Science and Media Technology
Gjøvik University College, 2015

Continuous User Identification

Mohammad Reza Mahmoudian Motlagh

May 2015

Abstract

The objective of this work has been to investigate the possibility of implementing a Continuous Identification system using Behavioural Biometrics, taking advantage of the high acceptability and low cost offered by this method. The Behavioural Biometrics chosen for this work is Keystroke Dynamics. Continuous Identification is proposed to be performed after a user is locked out in a Continuous Authentication system, utilizing the same keystroke dynamics.

Three features were considered when extracting the keystroke dynamics: duration, latency, and relative frequency of each keystroke action. The data was also categorized with respect to software context. Two distance metrics, namely Manhattan and Euclidean, were implemented and compared both via a Mean To Mean and via a One To Mean method. Two score fusion methods were utilized: weighted average mean with fixed weights and with variable weights. The analysis also included the effect of different data chunk sizes, simulating the number of actions before a user is locked out.

Th best results obtained was rank-1 identification accuracy rate of 60% after 50 actions and 72% after 1000 actions, when using duration-latency combination, with no categorization, Mean To Mean comparison, and Manhattan Distance. This setting was then applied on the results from a Continuous Authentication system. To the best of our knowledge, this work is the first to address Continuous Identification based on Behavioural Biometrics.

Acknowledgment

I would like to thank my supervisors Soumik Mondal and Patrick Bours, whose great support during the entire time has been indispensable to me. I especially thank Patrick Bours for introducing me to the very interesting subject of Behavioural Biometrics.

I would also like to offer my gratitude to my lovely family for their love and support throughout my life.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Topics Covered by The Project

In the past few decades, there has been a striking development in various areas of information technology. Computerization has revolutionized different aspects of life and information management is one of the areas which has been under notable changes and developments.

Inevitably, with these developments the need for security has also increased. Threats to confidentiality, integrity and availability of information [2] require different information security mechanisms in order to mitigate them. When it comes to security, one of the challenging areas that demand attention is access control. It is defined as :

> *"The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner"*[1]

Generally, to gain access to information resources, an authentication mechanism is performed. That is, a process of verifying if a claim for genuineness is true or not. There are different methodologies for implementing an authentication system. These are mostly classified into *Static* or *One-time* authentication, and *Continuous* or *All-time* authentication.

The simplest and most frequently used authentication mechanism is a static user name and password scheme. However, the password can be revealed, forgotten or stolen. Biometrics is one of the areas that can provide higher levels of security by taking advantage of uniqueness and permanence. Biometric Identification is a process of determining who is claiming to be the genuine user by means of utilizing Biometric characteristics of that person. These can be described as *Physiological* and *Behavioural* characteristics of a human. Examples for Physiological Biometrics are fingerprint, iris, face etc and for Behavioural Biometrics are the way a person types on a keyboard known as keystroke dynamics, mouse dynamics, gait etc.

In Biometric Identification, the identity of the person who claims to be the legitimate user is found out. Biometrics can be used for both Static and Continuous authentication. The advantage of Continuous Authentication over Static Authentication, as suggested by its name, is that the user is identified in a continuous manner in order to find out who he or she is.

1.2 keywords

Biometrics, Continuous Authentication, Biometric Identification, Biometric Authentication, Keystroke Dynamics

1.3 Problem Description

A Continuous Biometric Authentication (CA) system monitors the access control to a system constantly.

[1]RFC4949, page11, access control, part4

A user of a system is usually first authenticated by a Static Authentication process such as a username/password scheme or a Static Biometric Authentication system. A Static Authentication is a one-time authentication performed at initial log on to the system. In situations where the user leaves the computer device, there is a possibility of impersonation by an impostor user.

To avoid this, one way is to continuously monitor the user's behaviour such as typing rhythm [3] or the way he uses the mouse [4] etc. through a CA mechanism. At any time, the current user's behaviour is compared to the stored behaviour or template. A level of trust is defined which is continuously adjusted based on this comparison. With any impostor action the trust decreases and with any genuine action it increases. At some point, after a number of impostor actions the trust level falls under a certain limit and the user is locked out. To increase the accuracy of the model, it is recommended to use two or more Biometric factors so the authentication will be based on all these factors. The question that arises here is the identity of the locked out user.

It might happen that the genuine user is locked out by mistake. But more importantly, identifying the impostor user can play a significant role in forensics investigations of a cyber crime. Therefore, in addition to the CA process, there is need for a new concept of Continuous Identification process to identify the user who was just locked out. This master thesis will propose a way of designing and implementing a Continuous Identification system based on keystroke dynamics. To identify the impostor in a synchronized manner with a CA system, there is need to integrate the identification system into the CA system.

1.4 Justification and Benefits

In case of any impersonation attack on a system, identifying the attacker can be a great aid to the forensic investigation. In a CA system, an impostor access can be found out and the impostor user is locked out of the system, prohibiting him from attempting more illegal actions.

This helps the system to achieve an additional level of protection from such attacks and also if the account holder is the genuine user he will not be subject to defamation anymore. In addition, the use of Behavioural Biometrics, along with the high security characteristic of Biometrics, offers a more acceptable and cost efficient solution. It is acceptable due to the fact that all the authentication process is anonymous and does not require active interaction with the user. And, the general property of most Behavioural Biometrics is that there is no need for additional costly hardware, which makes this method cost efficient.

The Biometrics used in this work is the keystroke dynamics. The result of this research can boost the progress in practical use of CA in order to provide a more secure, low cost, accurate authentication solution.

1.5 Research Questions

1. Is it possible to implement a Continuous Identification system using behavioural characteristics of a person? If yes then:
2. To what extent can this system be implemented?
3. What Behavioural Biometric features can be utilized in such a system?
4. How is such a system integrated into a CA system?

1.6 Planned Contribution

The planned contribution of this research is to investigate the possibility of implementing a Continuous Identification module based on keystroke dynamics. The CA system is a keystroke dynamics based system and is based on the ideas mentioned in [5] and [6]. To design the identification system, it must be taken into account what keystroke features are required. Also the effect of software context should be considered, i.e. the software used by the user when the keystroke actions were taking place.

2 Biometrics in Authentication and Identification

2.1 Biometrics

Biometrics are unique features that make us perfectly distinguishable from other humans. Biometrics are instances of us and we can be recognized by them. Biometric Recognition, in a simple form , can be defined as recognizing an individual based on his/her Biometric characteristics.

The biggest advantage of using Biometrics over the other two traditional methods mentioned above is fraud resistance. It can be much easier to impersonate a person by stealing their password or their identity card rather than impersonating them by forging or mimicking their Biometric features.

There are two types of Biometric characteristics: Physiological and Behavioural. *Physiological* Biometric characteristics are those related to the physiological characteristics of a human being such as fingerprint pattern, iris pattern, facial features and DNA. *Behavioural* Biometrics are the characteristics related to behaviour of an individual. The way an individual talks (speech), walks (gait), types on a keyboard (keystroke dynamics), works with mouse (mouse dynamics) are some examples of Behavioural Biometrics.

Every Biometric feature must contain some properties to be considered for authentication. These properties [7] are,

- Universality: It exists for every individual
- Uniqueness: It is unique in every individual
- Permanence: It remains unchanged overtime
- Collectability: It can be quantified
- Performance: When used in recognition process, this process can be performed fast and produce results with high accuracy.
- Acceptability: Level of acceptability by people
- Circumvention: It is related to the fraud resistance

Table 1 is adopted from [7] and represents the degree of strength in each of the seven properties for some of the Biometric modalities.

Although all Biometric characteristics used for authentication have the properties mentioned above, the degree of strength in each property varies between the them. There are some modalities who have a high level of some property and are low in another. For example, Signature has a 'high' degree of circumvention which means it is easy for an impostor to forge a signature. However, it has a high degree of acceptability; probably because people are more willing to give their signature sample compared to e.g. their DNA sample.

Modality	Universality	Uniqueness	Permanence	Collectability	Performance	Acceptability	Circumvention
DNA	High	High	High	Low	High	Low	Low
Ear	Medium	Medium	High	Medium	Medium	High	Medium
Face	High	Low	Medium	High	Low	High	High
Fingerprint	Medium	High	High	Medium	High	Medium	Medium
Gait	Medium	Low	Low	High	Low	High	Medium
hand vein	Medium	Medium	Medium	Medium	Medium	Medium	Low
Iris	High	High	High	Medium	High	Low	Low
Keystroke	Low	Low	Low	Medium	Low	Medium	Medium
Retina	High	High	Medium	Low	High	Low	Low
Signature	Low	Low	Low	High	Low	High	High
Voice	Medium	Low	Low	Medium	Low	High	High

Table 1: Comparison between the Biometric modalities based on the Biometric properties

2.2 Biometric System

A Biometric system is defined as: "A system for the purpose of the Biometric recognition of individuals based on their behavioural and biological characteristics."[1]

There are 4 components in every generic Biometric system [8]:

- **Sensor**: For Biometric sample (data) collection such as fingerprint, iris, etc.
- **Feature Extraction**: For extracting Biometric features from the collected data
- **Comparison**: Extracted features are compared with the features stored in the database and a score/s is obtained
- **Decision-making**: Based on the comparison score, a decision is made on acceptance or rejection of an identity, or in case of identification the identity is built. [8]

Every Biometric system consists of ’at least’ the above components and works in at least one of the following three modes at a time:

- Enrollment
- Authentication
- Identification

[1]ISO/IEC 2382-37, pt, 37.02.03, First edition, 2012-12-15

2.2.1 Enrollment

In enrollment mode, a Biometric sample is collected through a user interface or sensor. Then a quality checking process performed where the quality of the sample is verified and the data is preprocessed in order to remove excess parts. Next, the features are extracted and they constitute a user's *template*. The template is then stored in the database of templates. An extra component called *Quality Checker* is involved in this process.

Figure 1 [7] illustrates a block diagram of enrollment process in a typical Biometric system.

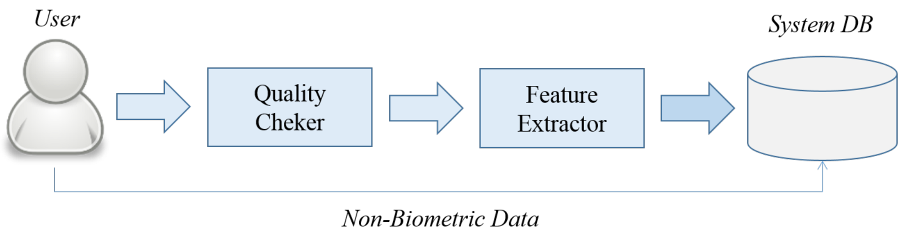


Figure 1: Enrollment process of a Biometric system [7]

There can be two types of enrollment, *Positive* and *Negative* [9]. *Positive* enrollment is a process where identities of authorized users of the system are created. These templates can be used later for authentication and positive identification of the users eligible to use the system. On the the other hand, *negative* enrollment are used to build the identity of the users, unauthorised to use the system [9].

2.2.2 Authentication

Biometric Authentication is a process of verifying an identity through Biometric characteristics. In authentication mode, a Biometric sample is collected through the sensor, pre-processed to remove any excess parts of it. From the pre-processed data, features are extracted. Then, through a *one-to-one* comparison algorithm, extracted features are compared with the features stored in the template belonging to the actual identity. Comparison process usually results in a score which represents the degree of similarity between the presented sample and the template of the actual identity. A threshold is defined for accepting or rejecting the claim to that identity. Finally, based on this score and the threshold, a decision is made on accepting or rejecting the claim [7].

Figure 2 is a block diagram of authentication process in a Biometric system.

There are possibilities where a false identity is accepted *(false acceptance)* or a true identity is rejected *(false rejection)*. The reason for a false acceptance or rejection could be a performance issue in some part of the authentication system. Hence, the performance of a Biometric Authentication system can be measured as *False Acceptance Rate (FAR)* and *False Rejection Rate (FRR)*. *FAR* is defined as ratio of number of authentications with identity claims falsely accepted to total number of impostor authentication attempts. *FRR* is defined as ratio of number of authentications with identity claims falsely rejected to total number of genuine authentication attempts [10].

Some research [11] provide with the system performance using FAR and FRR, whereas, some research [12] prefer to use *Equal Error Rate (ERR)*. It is defined as proportion of FAR to FRR at

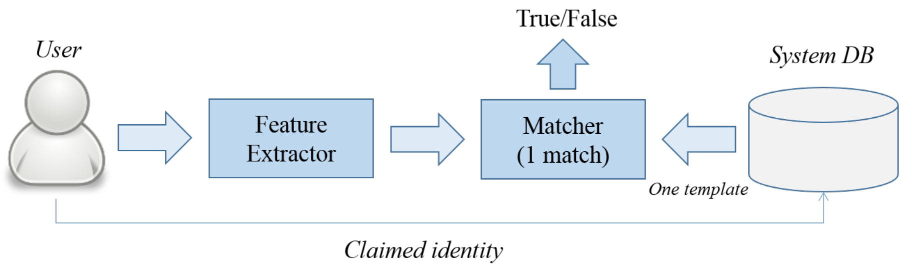


Figure 2: Authentication process of a Biometric system [7]

an operating point on ROC curve where FAR=FRR [10], that is the point of intersection of FRR and FAR curves on ROC.

2.2.3 Identification

Often, there is a confusion when using the terms *authentication* and *identification*. These two terms are frequently used as synonyms in many documents. But, they actually differ from each other. Authentication is "to verify an identity" [1]i.e. is this the person who he claims to be? Whereas, identification means to "prove one's identity" [1] i.e. who is this person? In identification, after sample collection, preprocessing and feature extraction, a *one-to-many* comparison is performed. That is, the sample of unknown user is compared to every template in the database. With every comparison a score is obtained. Based on the decision criteria e.g. the template with minimum comparison score [13] is selected as the *identified* template. There can also be a threshold for identification and if none of the scores fall below this threshold, a *not identified* verdict can be made. Figure 3 represents a block diagram of identification process in a Biometric system.

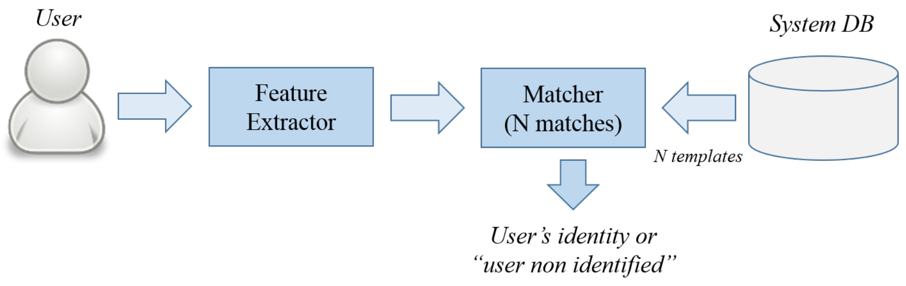


Figure 3: Identification process of a Biometric system [7]

Performance of an identification system can be determined by a metric called *Identification Rate* or *Identification Accuracy Rate*. It is defined as proportion of number of successful identifications to total number of identifications. Most of the research on Biometric Identification [14], report their performance using Identification Accuracy Rate.

[1]`http://www.oxforddictionaries.com`

2.2.4 Multi-modal Biometrics

There are many issues related to the accuracy and performance of uni-modal systems i.e. systems using only one Biometric characteristic. These issues include, the sensor capturing issues, overlapping of feature spaces among multiple users, spoof attacks etc [15].

These issues can be resolved by utilizing of *multi-modal Biometric systems*. A multi-modal Biometric system is the system that operates using either of the following or combinations of them:

- Multiple sensors such as two fingerprint sensors.
- Multiple Biometric characteristics such as keystroke and mouse.
- Multiple processes such as representation, comparison algorithms, etc. [16]

As explained in the previous section, there are 4 main components in a basic Biometric system. In a multi-modal Biometric system however, another component is added called *fusion*. Fusion process can be performed at various levels:

- Feature level
- Score Level
- Desicion level [17]

Figure 4 represents a multi-modal Biometric system which utilizes two types of Biometric factors: face and fingerprint. Fusion at all three levels are shown:

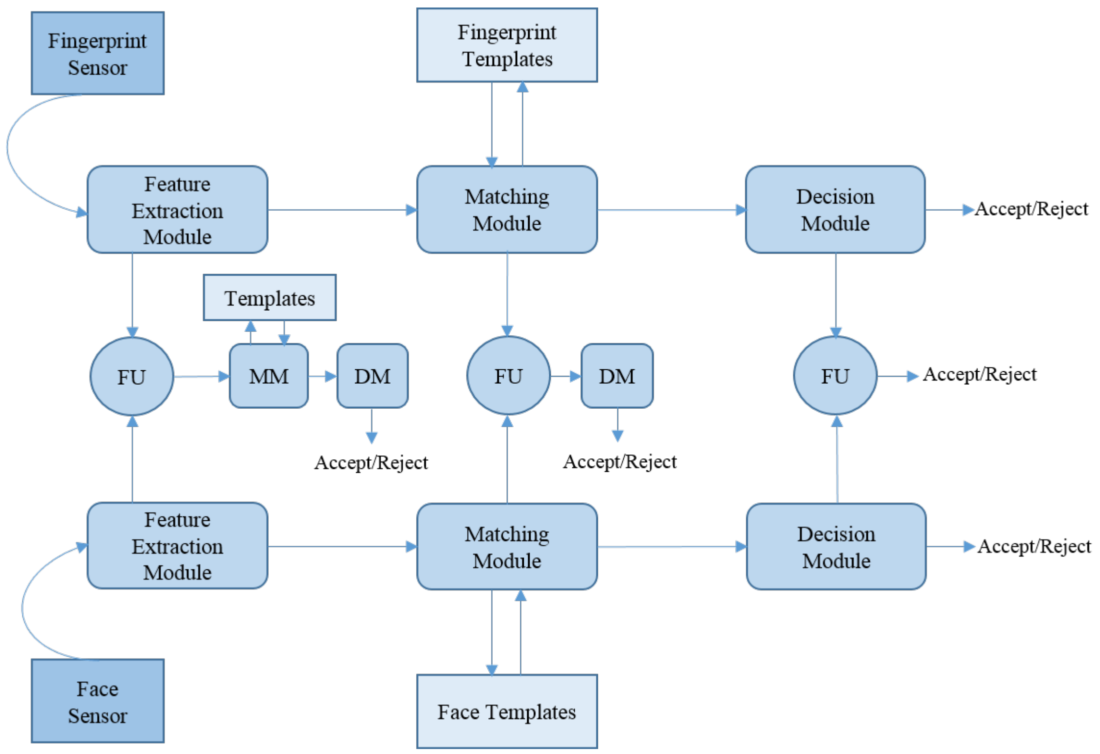


Figure 4: A multi-modal Biometric system with three levels of fusion [17]

Fusion at feature level includes combining of the features extracted from multiple Biometric characteristics and combine them into a single feature set. Fusion at score level, includes combining the scores obtained from separate comparison(matching) processes. Fusion at decision level is defined as combining the multiple decisions obtained as a result of multiple authentications into one [17].

3 Related Work

In this chapter, an overview of the related work is given in order to present the current state of the art. We shall review the past research on the Continuous Authentication(CA), Continuous Identification (CI), Keystroke Dynamics (KD) and methodologies used based on our research questions. Since the focus of this work is on Behavioural Biometrics we will emphasize more on Behavioural Biometrics and more specifically KD.

3.1 Keystroke Dynamics

Keystroke Dynamics is a Behavioural Biometrics which describes a person's typing rhythm [18]. One of the earliest research on application of KD in authentication was in 1980. By hypothesizing that telegraph operators have unique tapping styles, Gaines [19] performed a research in order to investigate the possibility to authenticate people by the way they type. Since then, there has been much research [20] [21] [22] [23] [24] [25] on this topic.

A person's typing can be characterized by various features obtained from the collected information such as timing information including time of pressing and releasing, amount of pressure, etc. Most of the data used in KD authentication is based on timing information. The two basic time based features are called *Duration*, which is the time between pressing down and releasing a key, and *Latency*, which is the time from releasing a key and pressing down the next key [3]. Other timing information include, *Up-Up* time which is the period between releasing a key and releasing the next key and *Down-Down* time which is the period between pressing a key and pressing the next key. However the most popular features are based on durations and latencies. There can also be other features such as the amount of pressure on the keys [21] or the features adapted from duration and latency such as duration of digraphs i.e. two simultaneous characters or trigraphs i.e. three simultaneous characters [26].

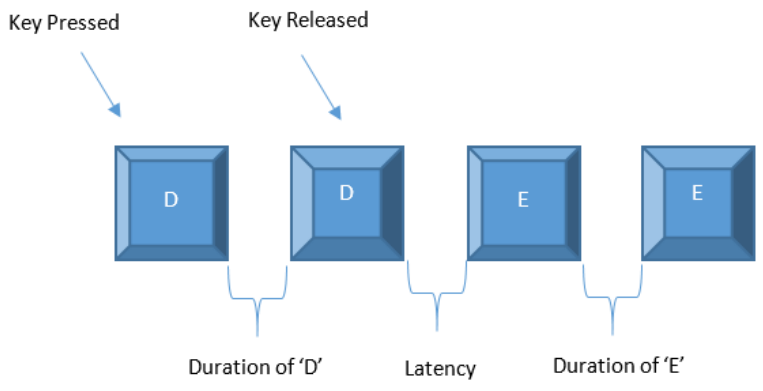


Figure 5: Keystroke Durations and Latency

3.1.1 Static Vs Free Text

There are two types of KD when it comes to type of input: *static* and *free text* [27]. *Static* input means the user input to the authentication process is fixed. The input is the same for every user such as a fixed user name and password for participant. Many works have considered fixed text input [28] [29] [18].

Clarke et al. [29] performed a research on authentication using numerical key paths on mobile phones. They collected a set of fixed PINs and numerical passcodes data from 16 participants. Using Neural Networks classifiers for the performance analysis they obtained an EER of 5.5% for 4 digit PIN and 3.2% for 11 digit passcodes.

In a work by Maxion et al. [18], focused on user authentication on mobile phones, authors analyzed data of 28 participants. Subjects were each asked to type a fixed 10 digit code 50 times. Choice of the fixed input was depending on various factors such as the positions of the digits on the keyboard and positions on the corresponding code. Using Random Forest classifiers they obtained an EER of 1.5%.

There also many works based on free text. In the analysis based on the free text, the input varies for everyone, so the input text typed is unexpected. The input text varies for every one and the user is free to enter anything. Analysis based on free text is mostly performed for a periodic or a CA [10]. Because, in a CA, a real-time analysis of keystrokes is performed. Hence, the next action of the user, that is the next input, is not predetermined. The text being dealt with is therefore a free text.

Gunetti et al. [11] believed that using short static text is not a good way of keystroke analysis. Because the timing information can be best analyzed when the frequency of typed characters is higher, thereby, achieving a higher performance rate. They performed a research on keystroke analysis with free text of 800 characters long on average. They defined two measures called A for measuring the degree of similarity and and R for measuring the degree of dissimilarity. According to [11] these two measures are complement of each other and perform the best when combined. Hence the best result they achieved when combining these two measures was an FAR of approximately 3.2% and an FRR of approximately 0.02%.

Monrose et al. [14] performed a research on KD authentication. They used both free text and static text data. Fixed data comprised of a set of words was given to the participants and in addition, participants could also type their own words. Using weighted probability classifiers they got their best performance with 90% correct recognition when comparing static input with static text template. When a static text was compared to free text, they got a 44% correct recognition rate and when authenticating using free text they got 23% of correct recognition rate. However, these performance rates were affected by various factors such as uncontrolled data collection environment and discarding around 50% of the collected data as a result of an outlier removal process.

In a research on keystroke analysis of free text in an uncontrolled environment, one of the factors is the type of the keyboard. Kang et al. [12] performed a research on keystroke authentications of long text on various keyboards such as a traditional keyboard, soft keyboard and mobile phone keyboard. They collected keystroke data of 35 people. Data consisted of long text inputs of more than 3000 characters. They found out that increasing the text length improves the performance as they had an EER of 5.6% when performing analysis with 1000 character text length on traditional keyboard compared to 24.1% when performing the same analysis with 100 characters.

One of the challenges regarding Behavioural Biometrics, especially KD, is that the user's behaviour is from time to time affected by various environmental factors such as noise, type of the hardware[25] and emotional factors such as stress [30]. Therefore, if for instance, a genuine user is stressed his typing cadence is also affected by this stress and if this situation occurs at a period of verification the user may be wrongly detected as impostor causing a *false* rejection.

With this assumption, several works[25][4][5] has been performed in an *Uncontrolled* environment, where there is no control on how the user types, the time, the place, his health conditions, hardware used etc.

3.2 Continuous Authentication and Identification

Authentication is not only limited to the login time where verifying an identity is done once only, i.e. Static Authentication. Static Authentication is usually performed at initial login or re-login processes, as opposed to *CA*, which is the way of constantly verifying an identity.

Similar to Static Authentication, CA can be performed using both physiological such as face [31] [32] and ECG [33] and behavourial modalities such as KD [3] and mouse dynamics [4] [34]. There have also been research which have used multi-modal Biometrics for CA citeSim2007 [35]. However, for some modalities such as fingerprint and iris, the process is very difficult. The reason is simply the collectability issues in a continuous manner regarding such modalities. Also, the cost of additional hardware is high. Hence one of the merits of Behavioural Biometrics i.e. the cost efficiency [36] makes behavioural modalities a better choice for CA.

In one of the earliest attempts to continuously verify the user's identity based on KD, Shepherd [36] introduced a continuous keystroke authentication system based on the timing information obtained by examining some typing characteristics of a user such as duration and latency. The possible features he introduced such as key pressure, typing error rates etc made his work a baseline for future research.

There are two different definitions of a CA system in terms of continuity. One definition is a system where the authentication is performed at every fixed period of time e.g. 1 minute actions e.g. after every 500 actions or based on other criteria such as *after a silence period* [37].

Zheng et al. [37], used this definition for implementing a CA system based on mouse dynamics. The system would verify the user identity in their defined unit of "one block" which is a series of number of move and click events. A problem with their proposed system was that the total time for verification was proportional to the length of block or more specifically the number of mouse movements and clicks in a block. Thus the length of block could be large thereby increasing the verification time.

Monaco et al. [23], introduced a "burst" CA system that would verify the user with a few keystroke actions after a period of silence. They believed that the chance of hijacking the session is higher after a pause period. Because this is usually the time when the actual user is not working at his station and is busy somewhere else. They obtained an EER of 1% when analyzing only 14 samples.

Ahmed et al. [25] tried to continuously authenticate users based on single key duration and digraphs. One of the challenges in when it comes to free text is the problem of missing entries where has not been used by the user. In order to solve this problem they used Neural Networks as prediction classifiers to predict these values based on the other keys recorded [25]. They obtained an FAR of 0.0152% and FRR of 4.82%

The above definition of CA in fact refers to periodic or a discrete authentication since authentication is performed between fixed periods and not within each period. Hence, if for instance, authentication takes place every minute or after 500 actions there might be a possibility of attacks during that one minute or before 500 actions since some impostor actions can be done within seconds [31].

3.2.1 Bours and Mondal Model

A more meaningful definition of CA in term of continuity, was first introduced by Bours [3]. Here the authentication is performed with every action by the user. This action can be a simple button press on a keyboard. In his paper Bours defined an action as a single key press and release. That is after every key press and release the authentication takes place [3]. Based on this definition, Mondal and Bours [4] modeled a CA system based on mouse dynamics. They used different features of the mouse such as acceleration, direction etc.

A CA system was designed by Bours and Mondal [5] based on the Bours definition and with a use of KD. In this system, a user is authenticated based on every keystroke action he performs. That is, there are comparisons on every user's action and the actual logged in user's template. If after several actions and based on the comparisons, the users characteristics are found to be very distant from the logged in user, he will be recognized as an impostor user and locked out of the system. To define the lock out criteria, a concept of *Trust* was defined. Initially a trust value would be set to the maximum, indicating the maximum trust of the system to the current user. Every action causes a change in the trust value where impostor actions reduce the trust and genuine actions either increase the trust value or if at maximum trust, it will remain at maximum. [5]

To measure the performance of this system two metrics called *Average Number of Impostor Actions (ANIA)* and *Average Number of Genuine Actions (ANGA)* were introduced [4]. ANIA is defined as the average number of actions that can be performed by an impostor before he is locked out. ANGA is defined as average number of actions that can be performed by a genuine user before he is locked out (falsely). The aim is to keep ANGA high so that a genuine user is never or very seldom locked out and keep ANIA to the lowest value possible so that an impostor is locked out as soon as possible.

This definition of CA provides a higher accuracy by lowering the decision making criteria to a more detailed level, and security by means of verification as early as possible. Because of the confidentiality, cost efficiency and anonymous monitoring nature indicating a good acceptability, CA has a great potential to be used as a complementary security measure to the Static Authentication techniques. Our assumed CA system to perform this research has been based on this model.

3.2.2 Continuous Identification

An important question is now: *who is the user just locked out?* To answer this question a new concept called *CI* is introduced. It can be defined as a Biometric Identification process where a locked out user is identified. Unlike CA process which is performed with every action, a CI is done only when a user is locked out.

Most of the Biometric Identification systems are static [38] [39]. However, There have been few attempts to identify a person based on his/her continuous biometric data. But, these works have been all based on physiological biometric modalities such as electrocardiogram (ECG) signals [40]. To the best of our knowledge, this work is the first to address CI based on Behavioural Biometrics.

4 Data Description

4.1 Data Collection

The Data set consists of data of 51 people, collected within 5 to 7 day. They were of different age, gender, profession and academic background and nationality. But all the participants were above 18 years of age. The environment where the data was collected was *uncontrolled* [6]. They could be anywhere, for example at home and do any activity using mouse and keyboard at any time. To collect the data we used a software called BeLT which was developed at GUC [1]. Participants were asked to perform their usual daily tasks during a 5 to 7 day period. The only request from them was to run the BeLT only when they are using the computer themselves. Therefore, in case somebody else was going to work with the same computer they could pause the program and resume as soon as they started to work again. However, we actually don't know if they have done the same.

4.1.1 BeLT Software

Behavioural Logging Tool (BeLT) is a GUI based Windows application which captures keystroke, mouse, and software interaction events as well as hardware events. Collected data can be saved offline on the client's computer or transmitted to a remote server. BeLT does not store sensitive data such as passwords [6]. We give a brief description of the BeLT functionality and the produced output format.

The main objective for development of BeLT is that there is a need for a logging tool that records Keystroke,mouse,software interaction and hardware events all together in a synchronized manner [1]. Most of data collection software such as RUI [41], MouseTrack [42] or WIDAM [43] work only with one of these modalities.

BeLT is a fast and a user friendly software. When started, it runs in the background and does not need any specific interaction unless it is required to pause or stop it. A *session* is defined as the period the BeLT is started until it is stopped or user loges of or shuts down the computer.

BeLT architecture consists of different modules: Graphical User Interface (GUI), data capturing module, data processing module, update service module and transmission module. Since explaining these modules in details will take us far from the scope of this research, we will only emphasize on some features of the BeLT and the collected data format. Due to the fact that we did not consider hardware and mouse events, we thus, ignore these two and describe the others. When BeLT is started, any event related to the hardware, software, keystroke and mouse actions is recorded simultaneously. Recording Time stamp is according to ISO-8601 and sampling period is 16 milliseconds [6].

Generally, for every event regardless of its type the following information are registered by BeLT:

- **Event ID**: A unique ID for each event in that session

- **Event Type**: Type of that event
- **Action**: Action performed in that event
- **Value**: Input value
- **Time**: Time of occurrence of that event
- **Relation**: Event ID for the parent event related to this event
- **Flag**: Depends on type of the event
- **Additional field**: If an event was repeated

We shall now describe the above properties with respect to each event type:

Keystroke Events

Table 2 shows the format for keystroke events recorded by BeLT. Each row indicates a different Event Type.

Event ID	Event Type	Action	Value	Time	Relation	Flag	Additional Fields
n	K	D	value	T	Event ID	flag	-
n	K	U	value	T	Event ID	flag	Count

Table 2: BeLT CSV format for KD events [1]

Event ID field is global and is independent of the event type. *Event type* is always 'K'. Generally there are two types of Keystroke events: *Key-Down* and *Key-Up*. Hence, *action* field is either 'U' indicating Key-Up or 'D' indicating Key-Down. *Value* field contains name of the pressed key. *Time* field contains the time where the key is pressed or released depending on the *action*. *Relation* is the field containing the Event ID of the parent event. If the action is 'D' then the *relation* is the software interaction Event ID under which the key is pressed. For every 'U' event there is an associated 'D' event since Key-Up event occurs as a result of a Key-Down event. Therefore, if the *action* is 'U', then the *relation* is the Event ID for the associated Key-Down event related to this Key-Up event. *Flag* field indicates if any combination keys have been used. Flag was a decimal equivalent of a 6 bit binary string. From right to left, bit 0 to 6 would signify *alt*, *ctrl*, *shift*, *Windows*, *CapsLock*, *NumLock* and *ScrollLock* respectively. If any of these keys were active at the time a key event occurred, the bit for that key would be set to 1, otherwise 0. This would give the binary string different values. For example in case of using only *alt*, the string would be 000001=1 and if *alt*+ *shift* was used, then it would be 000101=5. Finally, the last field contains additional information such as if the key was pressed and held, what is the equivalent number of keys when that key was pressed and released repeatedly. However, this option is available only for Key-Up events. [6].

Software Events

The second type of the event we are going to look at is the software event. Table 3 shows the format for software events generated by BeLT.

Event ID	Event Type	Action	Value	Time	Relation	Flag	Additional Fields
n	S	Type	Process name	T	Event ID	Elem type	Elem desc/ID/rectangle

Table 3: BeLT CSV format for Software events
[1]

Similar to Keystroke events, first element is the Event ID. An 'S' in the second field indicates that this is a software event and the third field indicates the event type. There are various types of software events [1] such as:

- **Object Change State (OCS)**: Change in the state of an element such as checking a radio button
- **Focus Changed (FC)**: When a software window or any other element is focused on
- **Visual Change (VC)**: When a software window is minimized, maximized or restored
- **Element Invoked (EL)**: When a button is pressed or any other trigger is fired.
- **Menu Opened (MO)**: A menu is opened or another menu item is focused on
- **Text changed (TC)**: Text is changed
- **Menu Mode Started (MMS)**: Viewing a menu for the first time
- **Window Opened (WO)**: A window is opened

Next field contains the process name for that software. Element ID contains the Event ID for the element that caused this software event to occur. For example, if clicking on a link which is a mouse click event has caused this software event to occur the the Event ID for that mouse event would be stored. Flag field's value depends on the type of software event. It can indicate the state for that element, or the type of an element according to Microsoft Control Type Identifiers (CTI)[1]. Similarly, the last field can also take different values. It can take the *Element Description* i.e. the purpose of the element, *Element ID* which is a unique number, identifying that element and *Rectangle* indicating the screen coordinates for that element. [1]

4.2 Pre-processing

In this section, the pre-processing of raw data before extracting features will be described. Raw files are the CSV files generated by BeLT . As explained before, the environment procedure for data collection was uncontrolled. Therefore, the amount of collected data varied between every participants.

4.2.1 Categorization

One of the objectives of this work was to see if the software application context affects the identification results. Hence, in addition to test with the overall data, we also decided to include

[1]`http://msdn.microsoft.com/en-us/library/windows/desktop/ee671198%28v=vs.85%29.aspx`. last accessed 12-05-2015

in the analysis the type of the software under which the user had typed the data. Initially, we created 6 categories, namely: browsing, chat, programming, documenting, gaming and unidentified applications. Unidentified category is the one where keystrokes belong to an unidentified software.

However, since the experiment was performed in an uncontrolled environment, there was a risk that there would not be sufficient amount of data collected for all the categories. Hence, we combined some of these categories and the following categories were finally obtained:

- **Internet**: Consist of keys typed when using browsers and chatting.
- **Documentation**: Consist of the keys typed when using word processors and other documentation software.
- **Others**: Consist of the keys typed when gaming, programming and using unidentified applications.

From the raw data, only keystroke events and Focus Change *FC* software events were required for the purpose of this work. As explained before, software *FC* event would be logged if focus was on that software window. Thus, any key-Down event would have that *FC* Event ID in it's relation field. This would indicate that any key typed, would be in that software context. For example, if there was an *FC* event related to *skype.exe* (belonging to *Skype*) with event_ID=12, then any Key-Down event *x* with *relation*=12 would indicate that *x* have been typed in the *Skype*.

By omitting other events such as mouse and hardware and the rest of software events, the pre-processed file would consist of software *FC* events followed by key stroke events in the same format as the raw file.

4.3 Feature Extraction and Template Creation

Before extracting features, software events had to categorized in one of the three categories mentioned above. Finding the category was a manual process. We extracted software process name from the value field of the software events in all the pre-processed files. Then, we searched the web to find the software name corresponding to the executable file name. For example, *chrome.exe* belongs to the *Google Chrome* browser. Therefore, this file belongs to the *Internet* category.

From the preprocessed files we extracted the following features:

- Duration of keys
- Latency of keys
- Relative Frequency of each key

4.3.1 Duration

Duration is defined as the timing difference between a *Key-Up* and *Key-Down* events for a single key.

Let *'k'* be any entered single key then the duration of *k* is as follows:

$$\mathrm{Dur}_k = \mathrm{TU}_k - \mathrm{TD}_k$$

Where *TU* is the Key-Up time and *TD* is Key-Down time for *k*.

Only the duration of alphabets was considered. The duration was case insensitive, for example 'A' and 'a' would be considered the same. As explained, the sampling period for recording was 16 milliseconds. Hence, if a key was pressed and released in less than this period, the Key-Up event would not be recorded by BeLT or the time for the Key-Up event would be equal to the time for the Key-Down event. In such cases the duration was considered to be 15 ms.

It is possible that a data point has a large deviation from other data points in a data set, thereby, increasing the standard deviation and consequently increasing the error rate. These data points are called *outliers*. Outliers are defined as "the observation/s that deviate much from other observations to arouse the suspicions that they are generated by another mechanism" [44].

In our case, the outliers would be very high or very low duration values. To detect such values there are many methods. One of these methods is the *Inter Quartile Range (IQR)*.

Consider having a data set of observations. if we sort these values from minimum (min) to maximum (max), then the midpoint is the *median* and the range will be $max - min$. The observation value at 25th% of data points is the 'first quartile (Q_1)' and the observation value at 75th% of data points is the 'third quartile (Q_3)'.

Inter Quartile range, in fact, refers to the observations 'closed' to median [45]. More precisely, it is calculated as:

$$IQR = Q_3 - Q_1$$

.

Figure 6 gives a graphical representation of Inter Quartile Range:

To detect the outliers we must detect the inter quartile boundaries:

$$lower\_bound = Q_1 - (IQR * k)$$

$$upper\_bound = Q_3 + (IQR * k)$$

where k is a non-negative constant. In our experiment we set $k = 1.5$.

Consequently, any observation x is an outlier if:

$$x > upper\_bound$$

or

[1]http://fashions-cloud.com/pages/i/interquartile-range-example/ last accessed on 12-05-2015

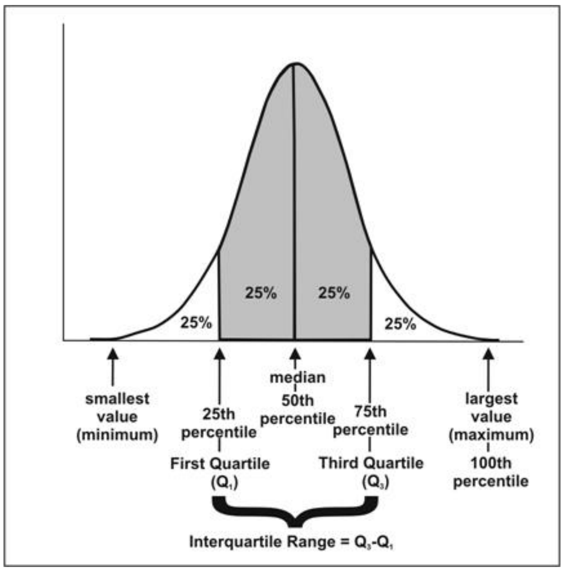


Figure 6: Inter Quartile Range(IQR)[1]

$$x < lower\_bound$$

We performed the outlier removal process on the extracted durations for each alphabet and each category. We then, created 4 templates of durations for every user, 3 for each category and the fourth category included the data without categorization. Each template was a 26×1 vector where rows indicate the corresponding number for each alphabet ('1' to '26' for 'a' to 'z'). The column included the mean of all durations for that alphabet.

4.3.2 Latency

Latency is defined as the difference between the Key-Down time for a key and Key-Up time for the next successive key. Let $key1$ be a key and $key2$ be the next successive key. The latency between $key1$ and $key2$ is defined as:

$$Lat_{key1,key2} = TD_{key2} - TU_{key1}$$

Where TD_{key2} denotes the Key-Down time for $key1$ and TU_{key2} denotes the Key-Up time for $key2$.

When extracting latency, we considered only pairs with the latency values less than or equal to 2000 milliseconds. A latency with more than 2000 milliseconds would indicate a deviation from a normal operation. For example, a user would type 'a' and after 5 minutes type 'b'. Then the latency between a and b would be 5 minutes which is not a normal amount of time for a latency. There were also cases where latency values would be negative. The reason was because in some situations, the the next key is pressed before the previous key is released.

As discussed previously, there were times when a Key-Up time was not available. For latencies we did the same strategy i.e. set $Lat = 15$. We also performed a similar outlier removal process on latencies.

Likewise to durations, we created 4 templates for latencies, 3 for each category and one for overall latencies without considering any categorization. Each template was a 26×26 vector which included mean values for latencies. The rows corresponded to numerical value of first letters of a pair ('1' to '26' for 'a' to 'z'), the columns corresponded to the same but for the second letter of the pair.

4.3.3 Frequency

For any letter 'n' we defined the frequency, as fraction of number of occurrences of 'n' in category 'cat' to total occurrences of all 26 letters in that category. This frequency is called the *Relative Frequency*:

$$\mathsf{Freq}_{cat} n = \frac{\mathrm{count}_{cat} n}{\sum_{n=1}^{26} \mathrm{count}_{cat} n}$$

For example, if in total there were 30 occurrences of 'c' in the *Internet* category and there total frequency of all keys in Internet was 600, then the relative frequency of 'c' in *Internet* would be 30/600=0.05

The relative frequency normalizes the frequency values for all the users to the range [01].

In case of frequencies, *deleted* keys were removed. Deleted keys are the keys that have been entered but later deleted by the user due to typing mistakes or any other reason. Similar to the other features, 4 templates were created for frequencies. Each template consist of a 26×1 vector. Rows represent the corresponding numeric value for alphabet letters and columns represent the relative frequency.

4.4 Data Separation

We allocated 35% of total data for template creation and the rest for testing. Pre-processing of test data is the same as training. However, our assumption of test data was a sequence of events, pre-processed so that the sequence of occurrence of these events is maintained. Since the CA system performs the comparison process for every single event, we had to maintain a similar file format for CI too.

Based on our assumption, three types of test files were created for each user. We will here describe the test data structure for durations and latencies in detail.

4.4.1 Duration Test Files

After pre-processing and feature extraction, a single test file was created for each user, number of rows varied between each user depending on the amount of data. Table 4 shows the format for duration test files along with an example of an entry:

Event ID	key	category	Duration
234	2	3	110
236	3	1	122
244	3	2	108

Table 4: Format and Example of a duration test data entry

The first column corresponds to the Event ID. The second column corresponds to the numerical equivalence of the alphabet, i.e. *a=1, b=2, ..., z=26*. The third column denotes the *category* from 1 to 3 and last column contains the duration values. There was no *mean* here and we included any single duration without outlier removal process.

A similar file to the duration test file was created but, it did not include deleted key records. We used this file for frequency test data and for testing with combinations of durations and frequencies.

Latency Test Files

The format for latency test file is a bit different from the duration's. Here, we deal with information regarding two keys and not one.

Table 5 shows the format for duration test files along with an example of an entry:

key1	key2	category	Latency	Dur(key1)	Dur(key2)
1	23	3	210	98	122
23	17	3	334	113	105
17	20	2	296	131	111

Table 5: Format and Example of a latency test data entry

The first and second columns correspond to the numeric equivalent of the keys. The third column indicates the category, the fourth column includes the latency value and the last two columns represent the durations of key1 and key2 respectively.

Not all keys present in the duration test file will also be present in the latency. The reason is simply individual keys with no successive key or a 'distant' successive key (i.e. $Lat > 2000ms$). Hence, for reasons to be explained in the Analysis chapter, the duration values were included again in the latency test file.

5 Methodology

In this chapter we are going to discuss the analysis process, the settings we used and the results we have achieved.

5.1 Period Separation

In order to simulate the situation, where after a period of actions the user is locked out, we performed the identification at different period lengths (chunks). Periods are simply defined as number of actions (n). Here, by n actions we mean n keystrokes and to be more specific n key-downs. We performed the comparison with 20 periods of length n with $n = 50, 100, 150, 200, ..., 1000$. This setting was fixed for every user. That means, we tested for every user with $n = 50$ then, with $n = 100$ and so on. Later, in order to see if it is possible to integrate the CI process into CA process instead of using fixed periods, for each user, we used the respective 'ANIA' and 'ANGA' values obtained from the CA process. We will explain this in detail in the section 5.5 .

5.2 Distance Metrics

To compare the test data with the template values we used the two distance formulae: Manhattan and Euclidean. These two distance metrics are both *Minkowski* distance metrics with different *p* values [46]. Depending on the feature and categorization we performed small modifications on these two distance metrics.

In case there was a categorization, the same formula would be used for each category, if there was test data belonging to that category in the test chunk. In case there was no categorization, simply the single test vector would be compared to the *Uncategorized* vector of the template.

The structure of template for *durations* is:

$$Tm_{dur}^{cat} = (td_1^{cat}, td_2^{cat}, td_3^{cat}, ..., td_{26}^{cat})$$

Where *cat* is the category number, td_i^{cat} is the duration template value for character *i* in *cat* which is $\mu_{dur(i)}^{cat}$ for duration.

Template structure of *frequency* is similar to that of duration:

$$Tm_{freq}^{cat} = (tf_1^{cat}, tf_2^{cat}, tf_3^{cat}, ..., tf_{26}^{cat})$$

Where, tf_i^{cat} is the frequency template value for character *i* in cat.

On the other hand, for Latencies the template structure looks different:

$$Tm_{lat}^{cat} = (tl_{1,1}, tl_{1,2}, tl_{1,3}, ..., tl_{26,26})$$

where $tl_{i,j}^{cat}$ is the *latency* between *i* and *j* in *cat*.

5.2.1 Manhattan Distance

Manhattan distance is a distance metric used to find the distance between two or more vectors such as feature vectors with equal dimension. It is used to find the shortest distance between the absolute values of coordinates of two points in the X-Y plane. In our case, the two vectors consist of test and template values.

The formula for *Manhattan* distance for duration and frequencies is as follows:

$$\mathrm{MD\_dist}(Ts^{cat}, Tm^{cat}) = \sum_{i=1}^{n} \left| test_i^{cat} - temp_{c_i}^{cat} \right|$$

Where $test_i^{cat}$ is the test value for *i* in *cat*, $temp_i^{cat}$ template value for *i* in *cat* and in case of OM method $n = 26$.

There are two issues regarding missing template and test data for some specific characters.

1. It is possible that test data entries for some characters are not available and hence the test value of those character in that chunk is 0. In this situation, we cannot calculate the above absolute distance for that character. To mitigate this, the absolute distance is:

$$\mathrm{dist}(test_i^{cat}, temp_{c_i}^{cat}) = \begin{cases} \left| test_i^{cat} - temp_{c_i}^{cat} \right|, & \text{if } test_i \neq 0 \\ 0, & \text{Otherwise} \end{cases}$$

2. If there is no template entry (i.e. tm_i) available for some specific character in some specific category, then the template entry associated with that character in *Uncategorized* template vector is used. If there is no entry in the Uncategorized vector too, then the mean of all nonzero entries in the Uncategorized template is used as tm_i. In comparison with no categorization, mean of nonzero Uncategorized template is considered in case the specified element is not available.

5.2.2 Euclidean Distance

Euclidean Distance Metric is similar to Manhattan but it is defined as the square root of sum of squared coordinate differences. It is in fact, the length of the shortest straight line from one point to the other. The general formula for Euclidean distance metric is:

$$\mathrm{ED\_dist}(Ts^{cat}, Tm^{cat}) = \sqrt{\sum_{i=1}^{n} (test_i^{cat} - temp_i^{cat})^2}$$

5.3 Comparison Methodologies

We performed the comparison with:

- Categories taken into account
- No categorization.

In category-wise comparisons, in addition to separate n actions we also separated these n values based on one of the three categories explained in section 4.2.1. Additionally, we performed the comparison without categories taken into account and compared the test input with the value stored in the Uncategorized template of each user. The reason behind this is to see how would the system perform when users are identified based on the application software they have used. Yet, we also performed the identification irrespective of the software being used.

For the purpose of comparison, we used two methods called *Mean to Mean Comparison* and *One to Mean Comparison*. The comparison process for each of the three features is similar with small differences. Thus, we explain the comparison process for duration in details as well as different combinations of the features. In all of these processes, comparison has been done when software categories have been considered and when not, using Manhattan and Euclidean distance metrics and with both comparison methods (except OM for frequency).

5.3.1 Mean to Mean Comparison

In *Mean to Mean (MM)* comparison, we compare the mean of durations or latencies in a chunk (period) of test data with the related template data. That means for every key in the test chunk, we take the mean value of the corresponding feature for all occurrences of that key. Hence there is a process of averaging before comparison.

Figure 7 shows the MM comparison of durations. Here, before comparison, by using an *averaging function*, test data becomes in a format similar to template that is it is converted to a 26×1 vectors for each category. After averaging, test data will contain mean duration of all occurrences of every key for each category in the test chunk. For example first two rows in the test chunk are both 2s and both belong to the same category. Suppose these are the only test data with these characteristics. Therefore, their mean, is the average test duration of 2 in category 2 of the test data.

5.3.2 One to Mean Comparison

In *One to Mean (OM)* comparison, each and every element in the test chunk is compared to the mean value stored in the template. In OM comparison without categorization, The process is similar to MM. But, we utilize the fourth test vector created by the averaging function. Figure 8 represents this process. For example, mean of all test durations for 3 is stored in the *Uncategorized* test vector and then compared to the value for 3, stored in the Uncategorized vector of the template.

Figure 9 displays the comparison between duration of each event k in a test chunk of length n of user *i* and the duration template of user *j* where i can be equal to j. The first 3 vectors in the template indicate the template values for each category. The last vector named as *Uncategorized* represents the template without categorization. The blue arrows indicate that which durations in the test chunk are compared to which elements in the template. During the feature extraction process,any letter (case insensitive) is converted to a digit with *a=1, b=2, ..., z=26*.

Hence, for example, the first letter in the test chunk is 2 which is equivalent to b (or B). According to the third column of the test vector, it must be compared with the value for 2 stored in the Category 2. n^{th} element in the test chunk is a 3 which must be compared to duration of 3 in Category 1.

Figure 10 represents the comparison with the same settings except that categories do not matter here. For example, all values of 2 regardless of their categories are compared to the

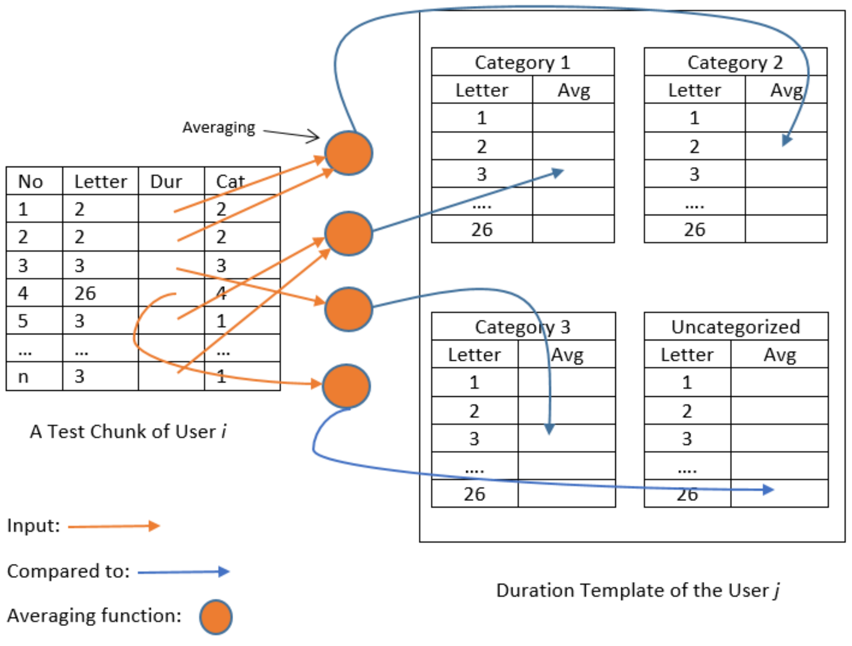


Figure 7: Comparison of durations using MM method with categorization

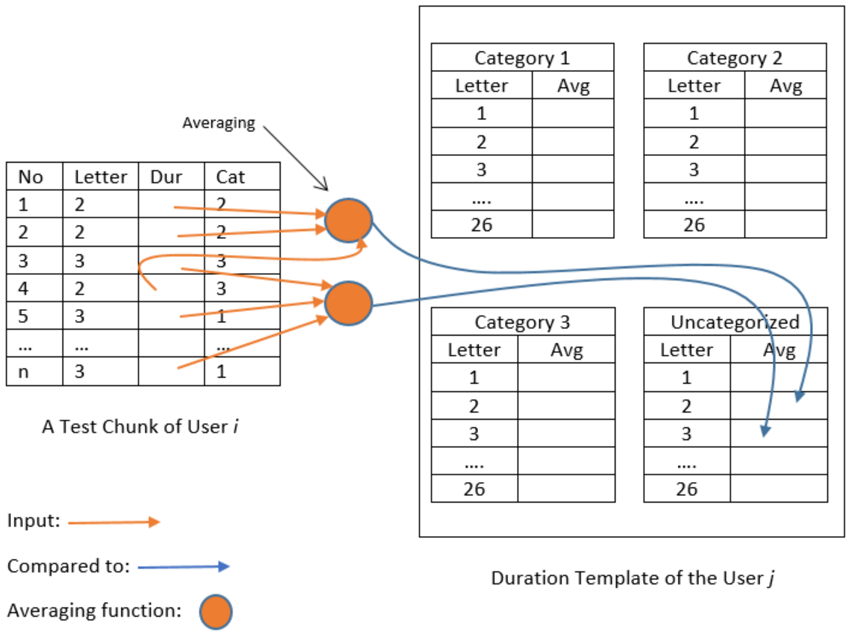


Figure 8: Comparison of durations using MM method without categorization

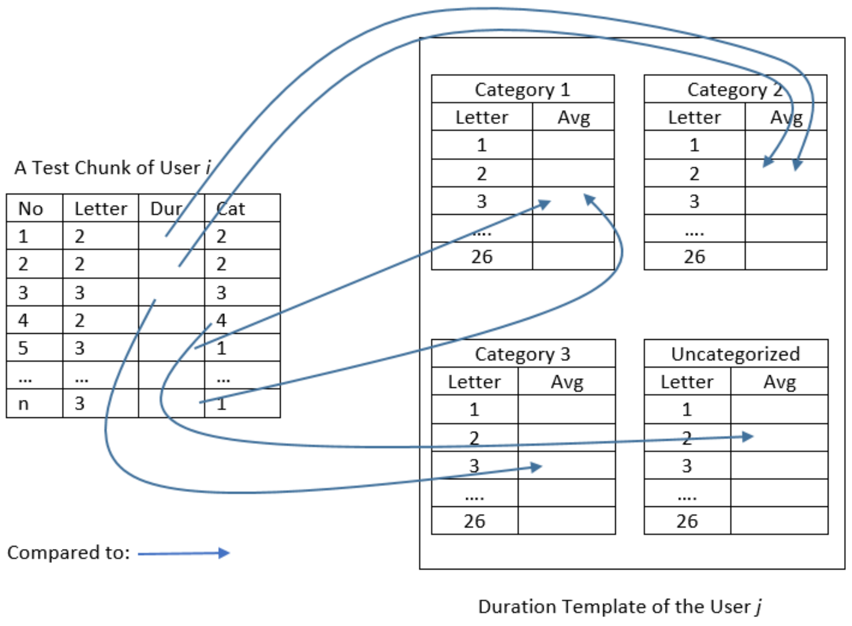


Figure 9: Comparison of durations using OM method and with categorization

duration value for 2, stored in the 'Uncategorized' vector.

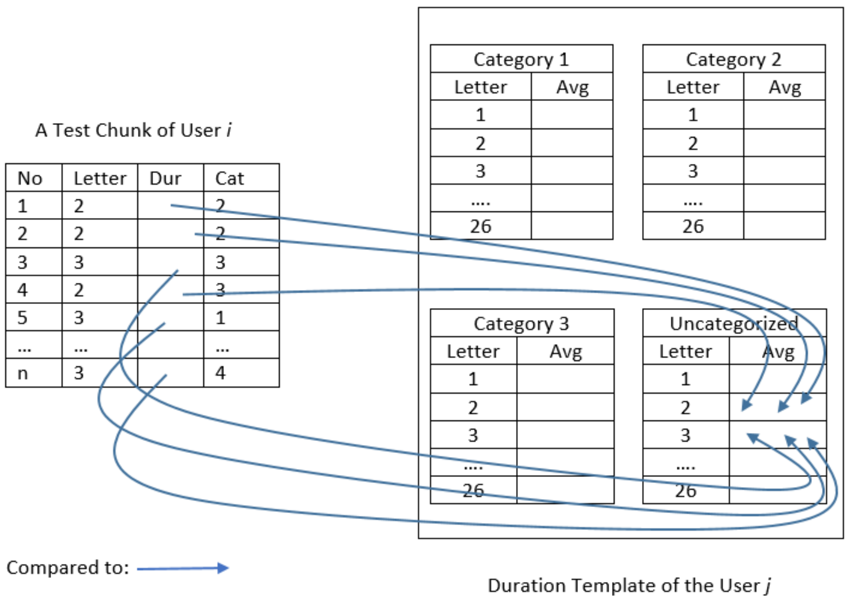


Figure 10: Comparison of durations using OM method and without categorization

OM and MM process are similar for the latencies. However, the structure of the template and test data are different. In test data, each record consists of the durations of the two characters and their latency. On the other side, there are four 26×26 vectors for the template. In OM, latency of each pair in the latency test file is compared to the related values in the template. In MM process, four 26×26 vectors are created from the test data containing the mean latency values.

In case of frequencies, we considered only the MM method. Because, frequency of one character is always one. In MM method for frequency, we calculated for each category, the *relative frequency* of each character with a nonzero absolute frequency. That is the number of occurrences of a character *x* in a category *cat* divided by total number of occurrences of each character in *cat*. From this we can also conclude, that the relative frequency of each character in the *Uncategorized* vector is number of occurrences of that character to the chunk size n which is the total number of frequencies in that chunk.

Combining the Features

We also performed the comparisons with two different combinations of features:

- Duration and Frequency
- Duration and Latency

We used the same configurations for testing process with minor modifications. For duration and frequency combination, since we could not apply the OM scheme on frequencies, we performed the analysis in two ways. First, we applied OM on durations and MM on frequencies and in the second method, we applied MM on both features. The final results were of different units and ranges, Hence, the overall distance score was obtained by multiplying the distance score obtained from each of individual comparison processes. That is:

$$\mathrm{dist}(Ts^{cat}_{dur,lat}, Tm^{cat}_{dur,lat}) = \mathrm{dist}(Ts^{cat}_{dur1}, Tm^{cat}_{dur1}) \times \mathrm{dist}(Ts^{cat}_{freq}, Tm^{cat}_{freq})$$

Figure 11 illustrates the first type of comparison process (OM-MM), when combining frequency and duration.

The comparison process is performed separately for each feature. Two distance calculation functions are the ones used for distance calculation with only durations or frequencies. The second type of comparison process (MM-MM) is also similar to this function with only duration distance calculation function changed.

When combining durations and latencies, as explained above, the aggregate distance is obtained by addition of comparison scores of duration of first letter, duration of the second letter and the latency. That is:

$$\mathrm{dist}(Ts^{cat}_{dl}, Tm^{cat}_{dl}) = \mathrm{dist}(Ts^{cat}_{dur1}, Tm^{cat}_{dur1}) + \mathrm{dist}(Ts^{cat}_{dur2}, Tm^{cat}_{dur2}) + \mathrm{dist}(Ts^{cat}_{lat}, Tm^{cat}_{lat})$$

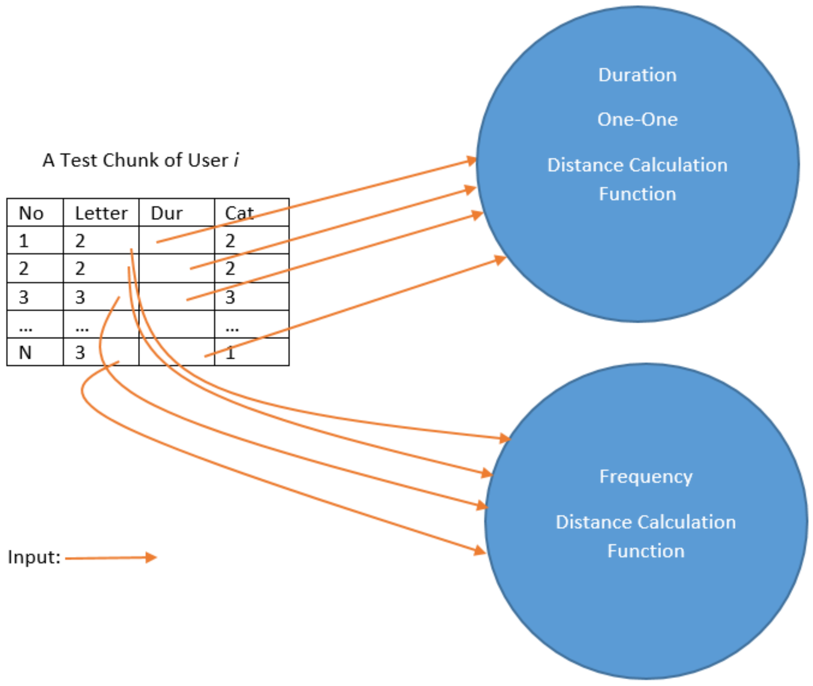
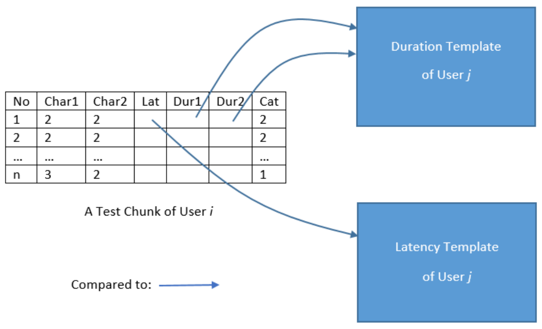
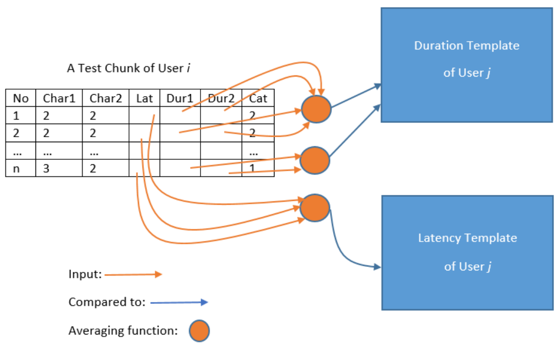


Figure 11: Comparison with duration and frequency combined



(a) OM Comparison Process



(b) MM Comparison Process

Figure 12: Comparison with duration and latency combined

5.4 Score Fusion and Decision

After the comparison process for a test chunk, if no categorization has been considered, the final score is same as the one produced. However, when considering categorizations, 3 scores are produced for each category. If there is no data from one category the score for that category will be 0.

In order to obtain a final score, which is defined as the score of user i from test chunk t, we must unify these three results. The solution is to fuse these scores into a unified score. Simplest way to achieve this is to calculate *Weighted Average Distance*. With respect to our work it is defined as:

$$\mathrm{totalDist} = \frac{\sum_{k=1}^{3} \mathrm{dist}_k * w_k}{\sum_{k=1}^{3} w_k}$$

where dist_{cat} is the score obtained from the comparison for k^{th} category and w_k is the associated weight. Weights can be set in two ways, choosing fixed weights or variable weights.

5.4.1 Weighted Average Mean with Fixed Weights

Weights indicate the influence of each score to the final value. Values with more weights contribute more to the final score. First, we decided to apply *fixed* weights as a combination of 1, 2 and 3. Initially, we gave the weight based on the availability of data i.e. the total amount of data (training and test) available with respect to each category. In this case the result would be $w_1 = 3, w_1 = 1, w_3 = 2$. However, to be more accurate, we performed a testing process with frequencies and chunk size of 50 when Manhattan distance was used. There were total $3! = 6$ distinct permutations of (1, 2, 3). Hence we performed the testing with each of the 6 permutations and the results obtained showed that the best weight permutation is : $w_1 = 3, w_1 = 2, w_3 = 1$. Table 6 shows the results for chunk size 50 obtained based on performance with frequency when MD, MM and these weight combinations are used:

w_1, w_2, w_2	ACC(%)
1, 2, 3	11.21
1, 3, 2	11.32
2, 1, 3	11.48
2, 3, 1	12.11
3, 1, 2	12.33
3, 2, 1	12.40

Table 6: Rank-1 results obtained for chunk size 50 using frequencies MM and MD

Results are quite close. However, regardless of this low difference, we selected 3, 2, 1.

5.4.2 Weighted Average Mean with Variable Weights

We also decided to analyze using variable weights. The weights varied depending on the frequency of characters belonging to each category in that chunk. For example, for a chunk size, $n = 50$ if 20 of the characters belong to category 1, then the weight for that category is 20, contributing more to the final score. It was also possible to have two equal weights for two categories.

5.4.3 Decision

The User with minimum of total score would be the identified user for any test chunk. Our criteria for identification decision was based on whether a user is identified correctly (denoted by 1) or not (denoted by 0) at every chunk. We calculated the accuracy rate for 8 ranks. That means if the test user was among the *r* identified users with minimum total score, it would be a correct identification for rank-*r* otherwise, it was an incorrect identification. Thus, for every chunk *ch* out of *m* chunks in a test data:

$$d = \begin{cases} 1 & \text{if correctly identified} \\ 0 & \text{if incorrectly identified} \end{cases}$$

Where *d* is the decision made.

From this, we obtained the identification accuracy rate for every user *k* and every chunk size *ch*.

$$ACC(r, k, ch) = \frac{\sum_i^m d_i}{m} * 100 \qquad , \forall d_i = 1$$

Where *r* is the rank number and *m* is the total number of identifications for that test data which is equal to the number of chunks in that test data.

The mean of these ACC values determines the system performance for every chunk size *ch*:

$$ACC(r, ch) = \frac{\sum_{k=1}^{51} ACC(r, k)}{51}$$

Therefore, we shall have the system performance defined as: identification accuracy rate for each of the 20 chunks.

5.5 Analysis with Continuous Authentication Data

In order to see the performance of the system for identification after a user is locked out, we also tried to use the results obtained from the CA process. For better understanding, we first describe the CA process purposed by [5] and then describe the CI analysis performed with the output from CA performance evaluation.

5.5.1 System Architecture

Figure 13 give an schematic view of the designed CA system by Bours and Mondal [5], where the red lines represent a possible integrated CI process.

During the enrollment phase, after capturing the KD data from the keyboard, different features related to CI and CA are extracted and stored in CI and CA template databases respectively. For authentication, the KD input is captured and after feature extraction enters into the matching (comparison) module. During the comparison process, the input is compared to the logged in user's template. Based on some threshold, *t* if the similarity score between the input sample and template input is lower than the *t*, the trust value is reduced and if it is higher than *t* then the trust value increases. In the decision module the trust value is compared with the lockout threshold *lt*. If the trust value is more than *lt*, the user can continue, otherwise he gets locked out. After lock out, from all the actions the locked out user has performed between previous lock out until now, the CI related keystroke features of the unknown user are extracted. Extracted features are then compared to all the templates in the CI database in order to identify the unknown user.

5.5.2 Performance Analysis

As explained before, the performance of the CA system was calculated in terms of ANIA and ANGA. For every user *k*, there exist an $ANGA_k$ indicating the average number of genuine actions can be done as user *k* before lock out and $ANIA_k$ indicating the average number of impostor actions that can be done as user *k* before lock out. Therefore, if $ANGA_k = \infty$ that means no

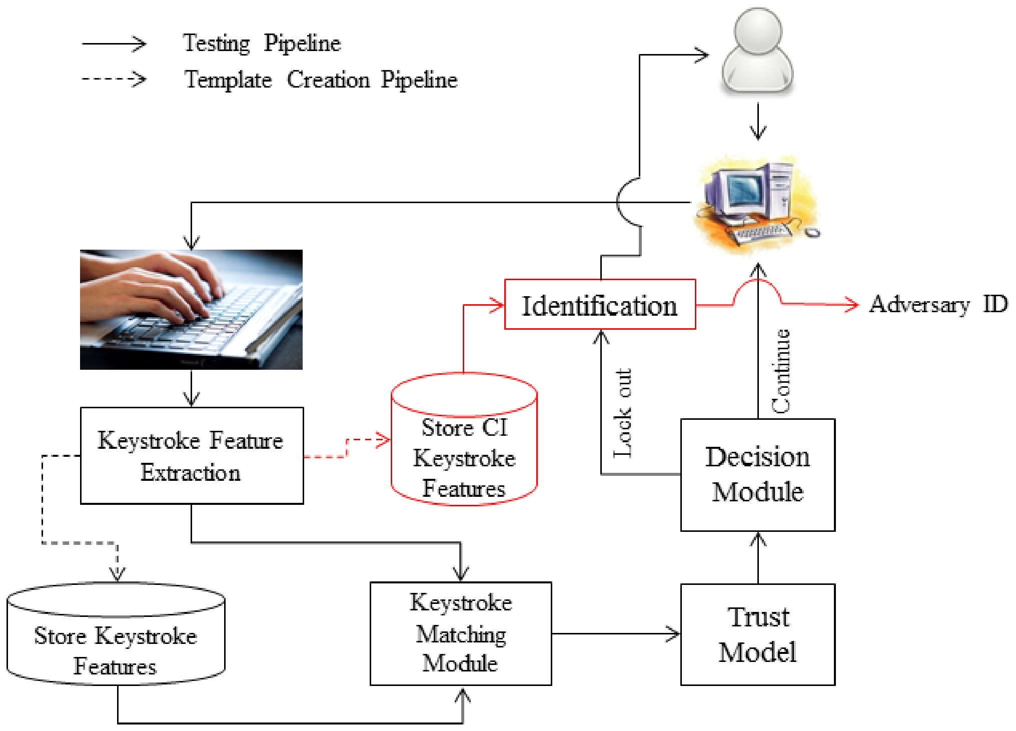


Figure 13: Countinuous Authentication model by Bours and Mondal [5]

value for $ANGA_k$, a genuine user has never been locked out [6]. As explained in section 5.1, we divided the test data into fixed number of actions and performed the testing. To simulate a more realistic situation, instead of using fixed chunk sizes we used $ANGA_k$ when comparing a test data of a user with his template and $ANIA_k$ when comparing the data with impostor templates. ANIA and ANGA values used from a CA result set produced by [6]. The CA process used the same set of data and was based on keystroke features.

For each user k , we performed a 51×50 impostor testing with chunk size= $ANIA_k$ and 'maximum' 51 genuine testing with chunk size= $ANGA_k$. If no ANGA value was available for user k meaning no lock out for genuine user, then we would not perform the genuine testing. Hence the actual number of computations was less.

6 Result Analysis

In this chapter, we will report the results obtained under various configurations. We mainly divide the results into those obtained with software context categorization and those without such a categorization. Then we will look at different results obtained for various comparison methods, features, distance metrics and score calculation methods. For each of the settings we computed the Identification Accuracy Rate (ACC) for each chunk size and for 8 ranks. To avoid repetition only selected results are presented here. The complete set of results can be viewed in Appendix A and B.

6.1 Results Based on Categorized Comparison

In this section, we present the results obtained from the analysis based on the software categorization, with various comparison methods, features, distance metrics and score fusion methods.

Detailed results corresponding to this section, can be found in A.1 and A.2.

6.1.1 Based on Distance Metrics

Table 7 displays the Rank-1 ACC for different chunk sizes. Analysis performed with both distance metrics using MM method when considering various features and when fixed weights are used for the score fusion.

As shown in the table 7, the results appear to show a difference between the two distance metrics under the same settings. The most pronounced difference is observed for latency. This has to do with the quality of the collected data which contained a non-negligible number of negative latencies, which would mean that the user had not fully released a certain key before pressing the next key. Some of these negative values were considerably large. Due to the non-linear nature of the ED, the large negative numbers affect the accuracy of the ED much more than MD.

The differences in other cases is neither significant nor consistent with respect to chunk size and features used. At some points it is low and at some points it is high. For example, if we consider the duration and frequency features combined, after 50 actions (chunk size) the difference between ACC obtained using MD and ED is only 0.6% . The difference is at highest after 1000 events (6.3%).

Given all the considerations above, we cannot conclude that MD in general offers a significant advantage over ED, especially without knowing the confidence interval for the calculations.

With this preliminary result, we can see that frequency results are notably low compared to other features with approximately 12.5% ACC after 50 actions (chunk size) and approximately 23.5%, after 1000 actions. This could indicate that frequency features we used, were not distinguishing different users to a sufficient level. A possible solution could be including more *Stylometric* features [47]. However, measuring various stylometric attributes when the nature of the data is not consistent is a complicated task which is out of scope of this research. As an

%	MD					ED				
Chunk	Dur	Lat	Freq	D-L	D-F	Dur	Lat	Freq	D-L	D-F
50	31.8	41.5	12.4	53.2	37.2	30.1	27.0	13.1	41.2	36.6
100	40.1	47.3	17.2	58.7	45.9	36.1	30.1	17.0	46.4	43.6
150	45.0	49.9	18.8	61.6	49.3	40.2	31.9	18.6	48.6	45.6
200	48.8	50.7	19.9	63.2	51.6	42.8	33.2	19.3	49.3	46.8
250	49.6	50.6	20.5	63.8	52.8	43.5	31.6	19.3	48.9	47.7
300	51.5	**53.5**	21.4	65.8	54.4	44.7	35.5	20.2	50.6	48.8
350	51.9	50.8	23.2	66.2	56.0	48.5	35.0	20.5	50.1	49.9
400	54.8	52.9	21.1	66.2	57.2	47.8	33.7	19.9	51.6	50.7
450	54.0	51.4	21.7	66.2	56.1	47.7	35.2	20.8	52.5	49.9
500	55.6	51.2	23.7	67.5	58.8	49.0	34.5	19.9	51.6	51.9
550	55.7	51.8	23.4	67.9	59.1	48.6	35.5	20.4	49.6	51.7
600	56.9	53.3	23.8	68.2	58.6	49.4	35.9	21.7	51.5	52.1
650	56.8	52.2	24.1	67.5	57.8	49.9	**36.6**	21.1	51.9	52.3
700	58.2	50.0	23.8	68.9	58.3	52.3	35.5	20.8	53.2	53.5
750	56.9	51.4	24.9	69.1	59.3	50.8	34.3	21.3	53.0	53.7
800	58.5	51.9	**25.8**	68.6	60.0	52.8	32.6	21.6	51.7	**55.4**
850	58.5	52.7	25.5	68.4	59.3	52.7	33.6	**22.3**	52.3	**55.4**
900	57.6	52.3	25.3	68.7	58.4	51.8	34.4	19.6	55.3	52.3
950	58.3	51.7	24.5	69.1	**60.5**	52.0	33.9	20.7	55.4	53.5
1000	**58.8**	52.7	23.4	**71.2**	60.0	**53.0**	34.8	21.3	**55.6**	53.7

Table 7: Rank-1 ACC(%) for MD vs ED distance metrics when considering various features

example, the language used was not specified to us since, the experiment was performed in an uncontrolled environment. Hence we used the frequency features just as an additional measure.

The biggest difference is observed between individual features performance vs combined features performance, which is consistent for all combinations and chunk sizes. Using duration and latency features combined there was 53% successful identification rate after 50 actions. This amount increased to 71.2% after 1000 events.

Another major observation is that increasing in performance was not linearly proportional to increase in the chunk size. In fact the ACC values seem to saturate beyond a certain chunk size.

6.1.2 Based on Comparison Method

Table 8 displays the rank-1 ACC obtained when considering two different comparison methods. As we explained before, we cannot perform OM comparison using frequencies but we performed the dur-freq analysis using OM on durations and MM on frequencies. Hence, we present the results obtained again with the similar configurations for 5 features, fixed weights. Here, we present the result of MD only. Columns related to MM are a repetition of columns for MD in table 7.

%	OM					MM				
Chunk	Dur	Lat	Freq	D-L	D-F	Dur	Lat	Freq	D-L	D-F
50	32.4	42.5	N/A	53.0	35.1	31.8	41.5	12.4	53.2	37.2
100	41.5	48.1	N/A	57.9	41.2	40.1	47.3	17.2	58.7	45.9
150	46.4	51.4	N/A	60.5	43.9	45.0	49.9	18.8	61.6	49.3
200	49.4	52.5	N/A	62.4	45.9	48.8	50.7	19.9	63.2	51.6
250	51.1	53.2	N/A	62.2	48.1	49.6	50.6	20.5	63.8	52.8
300	53.8	54.4	N/A	63.2	51.2	51.5	53.5	21.4	65.8	54.4
350	53.1	54.0	N/A	63.9	52.2	51.9	50.8	23.2	66.2	56.0
400	55.0	55.4	N/A	64.0	52.0	54.8	52.9	21.1	66.2	57.2
450	55.8	57.6	N/A	65.6	53.2	54.0	51.4	21.7	66.2	56.1
500	57.5	57.7	N/A	66.7	54.1	55.6	51.2	23.7	67.5	58.8
550	56.2	58.6	N/A	**68.2**	54.3	55.7	51.8	23.4	67.9	59.1
600	58.2	59.4	N/A	65.7	55.4	56.9	**53.3**	23.8	68.2	58.6
650	57.4	59.6	N/A	68.0	56.0	56.8	52.2	24.1	67.5	57.8
700	59.2	58.1	N/A	65.9	56.2	58.2	50.0	23.8	68.9	58.3
750	58.8	58.4	N/A	66.7	57.2	56.9	51.4	24.9	69.1	59.3
800	59.4	60.2	N/A	67.4	**58.8**	58.5	51.9	**25.8**	68.6	60.0
850	59.6	60.6	N/A	67.7	58.0	58.5	52.7	25.5	68.4	59.3
900	59.3	61.0	N/A	67.2	58.5	57.6	52.3	25.3	68.7	58.4
950	59.2	59.9	N/A	67.3	58.1	58.3	51.7	24.5	69.1	**60.5**
1000	**59.7**	**61.2**	N/A	67.8	58.3	**58.8**	52.7	23.4	**71.2**	60.0

Table 8: Rank-1 ACC for OM vs MM for various features

From the table, we can observe no significant differences between OM vs MM for the individual features performances. When using combined features ACC seems to be reduced for big chunk sizes. Again the differences are believed to be within the confidence interval for the calculations and no conclusion can be made about the possible advantages of MM over OM.

Consequently, the duration-latency combination once again shows the best performance with identification rate of 53.2% at chunk size 50 and that of 71.2% at chunk size 1000.

6.1.3 Based on Score Fusion Method

Finally, we can have a look at the results when considering different score calculations. As we explained previously, for a unified score calculation, we used *Weighted Average Mean* of the scores obtained for each category. The result we presented in tables 7 and 8 were based on the fixed weights ($w_{cat_1} = 3, w_{cat_2} = 2, w_{cat_3} = 1$). Table 9 represents the results obtained when considering fixed weights vs variable weights.

%	MM, Variable Weights					MM, Fixed Weights				
Chunk	Dur	Lat	Freq	D-L	D-F	Dur	Lat	Freq	D-L	D-F
50	31.8	41.8	12.7	53.1	37.4	31.8	41.5	12.4	53.2	37.2
100	40.2	47.5	17.8	58.6	47.3	40.1	47.3	17.2	58.7	45.9
150	45.1	49.4	20.2	61.4	50.8	45.0	49.9	18.8	61.6	49.3
200	48.7	50.8	22.4	63.1	53.3	48.8	50.7	19.9	63.2	51.6
250	49.8	50.5	23.5	63.0	55.3	49.6	50.6	20.5	63.8	52.8
300	51.6	**53.6**	25.8	64.9	56.4	51.5	53.5	21.4	65.8	54.4
350	51.5	51.2	26.6	65.4	58.2	51.9	50.8	23.2	66.2	56.0
400	54.2	52.2	25.7	65.4	59.3	54.8	52.9	21.1	66.2	57.2
450	54.7	52.2	26.0	65.6	59.6	54.0	51.4	21.7	66.2	56.1
500	54.9	51.7	29.0	66.6	61.3	55.6	51.2	23.7	67.5	58.8
550	55.6	52.2	27.7	66.1	62.4	55.7	51.8	23.4	67.9	59.1
600	57.0	52.1	30.5	67.2	62.4	56.9	**53.3**	23.8	68.2	58.6
650	56.4	53.4	31.9	66.8	62.4	56.8	52.2	24.1	67.5	57.8
700	58.1	49.9	32.4	67.1	63.2	58.2	50.0	23.8	68.9	58.3
750	56.1	51.0	30.9	64.3	63.5	56.9	51.4	24.9	69.1	59.3
800	58.0	51.3	32.4	66.9	63.4	58.5	51.9	**25.8**	68.6	60.0
850	57.0	52.5	32.4	**68.4**	62.9	58.5	52.7	25.5	68.4	59.3
900	58.5	52.3	**32.9**	68.0	62.8	57.6	52.3	25.3	68.7	58.4
950	60.2	53.4	32.0	67.6	64.0	58.3	51.7	24.5	69.1	**60.5**
1000	**60.3**	52.5	32.5	68.0	**64.7**	**58.8**	52.7	23.4	**71.2**	60.0

Table 9: Rank-1 ACC for different weights for score unification

Similar to Table 8, no general conclusion can be made about the possible advantages of fixed weights versus variable weights. Again, the duration-latency combination with fixed size weights performs best.

From the above results, we can see that the best configuration, in this case, has constantly been the use of the combined duration-latency features, when using MM comparison and fixed weights and Manhattan distance.

Figure 14 represents the accuracy rate changes with chunk sizes for 8 ranks when this setting is applied.

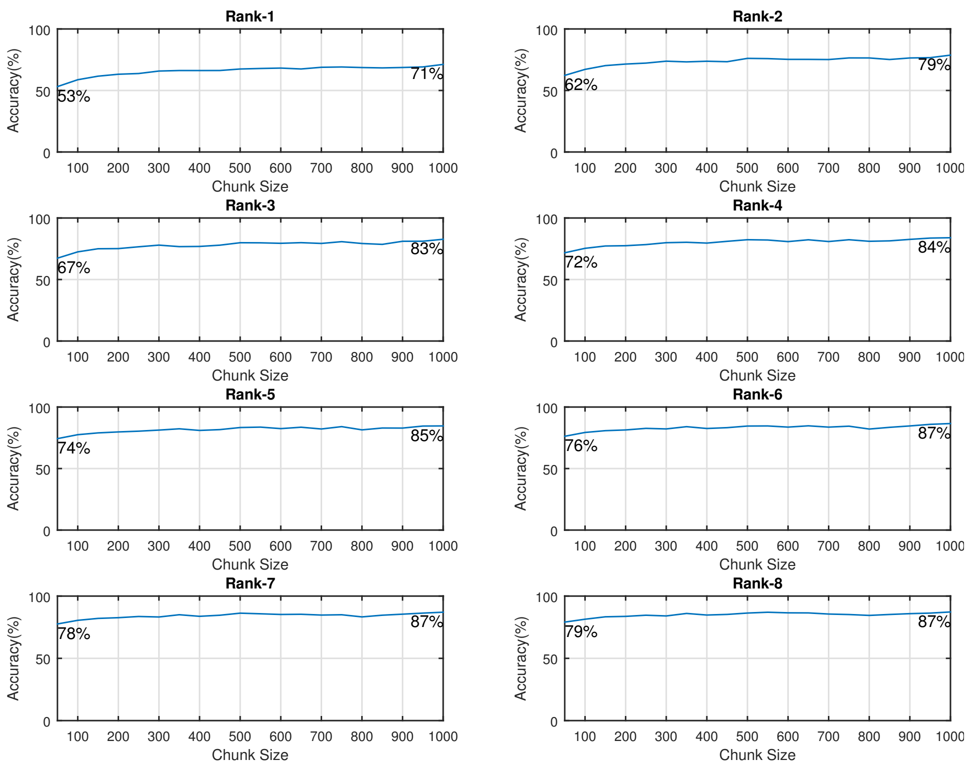


Figure 14: Identification Rate (ACC%) changes of the best setting at various chunk sizes for 8 ranks

As we can see, there is not much variation in ACC when increasing chunk size on higher ranks. The results in lower ranks tend to vary more with chunk size compared to the higher ranks. We can see that $\approx$ 80% of the identifications have been correct for rank-8 regardless of the chunk size.

6.2 Results without Categorization

For analysis without categorization, we did not have any score fusion, since there was only one result obtained with one category (i.e. Uncategorized) taken into consideration. Therefore, we performed the testing on various features, based on various distance metrics and comparison methods. Detailed results corresponding to this section, can be found in A.3 and A.4.

6.2.1 Based on Distance Metrics

We first tried to test with different distance metrics. Similar to the analysis with categorization, our baseline for the analysis was testing with all features when using MM for comparison and MD as distance metrics. We performed the testing under the same settings but with ED as well. Table 10 represents the results obtained by MD and ED.

%	MD					ED				
Chunk	Dur	Lat	Freq	D-L	D-F	Dur	Lat	Freq	D-L	D-F
50	31.3	48.6	14.0	60.0	38.0	30.4	31.0	14.4	47.1	36.4
100	40.3	56.2	18.4	65.2	46.8	36.3	34.5	18.8	50.9	43.9
150	45.0	57.6	20.5	66.9	50.6	39.7	36.3	20.5	52.0	46.8
200	47.6	59.5	21.9	69.6	53.8	42.7	36.4	22.4	55.0	49.8
250	50.7	59.4	24.3	68.7	56.0	44.4	36.2	23.4	54.1	51.2
300	51.3	60.0	27.4	69.4	57.1	45.8	37.5	25.4	54.8	52.4
350	53.2	58.5	26.9	68.8	58.1	48.4	37.4	26.5	56.3	54.0
400	54.3	61.2	29.3	70.8	60.3	48.7	36.8	27.8	55.5	54.5
450	55.2	59.8	27.9	68.7	58.7	48.2	37.8	27.8	56.3	55.2
500	56.2	60.2	30.3	68.6	59.6	48.4	36.5	29.1	56.4	55.5
550	57.9	**61.9**	30.7	69.6	60.2	51.1	36.4	29.8	54.2	56.9
600	56.6	59.9	31.8	69.1	60.9	51.1	**38.9**	31.8	55.8	57.2
650	58.5	58.7	31.6	70.7	61.8	51.2	38.8	31.3	56.8	58.1
700	59.0	58.2	33.0	69.4	62.2	52.1	35.1	32.3	57.2	59.3
750	60.9	57.9	32.8	67.7	61.9	53.2	35.4	33.6	54.3	58.1
800	60.2	59.2	32.5	72.1	62.5	52.7	33.5	31.4	57.1	57.7
850	60.1	58.0	34.5	70.3	61.5	**54.3**	35.0	32.7	57.4	59.1
900	58.9	58.6	35.8	69.9	63.1	53.6	35.7	**35.9**	56.8	59.6
950	61.5	61.0	35.9	70.3	62.1	52.8	34.9	32.8	58.8	58.1
1000	**62.4**	58.3	**37.0**	**72.9**	**63.8**	54.6	33.8	32.8	**60.8**	**61.2**

Table 10: Rank-1 ACC(%) for MD vs ED distance metrics when considering various features

According to the table, the results is much like the case with categorization. When comparing the results, when no categorization is considered, we got better performance with each feature compared to the corresponding feature in the categorized case. For example, the results for duration are higher when no categorization is done compared to the duration results using categorization. However, same deviation is observed for latency, and the lowest performance among the various features still belongs to the frequency with only 14% ACC after 50 events and 37% after 1000 events. Duration-latency combination has the best performance with 60% of successful identification rate after 50 events and almost 73% after 1000 events.

6.2.2 Based on Comparison Method

Furthermore, we performed the analysis using OM comparison method. Table 11 represents the comparison of results when using MM and OM methods. We present the results using MD only. OM could not be performed for frequencies.

%	OM					MM				
Chunk	Dur	Lat	Freq	D-L	D-F	Dur	Lat	Freq	D-L	D-F
50	30.4	50.0	N/A	60.0	35.6	31.3	48.6	14.0	60.0	38.0
100	36.3	56.0	N/A	64.6	42.0	40.3	56.2	18.4	65.2	46.8
150	39.7	59.9	N/A	66.2	44.8	45.0	57.6	20.5	66.9	50.0
200	42.7	62.3	N/A	69.2	46.9	47.6	59.5	21.9	69.6	53.8
250	44.4	62.5	N/A	69.6	48.7	50.7	59.4	24.3	68.7	56.0
300	45.8	63.7	N/A	69.1	50.1	51.3	60.0	27.4	69.4	57.1
350	48.4	63.7	N/A	69.2	50.1	53.2	58.5	26.9	68.8	58.1
400	48.7	65.6	N/A	70.2	49.8	54.3	61.2	29.3	70.8	60.3
450	48.2	64.9	N/A	71.2	50.8	55.2	59.8	27.9	68.7	58.7
500	48.4	66.1	N/A	72.6	52.6	56.2	60.2	30.3	68.6	59.6
550	51.1	67.8	N/A	**73.2**	51.8	57.9	**61.9**	30.7	69.6	60.2
600	51.1	66.2	N/A	72.5	53.1	56.6	59.9	31.8	69.1	60.9
650	51.2	66.4	N/A	72.4	54.5	58.5	58.7	31.6	70.7	61.8
700	52.1	65.9	N/A	71.8	53.9	59.0	58.2	33.0	69.4	62.2
750	53.2	67.9	N/A	72.3	55.2	60.9	57.9	32.8	67.7	61.9
800	52.7	67.6	N/A	72.6	54.5	60.2	59.2	32.5	72.1	62.5
850	54.3	64.6	N/A	71.2	54.4	60.1	58.0	34.5	70.3	61.5
900	53.6	66.9	N/A	72.1	55.6	58.9	58.6	35.8	69.9	63.1
950	52.8	**68.1**	N/A	70.9	55.8	61.5	61.0	35.9	70.3	62.1
1000	**54.6**	67.8	N/A	72.6	**56.4**	**62.4**	58.3	**37.0**	**72.9**	**63.8**

Table 11: Rank-1 ACC(%)for OM vs MM when considering various features and MD

The table shows a difference between the categorized and uncategorized data. Here MM performs better than OM for duration. Yet, similar to the analysis with categorization, the best performance for all chunks is obtained using combination of duration-latency.

Hence, the best performance obtained in no categorization scheme was using duration-latency pair when using MM method and MD as distance metrics.

Figure 15 represents the best performance for 8 ranks.

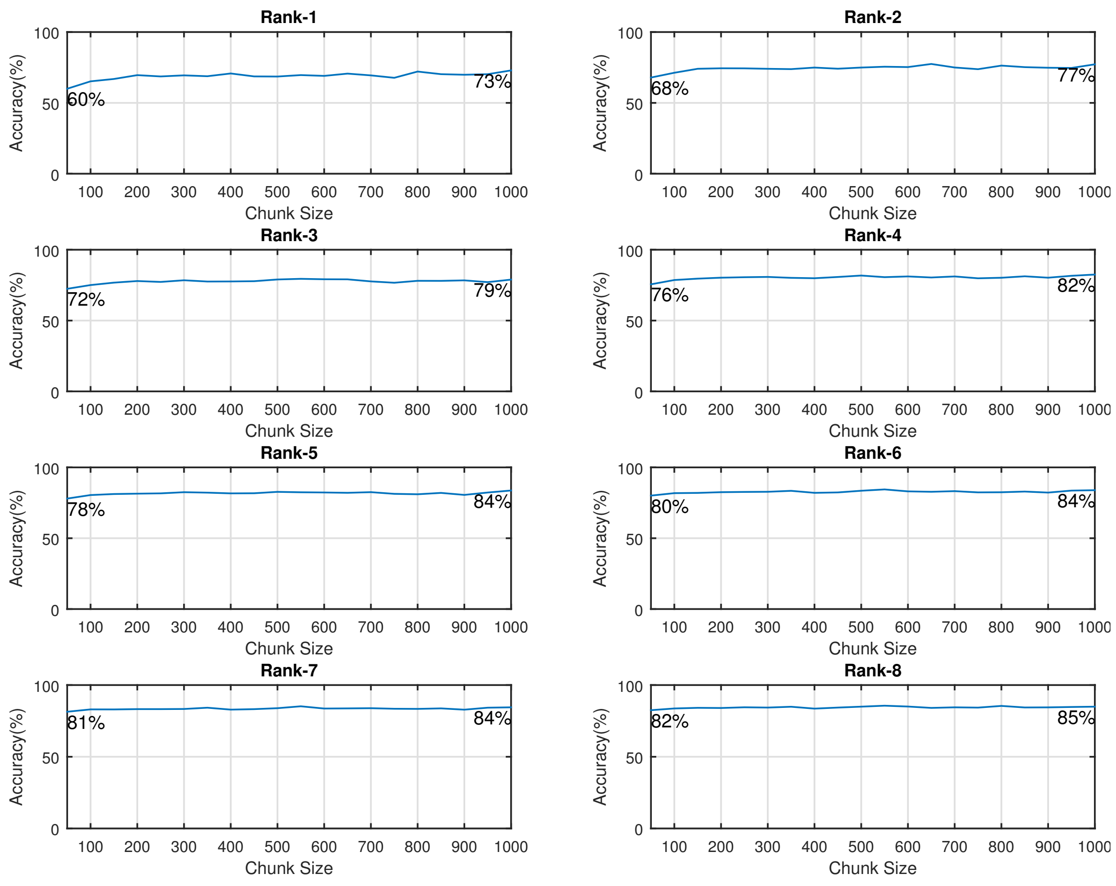


Figure 15: Identification Rate (ACC%) changes of the best setting at various chunk sizes for 8 ranks with no categorization

The graphs show, that in the first rank there is a large difference of ACC scores between chunk size 50 and 1000 meaning that overall, by increasing the chunk size the performance improves. However, in the rank 8 the graph tends to the constant function f(chunk size)=ACC, meaning that regardless of the number of events (between 50 and 1000), more than 82% of the identifications are correct.

From all the results obtained, we can see that overall, the results improve when no categorization has been performed. A very important reason could be the low universality of features resulting low frequency and sometimes unavailable template entries in our template set. When testing, sometimes, there was insufficient data for specific features available for some specific category, forcing us to use the respective *Uncategorized* entry for the same feature or even use the mean value as explained in the analysis part. Hence the intersection between the categorization and no categorization schemes increases thereby increasing the chance of similar results. Also, if we include more software interaction characteristics such as how a user deletes a file using shortcut keys or the delete button etc, along with the software category, it may provide more accuracy on how a specific user behaves on specific software category.

6.3 Results for Analysis with Continuous Authentication Data

As described in section 5.5, we performed an analysis based on ANIA_k and ANGA_k values. Table 12 represents the CA results obtained by [5] using only KD . We performed this analysis for 8 ranks with our best performed settings, i.e. using duration-latency combinations with no categorization, with MM comparison and MD. According to [5], there were 4 classes defined for CA analysis. First class (+/+) included the analysis with no genuine user was locked out and all impostors were locked out. Second class (+/−) included the results with no genuine user locked out and some impostors were not locked out. Third class (−/+), included the results where some genuine users were locked out but every impostor was detected. Finally, class four (−/−) consisted of the results where some genuine locked out and some impostors were not locked out.

		CA			CI							
Class	# Users	ANIA	ANGA	# imp.ND	ACC(1)	ACC(2)	ACC(3)	ACC(4)	ACC(5)	ACC(6)	ACC(7)	ACC(8)
+/+	43	123	∞		61.8	69.4	73.9	76.9	79.0	80.6	81.7	82.7
+/−	3	516	∞	3	**69.2**	**75.5**	**78.1**	80.0	81.3	**82.9**	**83.6**	**84.9**
−/+	5	219	1469		68.4	74.4	**78.1**	**80.2**	**81.8**	82.7	**83.6**	84.1
−/−												

Table 12: Rank-1 to Rank-8 ACC(%) for Continuous Identification based on CA data

As the table shows, on the first class with an ANIA=123, we obtained a rank-1 ACC=61.8% identification accuracy rate. In second class with ANIA=516, we got an ACC=69.2%, and finally in the third class with ANIA=219 and ANGA=1469, we got a rank-1 ACC=68.4%.

The complete set of results can be found in Appendix B.

6.4 Discussion

The summary of our findings by this research are:

- Fusion of two keystroke features resulted in significantly higher ACC compared to using individual features.
- Increasing the chunk size might improve accuracy to a certain point (ACC), beyond which we can observe a saturation behaviour in ACC when increasing the chunk size.
- Comparing the two distance metrics studied in this work, ED and MD, the only significant difference we could see was when using latency as individual feature. One conclusion is that, the ACC results with ED are affected more by the pre-processing procedure and effective outlier removal.
- The variations observed in the ACC due to different comparison methods, or score fusion methods were not large enough to present a strong indication that one method outperforms the other candidates.
- Categorization based on software context has not improved the ACC. Probably because, unavailable template entries for specific categories has been replaced with Uncategorized template entries.

Overall, pre-processing of the collected data when it comes to outlier removal, or post processing of the results to find an optimum choice of the right feature combinations, seem to affect the achieved results more than distance metrics or comparison methodology, or chunk size.

7 Conclusions

In this research, we introduced a novel concept called Continuous Identification (CI) through Keystroke Dynamics (KD). The main idea behind this concept is to identify a locked out user as a result of CA process. During CA process [5], a user's activity is continuously monitored in an action by action manner. That means, on every action performed, the identity of the individual using the system is verified against that of actual logged on user and a trust level is calculated. When this level falls below a lock out threshold, the session for that user is ended and he/she must re-authenticate to regain access to the system.

The most important application of CI is in forensics investigations. In a scenario where an impostor (possibly an adversary) has been locked out of the system, an interesting question is *who is he?* To answer this question, an identification process must be performed as soon as a user is locked out.

In order to perform our analysis, we used a data set of 51 individuals, collected within 5 to 7 days, where they performed their normal daily activities using any computer. Data collection environment was uncontrolled and they could perform any activity such as browsing, chat, programming, document typing, gaming etc. We performed an statistical analysis on the data set in order to assess the possibility for CI. We could not find any relevant related work on this concept. However, we could make use of various previous work on KD and CA.

In order to perform the analysis, we used two distance metrics: *Manhattan Distance (MD)* and *Euclidean Distance (ED)*.

As a secondary objective, we tried to perform our analysis with respect to the software context. That means to identify a user based on the software category he has been using while performing keystroke actions. Hence, in addition to analyzing the keystroke data as whole, we also categorized the keystroke data based on the used software. We created three categories:

1. For the Internet related software, such as browsers and instant messaging applications
2. Documentation, such as word processors
3. Other software

We then performed the comparison between the test and template within each category. But at the end, we achieved a better performance without categorization. We believe that the reason is because when creating templates based on categorization, frequency of various keys was not high, meaning that for many keys we did not have any data with respect to that category. For example, there could be many 'a's in the first category but no 'a' in the second category. Whereas, when analyzing without categorization, there was more availability of data. Here, the template would consist of data for all the keys.

We utilized several features such as, duration of single keys, that is the time between pressing and releasing a key, latency between the two keys which is the time between releasing a key until pressing the next key and relative frequency of single keys. We also used combinations of the mentioned features as pairs of duration-latency and duration-frequency. The best performance was achieved using duration-latency pairs and the lowest performance obtained when using only frequency. This showed that time based features provides more accuracy than frequency. In order to improve the performance for frequency, we can extend this feature to a complete set of Lexical Stylometry[47] features.

We also tried to perform the comparison between test and template data in two ways. First, comparing each of the features for action with the corresponding value in the template. This is called a One to Mean (OM) comparison. Then, we analyzed the performance by comparing the average of features in the test data to the corresponding value stored in the template. This method is called Mean to Mean (MM) comparison. We could not perform the OM comparison on frequency. However, when combining the features, we performed comparisons on each feature individually and then we combined the results. In order to combine the results, for duration-frequency pair we multiplied the individual results together. In order to combine the results for duration-latency pair, we added the individual results. The results showed no significant advantage of OM over MM or vice versa.

In case of comparisons with categorizations, we obtained three comparison scores for each category representing the distance of each user from the the test data. In order to unify the data we performed a fusion using Weighted Average Mean of the individual scores. We performed the analysis with fixed weights ($w_1 = 3, w_2 = 2, w_3 = 1$) and variable weights where each weight indicates the frequency of test elements in associated with that category in the test data. At the end, we cannot see any real difference between the ACC results when using fixed weights compared to the ACC results using variable weights.

Based on the analysis, best approach was MM comparison of duration-latency pairs without software categories taken into account, when MD was used. The performance was almost 61% of Identification Accuracy (ACC) rate after 50 events and almost 72% after 1000 events.

We also applied the mentioned best setting based on the results obtained using an existing CA performance analysis[5]. Using ANGA (if available) as chunk size when comparing a users data with his own template and ANIA when comparing with other templates. Based on the three classes of users defined in [5], we obtained an ACC of approximately 62% for the first class, 69% for the second class and 68% for the third class.

To summarize, an investigation on possibility of Continuous Identification was performed. Although the results are promising for future research, there is a need for improving the accuracy before integrating to a deployed CA system. One way of achieving this would be to include more Behavioral Biometric features.

8 Future Work

Based on the major findings as a result of this work, there are various directions for the future development on the subject of Continuous Identification.

Concerning this work there are various open issues that require further investigation:

1. According to the results, performance with no categorization of keystrokes based on software context was better than the analysis with categorization taken into consideration. A possible reason for this would be the missing template entries for various categories of a user's template. Should software categorization be ruled out as an ACC improvement method? Or can it be implemented in a more useful and constructive way? We believe, for instance, that acquisition of more data of various categories for template creation may improve the performance.
2. The findings showed that the performance was affected significantly by the pre-processing tasks and feature combining methods. Will introducing more keystroke features, such as digits, navigation keys, combination keys, etc, have a significant contribution to higher ACC? How are the different features best combined? What is the practical limit for the number of combined features?
3. How can we improve the outlier identification process? Shall all 'odd' entries such as high negative latencies be considered outlier? Can it be optimized based on the software categorization?
4. How can the person identification methodology investigated during this work, evolve to a CI system?

There are also several other ways to carry forward the research on CI systems. A few suggestions are presented here.

- Since, the model by Bours and Mondal has been extended to a multi-modal system including mouse dynamics[6], the research can also be extended to include mouse dynamics.
- The CI system will benefit enormously by utilizing Machine Learning.
- The performance we achieved was obtained when using a *closed-set identification*. A closed-set identification is defined as an identification where the the unknown identity is in database. On the other side, an *Open-set identification* is an identification where the unknown user may not be in the database of identities[1]. In order to find out if the person is not in the database of identities or not, an open-set identification can be performed.
- Another interesting activity will be to look into active attack scenarios and verify the performance of a deployed CA/CI system in live action.

[1]`http://www.technovelgy.com/ct/Technology-Article.asp?ArtNum=78` last accessed 30-05-2015

Bibliography

[1] Stenvi, Robin, overbo, Magnus, Johansen, & Lasse. 2013. Behaviour logging tool-belt. http://brage.bibsys.no/xmlui/handle/11250/143069. Visited May 2015.

[2] Gollmann, D. 1999. *Computer Security*. John Wiley & Sons, Inc.

[3] Bours, P. 2012. Continuous keystroke dynamics: A different perspective towards biometric evaluation. *Inf. Secur. Tech. Rep.*, 17(1-2), 36–43.

[4] Mondal, S. & Bours, P. 2013. Continuous authentication using mouse dynamics. In *International Conference of the Biometrics Special Interest Group (BIOSIG)*, 1–12. IEEE.

[5] Bours, P. & Mondal, S. 2015. Continuous authentication with keystroke dynamics. In *Recent Advances in User Authentication Using Keystroke Dynamics*, volume 2, 41–58. Science Gate Publishing.

[6] Mondal, S. & Bours, P. 2015. Continuous authentication in a real world settings. In *Eighth International Conference on Advances in Pattern Recognition (ICAPR)*, 1–6. IEEE.

[7] Jain, A., Ross, A., & Prabhakar, S. 2004. An introduction to biometric recognition. *Transactions on Circuits and Systems for Video Technology, IEEE*, 14(1), 4–20.

[8] Delac, K. & Grgic, M. 2004. A survey of biometric recognition methods. In *Proceedings Elmar 2004:46th International Symposium Electronics in Marine*, 184–193. IEEE.

[9] Bolle, R., Connell, J., Pankanti, S., Ratha, N., & Senior, A. 2003. *Guide to Biometrics*. SpringerVerlag.

[10] Banerjee, S. P. & Woodard, D. L. 2012. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1), 116–139.

[11] Gunetti, D. & Picardi, C. 2005. Keystroke analysis of free text. *ACM Trans. Inf. Syst. Secur.*, 8(3), 312–347.

[12] Kang, P. & Cho, S. 2015. Keystroke dynamics-based user authentication using long and free text strings from various input devices. *Information Sciences*, 308(0), 72 – 93.

[13] Tappert, C. C., Villani, M., & Cha, S.-H. 2009. Keystroke biometric identification and authentication on long-text input. *Behavioral biometrics for human identification: Intelligent applications*, 342–367.

[14] Monrose, F. & Rubin, A. 1997. Authentication via keystroke dynamics. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 48–56. ACM.

[15] Ross, A. & Jain, A. K. 2004. Multimodal biometrics: An overview. In *Proceedings of 12th European Signal Processing Conference (EUSIPCO*, 1221–1224.

[16] Prabhakar, S. & Jain, A. K. 2002. Decision-level fusion in fingerprint verification. *Pattern Recognition*, 35(4), 861 – 874.

[17] Ross, A. & Jain, A. 2003. Information fusion in biometrics. *Pattern Recognition Letters*, 24(13), 2115–2125.

[18] Maxion, R. & Killourhy, K. 2010. Keystroke biometrics with number-pad input. In *International Conference on Dependable Systems and Networks (DSN)*, 201–210. IEEE/IFIP.

[19] Gaines, R. 1980. *Authentication by Keystroke Timing: Some Preliminary Results*. Rand.

[20] Bours, P. & Fullu, C. 2009. A login system using mouse dynamics. In *Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP '09)*, 1072–1077. IEEE.

[21] Saevanee, H. & Bhattarakosol, P. 2009. Authenticating user using keystroke dynamics and finger pressure. In *6th IEEE Consumer Communications and Networking Conference (CCNC)*, 1–2. IEEE.

[22] Alsultan, A. & Warwick, K. 2013. User-friendly free-text keystroke dynamics authentication for practical applications. In *International Conference on Systems, Man, and Cybernetics (SMC)*, 4658–4663. IEEE.

[23] Monaco, J., Bakelman, N., Cha, S.-H., & Tappert, C. 2012. Developing a keystroke biometric system for continual authentication of computer users. In *European Intelligence and Security Informatics Conference (EISIC)*, 210–216. IEEE.

[24] Zhong, Y., Deng, Y., & Jain, A. 2012. Keystroke dynamics for user authentication. In *Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 117–123. IEEE.

[25] Ahmed, A. & Traore, I. 2014. Biometric recognition based on free-text keystroke dynamics. *Transactions on Cybernetics, IEEE*, 44(4), 458–472.

[26] Dowland, P. & Furnell, S. 2004. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In *Security and Protection in Information Processing Systems*, Deswarte, Y., Cuppens, F., Jajodia, S., & Wang, L., eds, 275–289. Springer US.

[27] Shanmugapriya, D. & Padmavathi, G. 2009. A survey of biometric keystroke dynamics: Approaches, security and challenges. *International Journal of Computer Science and Information Security*.

[28] Douhou, S. & Magnus, J. R. 2009. The reliability of user authentication through keystroke dynamics. *Statistica Neerlandica*, 63(4), 432–449.

[29] Clarke, N., Furnell, S., Lines, B., & Reynolds, P. 2003. Using keystroke analysis as a mechanism for subscriber authentication on mobile handsets. In *Security and Privacy in the Age of Uncertainty*, Gritzalis, D., De Capitani di Vimercati, S., Samarati, P., & Katsikas, S., eds, 97–108. Springer US.

[30] Vizer, L. M. 2009. Detecting cognitive and physical stress through typing behavior. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, 3113–3116. ACM.

[31] Pamplona Segundo, M., Sarkar, S., Goldgof, D., Silva, L., & Bellon, O. 2013. Continuous 3d face authentication using rgb-d cameras. In *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 64–69. IEEE.

[32] Janakiraman, R., Kumar, S., Zhang, S., & Sim, T. 2005. Using continuous face verification to improve desktop security. In *Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTIONS '05)*, 501–507. IEEE.

[33] Guennoun, M., Abbad, N., Talom, J., Rahman, M., & El-Khatib, K. 2009. Continuous authentication by electrocardiogram data. In *Toronto International Conference Science and Technology for Humanity (TIC-STH)*, 40–42. IEEE.

[34] Shen, C., Cai, Z., & Guan, X. 2012. Continuous authentication for mouse dynamics: A pattern-growth approach. In *42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 1–12. IEEE/IFIP.

[35] Traore, I., Woungang, I., Obaidat, M., Nakkabi, Y., & Lai, I. 2012. Combining mouse and keystroke dynamics biometrics for risk-based authentication in web environments. In *Fourth International Conference on Digital Home (ICDH)*, 138–145. IEEE.

[36] Shepherd, S. 1995. Continuous authentication by analysis of keyboard typing characteristics. In *European Convention on Security and Detection*, 111–114. IET.

[37] Zheng, N., Paloski, A., & Wang, H. 2011. An efficient user verification system via mouse movements. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 139–150. ACM.

[38] Rigas, I., Economou, G., & Fotopoulos, S. 2012. Human eye movements as a trait for biometrical identification. In *Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, 217–222. IEEE.

[39] Wenchao, W. & Limin, S. 2012. A fingerprint identification algorithm based on wavelet transformation characteristic coefficient. In *International Conference on Systems and Informatics (ICSAI)*, 1–3. IEEE.

[40] Matta, R., Lau, J., Agrafioti, F., & Hatzinakos, D. 2011. Real-time continuous identification system using ecg signals. In *24th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 001313–001316. IEEE.

[41] Kukreja, U., Stevenson, W., & Ritter, F. 2006. Rui: Recording user input from interfaces under windows and mac os x. *Behavior Research Methods*, 38(4), 656–659.

[42] Arroyo, E., Selker, T., & Wei, W. 2006. Usability tool for analysis of web designs using mouse tracks. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, 484–489. ACM.

[43] Gamboa, H. & Fred, A. 2003. A user authentication technic using a web interaction monitoring system. In *Pattern Recognition and Image Analysis*, Perales, F., Campilho, A., de la Blanca, N., & Sanfeliu, A., eds, 246–254. Springer Berlin Heidelberg.

[44] Hawkins, D. M. 1980. *Identification of Outliers*. Springer Netherlands.

[45] L.Sunitha, M.BalRaju, J.Sasikiran, & Ramana, E. 2014. Automatic outlier identification in data mining using iqr in real-time data. *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, 3(6), 7255–7256.

[46] Website, N. 2015. http://xlinux.nist.gov/dads//HTML/lmdistance.html. 18-05-2015.

[47] Abbasi, A. & Chen, H. 2008. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Trans. Inf. Syst.*, 26(2), 7:1–7:29.

A Appendix A

A.1 Results with Software Categorization and MM

A.1.1 Based on Distance Metrics with Fixed Weights

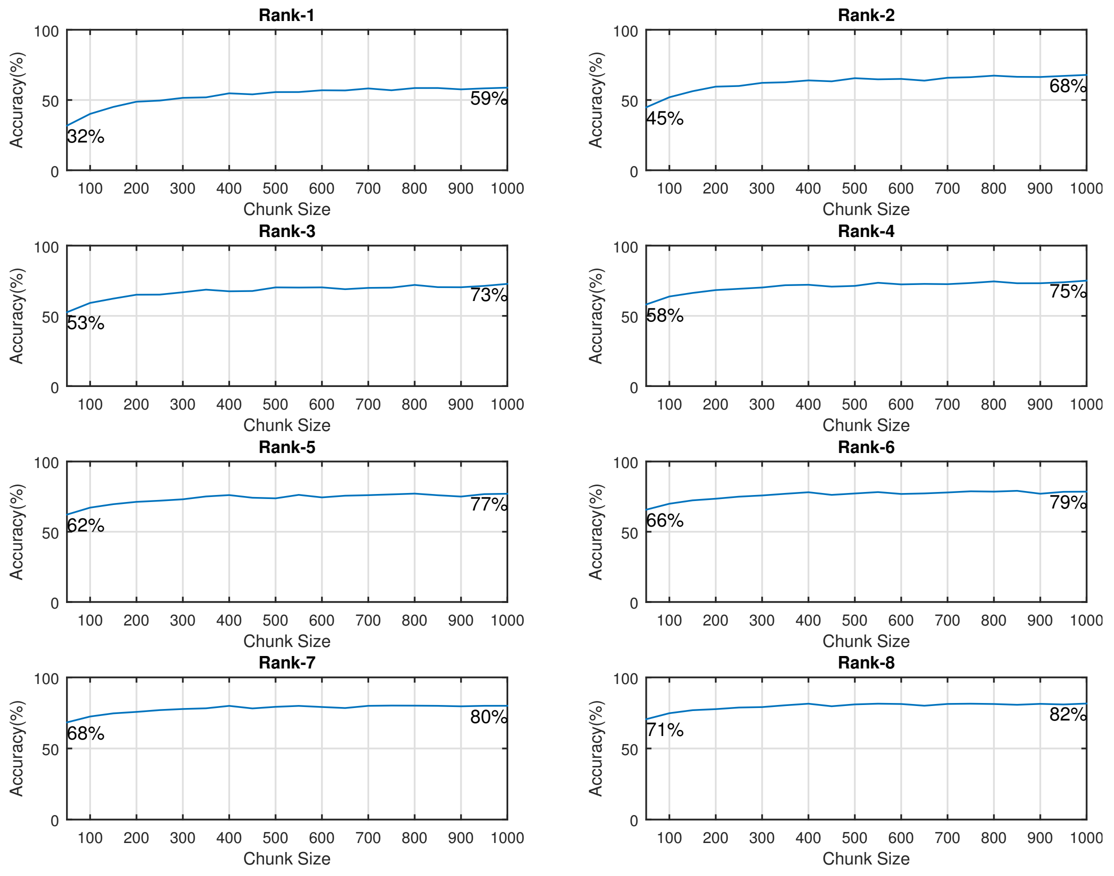


Figure 16: ACC(%) for **duration** vs chunk size for various rank, with **MM**, **fixed** weights, and **MD**

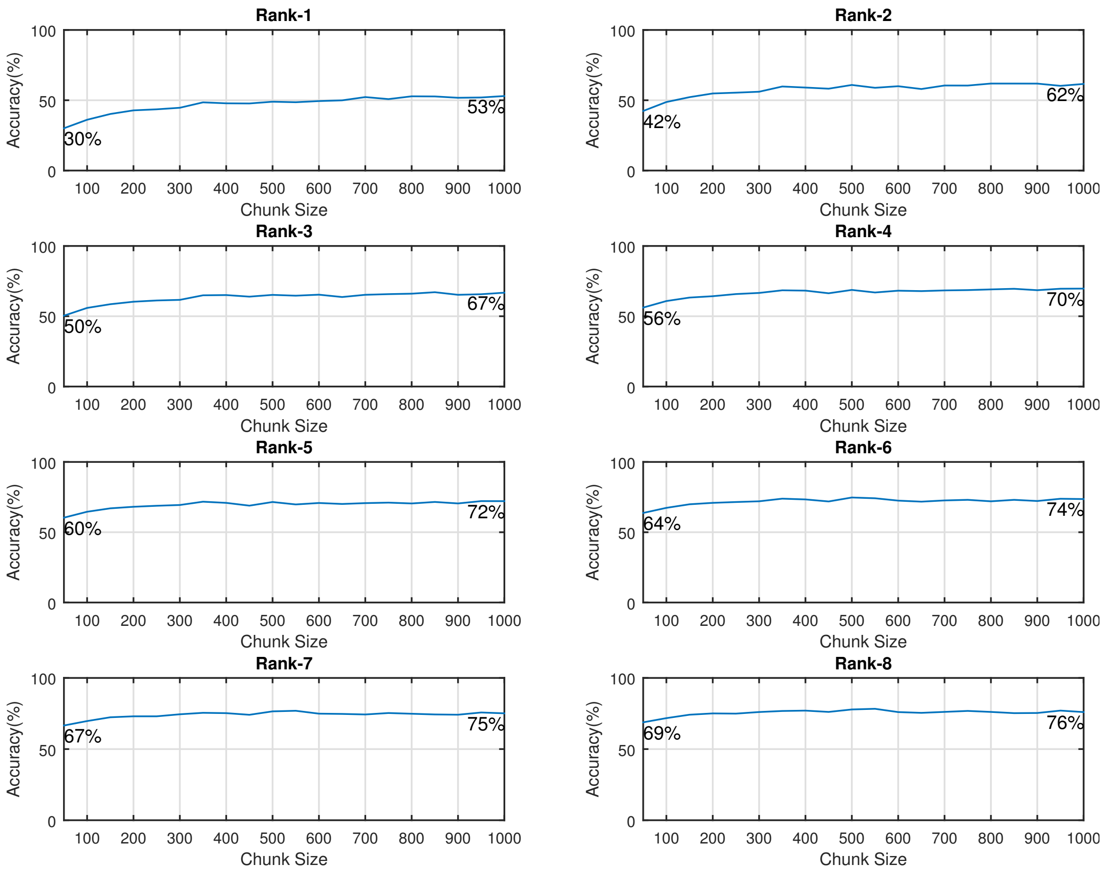


Figure 17: ACC(%) for **duration** vs chunk size for various rank, with **MM**, **fixed** weights, and **ED**

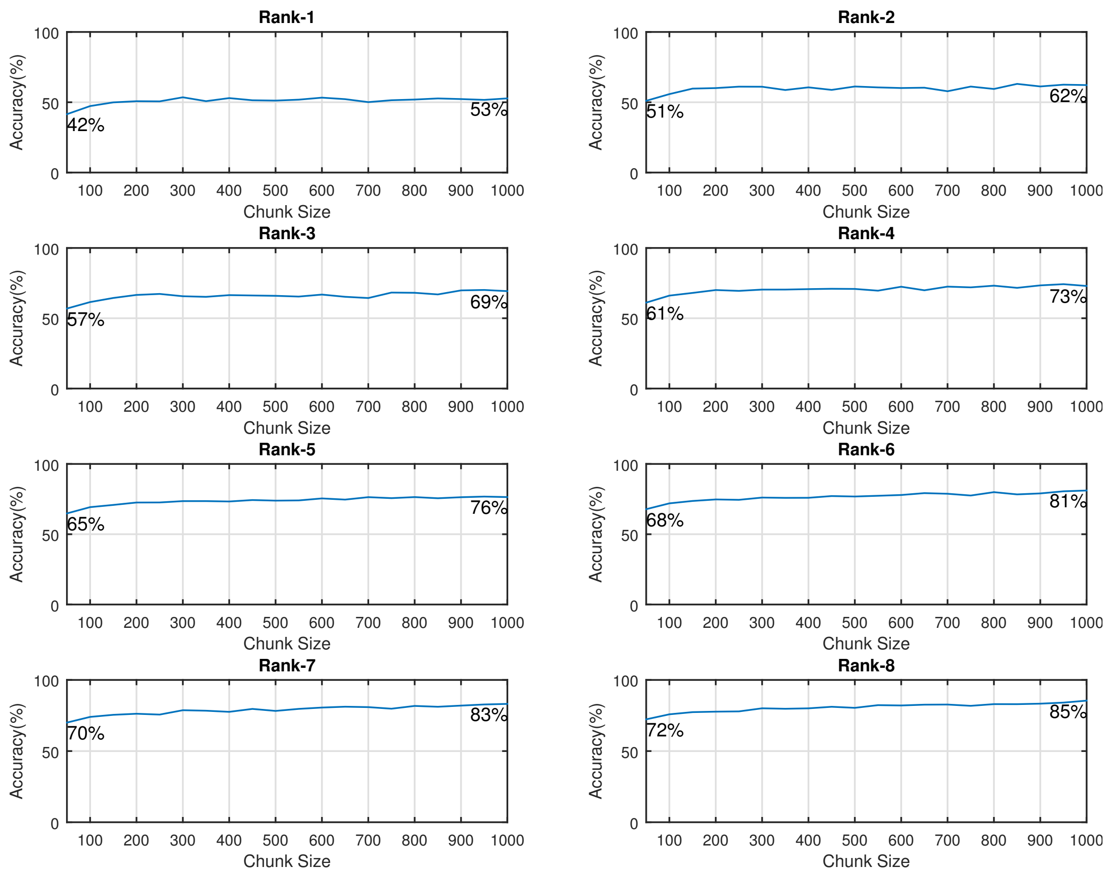


Figure 18: ACC(%) for **latency** vs chunk size for various rank, with **MM**, **fixed** weights, and **MD**

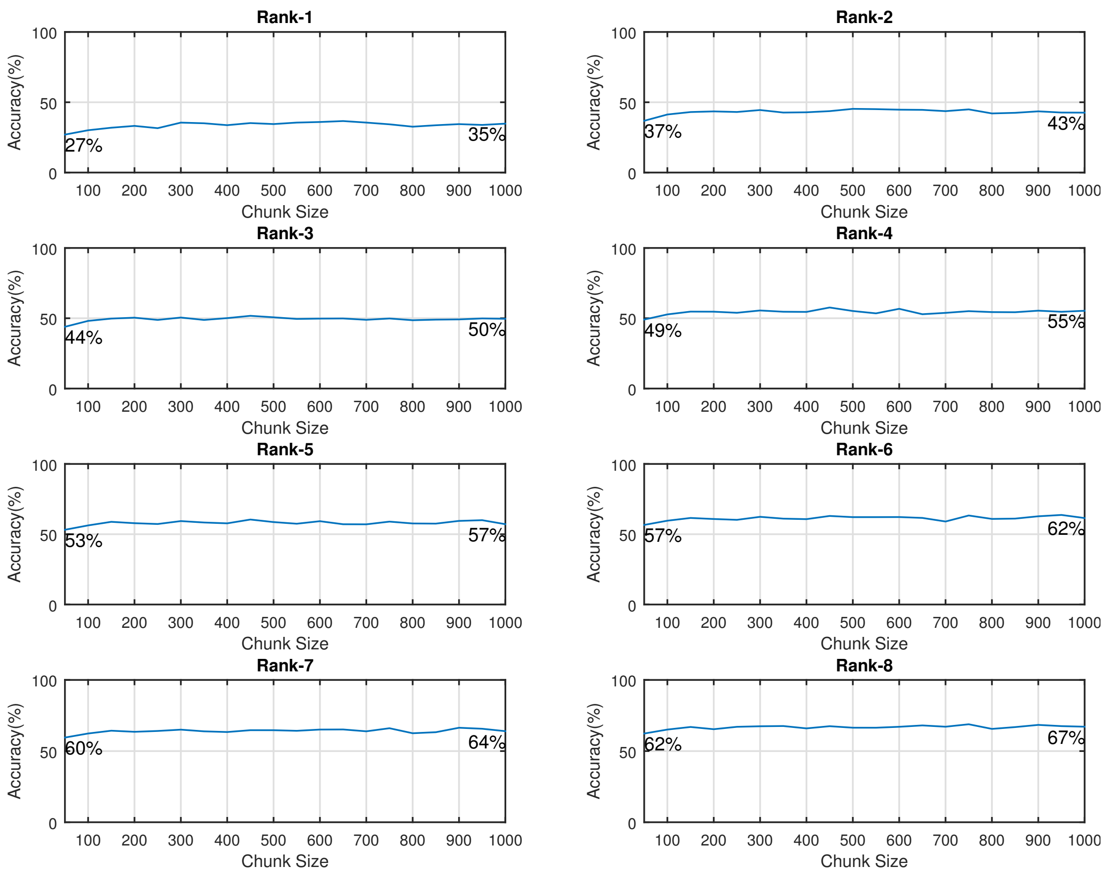


Figure 19: ACC(%) for **latency** vs chunk size for various rank, with **MM**, **fixed** weights, and **ED**

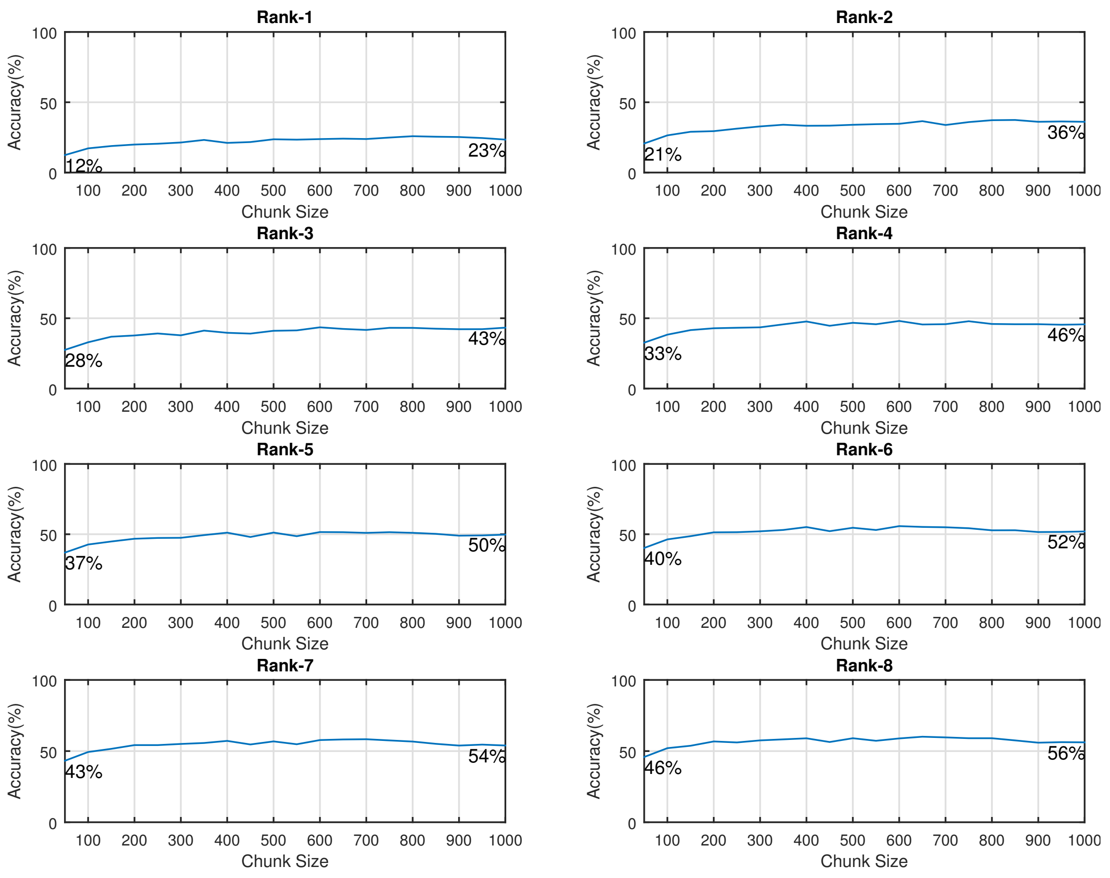


Figure 20: ACC(%) for **frequency** vs chunk size for various rank, with **MM**, **fixed** weights, and **MD**

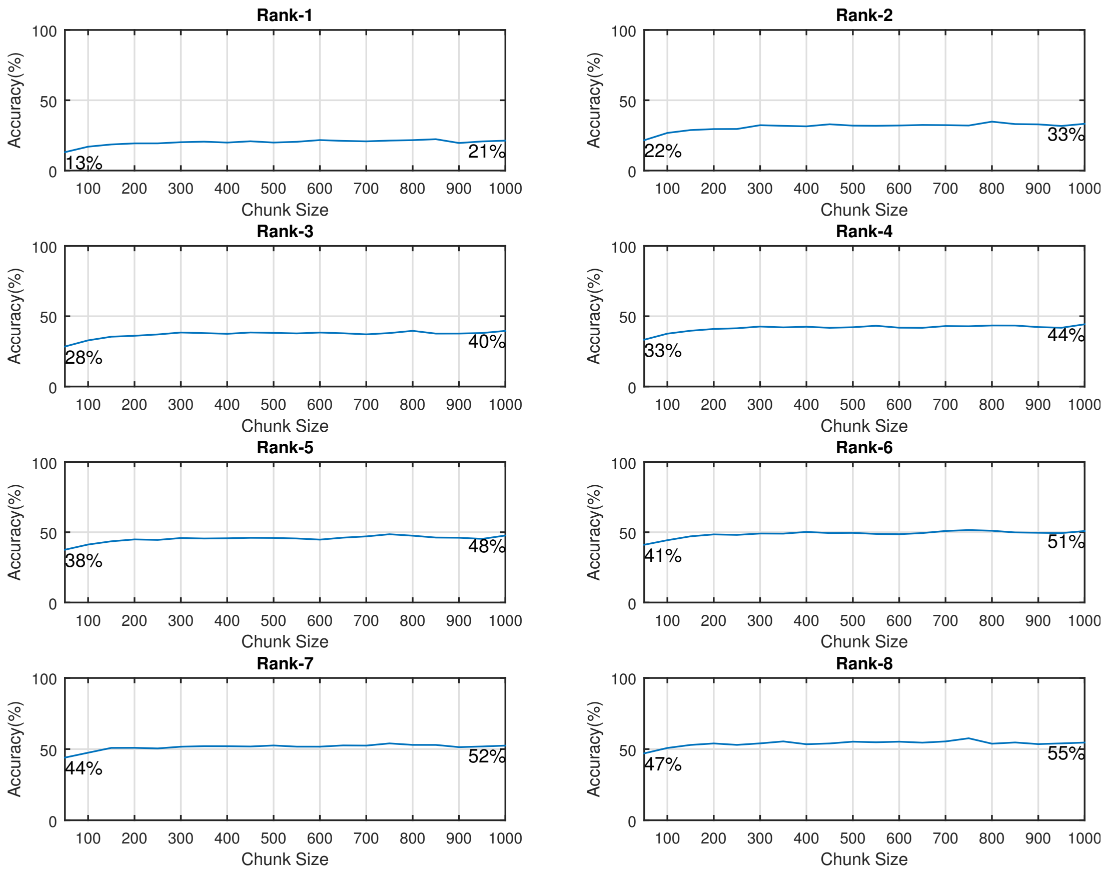


Figure 21: ACC(%) for **frequency** vs chunk size for various rank, with **MM**, **fixed** weights, and **ED**

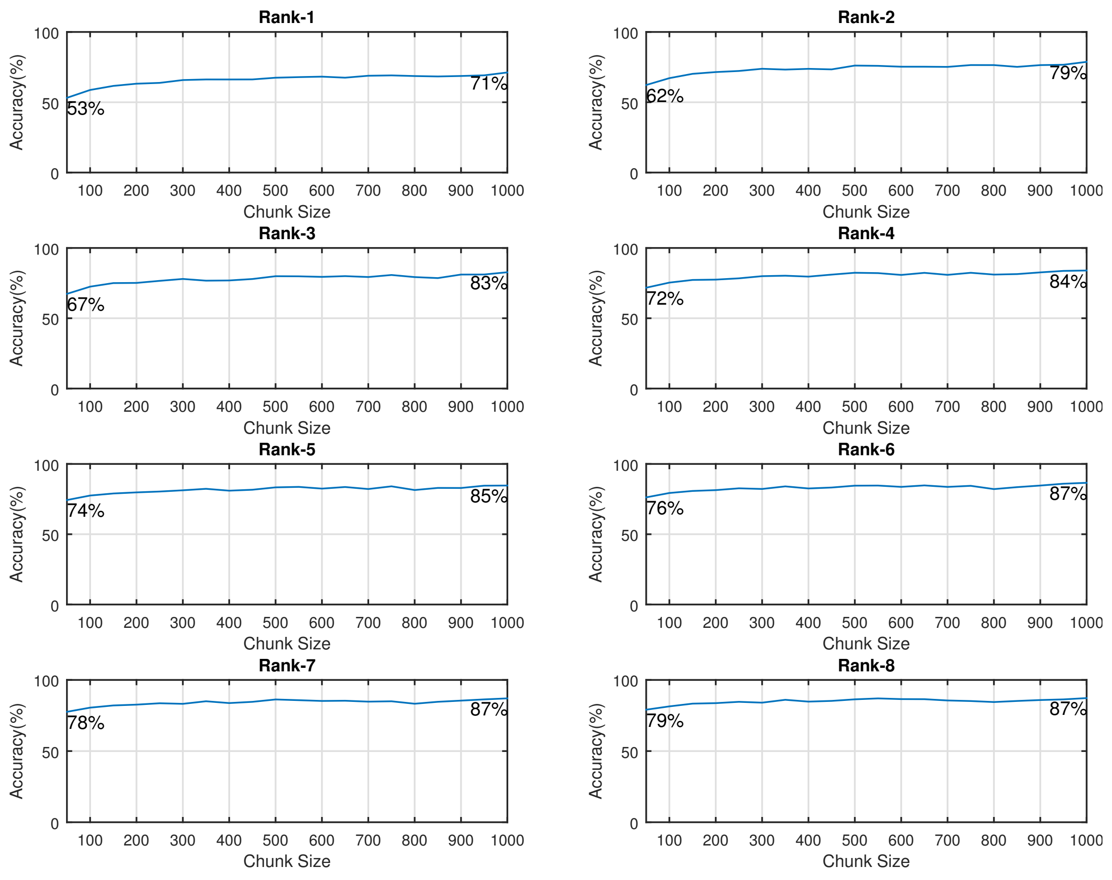


Figure 22: ACC(%) for **duration-latency** vs chunk size for various rank, with **MM**, **fixed** weights, and **MD**

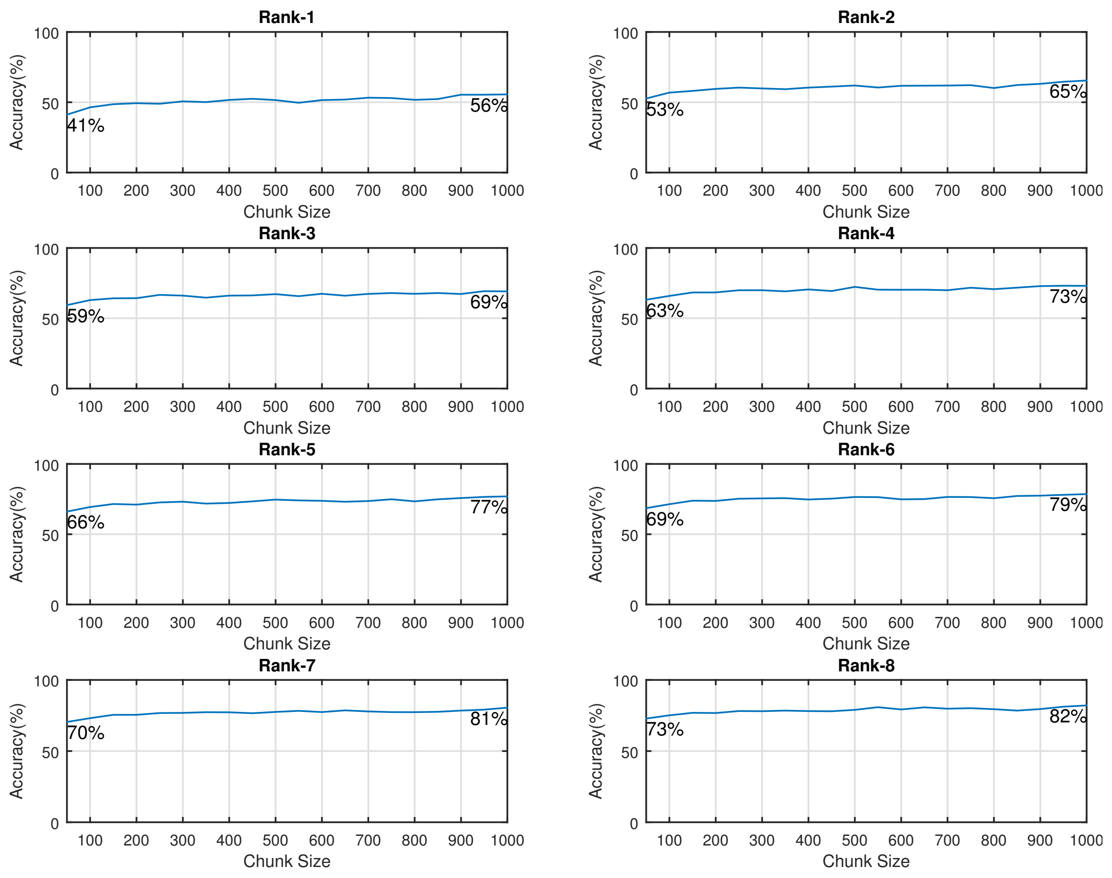


Figure 23: ACC(%) for **duration-latency** vs chunk size for various rank, with **MM**, **fixed** weights, and **ED**

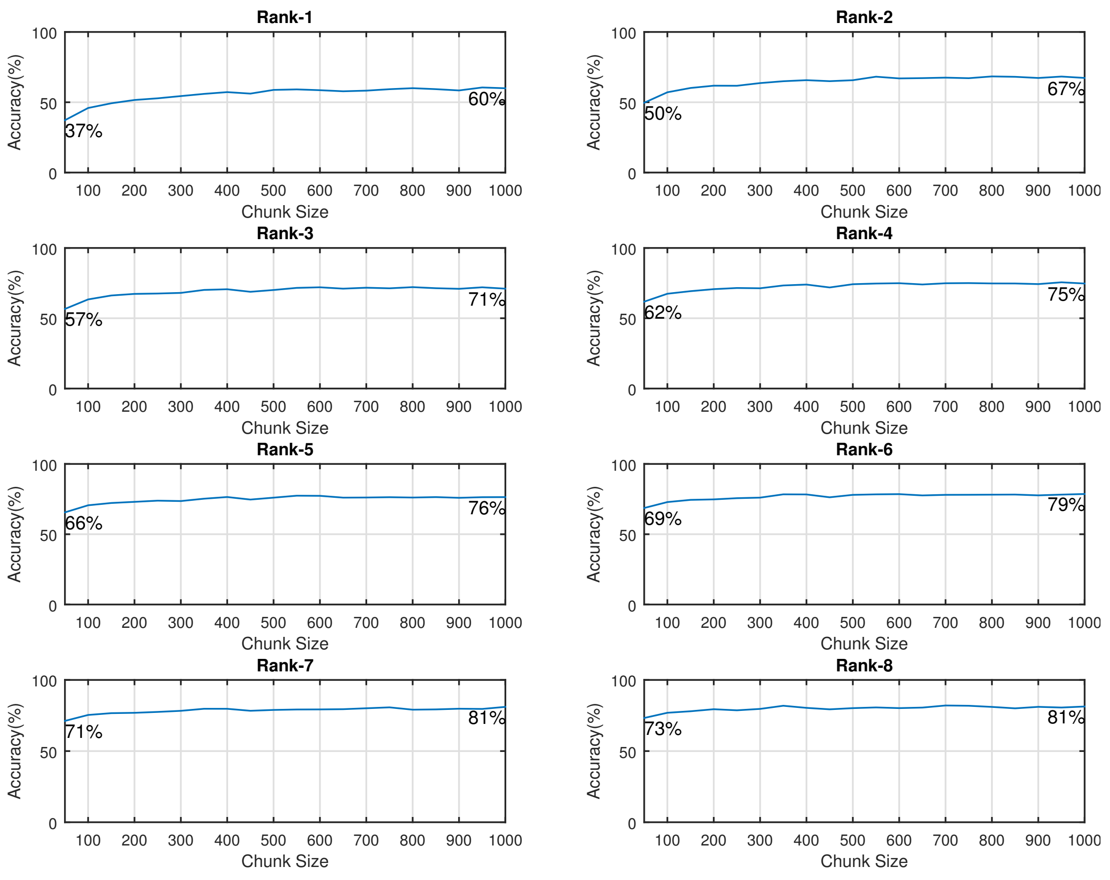


Figure 24: ACC(%) for **duration-frequency** vs chunk size for various rank, with **MM**, **fixed** weights, and **MD**

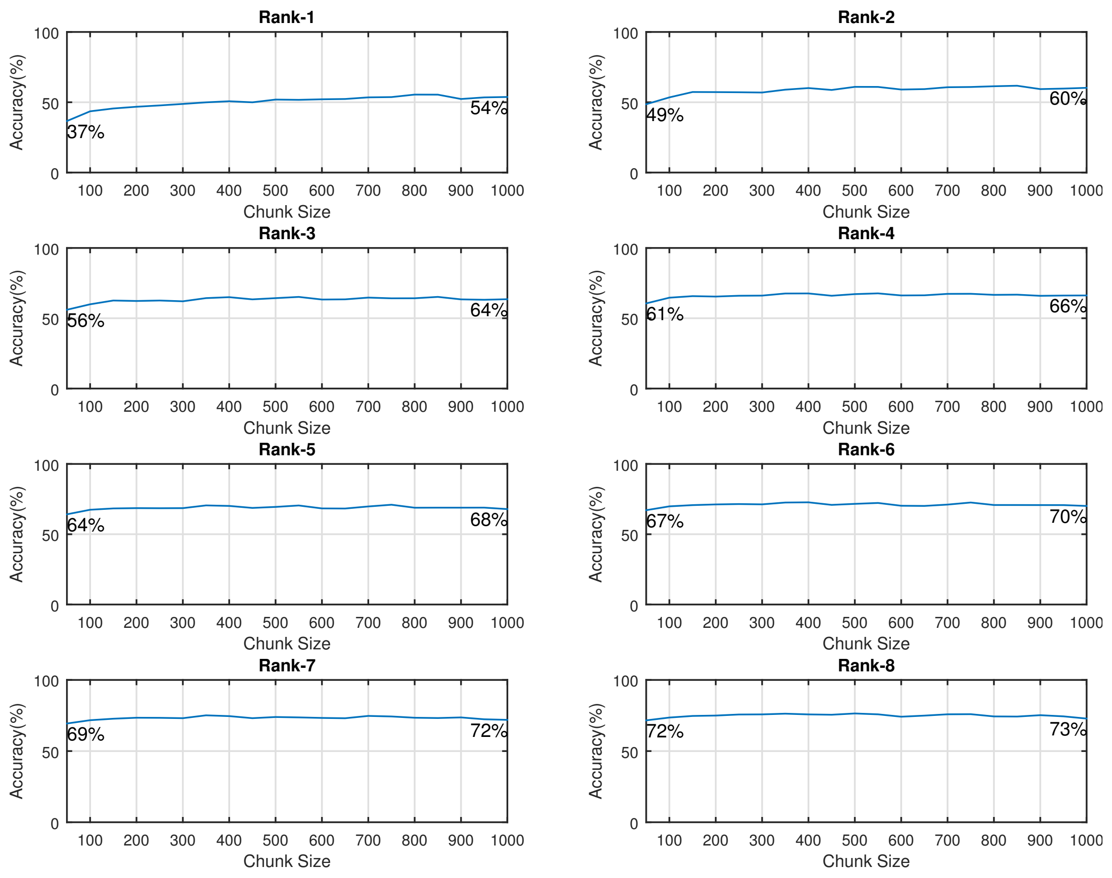


Figure 25: ACC(%) for **duration-frequency** vs chunk size for various rank, with **MM**, **fixed** weights, and **ED**

A.1.2 Based on Distance Metrics with Variable Weights

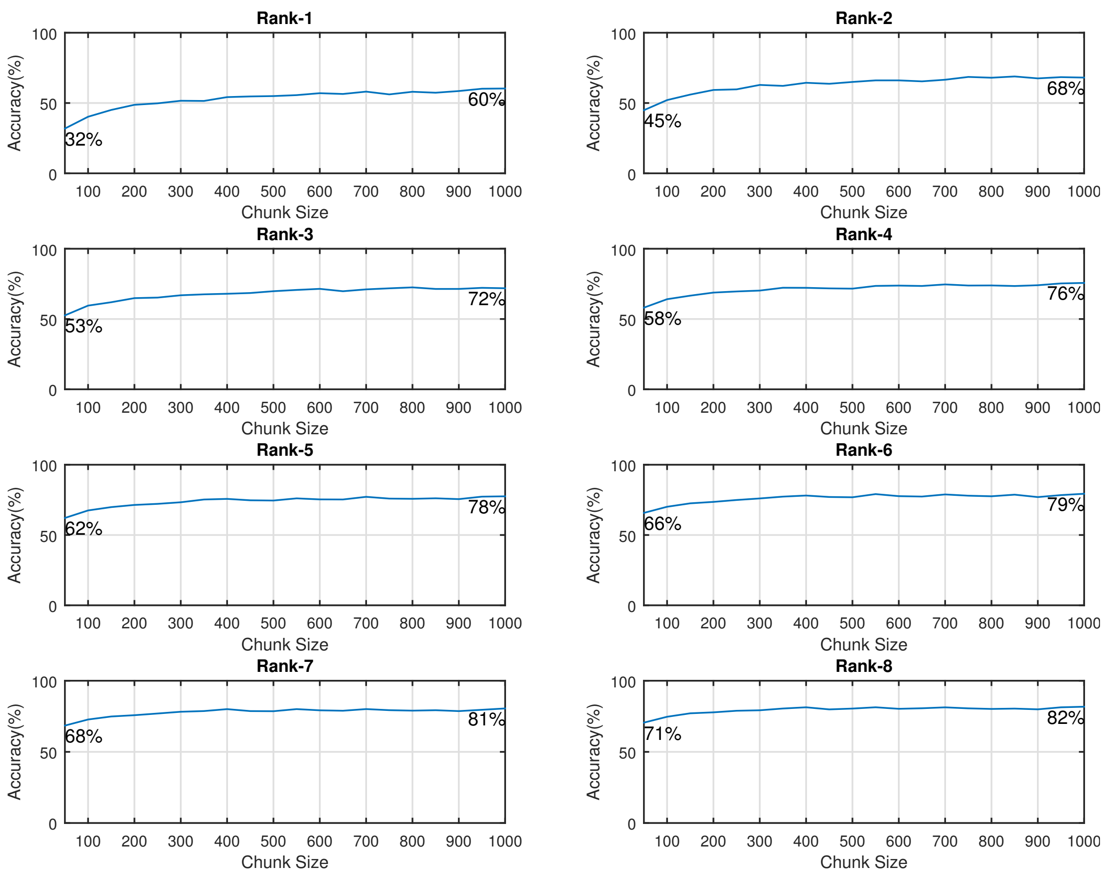


Figure 26: ACC(%) for **duration** vs chunk size for various rank, with **MM**, **variable** weights, and **MD**

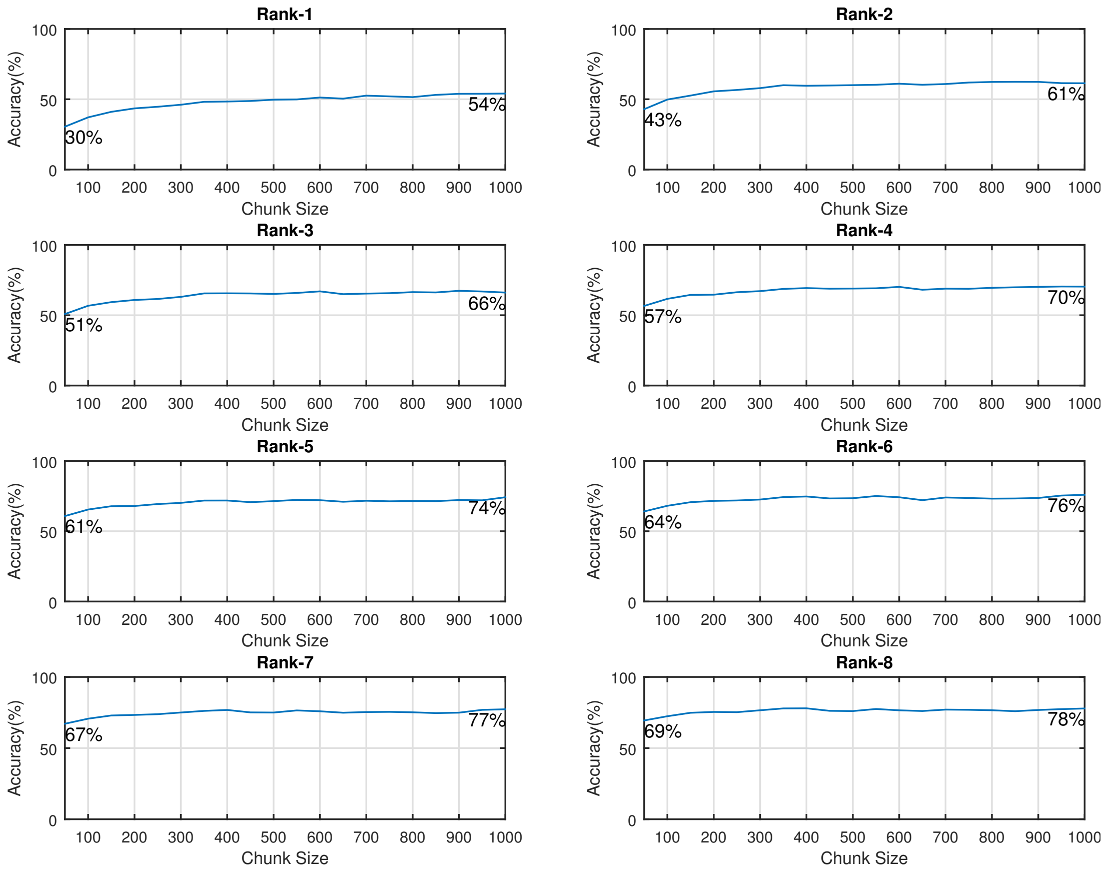


Figure 27: ACC(%) for **duration** vs chunk size for various rank, with **MM**, **variable** weights, and **ED**

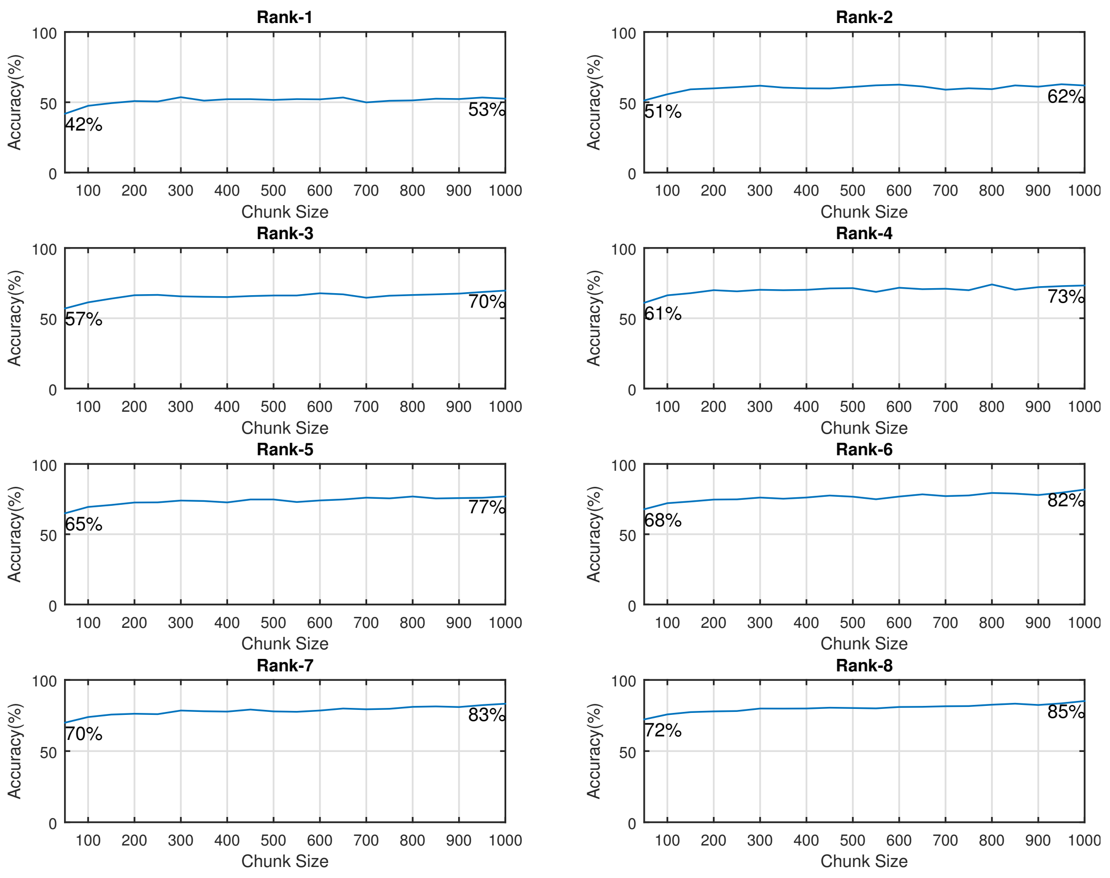


Figure 28: ACC(%) for **latency** vs chunk size for various rank, with **MM**, **variable** weights, and **MD**

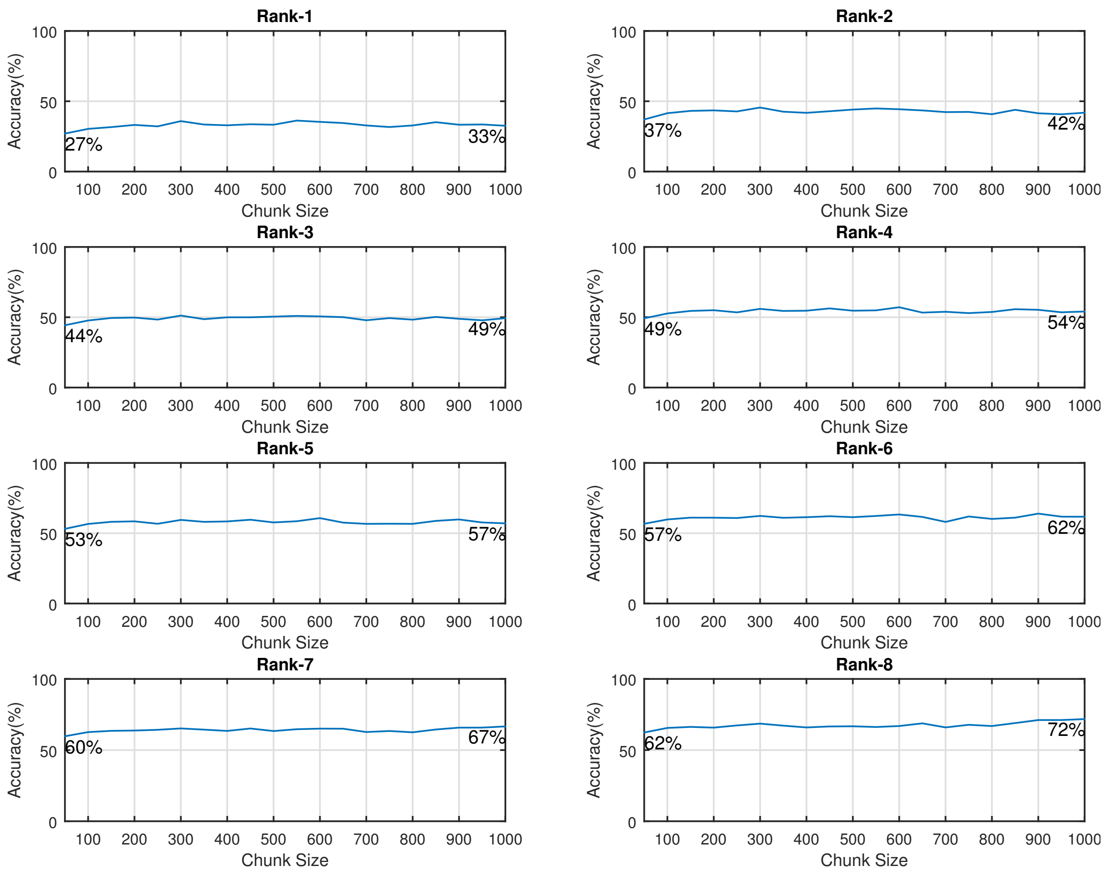


Figure 29: ACC(%) for **latency** vs chunk size for various rank, with **MM**, **variable** weights, and **ED**

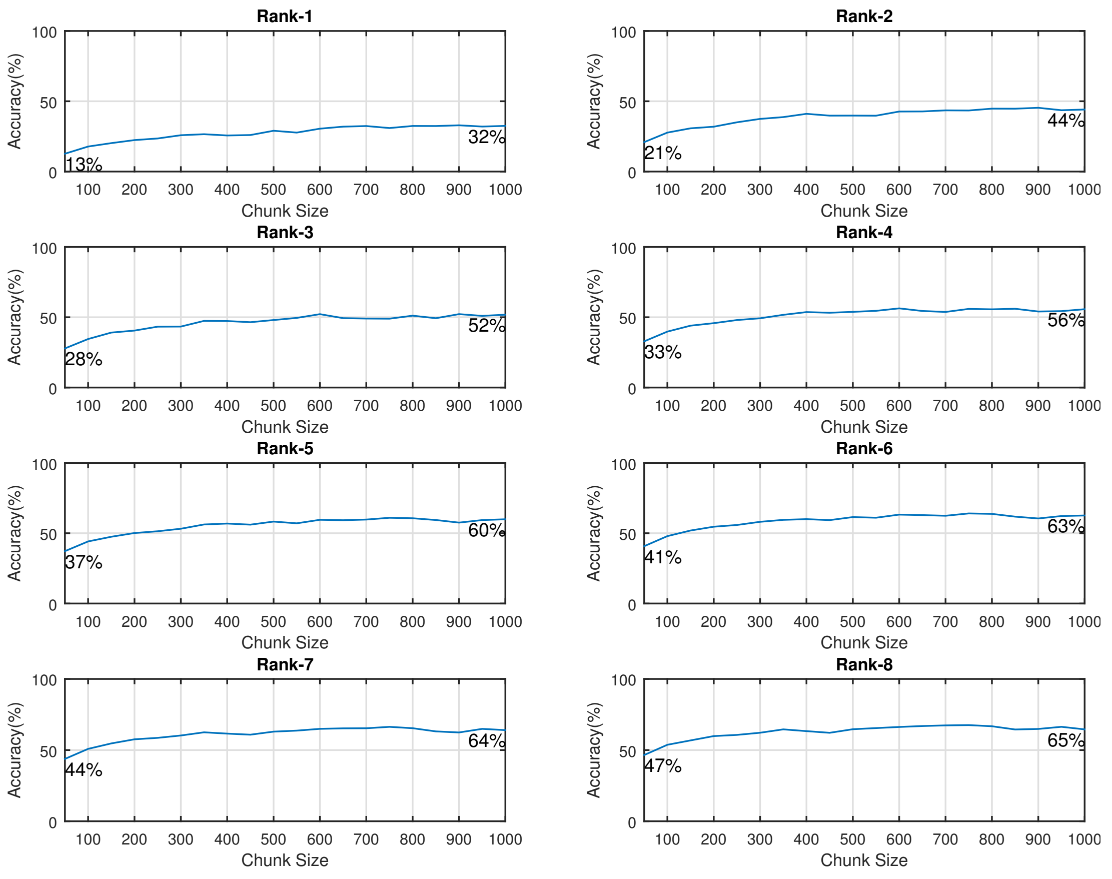


Figure 30: ACC(%) for **frequency** vs chunk size for various rank, with **MM**, **variable** weights, and **MD**

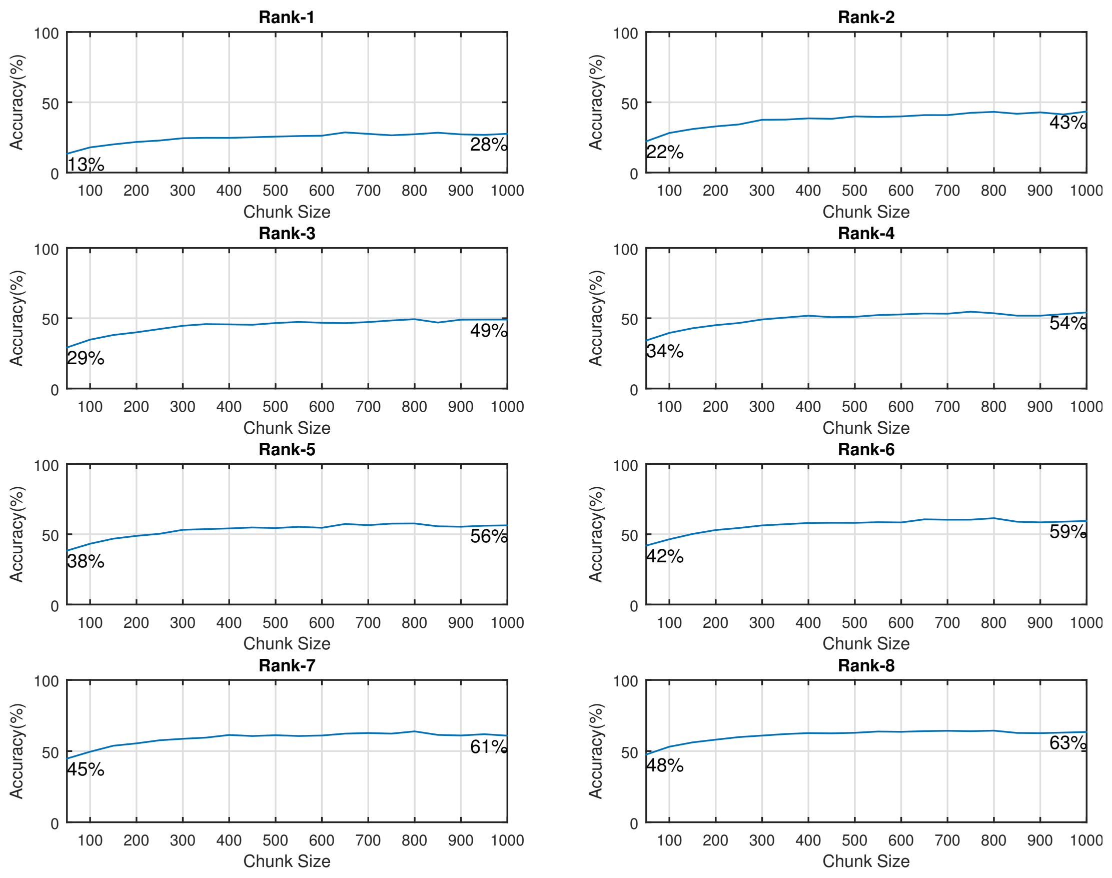


Figure 31: ACC(%) for **frequency** vs chunk size for various rank, with **MM**, **variable** weights, and **ED**

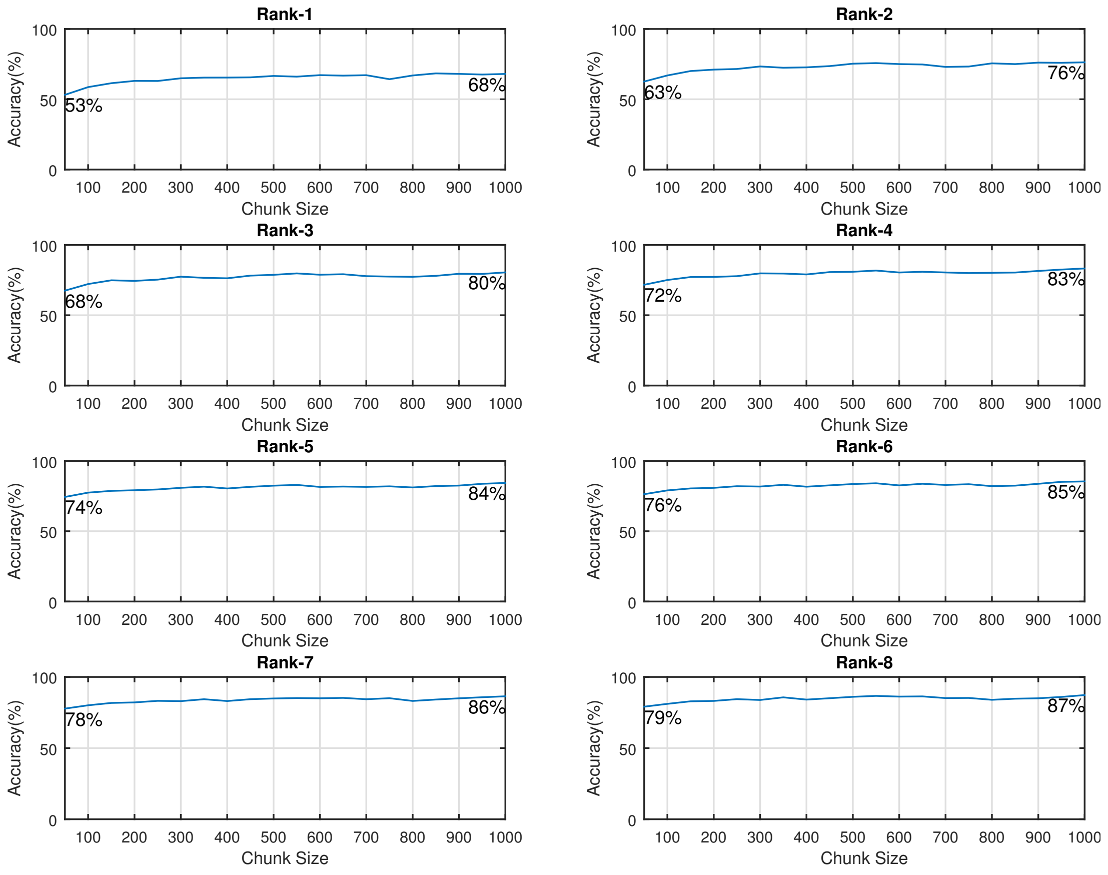


Figure 32: ACC(%) for **duration-latency** vs chunk size for various rank, with **MM**, **variable** weights, and **MD**

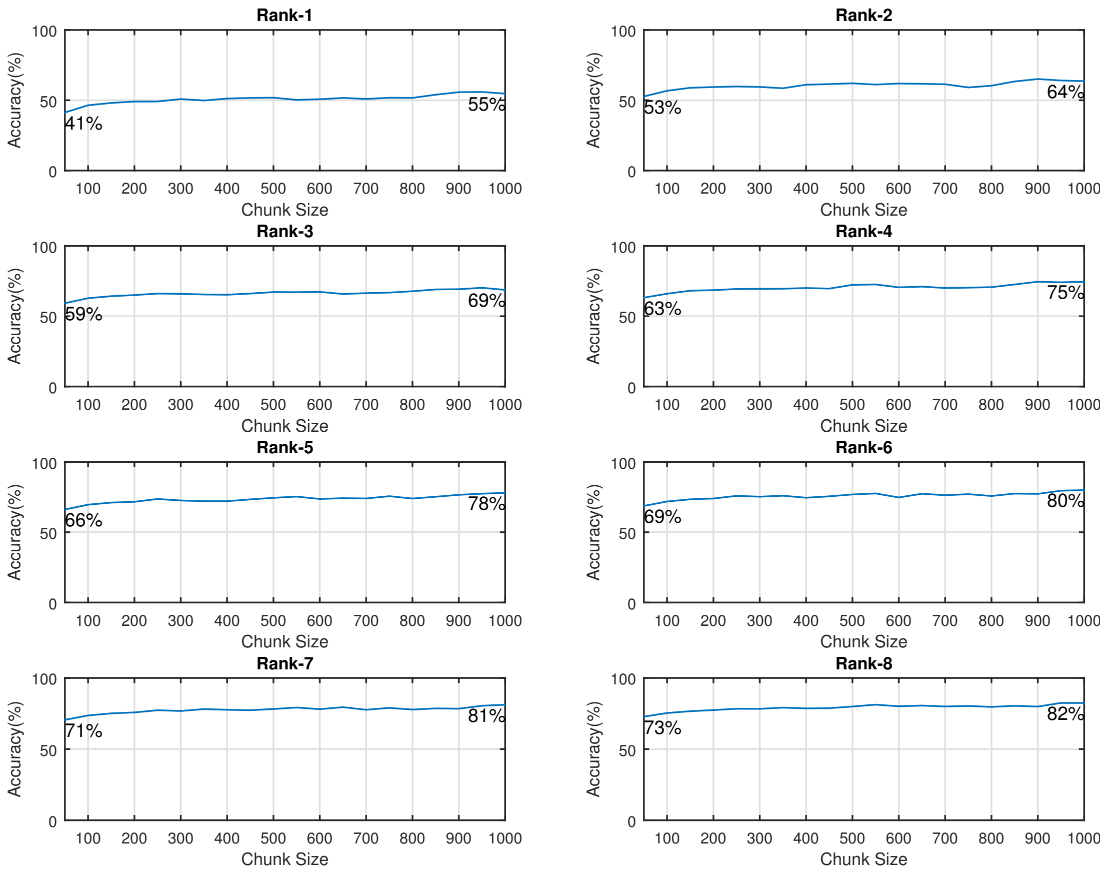


Figure 33: ACC(%) for **duration-latency** vs chunk size for various rank, with **MM**, **variable** weights, and **ED**

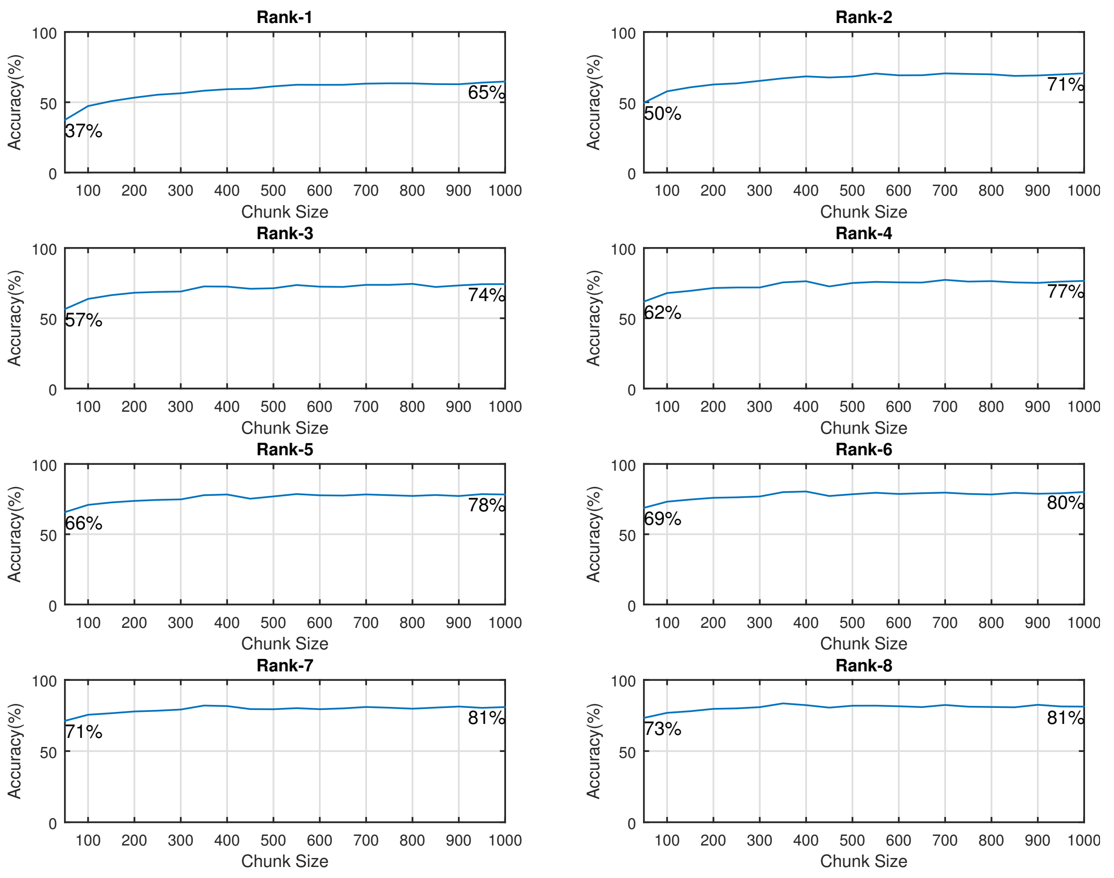


Figure 34: ACC(%) for **duration-frequency** vs chunk size for various rank, with **MM**, **variable** weights, and **MD**

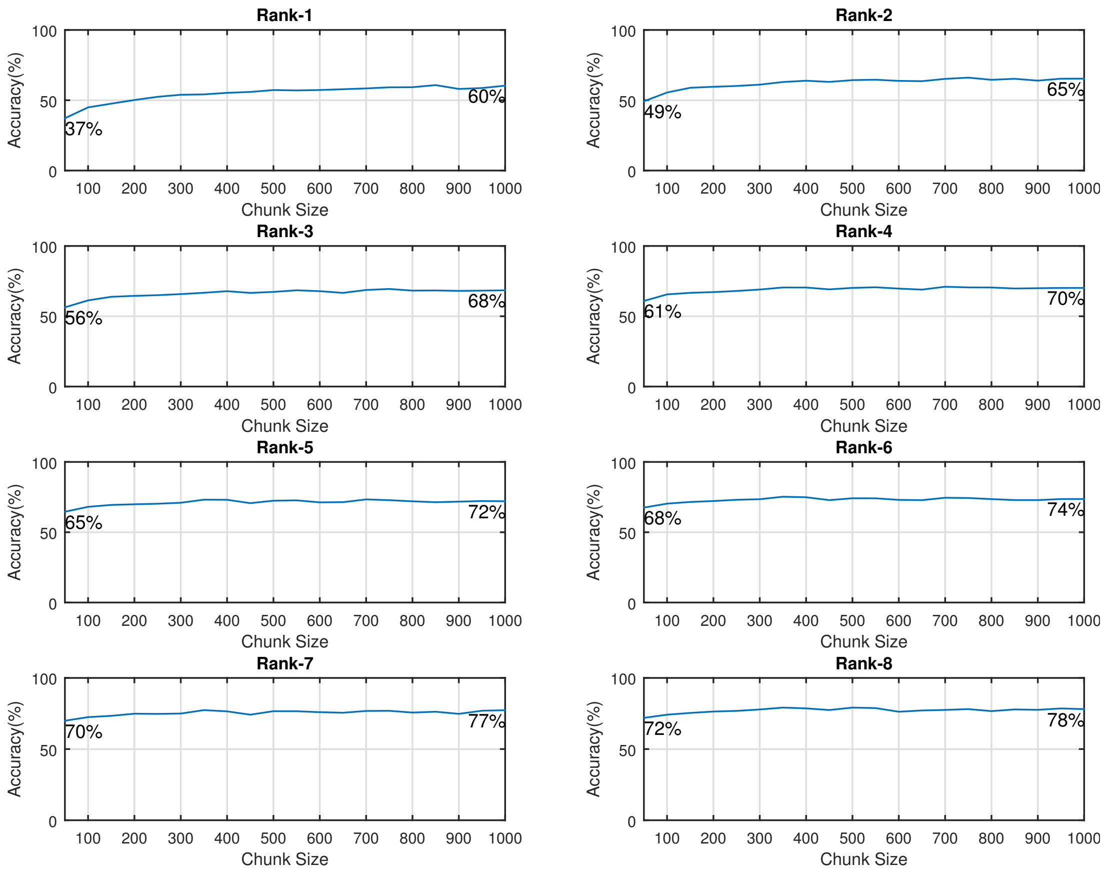


Figure 35: ACC(%) for **duration-frequency** vs chunk size for various rank, with **MM**, **variable** weights, and **ED**

A.2 Results with Software Categorization and OM

A.2.1 Based on Distance Metrics with Fixed Weights

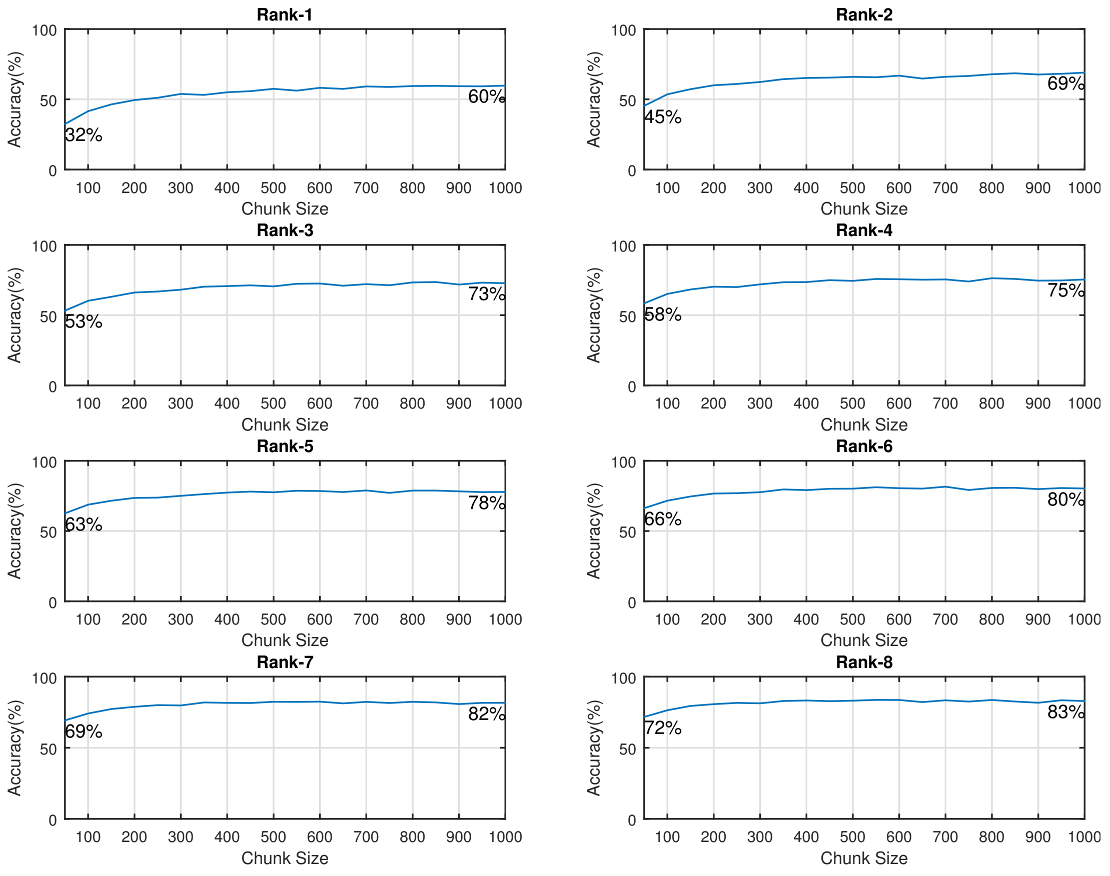


Figure 36: ACC(%) for **duration** vs chunk size for various rank, with **OM**, **fixed** weights, and **MD**

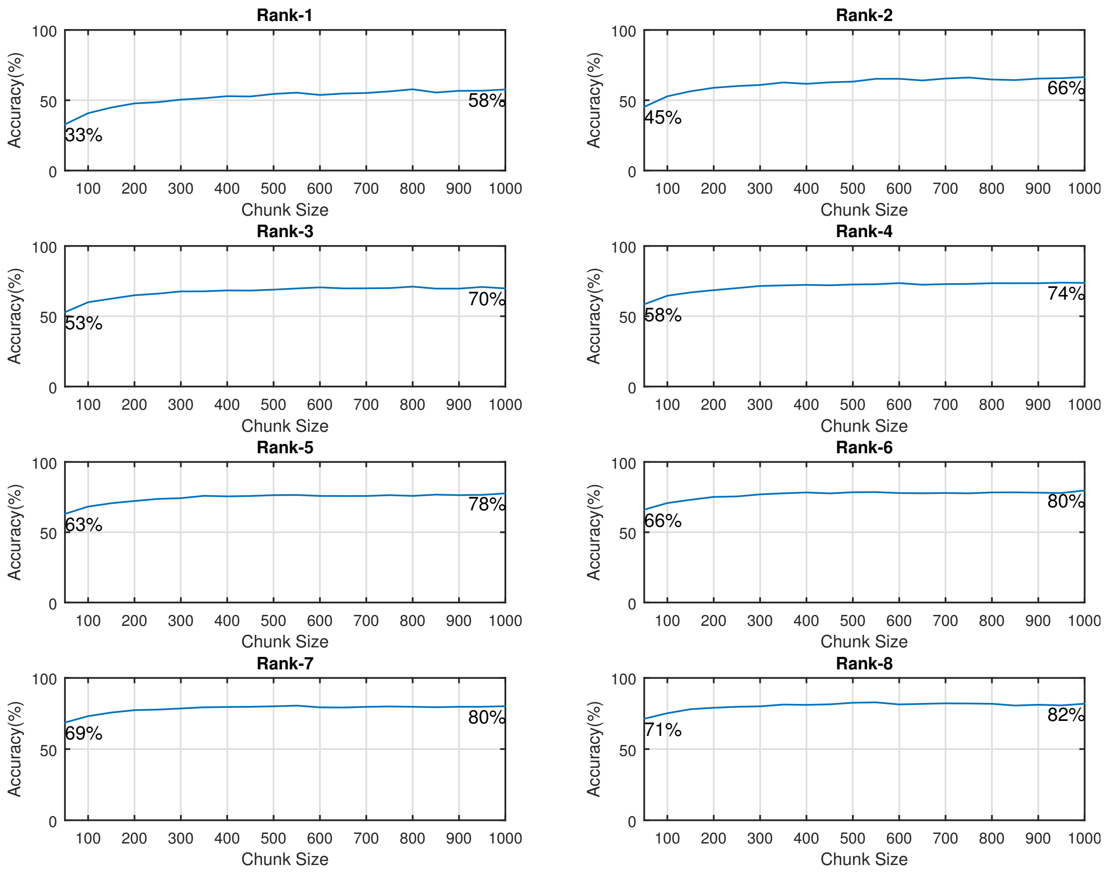


Figure 37: ACC(%) for **duration** vs chunk size for various rank, with **OM**, **fixed** weights, and **ED**

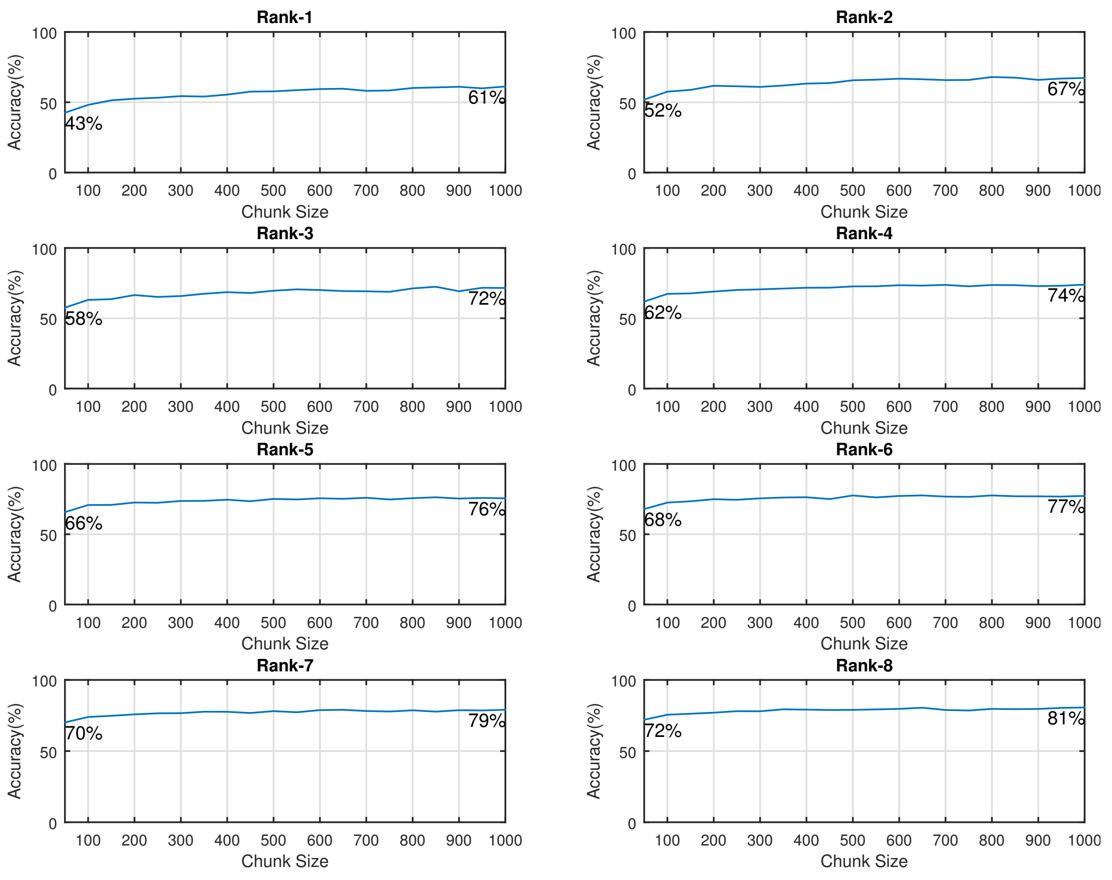


Figure 38: ACC(%) for **latency** vs chunk size for various rank, with **OM**, **fixed** weights, and **MD**

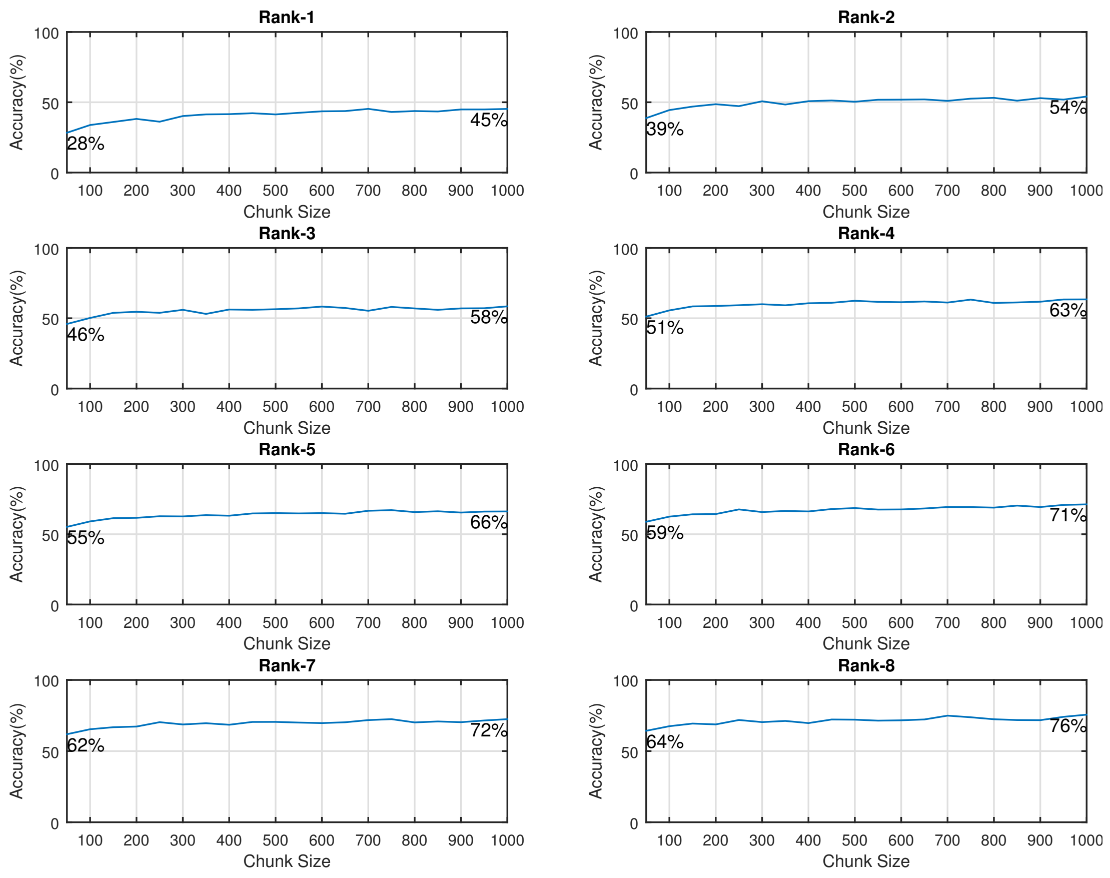


Figure 39: ACC(%) for **latency** vs chunk size for various rank, with **OM**, **fixed** weights, and **ED**

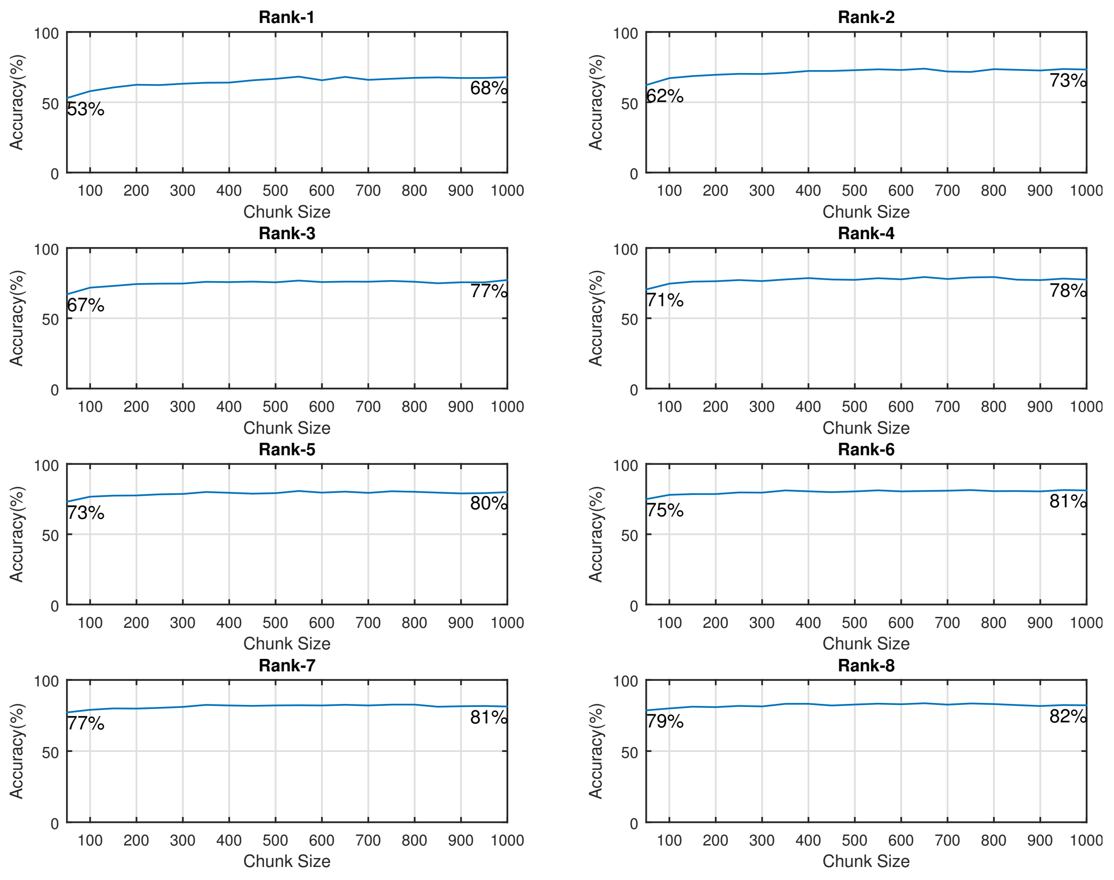


Figure 40: ACC(%) for **duration-latency** vs chunk size for various rank, with **OM**, **fixed** weights, and **MD**

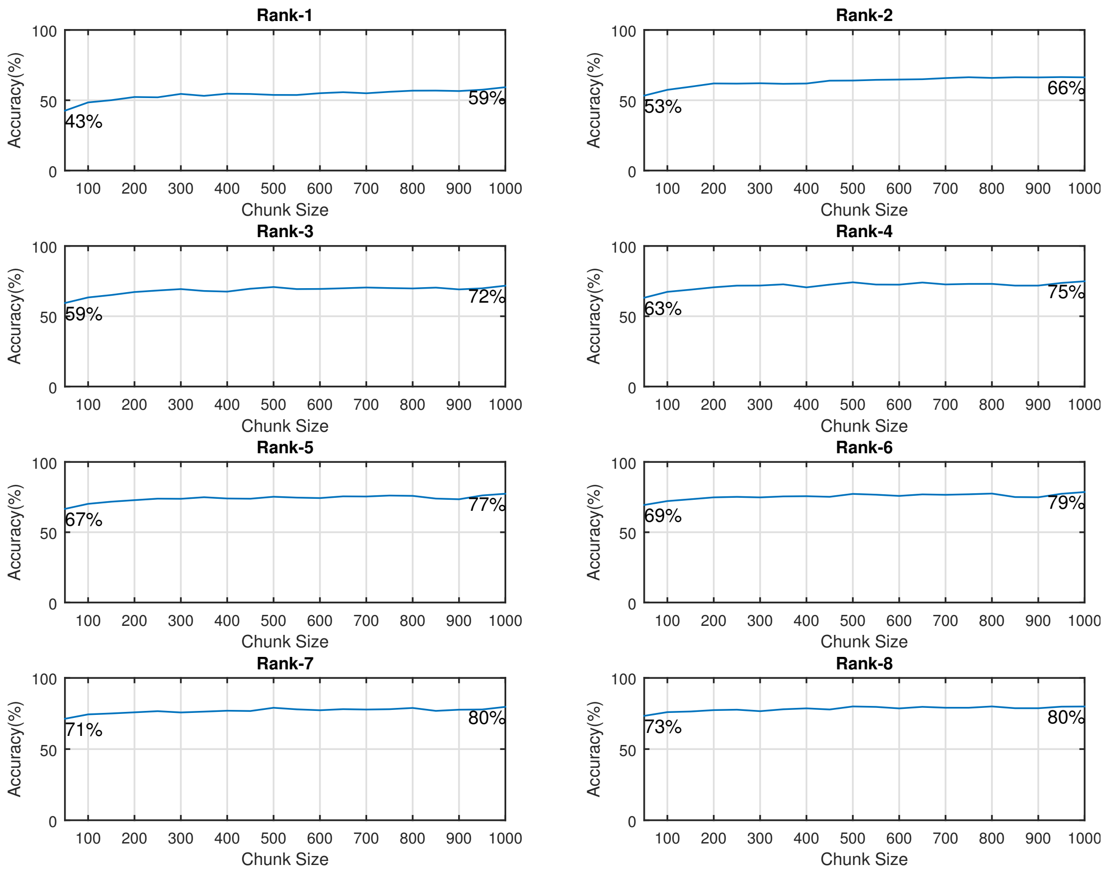


Figure 41: ACC(%) for **duration-latency** vs chunk size for various rank, with **OM**, **fixed** weights, and **ED**

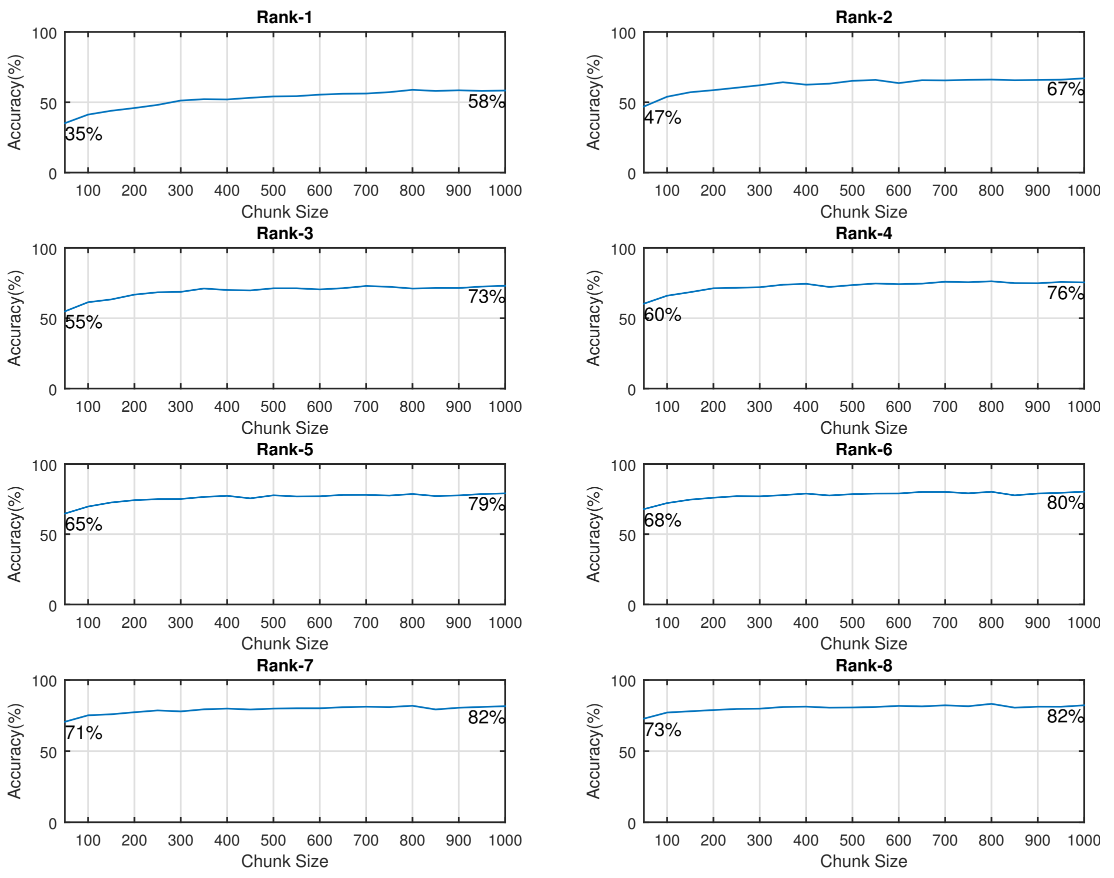


Figure 42: ACC(%) for **duration-frequency** vs chunk size for various rank, with **OM**, **fixed** weights, and **MD**

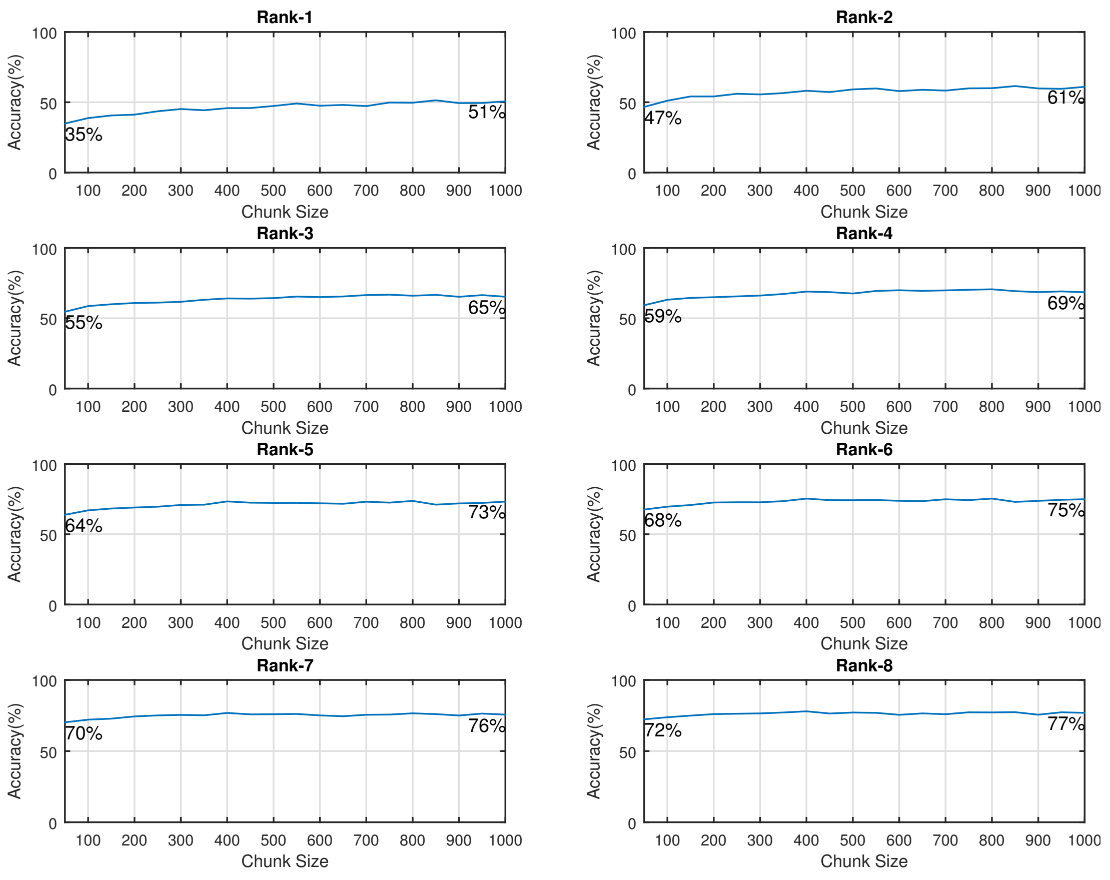


Figure 43: ACC(%) for **duration-frequency** vs chunk size for various rank, with **OM**, **fixed** weights, and **ED**

A.2.2 Based on Distance Metrics with Variable Weights

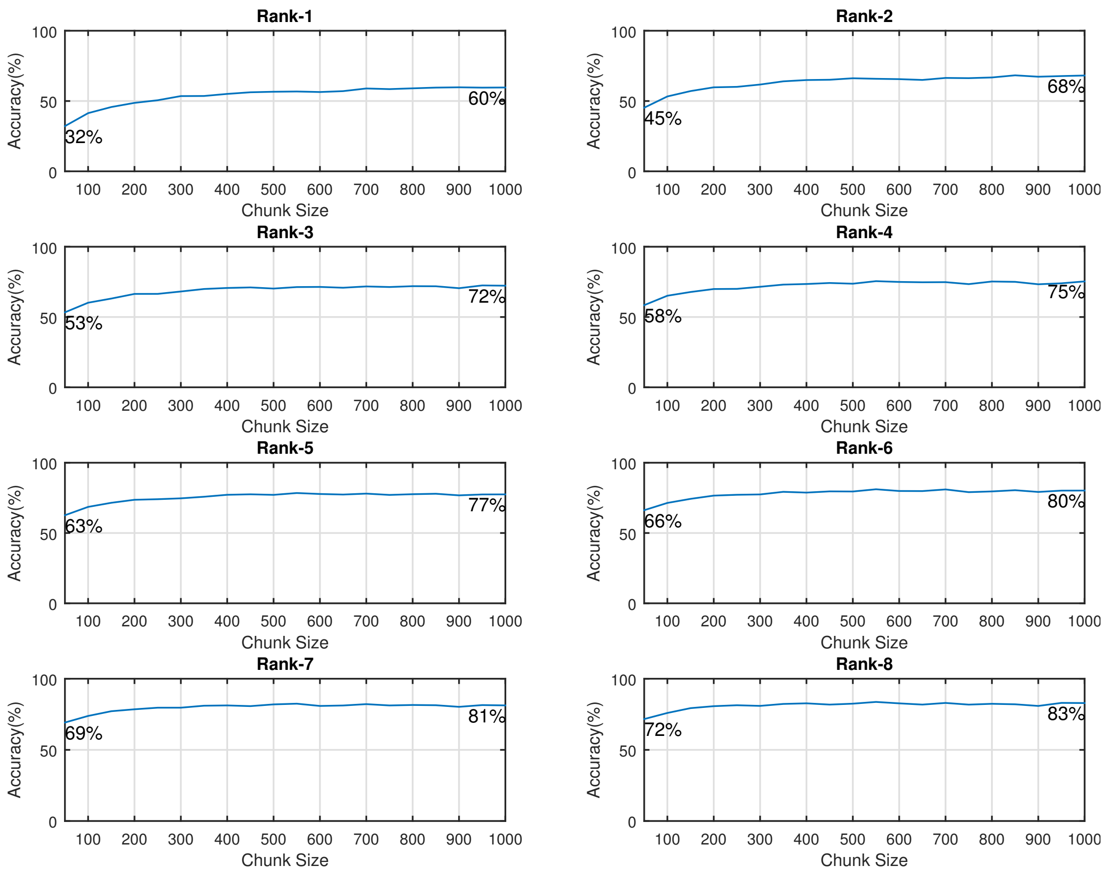


Figure 44: ACC(%) for **duration** vs chunk size for various rank, with **OM**, **variable** weights, and **MD**

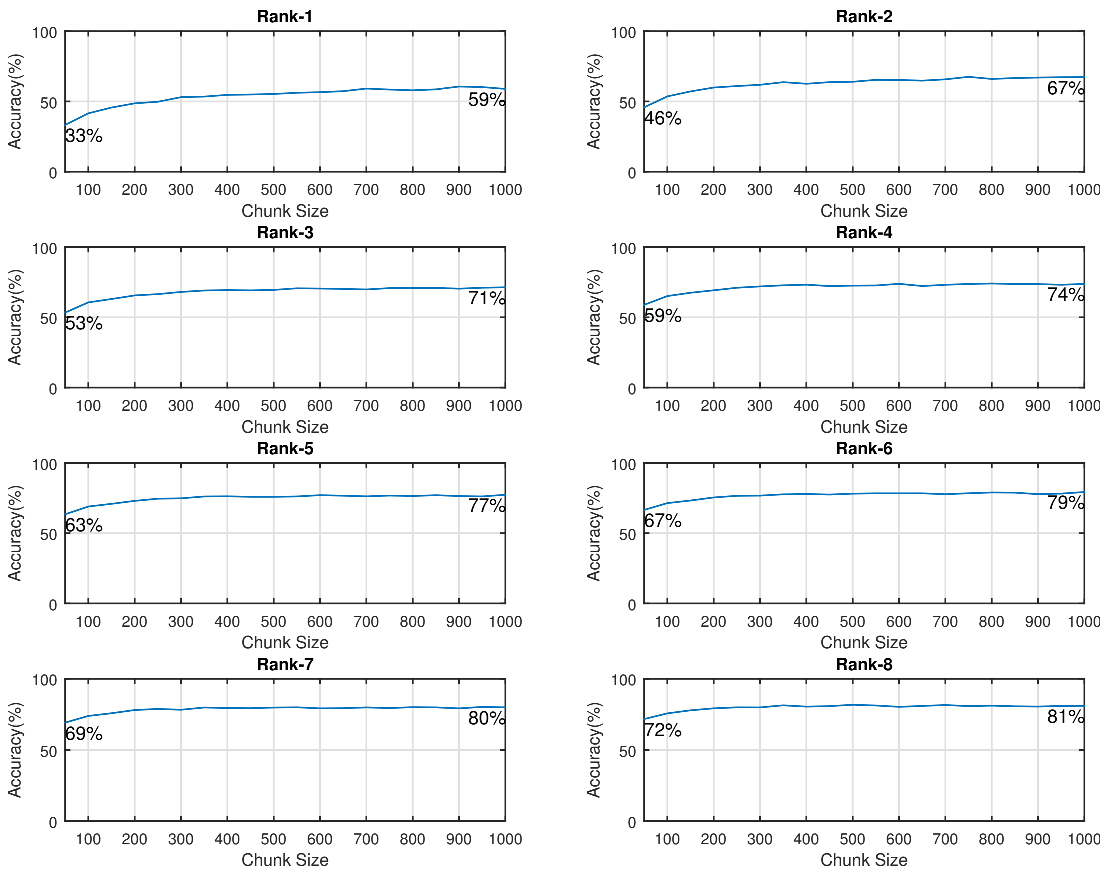


Figure 45: ACC(%) for **duration** vs chunk size for various rank, with **OM**, **variable** weights, and **ED**

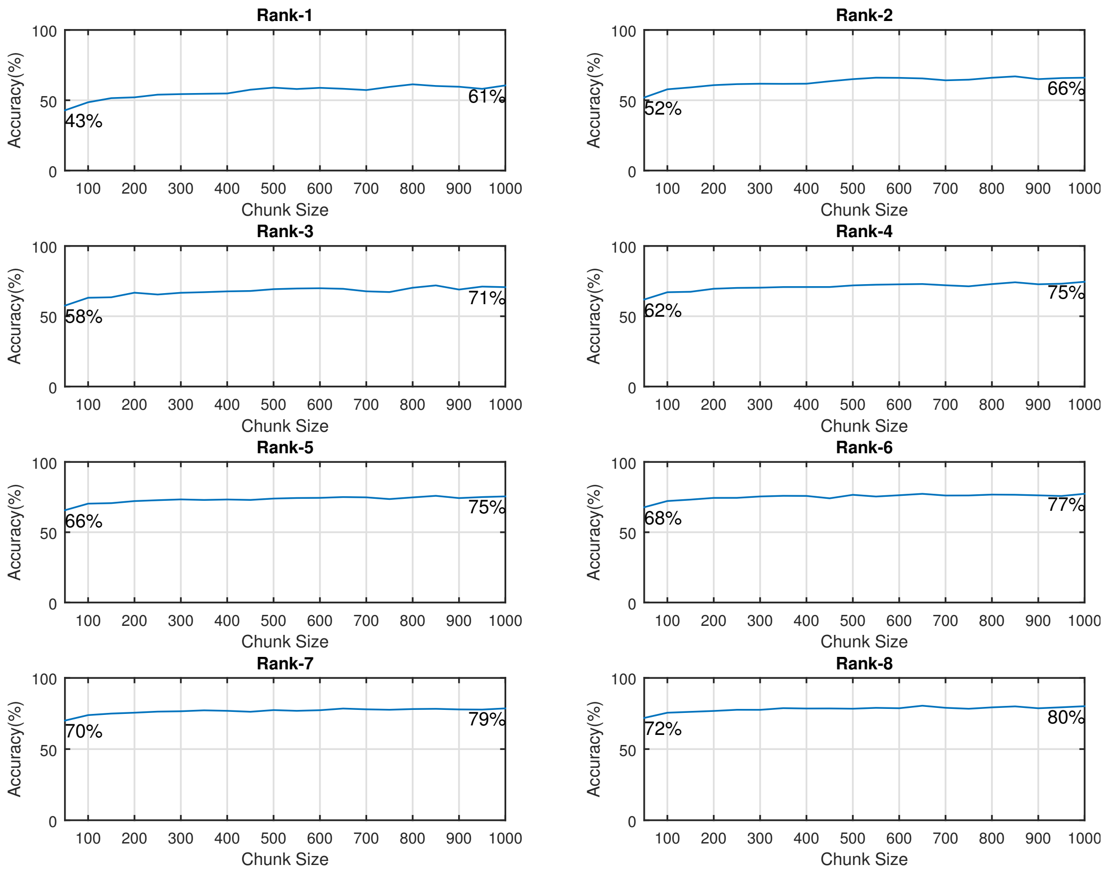


Figure 46: ACC(%) for **latency** vs chunk size for various rank, with **OM**, **variable** weights, and **MD**

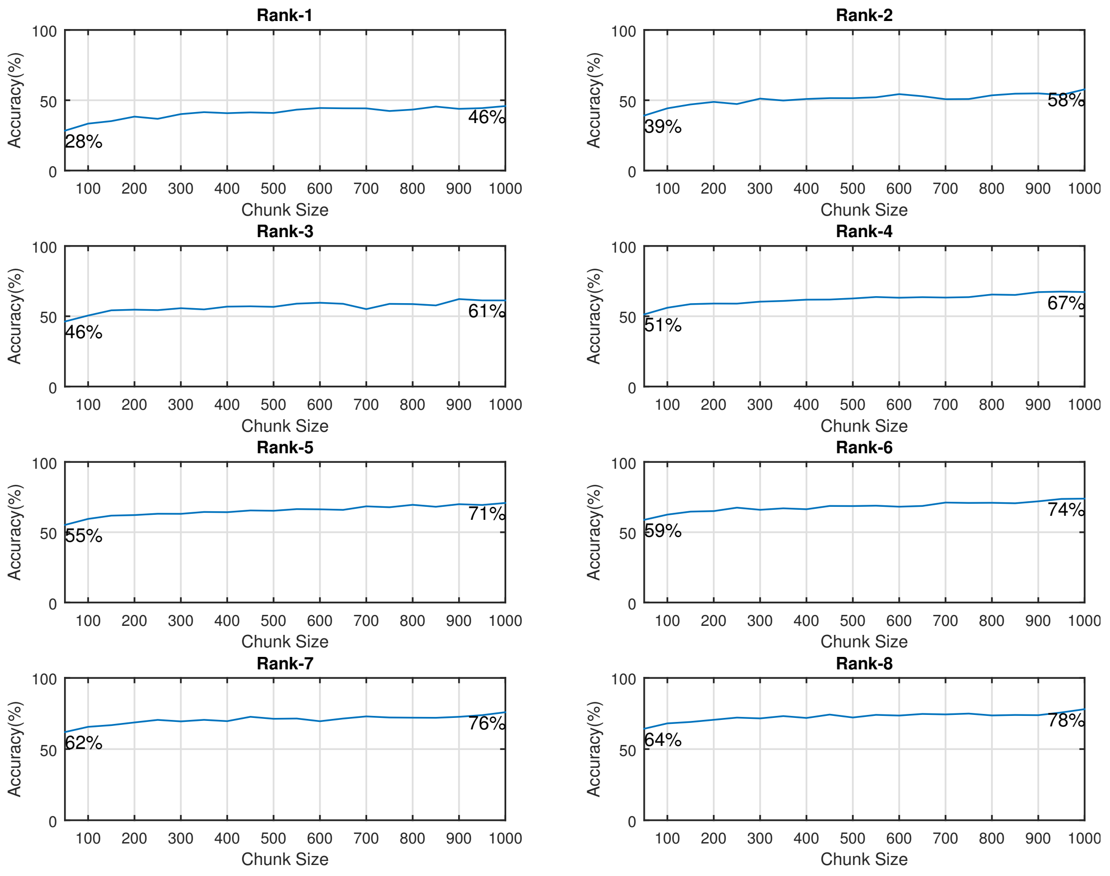


Figure 47: ACC(%) for **latency** vs chunk size for various rank, with **OM**, **variable** weights, and **ED**

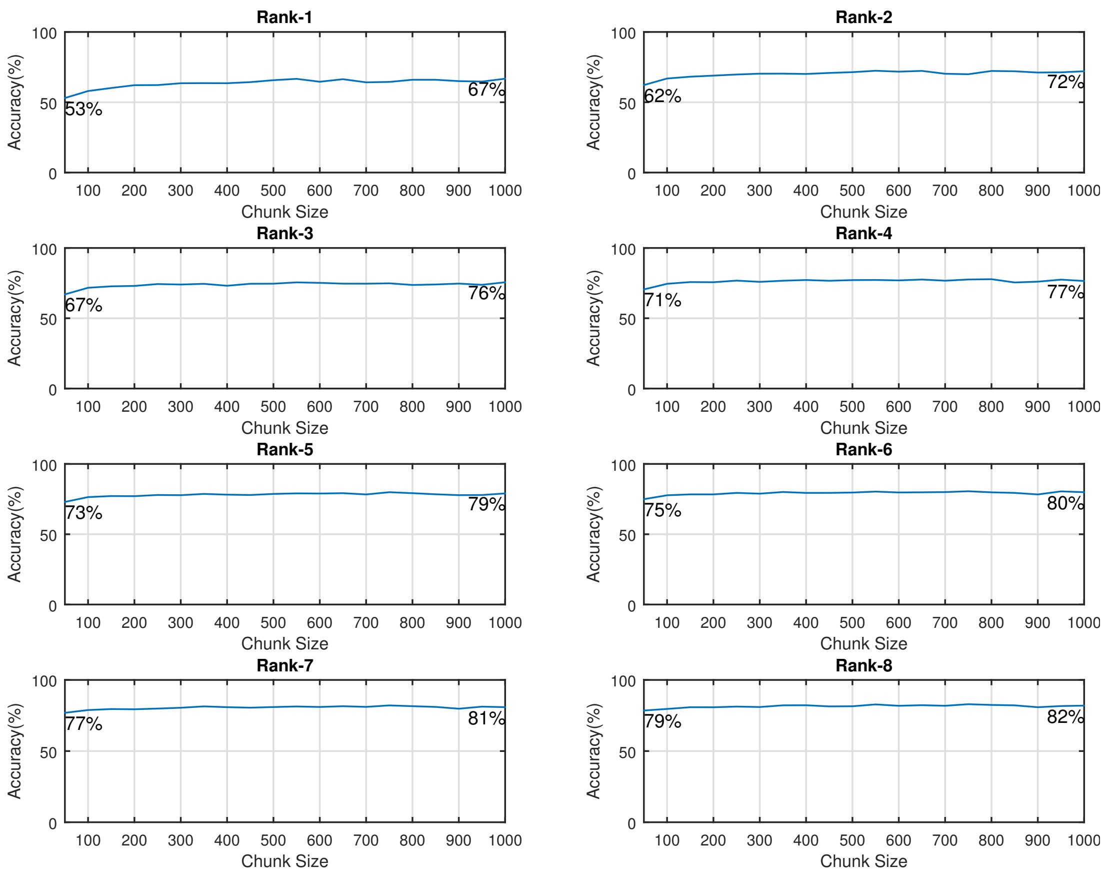


Figure 48: ACC(%) for **duration-latency** vs chunk size for various rank, with **OM**, **variable** weights, and **MD**

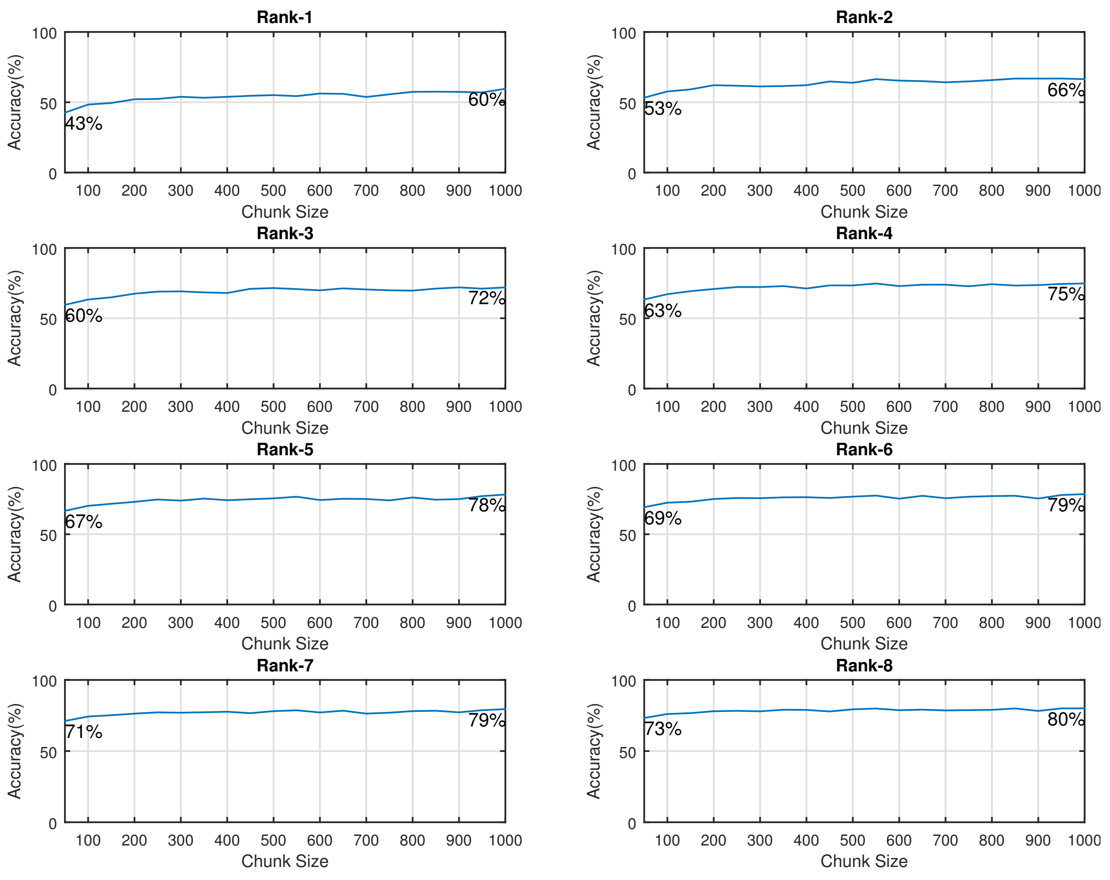


Figure 49: ACC(%) for **duration-latency** vs chunk size for various rank, with **OM**, **variable** weights, and **ED**

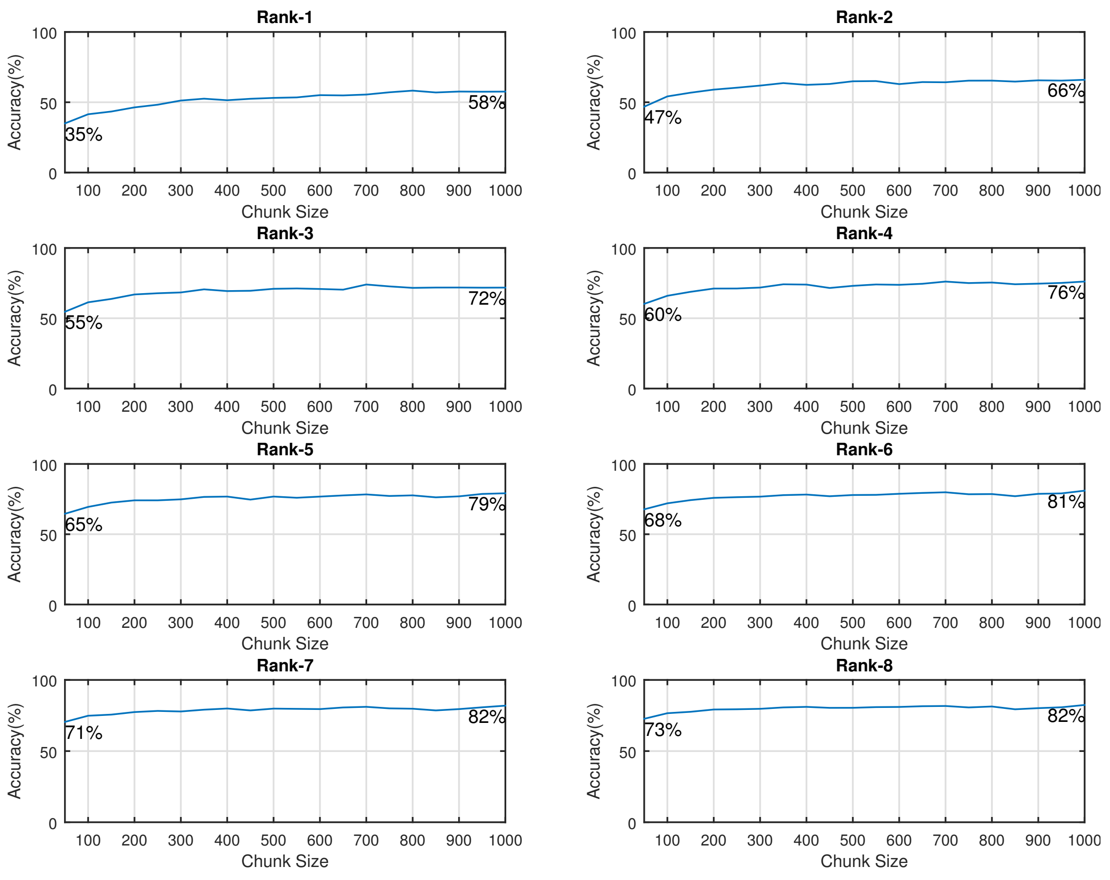


Figure 50: ACC(%) for **duration-frequency** vs chunk size for various rank, with **OM**, **variable** weights, and **MD**

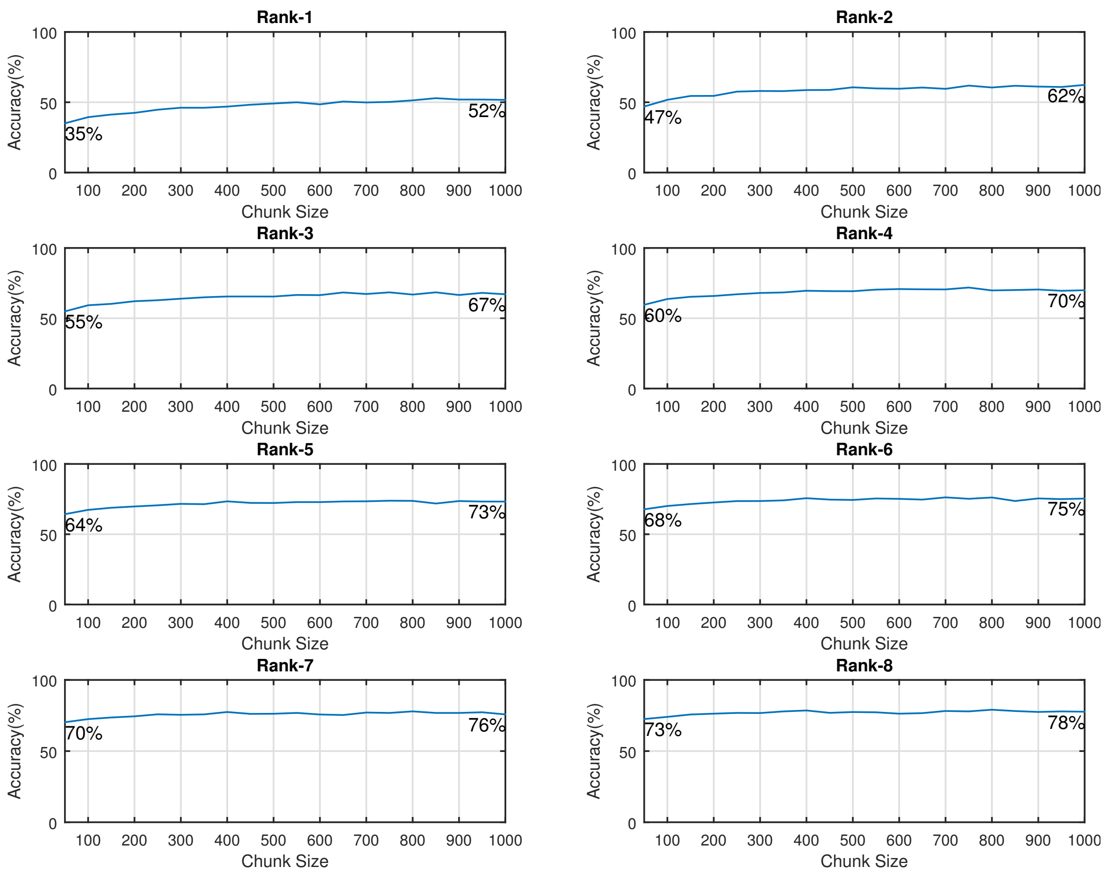


Figure 51: ACC(%) for **duration-frequency** vs chunk size for various rank, with **OM**, **variable** weights, and **ED**

A.3 Results without Software Categorization and MM

A.3.1 Based on Distance Metrics

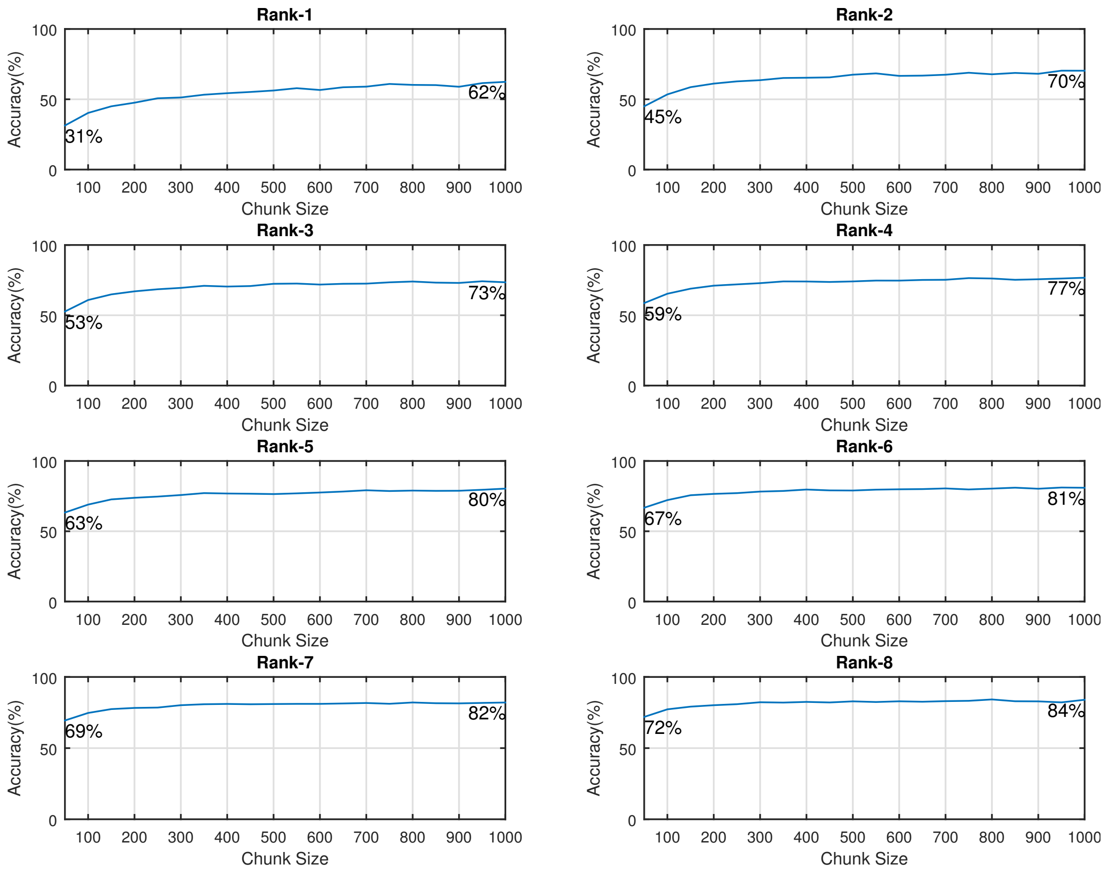


Figure 52: ACC(%) for **duration** vs chunk size for various rank, with **MM**, **uncategorized**, and **MD**

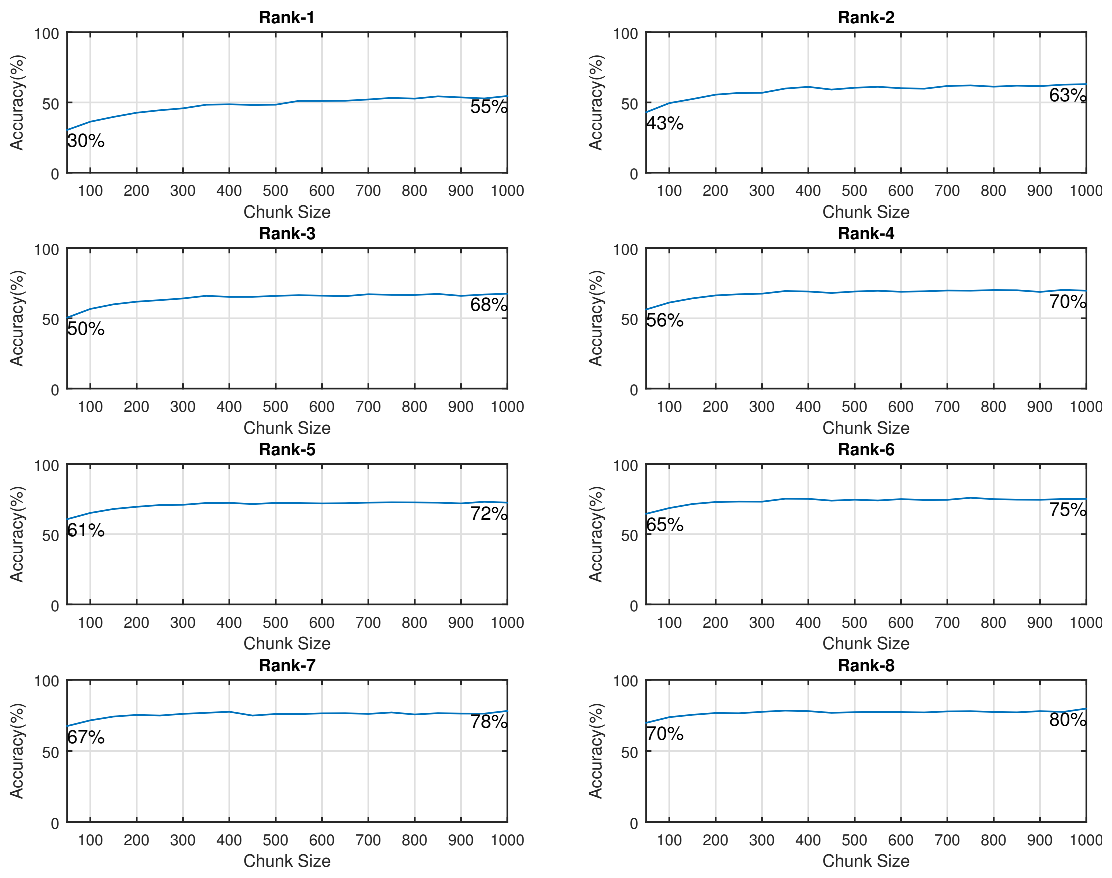


Figure 53: ACC(%) for **duration** vs chunk size for various rank, with **MM**, **uncategorized**, and **ED**

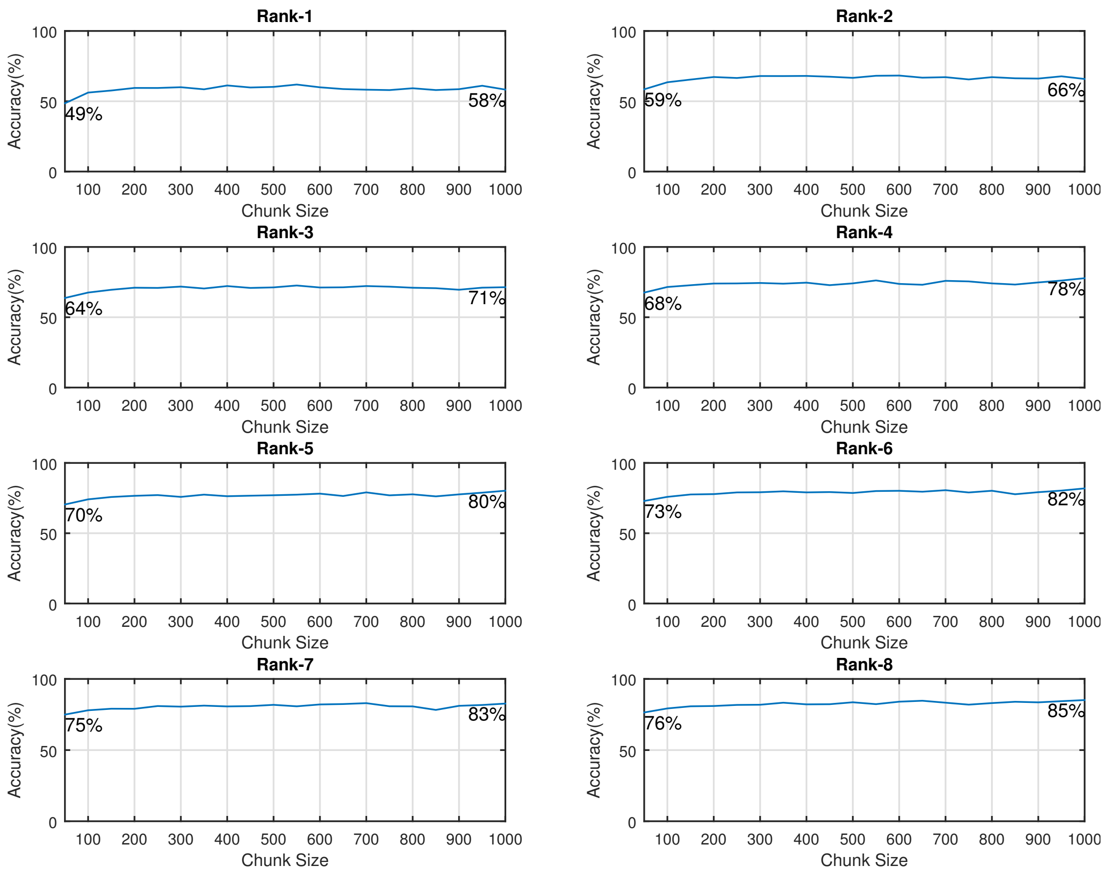


Figure 54: ACC(%) for **latency** vs chunk size for various rank, with **MM**, **uncategorized**, and **MD**

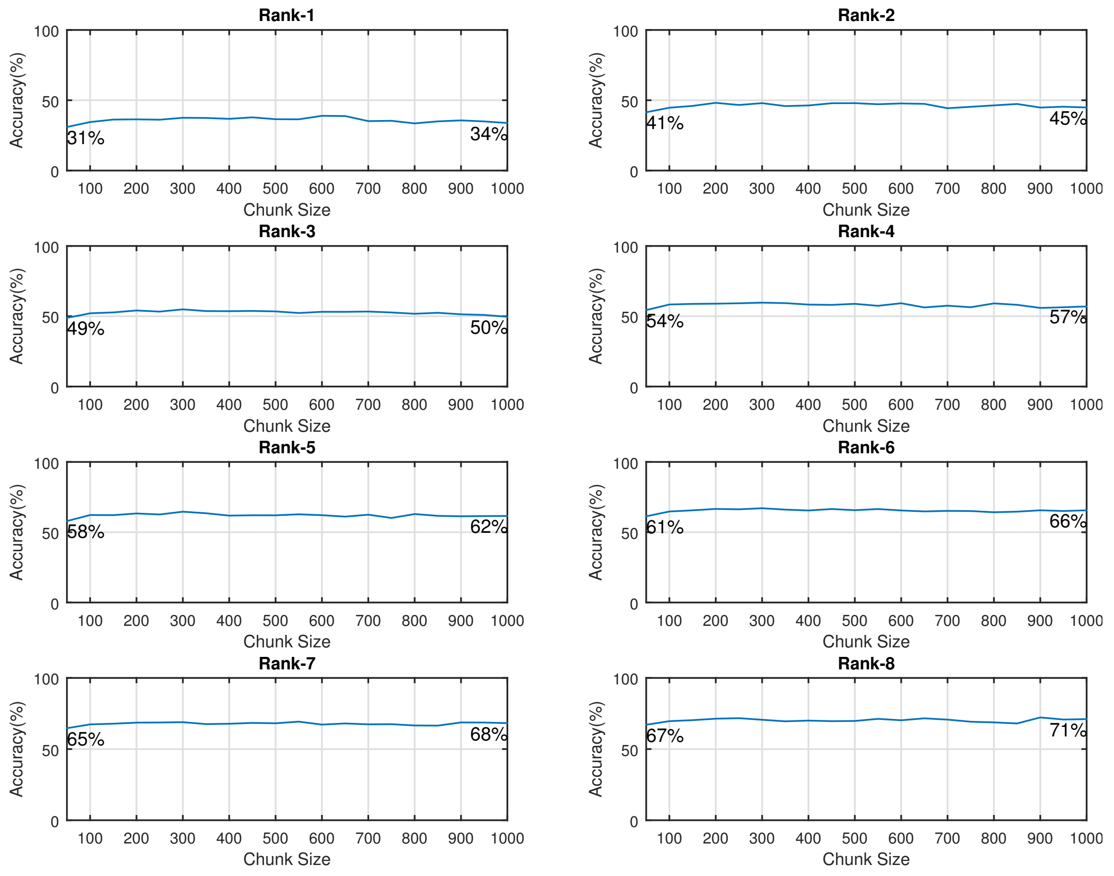


Figure 55: ACC(%) for **latency** vs chunk size for various rank, with **MM**, **uncategorized**, and **ED**

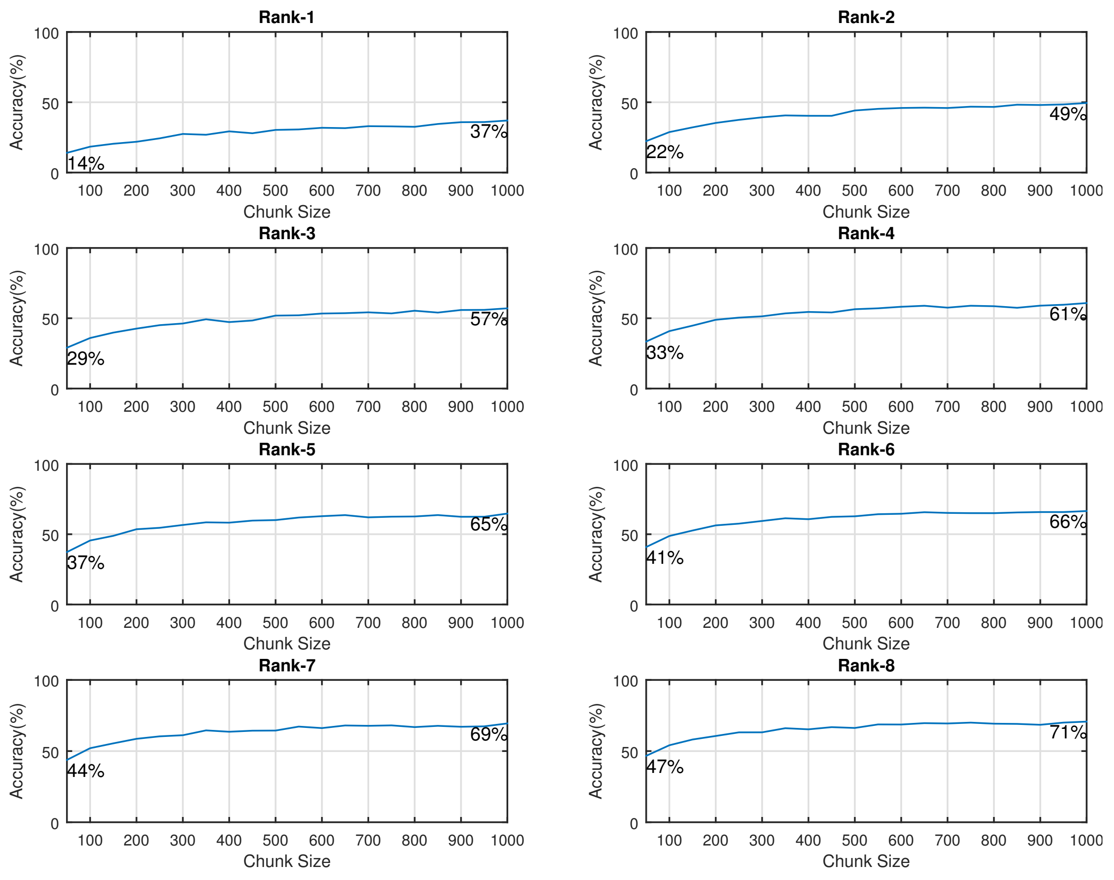


Figure 56: ACC(%) for **frequency** vs chunk size for various rank, with **MM**, **uncategorized**, and **MD**

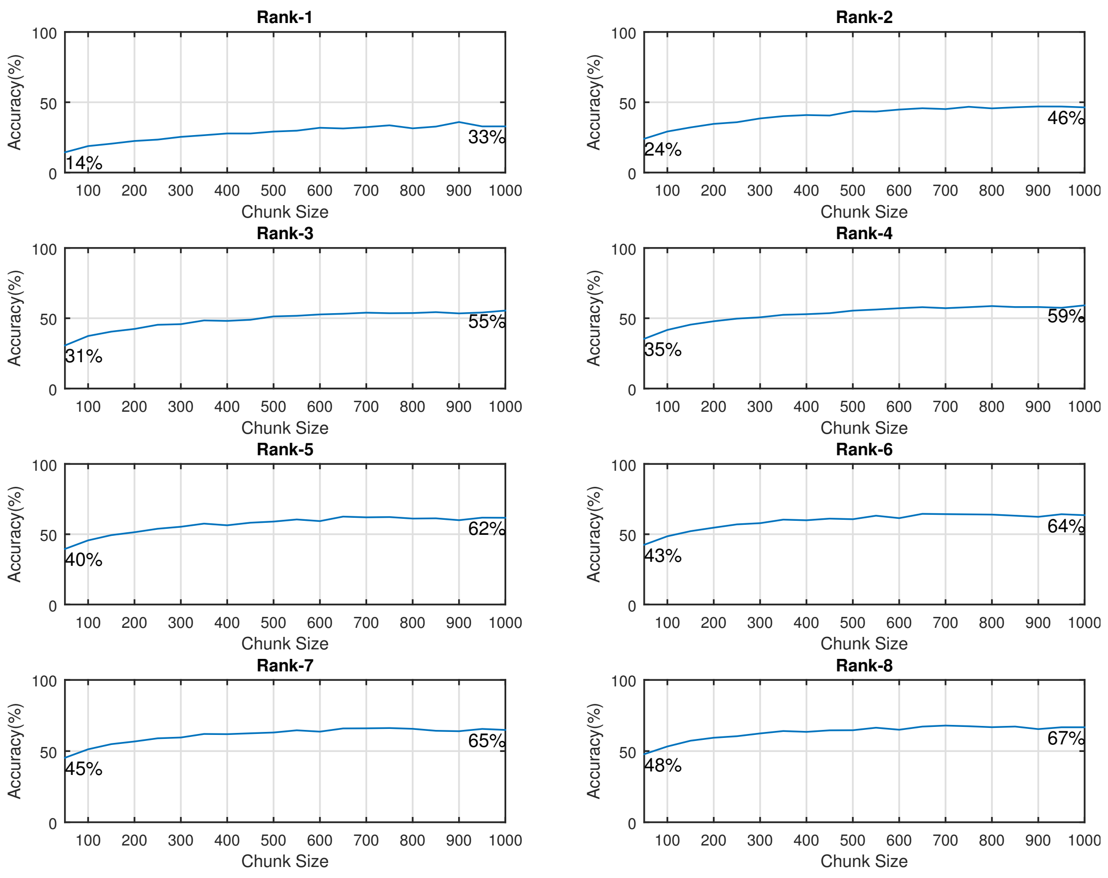


Figure 57: ACC(%) for **frequency** vs chunk size for various rank, with **MM**, **uncategorized**, and **ED**

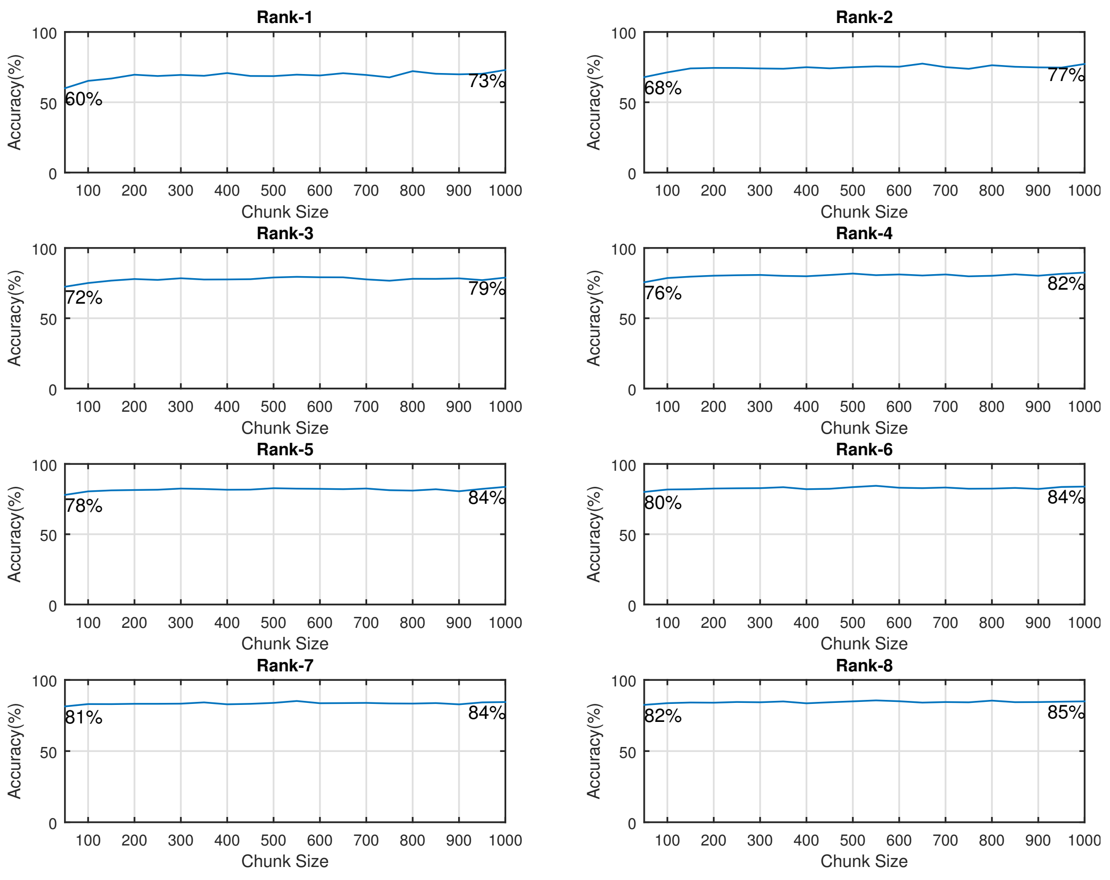


Figure 58: ACC(%) for **duration-latency** vs chunk size for various rank, with **MM**, **uncategorized**, and **MD**

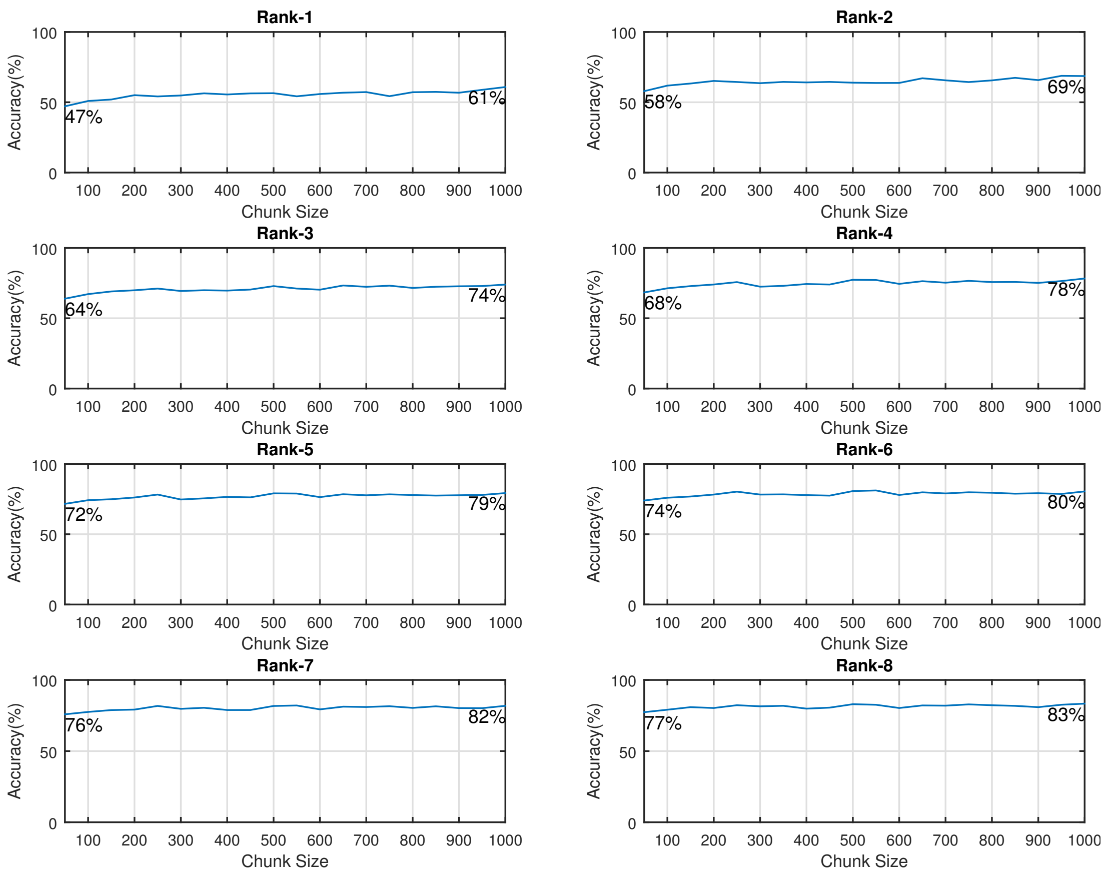


Figure 59: ACC(%) for **duration-latency** vs chunk size for various rank, with **MM**, **uncategorized**, and **ED**

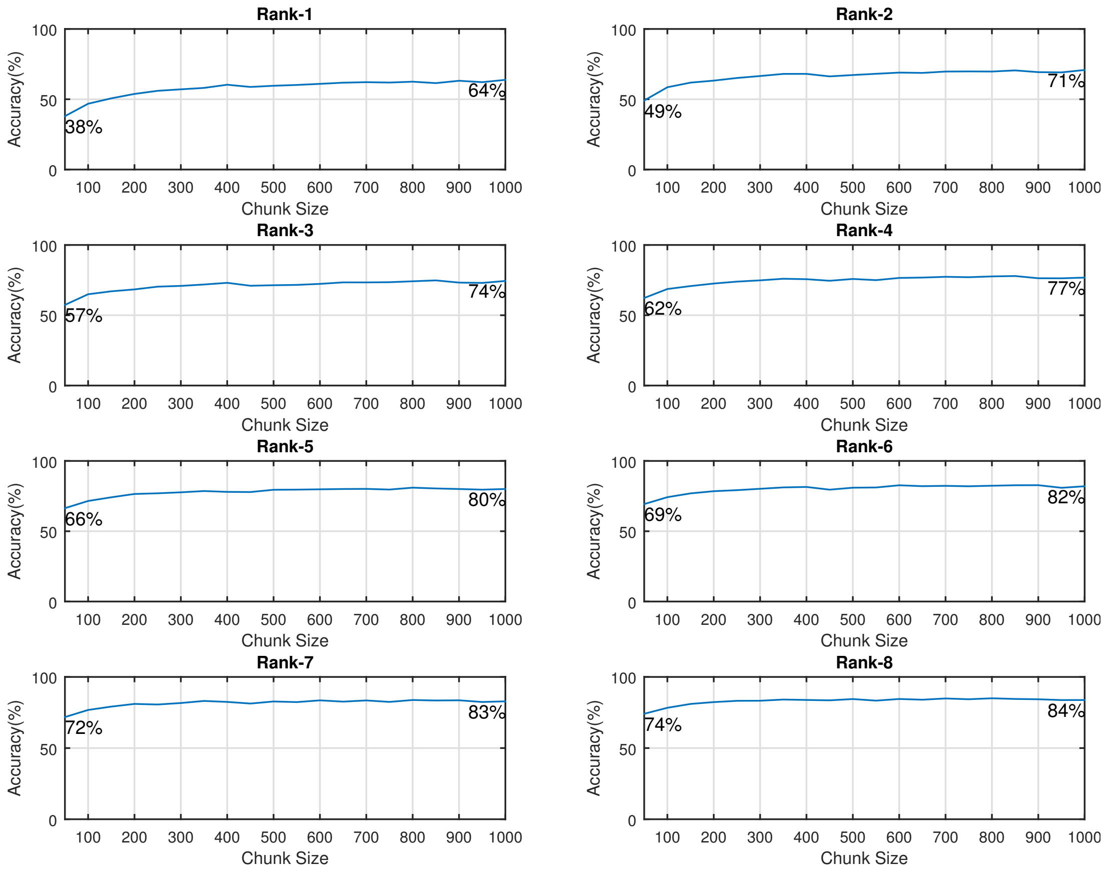


Figure 60: ACC(%) for **duration-frequency** vs chunk size for various rank, with **MM**, **uncategorized**, and **MD**

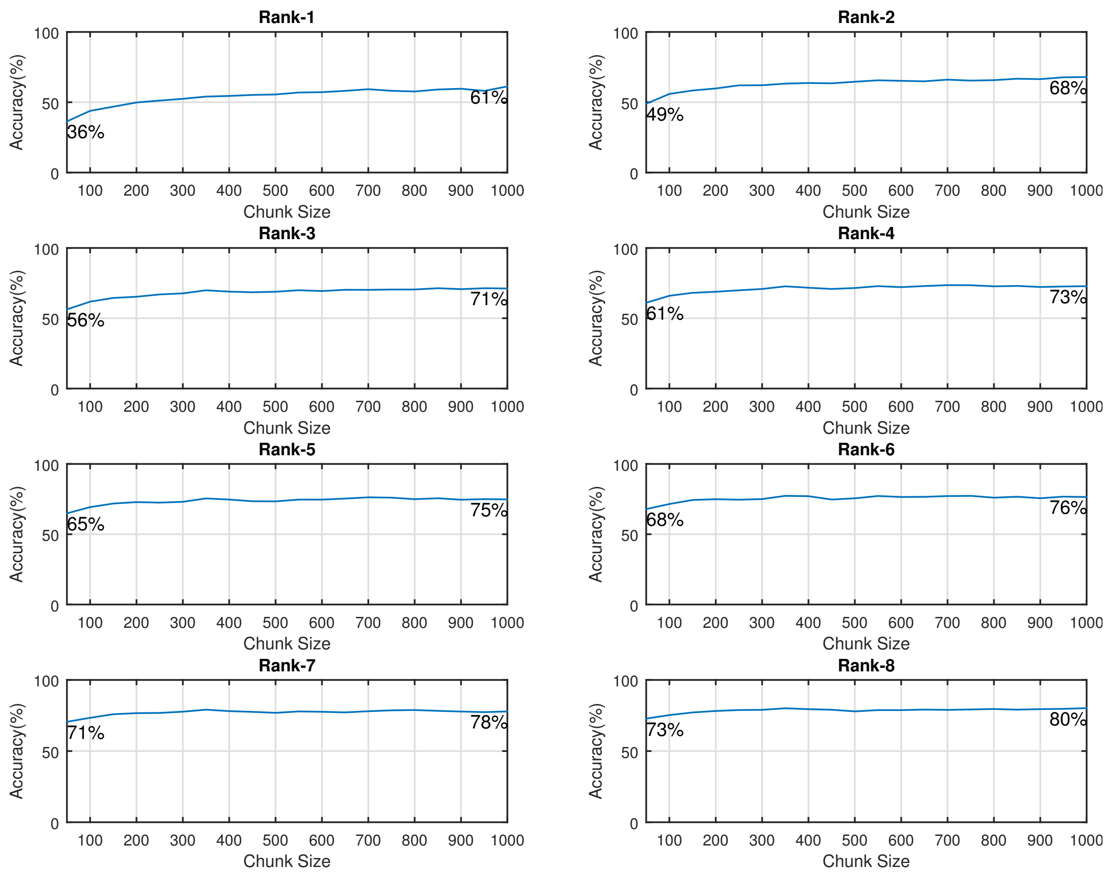


Figure 61: ACC(%) for **duration-frequency** vs chunk size for various rank, with **MM**, **uncategorized**, and **ED**

A.4 Results without Software Categorization and OM

A.4.1 Based on Distance Metrics

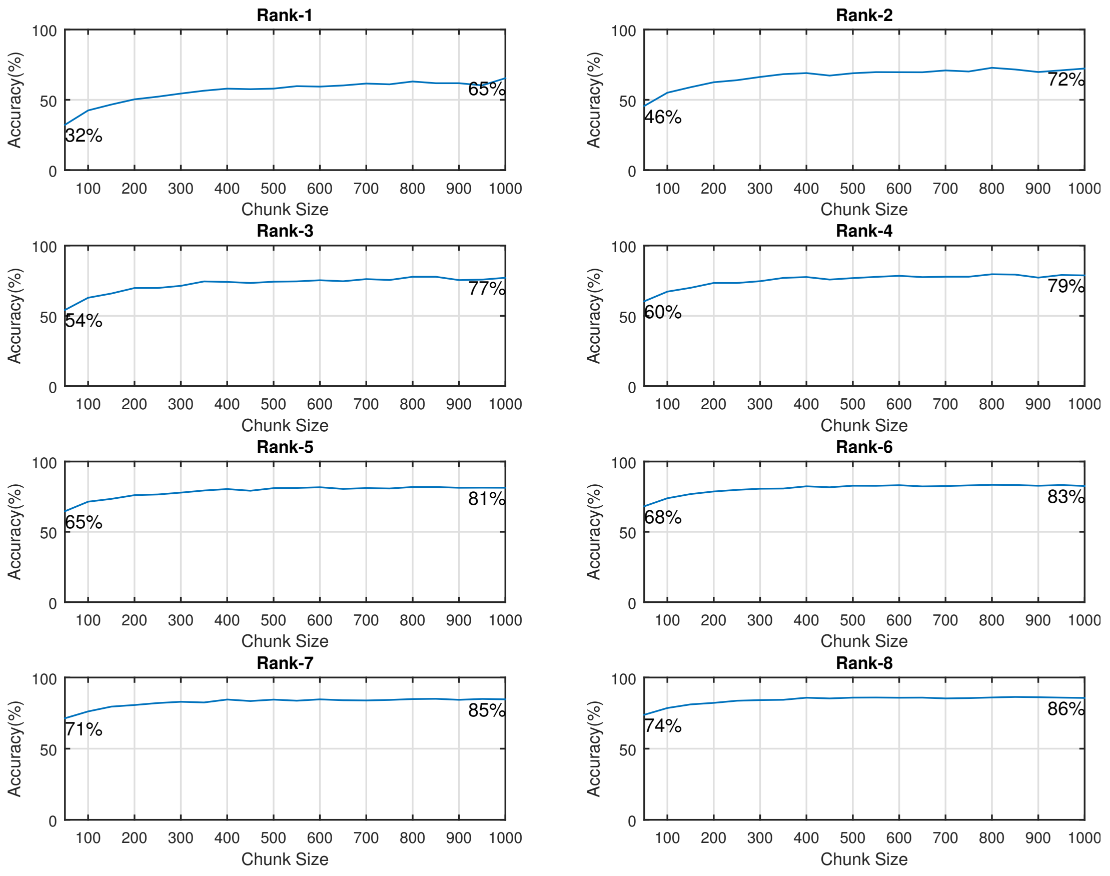


Figure 62: ACC(%) for **duration** vs chunk size for various rank, with **OM**, **uncategorized**, and **MD**

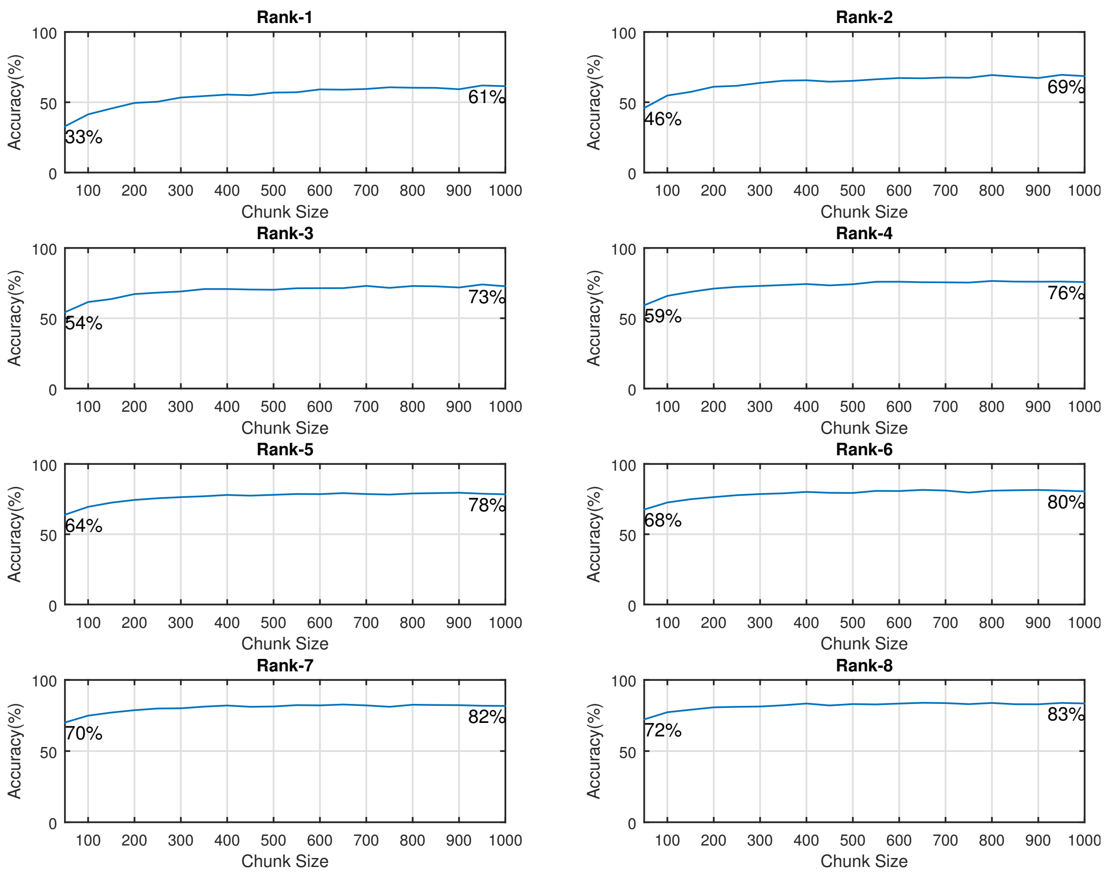


Figure 63: ACC(%) for **duration** vs chunk size for various rank, with **OM**, **uncategorized**, and **ED**

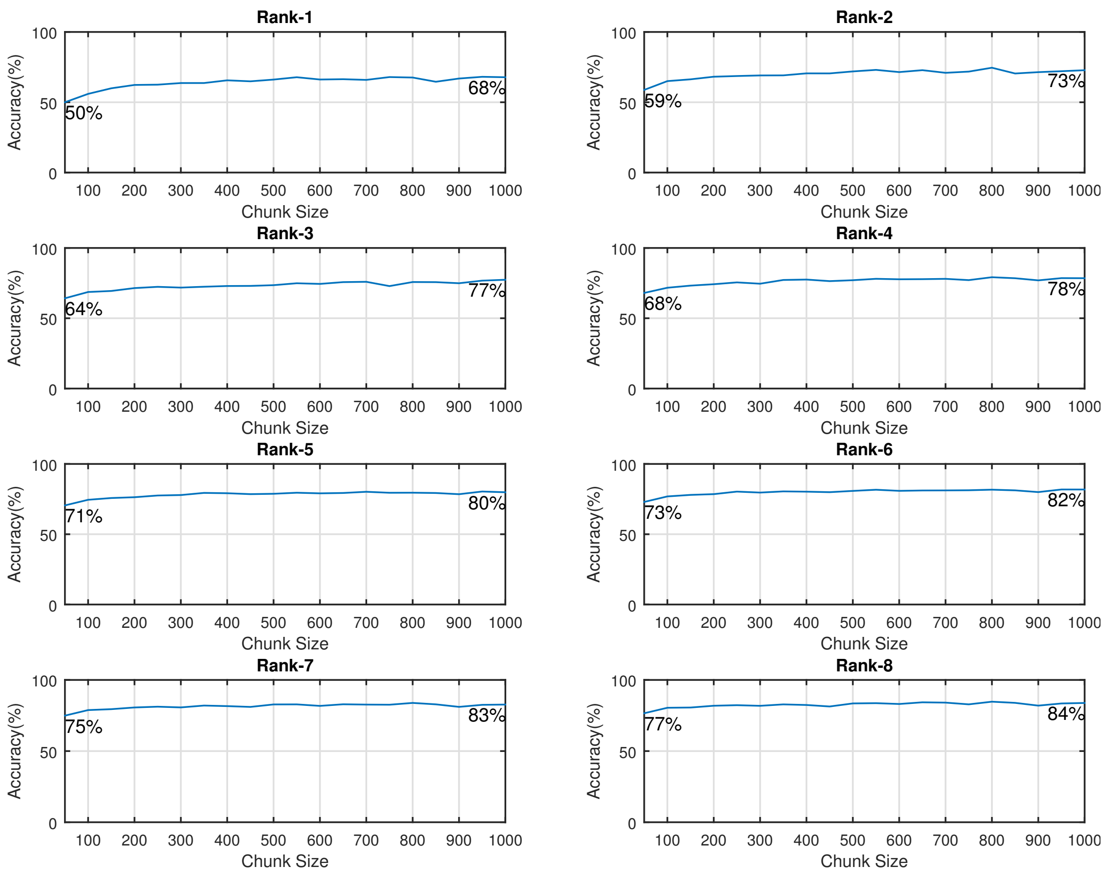


Figure 64: ACC(%) for **latency** vs chunk size for various rank, with **OM**, **uncategorized**, and **MD**

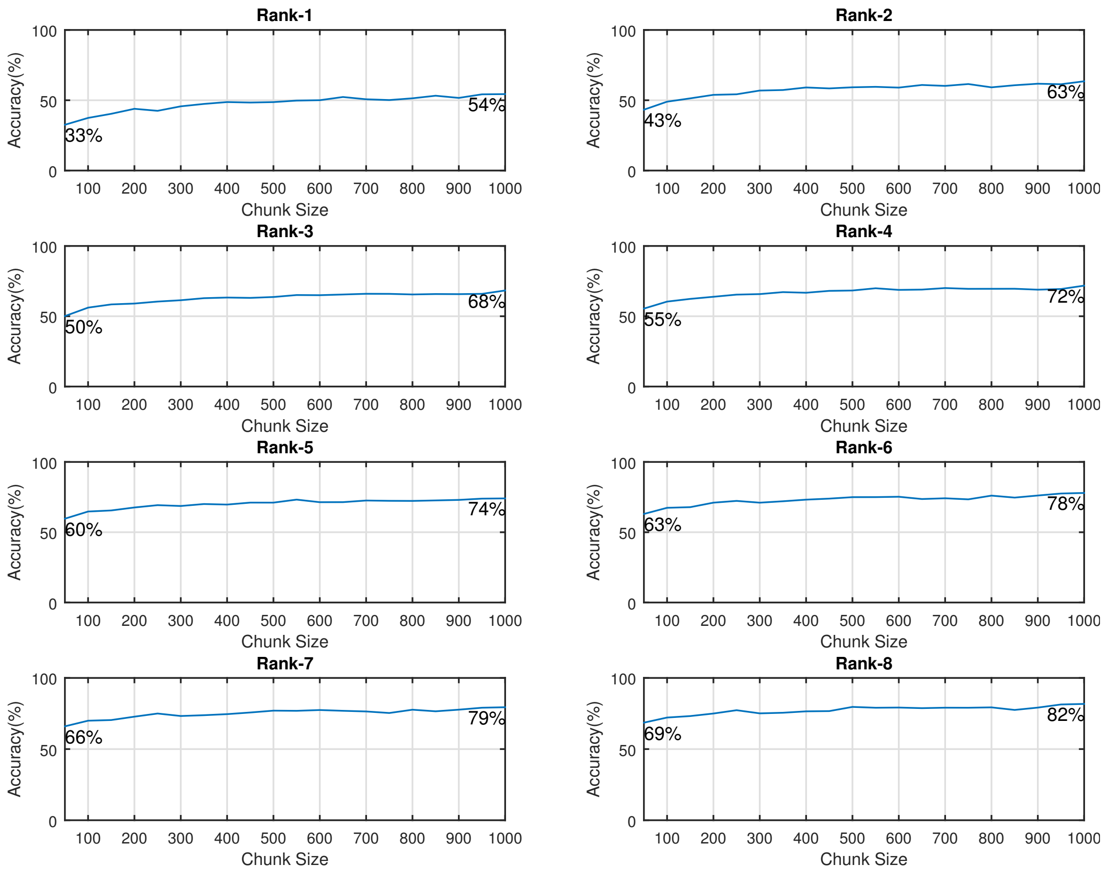


Figure 65: ACC(%) for **latency** vs chunk size for various rank, with **OM**, **uncategorized**, and **ED**

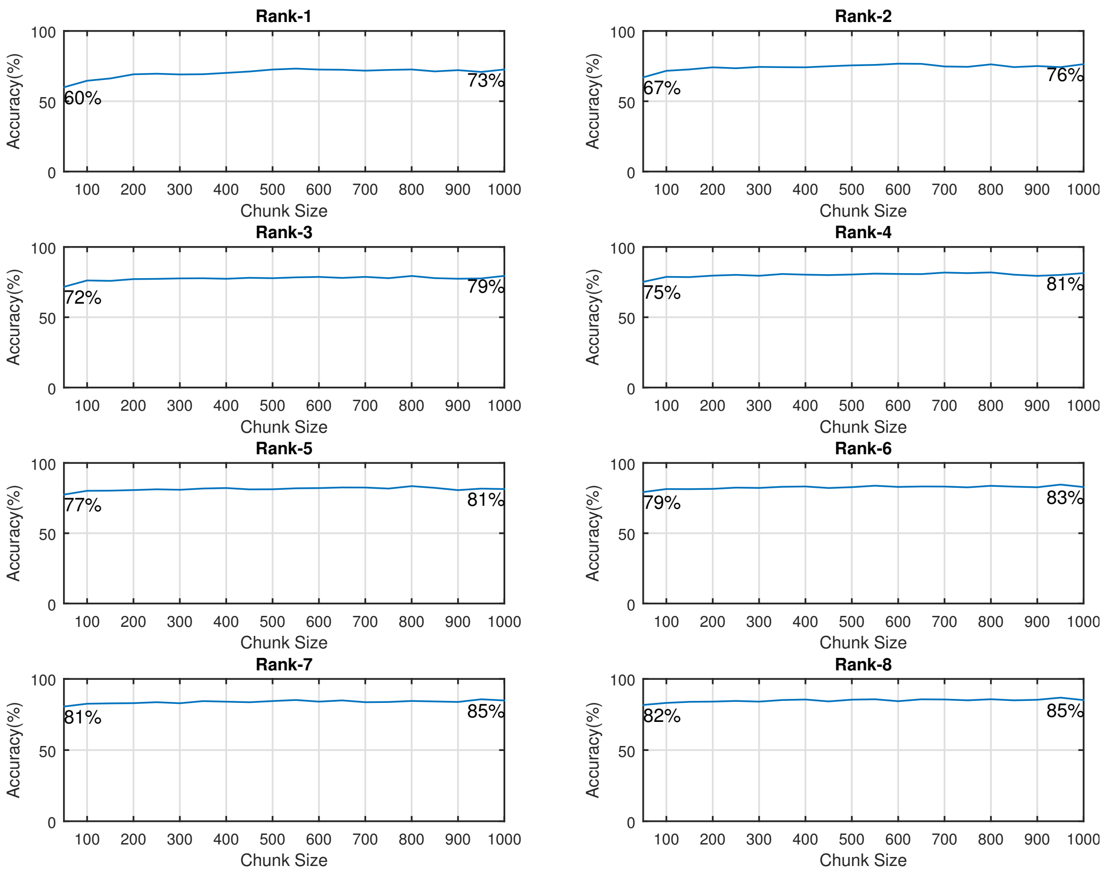


Figure 66: ACC(%) for **duration-latency** vs chunk size for various rank, with **OM**, **uncategorized**, and **MD**

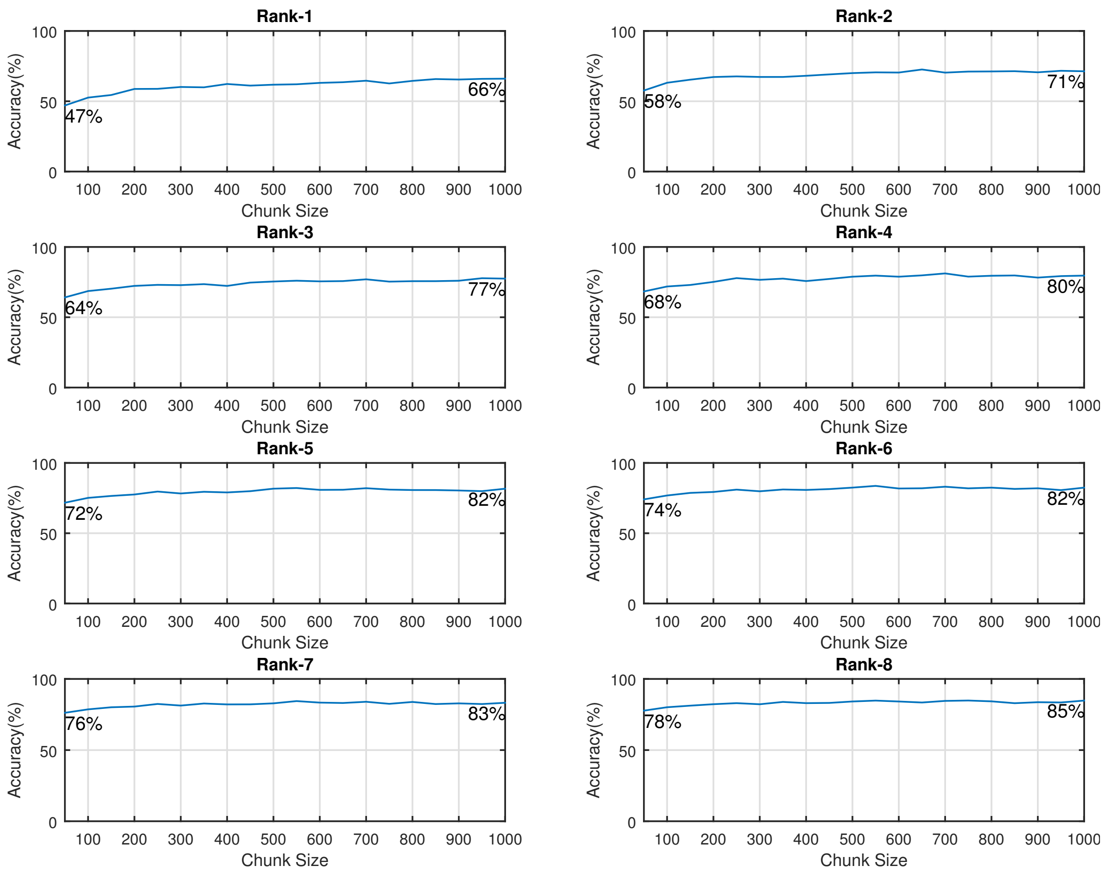


Figure 67: ACC(%) for **duration-latency** vs chunk size for various rank, with **OM**, **uncategorized**, and **ED**

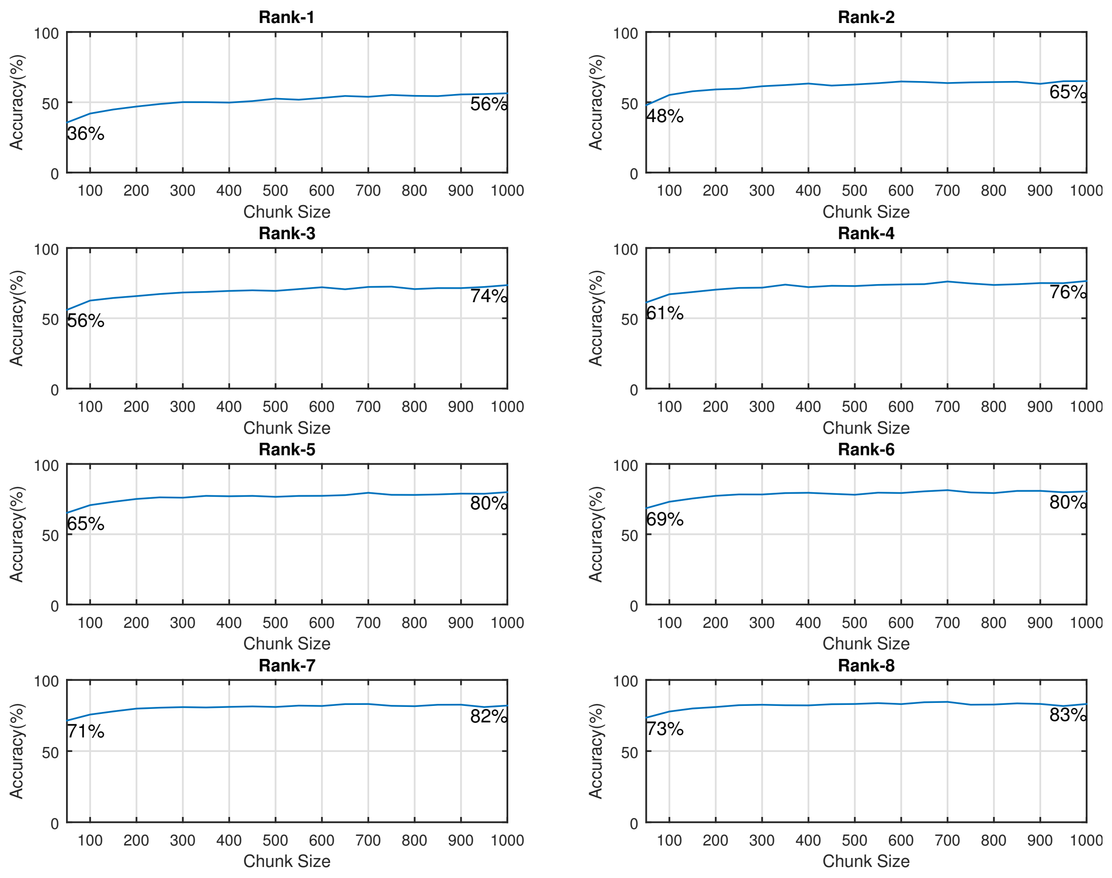


Figure 68: ACC(%) for **duration-frequency** vs chunk size for various rank, with **OM**, **uncategorized**, and **MD**

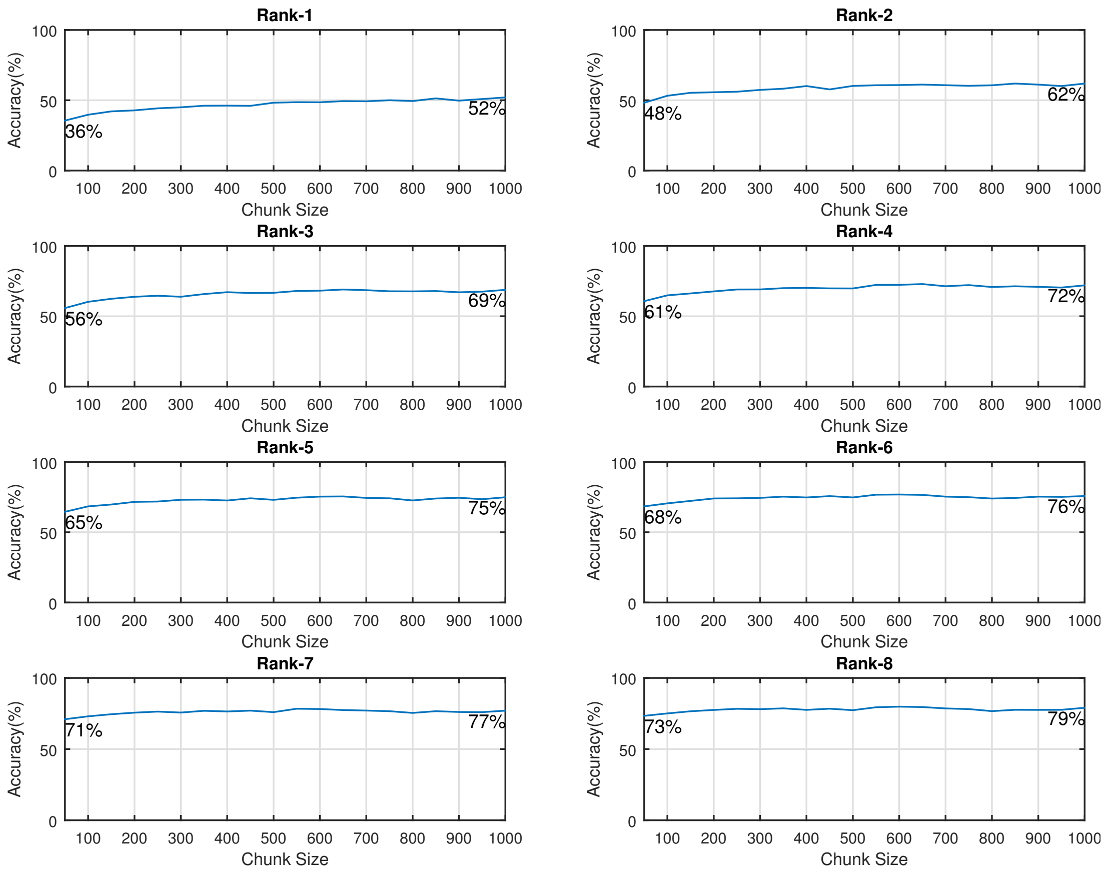


Figure 69: ACC(%) for **duration-frequency** vs chunk size for various rank, with **OM**, **uncategorized**, and **ED**

B Appendix B

B.1 Results Based on CA Data of Class(+/+)

No	R-1	R-2	R-3	R-4	R-5	R-6	R-7	R-8
1	61.1	68.0	73.1	76.4	78.9	80.4	81.5	82.4
2	69.1	75.4	78.7	80.4	81.6	82.3	82.9	84.3
3	63.4	71.4	75.3	78.2	80.5	81.7	82.8	83.5
4	65.0	71.9	75.9	78.5	80.4	81.4	82.4	83.5
5	64.2	71.3	74.9	78.5	80.5	81.9	83.0	84.0
6	62.6	69.5	74.2	77.4	79.5	81.0	82.5	83.3
7	59.9	67.8	72.1	75.4	77.5	79.5	81.0	81.9
8	61.0	68.5	73.2	76.5	78.8	80.4	81.5	82.8
9	70.4	76.0	79.3	81.8	83.2	84.0	85.2	85.5
10	59.0	67.6	72.5	75.5	78.1	80.1	81.4	82.6
11	69.2	74.1	77.7	79.9	81.6	83.1	83.6	84.1
12	65.0	71.4	76.0	78.6	80.6	81.9	82.6	83.8
13	70.8	74.0	77.9	81.3	82.0	83.2	83.4	83.9
14	66.9	73.5	76.7	79.7	81.2	82.0	82.6	83.6
15	53.4	63.6	69.2	72.7	75.3	77.5	79.1	80.4
16	67.5	73.7	77.4	79.5	80.6	81.4	82.1	83.4
17	64.9	72.0	76.3	79.5	81.2	83.1	84.3	85.2
18	62.7	69.6	74.4	77.6	79.4	81.2	82.1	83.1
19	60.6	68.3	73.0	76.4	78.5	80.3	81.6	82.7
20	59.9	67.8	72.3	75.5	77.9	80.0	81.3	82.4
21	65.5	72.2	76.0	78.5	80.3	81.7	82.9	84.1
22	57.6	65.7	71.0	74.7	77.2	78.8	80.6	81.5
23	66.1	73.0	76.8	78.8	80.4	81.7	82.5	83.3
24	64.8	71.5	75.3	78.0	79.5	81.6	82.8	83.8
25	64.4	71.2	75.6	78.2	80.1	81.3	82.4	83.4
26	60.5	68.4	72.8	76.0	78.0	79.8	81.4	82.3
27	65.3	72.9	76.6	78.8	80.6	81.9	82.6	83.4
28	67.7	74.0	77.7	80.2	81.7	83.0	83.9	84.9
29	48.6	59.3	65.5	69.5	72.5	75.0	76.9	78.5
30	46.3	57.8	63.8	68.0	71.4	74.1	76.0	77.6
31	59.3	67.3	71.9	75.1	77.6	79.7	81.0	82.1
32	53.3	63.4	69.1	72.6	75.3	77.5	79.1	80.4
33	67.0	73.1	77.2	79.4	81.0	81.9	82.9	83.5
34	58.1	65.9	71.3	74.9	77.4	79.0	80.7	81.7
35	51.4	62.0	67.4	71.4	74.3	76.5	78.1	79.6
36	66.6	74.2	78.5	80.8	82.6	84.1	84.8	85.5
37	59.3	67.9	72.2	75.7	78.2	80.0	81.3	82.2
38	48.2	59.0	65.1	68.8	71.8	74.2	76.2	77.5
39	63.6	70.6	75.2	79.1	81.0	82.6	83.3	84.0
40	67.8	73.3	77.0	80.2	81.3	82.3	83.1	83.5
41	64.1	71.5	75.5	78.5	80.4	81.6	83.1	83.9
42	64.0	71.2	75.2	78.4	80.2	81.7	82.9	83.7
43	53.0	63.0	68.8	72.3	75.1	77.2	78.8	80.2

Table 13: ACC(%) Based on CA Data Class(+/+)

B.2 Results Based on CA Data of Class(+/−)

No	R-1	R-2	R-3	R-4	R-5	R-6	R-7	R-8
1	71.7	76.3	77.8	79.8	81.1	83.0	83.5	85.6
2	66.3	73.0	76.4	78.7	80.4	81.7	82.5	83.3
3	69.5	77.2	80.0	81.5	82.4	83.9	84.8	85.6

Table 14: ACC(%) Based on CA Data Class(+/−)

B.3 Results Based on CA Data of Class(−/+)

No	R-1	R-2	R-3	R-4	R-5	R-6	R-7	R-8
1	68.8	74.9	78.8	80.5	81.7	82.9	83.5	83.8
2	69.2	74.8	78.5	80.2	82.5	83.4	84.5	85.2
3	67.3	74.3	76.9	79.5	81.6	82.3	83.4	84.1
4	68.7	74.7	78.4	80.5	81.9	82.7	83.3	83.8
5	68.0	73.3	77.9	80.1	81.4	82.2	83.2	83.7

Table 15: ACC(%) Based on CA Data Class(+/−)