Classifying Arabic Fonts Based on Design Characteristics: PANOSE-A

Jehan Janbi

A Thesis
In
the Department
of
Computer Science & Software Engineering

Presented in Partial Fulfillment of the Requirements
For the Degree of
Doctor of Philosophy (Computer Science) at
Concordia University
Montreal, Quebec, Canada

July 2016

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Jehan Janbi

Entitled: Classifying Arabic Fonts Based on Design Characteristics: PANOSE-A

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Rabin Raut	Chair
Dr. Farida Cheriet	External Examiner
Professor Nathalie Dumont	External to Program
Dr. Peter Grogono	Examiner
Dr. Tonis Kasvand	Examiner
Dr. Ching Y. Suen	Thesis Supervisor

Approved by

Chair of Department or Graduate Program Director

30/08/2016

Dean of Faculty

Abstract

In desktop publishing, fonts are essential components in each document design. With the development of font design software and tools, there are thousands of digital fonts. Increasing the number of available fonts makes selecting an appropriate font, which best serves the objective of a design, not an intuitive issue. Designers can search for a font like any other file types by using general information such as name and file format. But for document design purposes, the design features or visual characteristics of fonts are more meaningful for designers than font file information. Therefore, representing fonts' design features by searchable and comparable data would facilitate searching and selecting a desirable font. One solution is to represent a font's design features by a code composed of several digits. This solution has been implemented as a computerized system called PANOSE-1 for Latin script fonts. PANOSE-1 is a system for classifying and matching typefaces based on design features. It is composed of 10 digits, where each digit represents a specific design feature. It is used within several font management tools as an option for ordering and searching fonts based on their design features. It is also used in font replacement processes when an application or an operating system detects a missing font in an immigrant document or website. Currently, PANOSE-1 is only defined for fonts that have Latin characters. Therefore there is a need for providing a model that describes and classifies fonts with Arabic characters.

In this research, a new model PANOSE-A is defined to extend PANOSE-1 coverage to support Arabic characters. The model defines eight digits in addition to the first digit of PANOSE-1which indicates the font script and family type. Each digit describes a visual or a design feature and takes value between 0-15. The meaning of 0 and 1 values is similar to what is defined for PANOSE-1. Each of the remaining values indicates a specific variation of its represented feature. Weight and contrast are two essential features in any font design. Two digits of the models describe the common variations of the weight and contrast features for text fonts. Another four digits describe the shape of some strokes that usually vary in their design between fonts. One digit describes the end shape of terminal strokes using three letters with different terminal strokes. Another digit describes the shape of the bowl stroke while the third digit describes the shape of curved stroke. The thresholds that used to define each shape class are taken from Naskh calligraphy guidelines. The fourth digit describes the shape of rounded strokes with enclosed counter. The shape classes of this digit are based on how the counter shape is similar to one of five geometric shapes. These basic geometric shapes are triangle, square, rectangular, oval and circle. The last two digits describe the characteristics of two important vertical references of the Arabic font design which are tooth and loop heights. The reliability of the model was evaluated by conducting two clustering processes on 30 fonts of Naskh style. The proposed PANOSE-A model was used to construct a similarity matrix for one the clustering processes while the other clustering process used a similarity matrix that was produced by using a font matching tool. The result clusters of the two clustering processes have been evaluated by silhouette coefficient. Silhouette is a method to measure data consistency validation within clusters. It indicates how objects are similar in their own cluster compared to other clusters. Clustering result based on the similarity matrix produced by font matching tool got 97.04 while using clustering result based on the similarity measured by PANOSE-A model got 98.24.The similarity between the results of the two clustering processes has been estimated, indicating that the model succeeded in classifying 85% of the fonts as a font matching tool.

Acknowledgment

I would like to thank my supervisor Dr. Ching Y. Suen for his patience, guidance and support throughout the years of this research. Thanks to Prof. Nathalie Dumont also, who has dedicated her time to provide valuable suggestions and guidance. Thanks to Dr. Marleah Blom and Mr. Nicola Nobile for all their cooperation and endless help to facilitate all what my study required from documents, data, tools and writing this thesis.

Thanks to the Qassim and Taif universities for giving me the opportunity to study in Canada. Thanks to Saudi cultural bureau and International student office for their support in any difficulties I faced during the study or the expatriation.

Special thanks for my kids, Abdulaziz and Jorey, for their understanding of my preoccupation and their cooperation to ease all critical times during the past years. Thanks to my mother for her caring, encouraging and support despite the distance and her needs for us to be around her. Thanks for Dr. Fared Felemban who supported me and put the first steps to take my study to a higher level.

Thanks also to Dr. Mamoun Sakkal from SAKKAL DESIGN and Mr. Hasan Abu Afesh from Hiba Studio, for donating part of the material used in this research. Also thanks to them with, the typographers and the graphic designers, Mr. Khalid M. Khalid and Mr. Mike Sabbagh, for their comments and suggestions that have greatly helped this research.

Table of Contents

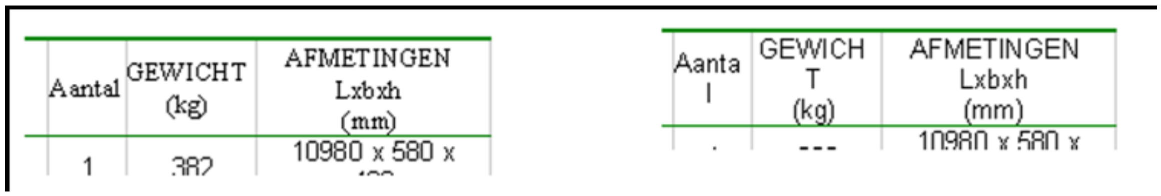
LIST OF FIGURES

LIST OF TABLES

Chapter 1 : Background

1.1. Introduction

There are thousands of digital fonts for different languages in the current digital world. Due to the huge number of available digital fonts, the need for controlling and organizing fonts has become an important issue. Most font management tools provide the capability to sort and to organize font files according to different kinds of font file information such as name, format, style, family name, weight, designer, manufacture, vendor and version. Within desktop publishing environments, designers mainly make font selection decisions based on fonts' appearance and how the selected fonts are visually harmonized with each other to achieve the intended design goal. Another issue is related to designing portable documents. In general, fonts used in a document are not attached to the document; the quality of the document may then change if the document is opened on other machines that do not have the same fonts. Operating systems and most software apply font substitution processes to replace missing fonts. Selecting the best substitution depends on finding the best match for the missing font. Substituting a missing font with an inappropriate one may completely change the appearance of the document, as in the example shown in Figure 1. Moreover, the substituted fonts may not have the same character sets as the missing fonts, causing information loss as in the example shown in Figure 2. Therefore, there is a need to find a method to deal with fonts according to their visual appearance and design characteristics.



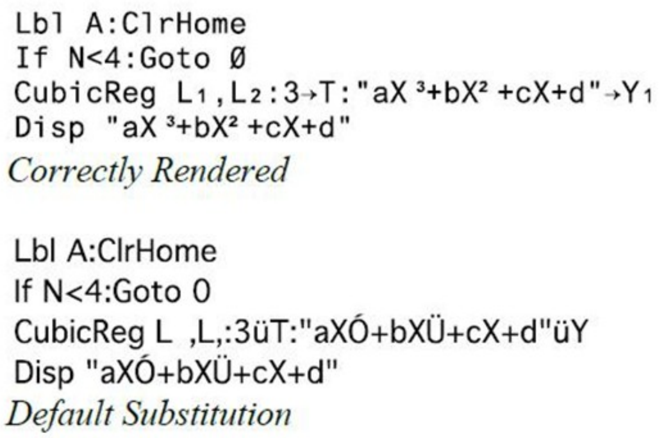
Aantal	GEWICHT (kg)	AFMETINGEN Lxbxh (mm)
1	382	10980 x 580 x

This is the text in Times New Roman, as designed

Aanta l	GEWICH T (kg)	AFMETINGEN Lxbxh (mm)
.	---	10980 x 580 x

When a font is missing and it is substituted with Arial, the words no longer fit into the column any more.

Figure 1: Example of document appearance changing with a substituted font [43]



```
Lbl A:ClrHome
If N<4:Goto Ø
CubicReg L₁,L₂:3→T:"aX³+bX²+cX+d"→Y₁
Disp "aX³+bX²+cX+d"
```

Correctly Rendered

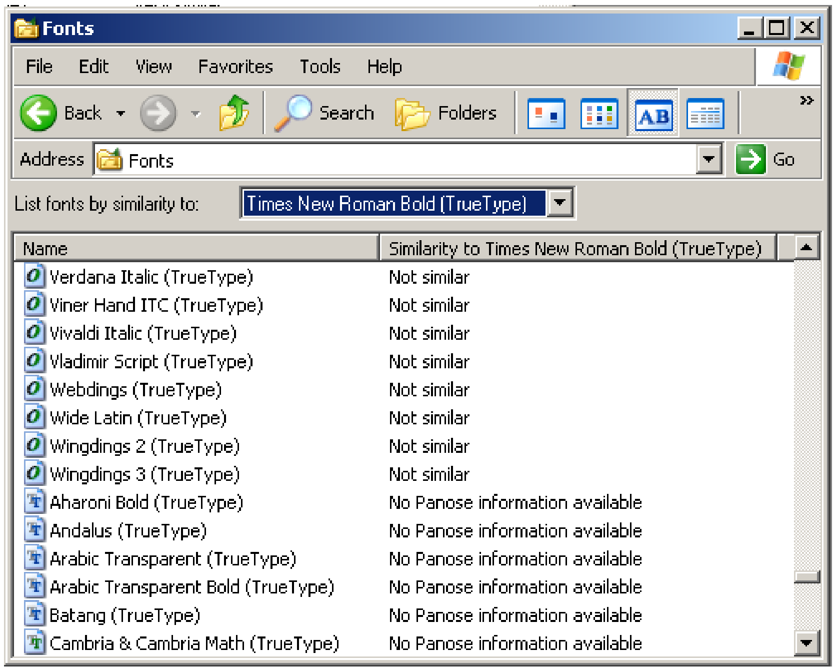
```
Lbl A:ClrHome
If N<4:Goto 0
CubicReg L ,L,:3üT:"aXÓ+bXÜ+cX+d"üY
Disp "aXÓ+bXÜ+cX+d"
```

Default Substitution

Figure 2: Example of losing information when Ti83Pluspc font is substituted [43]

One way that enables comparing, sorting or grouping fonts based on their visual characteristics is to represent these characteristics by comparable data that can be attached to font files. This way is based on encoding the visual characteristics into a number composed of several digits. Each digit represents a specific visual characteristic, which takes one or more values indicating its specific variation. IBM and PANOSE-1 are computerized systems that have implemented this idea for Latin script. The exact definition of IBM and PANOSE-1 numbers and the meaning of their digits are reviewed and explained in detail within chapter 3. Usually, these numbers are determined by font designers or vendors and are stored in font files. Several font management tools, such as MainType and Typograf, are able to display this information and use it to sort and compare fonts. These numbers are also used by operating systems and software to find the best substitution for missing fonts.

Although we can find fonts that contain character sets for Latin and Non-Latin scripts, the basic design characteristics defined in IBM and PANOSE-1 are mainly for Latin characters. For Arabic fonts no work has been done to encode their visual characteristics into a number or any comparable data format. This increases the effects of the missing font problem for documents containing Arabic text. For example, in Microsoft software, usually there is no suitable substitution for the missing Arabic font because most current fonts do not have any PANOSE-1 information, as in the example shown in Figure 3. When a font is missing, the Microsoft office substitutes it with Arial, which is the default option when no match can be found as the example in Figure 4.

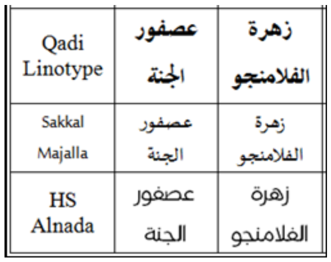


Fonts

List fonts by similarity to: Times New Roman Bold (TrueType)

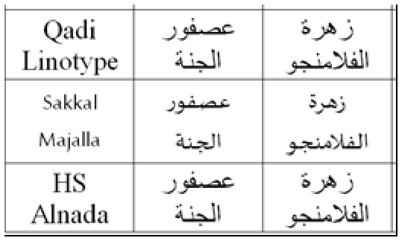
Name	Similarity to Times New Roman Bold (TrueType)
Verdana Italic (TrueType)	Not similar
Viner Hand ITC (TrueType)	Not similar
Vivaldi Italic (TrueType)	Not similar
Vladimir Script (TrueType)	Not similar
Webdings (TrueType)	Not similar
Wide Latin (TrueType)	Not similar
Wingdings 2 (TrueType)	Not similar
Wingdings 3 (TrueType)	Not similar
Aharoni Bold (TrueType)	No Panose information available
Andalus (TrueType)	No Panose information available
Arabic Transparent (TrueType)	No Panose information available
Arabic Transparent Bold (TrueType)	No Panose information available
Batang (TrueType)	No Panose information available
Cambria & Cambria Math (TrueType)	No Panose information available

Figure 3: Result of searching for similar fonts for Time New Roman shows no PANOSE for Arabic fonts



Qadi Linotype	عصفور الجنة	زهرة الفلامنجو
Sakkal Majalla	عصفور الجنة	زهرة الفلامنجو
HS Alnada	عصفور الجنة	زهرة الفلامنجو

Original table shows name of three Arabic fonts and sample of each font



Qadi Linotype	عصفور الجنة	زهرة الفلامنجو
Sakkal Majalla	عصفور الجنة	زهرة الفلامنجو
HS Alnada	عصفور الجنة	زهرة الفلامنجو

When the document is opened in a machine where Qadi Linotype and HS Alnada are missing, they were replaced with Arial font.

Figure 4: Example of document change when fonts are missing

Based on the above reasons, this research has been conducted to propose an extension of the PANOSE-1 system to support Arabic script. The main focus of this original idea is related to Arabic fonts designed for text. The proposed model describes Naskh style design characteristics, which are most suitable for text. The model efficiency has been evaluated by testing its function within clustering a set of fonts according to their similarity. The results of the clustering are compared with the result of another clustering on the same set, where the similarity between pair fonts are evaluated using font matching tools. The results of both clustering processes were compared and evaluated using a numerical measure, which indicates the degree of similarity between them.

1.2. Thesis Outline

The thesis is organized as follows. The next section reviews the most prominent characteristics of Arabic script. It reviews how sounds are represented to form alphabets in Arabic. It is also explains how words are cursively constructed and how certain letters are displayed in different forms, called ligature, when they appear after each other. The last section of this chapter reviews the difference between calligraphy and typography and ends by reviewing six common calligraphy styles. The second chapter is intended to provide details about digital fonts and Arabic typeface design characteristics. It explains typography terminologies that are commonly used by designers. It describes the anatomy of Arabic letters, which helps to describe typeface design features. It also reviews current technology of font file formats and how characters in a font file are represented. The third chapter reviews two computerized classification systems for Latin script. One is the IBM classification system, which is composed of two digits. It is defined by IBM and can be exclusively extended or modified by IBM. The other system is the PANOSE-1 system, which is composed of ten digits. It is defined to describe four Latin font families, which are text, handwritten, decorative and pictorial. The fourth chapter describes the proposed extension for PANOSE-1, named PANOSE-A, to support Arabic text fonts. It defines the meaning of each digit, as well as required measurements and classification algorithms used to assign appropriate values. The fifth chapter describes an analytical experiment to evaluate the efficiency of the proposed system to classify similar fonts. It describes the tools, methodology and the results of the experiment. It concludes with a discussion and outlines future work to improve the proposed PANOSE-A system.

1.3. Arabic script

Arabic script is the third most commonly used writing system in the world after Latin and Chinese scripts. It is adopted by several languages such as Arabic, Farsi, Pashto Urdu, Dari, Malay, and Old Turkic. For examples of how other languages use Arabic script see Figure 5. The direction of writing and reading in Arabic goes from the right toward the left. There is an exception for numbers that consist of more than one digit, where the direction of reading and writing goes from the left toward the right just like in Latin [1] [2]. This has led some to call Arabic a bidirectional script.

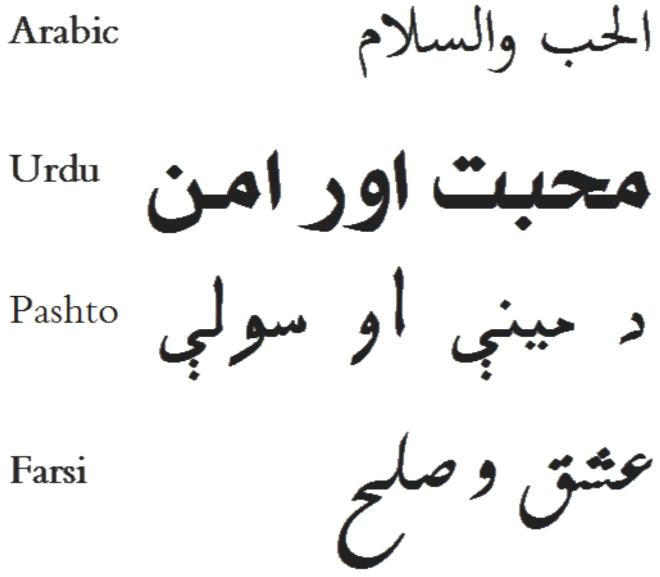


Figure 5: "Love and Peace" phrase in four languages using Arabic script

1.3.1. Alphabet

Arabic consists of three systems for sound representation, which are basic letterforms, diacritic dots and vocalization marks. The first two systems are indispensable [3]. They are combined to make the Arabic alphabet. The basic letterform consists of 18 basic shapes [4]. These shapes are shown in Figure 6.

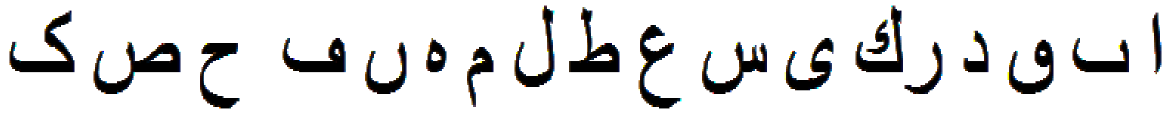
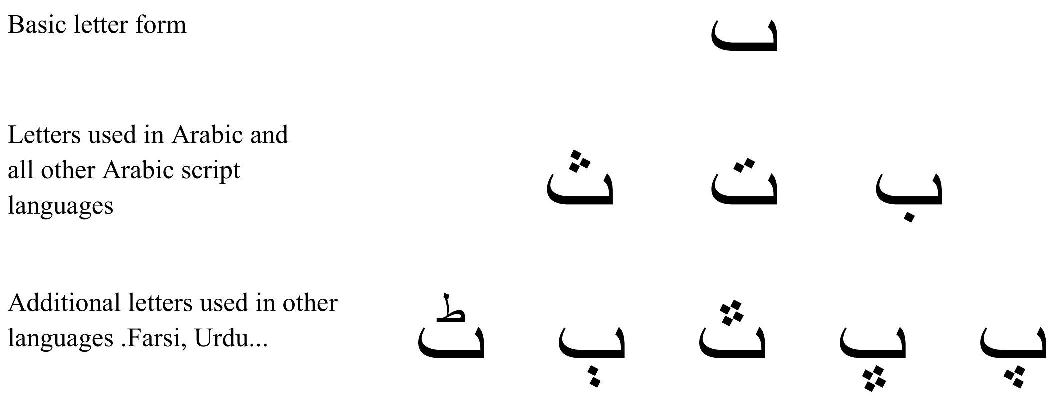


Figure 6: The 18 basic letterforms in the Arabic alphapet

Diacritic dots are used to create new letters, from the same basic letterforms, to represent particular sounds [5] [6]. For example, the basic shape 'ٮ' forms three different letters in Arabic language which are 'ب', 'ت' and 'ث' .The visual appearance of each letter differs in the count and position of diacritic dots, which appear above or below the basic form. For Arabic language, by using basic letter forms and diacritic dots, 28 letters are composed, which represent the consonants only [4]. These 28 letters are shown in Table 1. Three of the letters, Alif 'ا', Waaw 'و' and Yaa 'ي', are pronounced as long vowels depending on the contents of word. Other types of diacritics, called miniature letters, are used in addition to the diacritic dots to represent more sounds in non-Arabic languages e.g. Farsi, Pashto Urdu, Dari, and, Malay, Old Turkic, Uighur. Examples are illustrated in Figure 7 [5] [6].



Basic letter form
ٮ
Letters used in Arabic and all other Arabic script languages
ث ت ب
Additional letters used in other languages .Farsi, Urdu...
ٹ ٻ ٿ ڀ پ

Figure 7: Using diacritics dots and miniature letters to create additional letters

Table 1: The 28 Alphabets in Arabic and their contextual letterforms

Letter Name	Transliteration	Pronunciation & English Equivalent	Contextual forms			
			Final	Middle	Initial	Isolated
alif	Ā	-	ـا	-	-	ا
baa	B	B as in bake	ـب	ـبـ	بـ	ب
taa	T	T as in take	ـت	ـتـ	تـ	ت
thaa	ṯ	th as in thin	ـث	ـثـ	ثـ	ث
jiim	Ǧ	j as in joke	ـج	ـجـ	جـ	ج
haa	h	no equivalent	ـح	ـحـ	حـ	ح
khaa	ẖ (also kh, x)	no equivalent	ـخ	ـخـ	خـ	خ
daal	d	d as in day	ـد	-	-	د
dhal	ḏ (also dh)	th as in this	ـذ	-	-	ذ
raa	r	r as in car	ـر	-	-	ر
zaay	z	z as in zeal	ـز	-	-	ز
siin	s	s as in snake	ـس	ـسـ	سـ	س
shiin	š(also sh)	sh as in shake	ـش	ـشـ	شـ	ش
saad	ṣ	emphatic s	ـص	ـصـ	صـ	ص
daad	ḍ	emphatic d	ـض	ـضـ	ضـ	ض
taa	ṭ	emphatic t	ـط	ـطـ	طـ	ط
dhaa	ẓ	emphatic dh	ـظ	ـظـ	ظـ	ظ
ayn	‘	no equivalent	ـع	ـعـ	عـ	ع
ghayn	ġ (also gh)	no equivalent	ـغ	ـغـ	غـ	غ
faa	f	f as in face	ـف	ـفـ	فـ	ف
gaaf	q	emphatic k	ـق	ـقـ	قـ	ق
kaaf	k	k as in key	ـك	ـكـ	كـ	ك
laam	l	l as in leaf	ـل	ـلـ	لـ	ل
miim	m	m as in make	ـم	ـمـ	مـ	م
nuun	n	n as in none	ـن	ـنـ	نـ	ن
haa	h	h as in hat	ـه	ـهـ	هـ	ه
waaw	w/ū/aw	w as in wake	ـو	-	-	و
yaa	y/ ī/ay	y as in yell	ـي	ـيـ	يـ	ي

1.3.2. Diacritic Marks

The third sound representation system for Arabic script is vocalization marks, which is known as Tashkil. It consists of symbols that represent short vowels. These marks are usually placed above or below letters. There are in total 12 marks [6] [7]. They are used to clarify the precise pronunciation of the letters. They are not frequently used in newspapers, reports, or books. Often, they are eliminated for simplicity and to save of spaces between lines. On the other hand, they are necessary within special texts to aid proper reading. For example, they are used in a text for new language learners like children when they start to learn how to read. They are also intensively used in the holy text of the Quran to insure exact pronunciation. An example is shown in Figure 8. In case of homographs, which are words that have the same spelling but different sounds, vocalization marks are used to avoid ambiguity and aid in proper pronunciation. In another case, they are used to differentiate words that share the same consonantal root as shown in the example in Figure 9. Moreover, they are sometimes used to indicate the grammatical functions of words [6] [7].

Table 2: Vocalization marks of Arabic script

Name	Shape	Name	Shape
Fathah	◌َ	Dagger alif	◌ٰ
Tanwin fathah	◌ً	waslah	◌ٱ
Dammah	◌ُ	kasrah	◌ِ
Tanwin dammah	◌ٌ	Tanwin kasrah	◌ٍ
Sukun	◌ْ	hamzah	◌ٔ
Shaddah	◌ّ	Maddah	◌ٓ

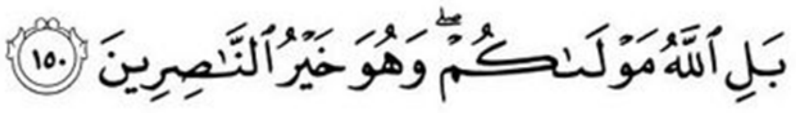

بَلِ ٱللَّهُ مَوْلَىٰكُمْ ۖ وَهُوَ خَيْرُ ٱلنَّٰصِرِينَ ١٥٠

Figure 8: A sample phrase from the Quran which includes vocalisation marks

1.3.3. Cursive

In Arabic, letters that form a word are always connected in a cursive style, whether they are handwritten or printed. Unlike Latin, the characters that form a word can be written independently or connected. In general, the trend of using the independent style is dominant due to its higher legibility. The cursive style may be used with Latin just for aesthetic or imitating handwriting styles.

In Arabic, not all letters have the same connectivity rules. They take four different forms according to their position in a word. Generally, a letter will be joined only from the left side if it is the first letter of a word. If a letter is the last letter of a word, it will be joined only from the right side. A letter will be joined from both sides if its position is in the middle of a word. This left connectivity rule, however, is not the same for all letters. Six letters have exceptions to this rule. These letters are Alif 'ا', Daal 'د' , Dhal 'ذ' , Raa 'ر' , Zaay 'ز' and Waaw 'و'. They are never joined from the left side even when they are positioned in the middle or at the beginning of a word. Table 1 lists all letters and their four forms, isolated, initial, middle and final form.

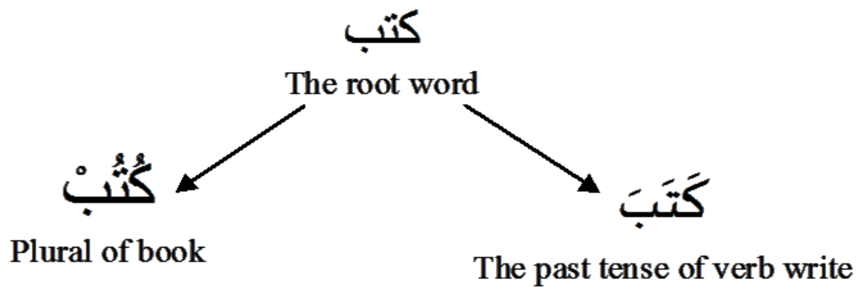


Figure 9: Example of two words that share the same root

1.3.4. Alternates (Allograph)

In Arabic writing, in some cases a letter can have different shapes depending on the letter that follows it. For example, as shown in Figure 10, the initial form of Baa 'بـ' can take different allograph shapes according to its left neighboring letter [8]. These allograph shapes enhance legibility and using them is admirable in Arabic writing [4] [9].

Figure 10: Initial Baa's different forms according to neighboring letters [11]

1.3.5. Ligatures

A ligature is when a combination of two or more letters is represented by a single glyph. It is used for several different purposes. In Latin script, it may used to represent new letters such as the letter 'œ' in French language, which is the combination of letter 'o' and 'e' [10]. Sometimes it is used to enhance legibility. For example in Figure 11-a, the ligature 'fi' is used to improve the appearance of the combination of letters 'f' and 'i', where the teardrop part at the top of 'f' interferes with the dot of the letter 'i'. In Arabic script, using ligatures is very common. Some are mandatory such as the 'لا' ligature shown in Figure 11-b. It is one of the most commonly used ligatures in Arabic script. Some references suggest it should be added to the alphabet set. Table 3 below shows other ligatures that are mostly used for stylish purposes.

f + i → fi → ﬁ

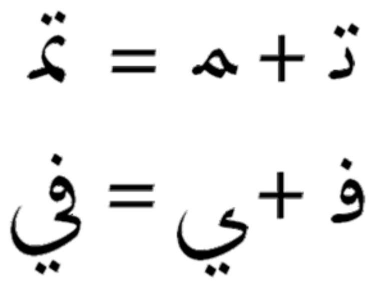
(a)

ا + ل ← لا

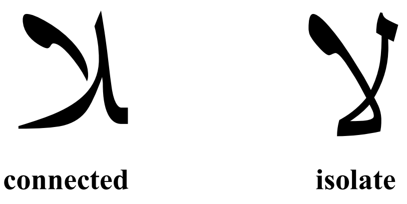
(b)

Figure 11: Example of a ligature: (a) in Latin (b) in Arabic

Ligatures in Arabic script are created not only horizontally. They can be created diagonally or even vertically where letters are stacked on top of each other as the example in Figure 12-a. Some letter combinations have different ligatures based on their position within the word. As an example in Figure 12-b, the isolated ligature 'لا' shape is different than the connected one.



(a y



(b) Different ligature shapes based on position in a word

Figure 12: Examples of ligature

Table 3: Samples of other common ligatures in Arabic script

Without Ligatures	Ligatures
الله	ﷲ
محمد	محمد
نبي	نبي
يثرب	يثرب

1.3.6. Kashidah

The Kashidah is a glyph but it is not for a separate character [4]. The distinction between a character and a glyph is explained in next chapter. Kashidah is a small flowing curve that stretches a character or acts as a junction between two characters in curvilinear way, as in the example shown in Figure 13. Its shape can take several degrees of extension depending on the characters and their contextual forms. Figure 14 shows an example of different levels of using Kashidah.

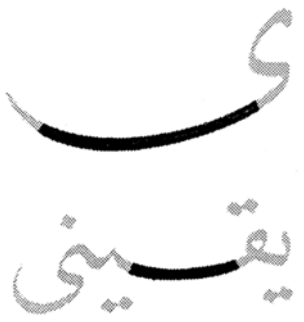


Figure 13: Kashidah in a letter (top) and between two letters in a word (below) [4]

Kashidah is used for various purposes. One of its common uses is for text justification. Modifying the space between words and letters is one text justification method used in Latin. In Arabic, since the letters are naturally connected and spaces between letters cannot be added, the kashidah is used instead. It is also used for aesthetic purposes and to respect some calligraphy styles that tend to shape individual letters to give them the best fit within words.

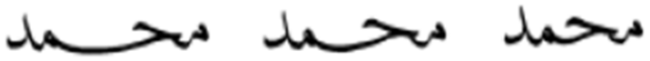


Figure 14: Different levels of Kashidah

1.4. Calligraphy and Typography

Calligraphy and typography are the art of arranging and setting type [3]. Typography refers to the style and the arrangement of machine production or printed letterforms while calligraphy refers to the style and arrangement of handwritten letterforms. Typography can be referred to as letterforms produced by movable metal types in printing machines as well as digitally drawn and produced letterforms using computers. In calligraphy, letterforms that create words are produced using a pen or brush with ink to write on several materials such as paper, leather, or wood. A sample of Arabic calligraphy is shown in Figure 15.

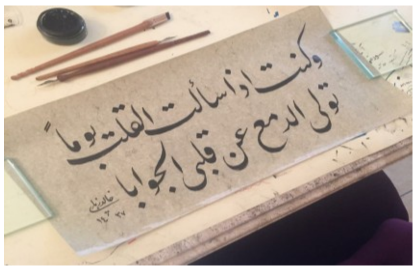


Figure 15: Calligraphy on hand-dyed paper

Arabic calligraphy has been used for years before the appearance of Islam in the 7th century. Its major development started after Islam and Arabic calligraphy has become the core of Islamic Art due to its aesthetic quality. Among all different calligraphy styles that have evolved throughout history, there are six major styles that have been established and continue to be used, which are Kufi, Thuluth, Naskh, Nastaliq, Diwani and Rukaa. Examples of each style are shown in Figure 16. Before the development of typography, calligraphy was the method of writing and copying books. It was common to use calligraphy for art and to decorate buildings. These days, calligraphy is mainly used for aesthetic purposes and art. Each of these six styles of Arabic calligraphy has a different visual appearance that makes it distinct from the others.

Kufi style was established, in a city named Kufah, in Iraq in the 6th century, many years before Islam. After Islam, it was formally used to transcribe the Quran. Generally, it is a geometrical style. Its visual appearance is distinct due to the combination of angular square lines, bold circular forms and squarish letter proportions. Its strokes are monumental bold with short ascender and decender as well as extended horizontal strokes [3] [4].

Thuluth style was created in the 7th century and became a fully standardised style at the end of the 9th century. Originally, it was a large display style. It was often used for titling and epigrams more than for long-running texts. Then, it was refined into a lighter and more elegant style to be used in writing. Its appearance is distinct by having thin, curved and oblique strokes. In addition, 1/3 of each letter slopes. That is where it got its name from, as the word 'thuluth' is the Arabic word for one third [3] [4] [11].

Naskh style is named from the Arabic verb for copying, which is 'نسخ'. This style is derived from Thuluth style and has evolved to its own form within the 10th century. It is a simpler and more legible style, especially in small sizes. Its lines are thin and naturally round. Naskh has become the most popular script for book copying and has become a familiar style for writing the Quran [3] [4] [11] [12].

Nastaliq style, also known as Farsi, developed in Iran in the 15th century. Its name came from two words, Naskh and Ta'liq 'تعليق', which mean hanging letters above each other. This makes the word space more dense, which creates sophisticated and structured text lines. It is used extensively in copying romantic and mystical epics in Iran. It has deep curved horizontals and is sloped to the right in contrast to the all other styles. It uses wide horizontal swashes frequently. [3] [4] [11].

Diwani style was established in the 15th century in Turkey. It was used originally to write official Ottman Sultans' responses. This style does not follow the baseline rules. It heavily uses curves in one or two directions. Connections between letters can be slanted or vertical. There is no specification of which letter should go above or below. The harmony of the letters in words is most important to position letters together. Vocalisation marks are not used with this style. All the letters are slanted downward [3] [4] [11].

Rukaa is the simplest style for everyday handwriting and has a round fluid style. It was invented in the 9th century but, due to its simplicity, it has become the favourite style in the eastern Arab world for everyday writing. Its words are dense and ligature structure. Its main characteristics are horizontal flatish letterforms. It has a thick baseline and short horizontal strokes, ascenders and decenders [3] [4] [11].

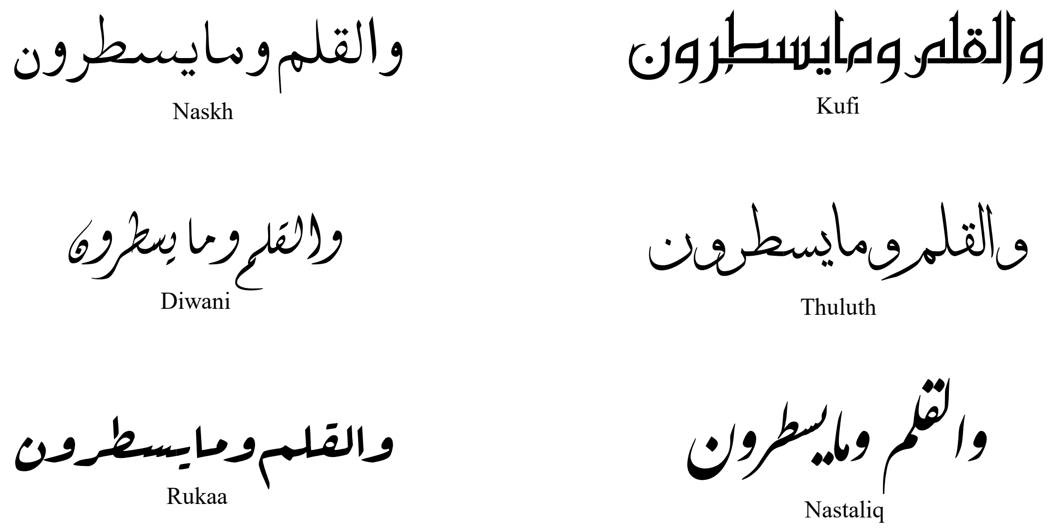


Figure 16: Examples of common Arabic calligraphy styles

Chapter 2 : Digital Typeface

2.1. Typeface and Digital Type

A typeface is a set of characters that share common design features. It is a distinctive and abstract design concept that determines the characters’ visual appearance. Each character can have more than one visual symbol to represent it. Each individual symbol is called a glyph. Usually, a character can be represented by several glyphs. For example in Latin script, character ‘A’, in addition to an uppercase form, it can be displayed in a lower case form with glyph ‘a’, an ordinal ‘^{a}’, or a small cap ‘ᴀ’. The design is described in terms of specialized typographical terminologies that will be explained in detail in the following section.

A typeface can have several styles based on design variations of its design characteristics. For example, in Latin typefaces, common known styles are roman, italic and bold. The roman style is the fundamental style, where the characters appear vertically. The variation of the angle will change the typeface style to italic. When a roman character is slanted without redesigning, the oblique italic style is obtained. When characters are redesigned to be slanted in a cursive style, we get cursive italic style [13]. Many distinct fonts can be obtained from a particular typeface. Several variations in average character width or oblique can be associated with the same typeface design.

The bold style can be obtained by changing the character weight. The most common weight variations are light, medium and bold. Other variations can be obtained, such as extra light, semi light, semi-bold and extrabold or black. Another variation within one typeface can be obtained by changing character width. If the character width becomes narrower than roman, the style is called condensed. If it becomes wider it is called extended. Another variation can be obtained such as Ultra Condensed and Ultra Extended. A Typeface family is the complete set of all the sizes and styles of typeface forms [14].

The size of the font is measured by two basic units. The first unit is point (1/72 inch), which is used to measure character height and spaces between lines and paragraphs. The second unit is pica (12 points) which is used to measure the width of columns or spaces between columns. Em and en are two other measures, which are commonly used when describing spaces between characters and words or the space of a special character, such as long dash e.g. (em-dash) in page design. Em is derived from the width of capital ‘M’ of a given typeface. It is equal to the square of the type size. For example, if the type size is 48 pt, the em is quad of 48 pt, 48 pt height by 48 pt wide. The en is half of the em [3] [15].

When the typeface design is obtained for a set of characters in a particular size, weight, width and slant, it is called a font [10] [16]. The contents of the characters set vary depending on the font’s purpose, usage and whether it is a multi-script font. For Arabic, the basic set of characters consist of the free-standing letters, their alternative letterforms, diacritics dots and vocalization marks. Also, it may contain a ligature set. Other additional characters of non-alphabetic sets like signs, symbols, decorative motifs and pictograms may be included [3].

2.2. Anatomy of Arabic Letters

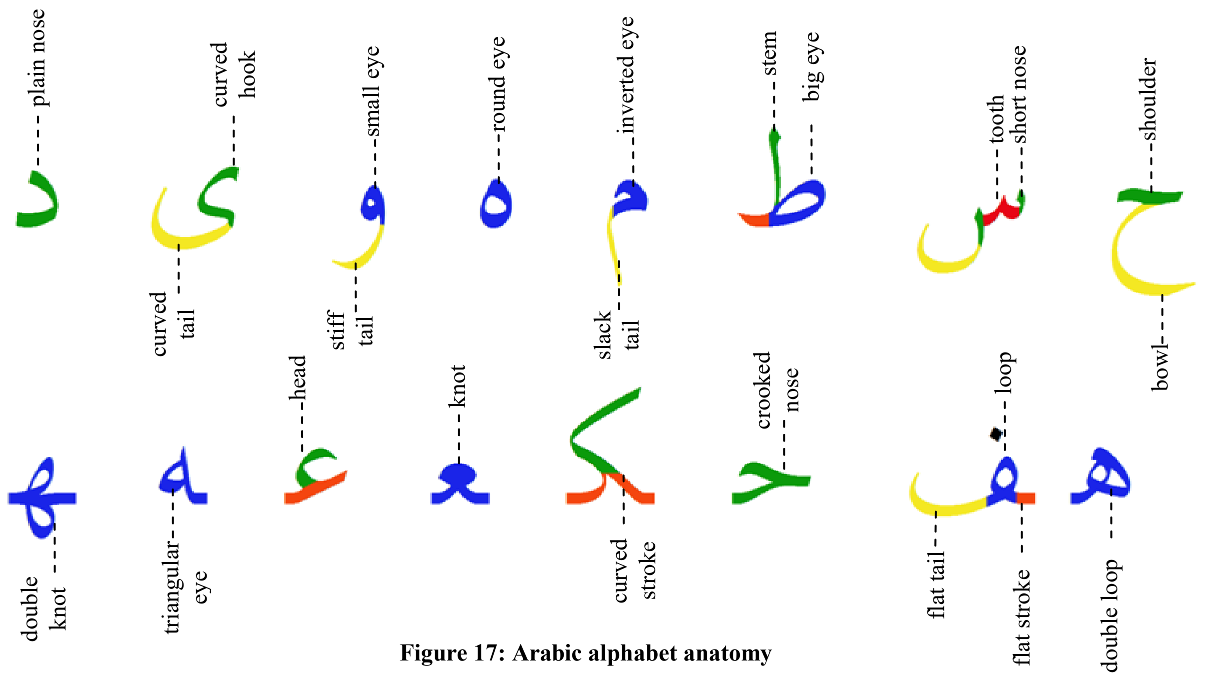


Figure 17: Arabic alphabet anatomy

Type designers use specific terminologies to describe the general structure of alphabet letterforms. They provide basic parameters and help define certain typographical aspects for the important parts of all letters. Those parameters are applied for traditional and digital forms of typography because they resemble abstract font graphical characteristics. They are taken from different sources including calligraphers’ techniques, engraver art (counter), geometry (apex, vertex) and anatomy analogy (arm, eye). Until know, there are no accepted conventions about terminologies for Arabic letters [3]. In this thesis, the terminologies in [3] [4] will be used due to their comprehensiveness for all basic letterforms. In [4], strokes for Naskh and Kufi styles are categorized into six groups. These groups are beginning, middle, ending, closed shape, diacritics and connecting strokes. Figure 17 illustrates these terminologies applied to Arabic script. A detailed description for each group will be provided.

2.2.1. Beginning Strokes

As the name of this set indicates, these strokes appear mostly at the beginning of some letters.

- **Stem:** A vertical stroke that appears in letter Alif 'ا'. Lam 'ل', Kaaf 'ك ', Taa 'ط'
- **Short Nose**: A small vertical stroke that extends above the horizontal stroke. It appears at the beginning of letter Baa 'ب' and Siin 'س'. It is common to call this part a tooth even if it is not identical to the ones explained below.
- **Plain Nose:** This stroke is used to represent the letter Daal 'د'.
- **Crooked Nose:** This stroke appears in the middle and the initial form of the letter Haa 'حـ'and other letters that share the same basic letterform.
- **Head:** The elliptical part that appears at the top of the letter Ayn 'ع'.
- **Shoulder:** The flat horizontal stroke that is located above the bowl part of letters such as Khaa 'خ' and Haa 'ح'.
- **Zig-Zag:** The stroke with two corners at different angles. It appears in initial and middle forms of letter Kaaf 'كـ'.
- **Curved hook:** This stroke appears in letter Yaa 'ي' in its isolated form.

2.2.2. Middle strokes

The strokes of this group usually appear between two strokes within the same letter or in middle form of some letters.

- **Tooth:** The short vertical stroke extends upwards, as in the letter Siin 'س' or as in the middle form of the letter Taa ' ـتـ '.

2.2.3. Ending Strokes

The strokes of this group usually appear in final and isolated forms of letters.

- **Bowl:** The elliptical or round stroke of letterforms, such as in Haa ‘ح’ and Ayn ‘ع’.
- **Slack Tail**: The straight stroke of letterforms that extend vertically below the baseline, as in the isolated letter Miim ‘م’.
- **Stiff Tail:** The sharp curved stroke of letterforms that extend below the baseline, as in the isolated letter Waaw ‘و’.
- **Curved Tail:** The upward curved stroke that appears as the terminal part of letterforms, as in the isolated letters Nuun ‘ن’, Yaa ‘ي’ and Siin ‘س’.
- **Flat tail:** The flat stroke that extends horizontally along the baseline of letterforms, as in the letters Baa ‘ب’ and Taa ‘ت’.

2.2.4. Closed Shape

Closed Shape strokes have one counter or two in some letters that are entirely enclosed by a letterform.

- **Knot:** The part of letterforms that is filled and closed such as the middle form of the letter Ayn ‘ـعـ’. Another variation is called the double knot as in middle Haa ‘ـهـ’.
- **Eye:** The enclosed counters in letterforms. There are different types according to their shapes, such as the small eye in letter Waaw ‘و’, big eye in Saad ‘ص’, round eye in isolated Haa ‘ه’, triangular eye in final Haa ‘ـه’ and inverted eye in initial Miim ‘مـ’.
- **Loop:** The part of letterform that is shaped like a loop, which encloses a counter, as in the medial form of the letter Faa ‘ـفـ’. There is another variation of loop, which is the the double loop as in initial Haa ‘هـ’.

2.2.5. Connecting Strokes

Connecting strokes are the strokes that connect two glyphs. They take two shapes, flat and curved.

2.2.6. Diacritics

Diacritics are the strokes that represent the diacritics dots and vocalization marks that are illustrated in the previous chapter. The dots can be single, double or triple.

2.3. Designing Arabic Typefaces and Font Formats

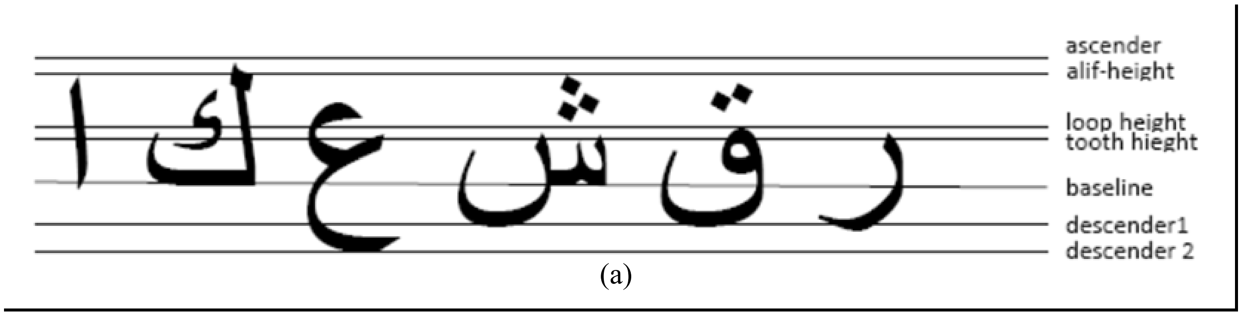
2.3.1. Design characteristics

Designing a typeface is a creative process that can be carried out in several stages. Initially, general design considerations need to be determined. Although the inspiration can come from any source, the final goals and usages of the design may impose some limitations on the design process. There is no font that can be ideal for all printing or display situations. The font designed to be used in newspapers may not the more appropriate for highway signage. For example, display fonts, which are used for titling and for short introductions, are not intended to be used in a very small size. In contrast, text fonts, which are usually used for newspapers and books, are designed to be used in small sizes as a way to enhance legibility and readability of long running passages [17].

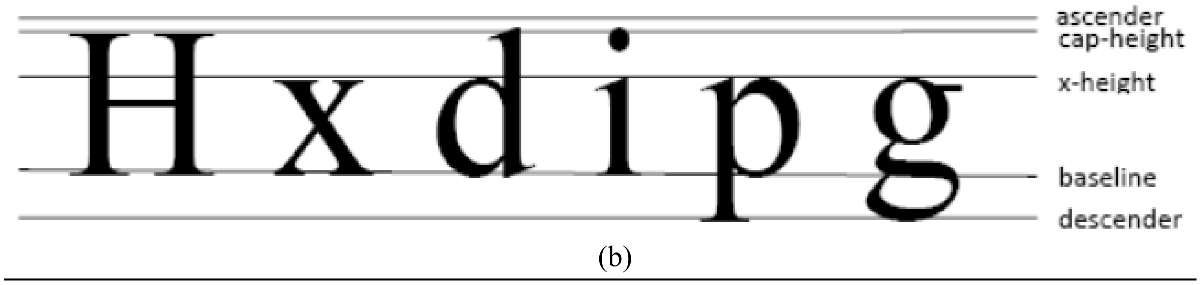
After determining the inspiration and the usage of the font, the basic visual characteristics of the design need to be defined. These characteristics will work as basic guidelines throughout the design process to obtain the consistency and harmonization between all glyphs within a font [3]. One important issue in font design is to match characteristics that are supposed to be the same on different output devices with different resolutions.

The design characteristics are determined at the beginning of the design process. For Latin, it is conventional to have five vertical height references, as illustrated in Figure 18-b. For Arabic, similar references are used with modifications suitable for Arabic script characteristics. Arabic is less restricted than Latin. Therefore, references are not limited to five. There could be up to twelve references depending on the rules the designer wants to obtain in the design [18]. The basic references for any Arabic typeface design, illustrated in Figure 18-a, are defined as follows:

- **Baseline:** An imaginary horizontal line on which most of the letters sit. It is a main vertical reference in designing a typeface; hence it is the base of determining the other vertical references.
- **Tooth-height:** The distance from the baseline to the height of letter with teeth, as in letters Siin 'س' or Baa 'ب'. The design can have one or more tooth heights. It is dependent on how much the design imitates particular calligraphy styles [3].
- **Loop-height:** The distance from the baseline to the height of a letter with a loop, such as Gaaf 'ق' or Faa 'ف'. Like the tooth-height, the design can also have one or more loop heights [3].
- **Alif-height:** The extended distance above the baseline of letter Alif 'ا' [3].The analog of this reference for Latin script is cap-height; which determines the height of uppercase letterforms. Cap-height is usually measured on the capital letter 'H'.
- **Ascender:** For Latin, it is the part of lower case letters that rise above the height of lowercase 'x', which is called x-height [10]. For Arabic, it is the part of some letters that extend above the tooth and loop-heights, as in letters Taa 'ط' and Kaaf 'ك'. Since loop and tooth may not be at the same height, the design can have more than one ascender**.**



(a)



(b)

Figure 18: Basic design vertical references (a) for Arabic (b) for Latin

- **Descender:** The part of letters that extends under the baseline. For Arabic, it defines the height of the bottom part of letterforms in isolated and final forms. A design can have one or more descenders. One is for letters with bowl, such as Haa ‘ح’ and Ayn ‘ع’, the other is for letters with a curled tail, such as Saad ‘ص’ , Siin ‘س’ and Nuun ‘ن’, while another is for letters with stiff tails, as in Waaw ‘و’ and Raa ‘ر’.

In addition to the previous vertical references, other design characteristics are essential for the design process. These characteristics are related to stroke size, orientation and the relationship between them. These characteristics are described as follows:

- **Weight:** This is one of the most important characteristics in typeface design. It is about the heaviness or blackness of the letterforms [16]. For Latin, it can be formally expressed by the ratio between the thickness of the vertical stem and the x-height. For Arabic, It can be defined as the thickness of the stem stroke of letter Alif ‘ا’ [4]. The stem widths, in some designs, vary from letter to letter. Therefore, the weight is described as the average value to approximate the measurement [16]. The larger the value of the weight, the darker each letter will be. Formulas are approximation of what people describe visually.
- **Contrast:** This is an important typography design parameter. In Latin typography, contrast is defined as the relationship between vertical and horizontal stroke thickness [16]. For Arabic, it is defined as a ratio between the thickest and thinnest parts of a character [3]. The larger the ratio, the higher is the contrast. When the ratio is low, it means that the contrast is close to unity.
- **Proportions:** It is about the relationships letter shapes have with each other. For Latin, the proportion can be determined by the geometric rules that are used to construct letters. The proportion can also be determined by the general letters horizontal extent where it can be condensed, even width or expanded. For Arabic, font proportion is defined as the ratio between character width and character height as defined in [3]. This type of proportion can be initially determined when designing the letter Siin ‘س’ [3]. Other type of proportion is determined based on the horizontal extent variations as in Latin. Although variations of advance width proportion create new fonts of the same family, there few Arabic designers do that [4]. Other types proportions can be defined between stroke thickness and open counters, which determine the overall color [3] [19].

Although the quality of the font is evaluated in the context of the final design, it is important to test basic design characteristics with test words before designing the whole set of characters. The designer needs to estimate the overall color of typeface and whether its elements visually interact in harmony. For Arabic, there are no standard conventions of what test words or sentences should be used. In [3], the author indicates that any test sentence that includes the essential basic letter shapes can be used to estimate the overall color of the text. They suggest pangram "هي كطوع المشعلان" be used as a test sentence. For the test words for Arabic, it is important to have letters with ascenders like in Alif 'ا', Khaf 'ك'and Taa 'ط', desscenders like in Raa 'ر' , Noun 'ن', Haa 'ح'and Ayn 'ع', eyes like in Waaw 'و', Loop like in Faa 'ف' and teeth like in Siin 'س' and Baa 'ب'.

Since Arabic letters have a variety of glyphs, it is common to start by designing free-standing letters. In general, character shapes can be partly copied to create new ones. Therefore, based on the design of isolated letters, other letterforms can be designed. Diacritic marks, which include dots and vocalization marks, can be then designed. After that, depending on typeface style, the needed ligatures are designed. The final set consists of non-alphabet set signs, symbols, decorative motifs and pictograms.

2.3.2. Font Format

A font file is the container of all glyphs of the character set. The format of the file is varies according to the technology used to store and represent glyphs. Based on the file format, the system uses a suitable engine to render a font on an output device. Generally, bitmap and outline are the two basic techniques used to represent glyphs. The two techniques differ in their storage, scaling efficiency and their typographic utilities. Bitmap is a very old technique and it is no longer commonly used. In this technique, each glyph is resembled by dot configuration in a matrix of white and black pixels, which define which pixels will be turned on when the font is used. Fonts that use this technique are practical for rendering on screen and are easier to use. On the other hand, they are not scalable and require complete character sets for each specific size. Therefore, more memory is needed to store fonts' files.

The outline technique uses a better way than bitmap, to represent the glyphs of characters. It only describes the curves within a glyph outline by a set of cubic or quadratic parametric equations called Bèzier curves [15]. Font files that use the outline technique requires less memory and can be easily scaled into different sizes. In addition, it is possible to produce different styles of a font by applying affine transformation such as rotation and shearing.

When outline fonts need to be represented on screen, a glyph representation needs to be transformed into bitmap. The outline fonts are supported by special instructions, called hinting, to enhance the quality of glyph bitmap configuration at any size or output resolution. There are overall hinting instructions and instructions for each specific glyph. Generally, in addition the glyphs' outlines, a font file includes a lot of tables that makes the font smarter to adapt to several situations. Each one of those tables has a specific role. The Type and number of tables differ according to the font format. Under this technology, several types of font formats have been developed such as Type1, TrueType and OpenType.

The Type1 format is based on PostScript, which is a page description language developed by Adobe. This language was initially invented for printing. It is used to describe a whole page and put that description into a digital file that is used to print the page at any resolution without losing much of its quality. In the Type1 font format, a set of PostScript special operators are used to describe the glyphs outlines.

Apple and Microsoft have also worked together to develop a TrueType font format. TrueType uses quadratic curves to represent glyphs whereas Type1 uses cubic ones. Therefore, it is possible to convert TrueType to Type1 without loss. The TrueType format is more complicated and come with more instructions than Type1. These instructions are stored in tables. A font file can contain more than 40 tables; some are required while others are optional. In Table 4, some of the required tables in TrueType are listed.

Table 4: Required tables for TrueType and OpenType fonts

Name	Description
name	Contains font general information
cmap	Contains multiple mapping tables, between characters and glyphs, from Unicode and 8-bits ASCII encoding.
head	Contains overall font matrices and font checksum.
hmtx	Contains horizontal metrics such as advance width and left side bearing for each glyph.
hhta	Contains overall horizontal metrics
hhea	Contains overall advance widths for hinted glyphs.
maxp	Indicates complexity of glyphs and hints.
OS/2	Contains information for the operating system such as codepoint ranges covered by glyphs, line spacing, weight, style, overall appearance
glyf	Contains glyphs' outlines and hints.
post	Contains information needed when font used in PostScript environment.

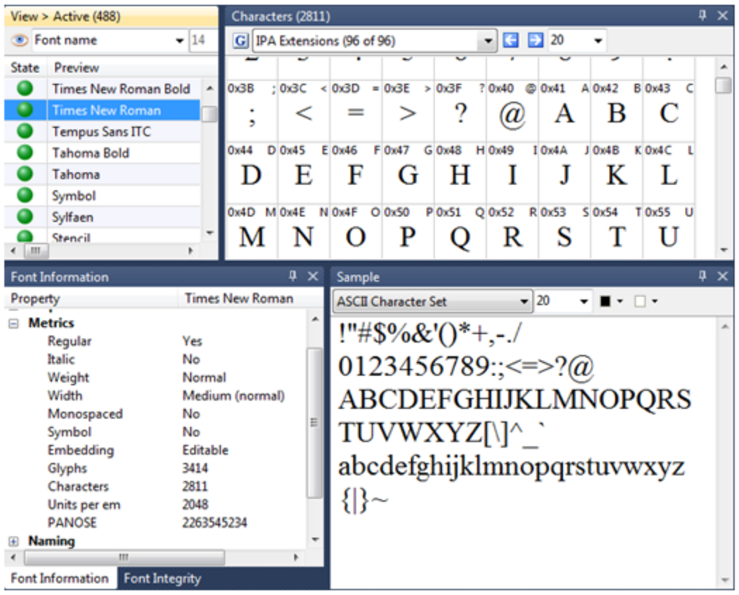
The OpenType font format was invented by Adobe and Microsoft together. It combines the advantages of both TrueType and Type1 formats and includes other additional features. It supports complex scripts and contextual analysis processes that display the appropriate glyphs based on proceeding and following ones. The OpenType font format is based on the Unicode encoding system. This code is a universal character code system that allows putting more than one script in the same font file. This enables creating more complex fonts that could work on most computer platforms. OpenType fonts can contain over 65000 different glyphs in addition to detailed instructions on how these glyphs should appear in a text. Ligatures, small caps, or alternate letters can appear automatically if the option is activated in a font. It has the required tables exit in TrueType and an addition to several optional tables. Table 5 lists some of these tables that are used for Arabic fonts [9] [20].

Table 5: Additional tables for OpenType

Name	Description
GSUB	Contains glyphs substitution information. Type 1, map default alphabetic forms to corresponding form such as initial, middle or final. Type 3, substitute a character with corresponding form with swash. Type 4, replace consecutive glyphs with a ligature glyph.
GPOS	Contains glyph positioning information. Type 2, adjust space between glyphs. Type 3, adjust the position of cursive characters. Type 4, adjust the position of marks in relation to character glyphs or ligature.
BASE	Contains baseline information.
JSTF	Contains information for paragraph justification.

Chapter 3 : Latin Typeface Classification

In desktop publishing and web design, selecting appropriate typefaces has a great impact on the design. Nowadays, there are thousands of typefaces to choose from. Classifying typefaces into groups will narrow down the search space and facilitate the font selection process. Typefaces can be classified according to historical aspects, particular functions or specific design characteristics.



Times New Roman Bold
Times New Roman
Tempus Sans ITC
Tahoma Bold
Tahoma
Symbol
Sylfaen
Stencil

Property	Times New Roman
Metrics	
Regular	Yes
Italic	No
Weight	Normal
Width	Medium (normal)
Monospaced	No
Symbol	No
Embedding	Editable
Glyphs	3414
Characters	2811
Units per em	2048
PANOSE	2263545234
Naming	

!"#$%&'()*+,-./
0123456789:;<=>?@
ABCDEFGHIJKLMNOPQRS
TUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz
{|}~

Figure 19: Example of MainType windows that displayed Time New Roman font information

For Latin, two computerized classification systems have been developed based on design characteristics. In general, typeface design features are encoded into numbers that are compared and sorted. Usually the numbers are determined by typeface designers or vendors and are stored in font files. Several font management tools, such as MainType and Typograf, are able to display this information as well as sort and compare fonts based on their attached encoded numbers as shown in the example in Figure 19. The two computerized classification systems are IBM and PANOSE-1 which will be reviewed in more detail in the next section.

3.1. IBM Classification System

This number is used by font designers and suppliers. It is used in selecting an alternate font when the requested one is not available. It classifies fonts based on their appearance. It gives the font a number which consists of two digits. The first digit represents the IBM font class and the second digit represents the IBM font sub-class parameter. The first digit represents the most general class while the second digit represents the most specific sub-class. The values of these two digits are registered values assigned by IBM to each family font. It classifies fonts according to their appearance but does not identify typeface variation, designer, supplier, or size. It is possible to have a font classified into more than one IBM class or sub-class [21]. This information is inserted into an OS/2 table of TrueType fonts under the entry sFamilyClass. It contains 10 classes; each class has its own sub-classes [15]. Figure 29, on page 39, summarizes all the values for classes and sub-classes within this system. Each class and sub-classes can take a value between 0 and 15. The 0 value for sub-class digit means that the font has no design sub-classification while 15 indicates miscellaneous for designs that are not covered by other sub-classes.

Class ID=1: Old Style Serifs

The style of fonts within this class is based on Latin printing used between 15th and 17th century. The style is characterized by mild diagonal contrast and bracketed serifs. Under this class, eight sub-classes are defined while the rest are reserved for the future use by IBM.

Sub-class 0: No Classification

This is assigned when the font has no design sub-classification.

Sub-class 1: IBM Rounded legibility

This is characterized by large x-height, short ascender and short decender.

Sub-class 2: Garalde

This is characterized by medium x-height and tall ascender. An example of this sub-class is the ITC Gramond family. A sample is shown in Figure 20.

Sub-class 3: Venetian

This is characterized by medium x-height, monotone appearance and a sweeping tail.

Sub-class 4: Modified Venetian

This is characterized by large x-height, monotone appearance and a sweeping tail. An example of this sub-class is the Linotype Palatino family. A sample is shown in Figure 20.

Sub-class 5: Dutch Modern

This is characterized by large x-height, wedge shape serif and a circular bowl appearance.

Sub-class 6: Dutch Traditional

This is characterized by large x-height, wedge shape serif and a circular bowl appearance. An example of this sub-class is the IBM press Roman family.

Sub-class 7: Contemporary

This is characterized by small x-height with light stroke and serif. An example of this sub-class is the University family. A sample is shown in Figure 20.

Sub-class 8: Calligraphic

This is characterized by calligraphy fine handwriting style while maintaining the old style appearance.

Sub-classes (9-14):

This reserved for future use and will be assigned only by IBM.

Sub-class 15: Miscellaneous

This is for miscellaneous designs that are not covered by the other sub-classes.

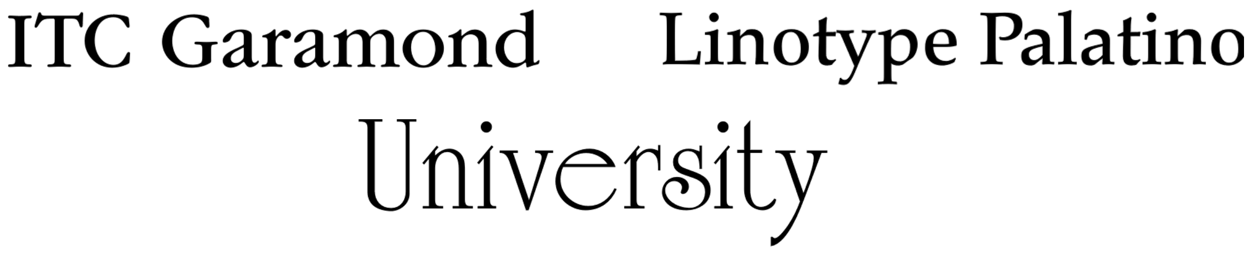


Figure 20: Example of fonts of Old Style Serifs Class ID=1

Class ID=2: Transitional Serifs

The style of fonts in this class is based on Latin printing used between 18th and 19th century. It is characterized by bracketed serifs and clear vertical contrast, where the vertical strokes thickness is heavier than the horizontal strokes. Under this class, two sub-classes are defined while the rest are reserved for future use to be assigned by IBM.

Sub-class 1: Direct Line
This is characterized by medium x-height with fine serifs. All uppercase letters are approximately the same width. An example of this sub-class is the Monotype Baskerville family. A sample is shown in Figure 21.

Sub-class 2: Script
This is characterized by a hand written script appearance that maintains the Transitional Direct Line style. An example of this sub-class is the IBM Nassim family for Arabic script. A sample is shown in Figure 21.

Sub-classes (3-14):
These sub-classes are reserved for future use by IBM.

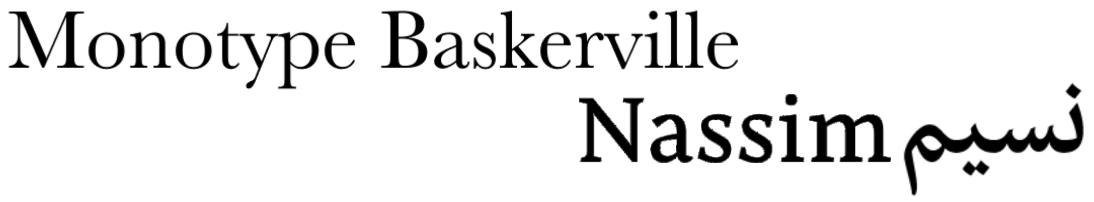


Figure 21: Example of fonts of Transitional Serifs Class ID=2

Class ID=3: Modern Serif

The style of fonts in this class is based on Latin printing used within the 20th century. It is characterized by extreme contrast between the thickest and thinnest parts of strokes. Under this class, two sub-classes are defined while the rest are reserved for future use by IBM.

Sub-class 1: Italian
This is characterized by medium x-height with hairline serifs. As an example of this sub-class is the Monotype Bodoni family. A sample is shown in Figure 22.

Sub-class 2: Script

This is characterized by a hand written script appearance that maintains the Modern Italian style. An example of this sub-class is the IBM Narkissim family for Hebrew. A sample is shown in Figure 22.

Sub-classes (3-14):

This reserved for future use by IBM.

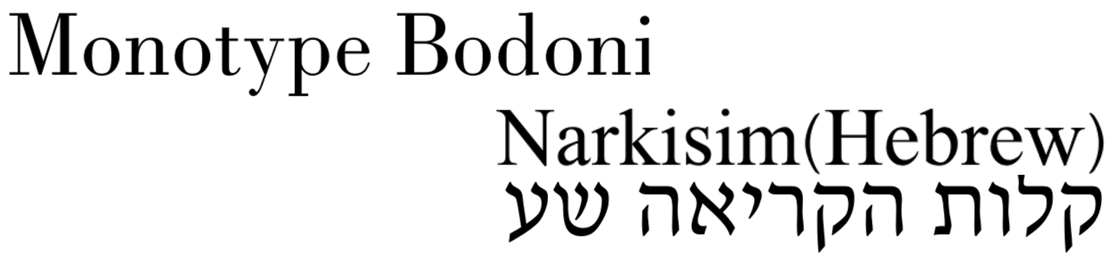


Figure 22: Example of fonts from Modern Serif class

Class ID=4: Clarendon Serifs

The style of fonts in this class is a variation of Old and Transitional Serifs styles. Generally, it is characterized by large x-height and bracketed serifs. Its vertical contrast is mild. Under this class, seven sub-classes are defined while the rest are reserved for future use by IBM.

Sub-class 1: Clarender

This is characterized by having serifs and strokes of almost similar weight. An example of this sub-class is Linotype Clarendon. A sample is shown in Figure 23.

Sub-class 2: Modern

This is characterized by having serifs thinner than the strokes. Its strokes' weights are lighter than Traditional style. An example of this sub-class is Monotype Century Schoolbook. A sample is shown in Figure 23.

Sub-class 3: Traditional

This is characterized by having serifs with lighter weight than the strokes. An example of this sub-class is the Monotype Century family. A sample is shown in Figure 23.

Sub-class 4: Newspaper

This sub-class is similar to the previously described Traditional sub-class but has a simpler style. An example of this sub-class is Linotype Excelsior. A sample is shown in Figure 23.

Sub-class 5: Stub Serif

This is characterized by having short stub serifs with bold stems. An example of this sub-class is the Cheltenham family. A sample is shown in Figure 23.

Sub-class 6: Monotone

As indicated by the name, it is characterized by having monotone stems. An example of this sub-class is the ITC Korinna family. A sample is shown in Figure 23.

Sub-class 7: Typewriter

This is characterized by having strokes with moderate thickness with a typewriter style. An example of this sub-class is the Prestige Elite family. A sample is shown in Figure 23.

Sub-classes (8-14):

These sub-classes are reserved for future use by IBM.

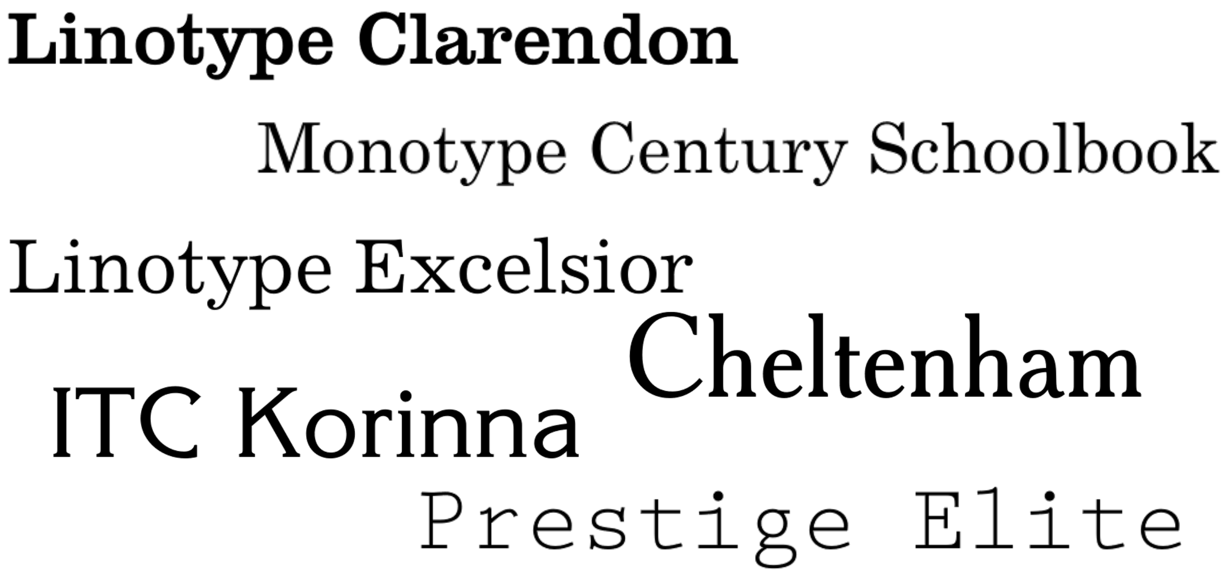


Figure 23: Example of fonts from Clarendon Serifs class

Class ID=5: Slab Serifs

The style of fonts in this class is characterized by a square transition between the strokes and the serifs. Under this class, five sub-classes are defined while the rest are reserved for future use by IBM.

Sub-class 1: Monotone

This is characterized by large x-height in addition to serifs and strokes of equal weights. An example of this sub-class is ITC Lubalin. A sample is shown in Figure 24.

Sub-class 2: Humanist

This is characterized by medium x-height and the serif is lighter than the stroke. An example of this sub-class is the Candida family. A sample is shown in Figure 24.

Sub-class 3: Geometric

This is characterized by large x-height, as well as serifs and strokes of equal weights with geometric design using circles and lines. An example of this sub-class is Monotype Rockwell. A sample is shown in Figure 24.

Sub-class 4: Swiss

This is characterized by large x-height, as well as serifs and strokes of equal weights with more white spaces. An example of this sub-class is the Linotype Serifa family. A sample is shown in Figure 24.

Sub-class 5: Typewriter

This is characterized by large x-height, as well as serifs and strokes of equal weights with geometric design style. As an example of this sub-class is IBM courier. A sample is shown in Figure 24.

Sub-classes (6-14):

These sub-classes are reserved for future use to be assigned only by IBM.

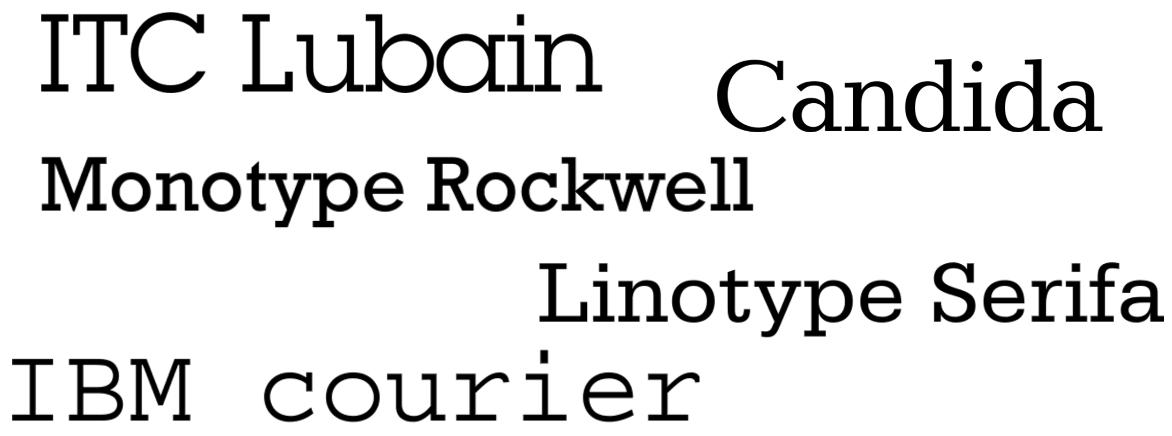


Figure 24: Example of fonts of Slap Serifs class

Class ID=6:

This class is reserved for future use by IBM.

Class ID=7: FreeForm Serifs

Fonts of this class also include serifs but have more freedom in the design, which is not classified under other types of serif classes. Under this class, one sub-class is defined while the rest are reserved for the future use by IBM.

Sub-class 1: Modern
This is characterized by medium x-height, light contrast between strokes and round full design. An example of this sub-class is the ITC Souvenir family. A sample is shown in Figure 25.

Sub-classes (2-14):
This reserved for future use only by IBM.

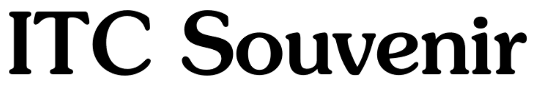


Figure 25: Example of font of Free-Form Serifs class

Class ID=8: Sans Serif

As the name indicates, fonts of this class do not have serif. Under this class, eight sub-classes are defined while the rest are reserved for the future use by IBM.

Sub-class 1: IBM Neo-grotesque Gothic
This is characterized by large x-height and uniform stroke thickness. Also, it is characterized by a simple one story design with hand tuned and medium resolution. An example of this sub-class is the IBM Sonoran Sans Serif family. A sample is shown in Figure 26.

Sub-class 2: Humanist
This is characterized by medium x-height and light contrast with classic roman letterform. An example of this sub-class is the Linotype Optimal family. A sample is shown in Figure 26.

Sub-class 3: Low-x Round Geometric
This is characterized by low x-height, monotone stroke weight and a round geometric design. An example of this sub-class is the Fundicion Tipograficia Neufville Futura family. A sample is shown in Figure 26.

Sub-class 4: High-x Round Geometric
This is characterized by high x-height and uniform stroke thickness with a round geometric design. An example of this sub-class is the ITC Avant Grade Gothic family. A sample is shown in Figure 26.

Sub-class 5: Neo-grotesque Gothic
This is characterized by high x-height, uniform stroke thickness and simple one story design. An example of this sub-class is the Linotype Helvatica family. A sample is shown in Figure 26.

Sub-class 6: Modified Neo-grotesque Gothic
This subclass is similar to Neo-grotesque Gothic with a different design for letters ‘G’ and ‘Q’. An example of this sub-class is the Linotype Universe family. A sample is shown in Figure 26.

Sub-classes 7-8:

These sub-classes are reserved for future use by IBM.

Sub-class 9: Typewriter Gothic

This subclass is similar to Neo-grotesque Gothic with modern stroke thickness and typewriter style. As an example of this sub-class is the IBM Letter Gothic family. A sample is shown in Figure 26.

Sub-class 10: Matrix

It is characterized by a simple design for a dot matrix printer. As an example of this sub-class is the Matrix Gothic family. A sample is shown in Figure 26.

Sub-classes 11-14:

These sub-classes are reserved for future use by IBM.

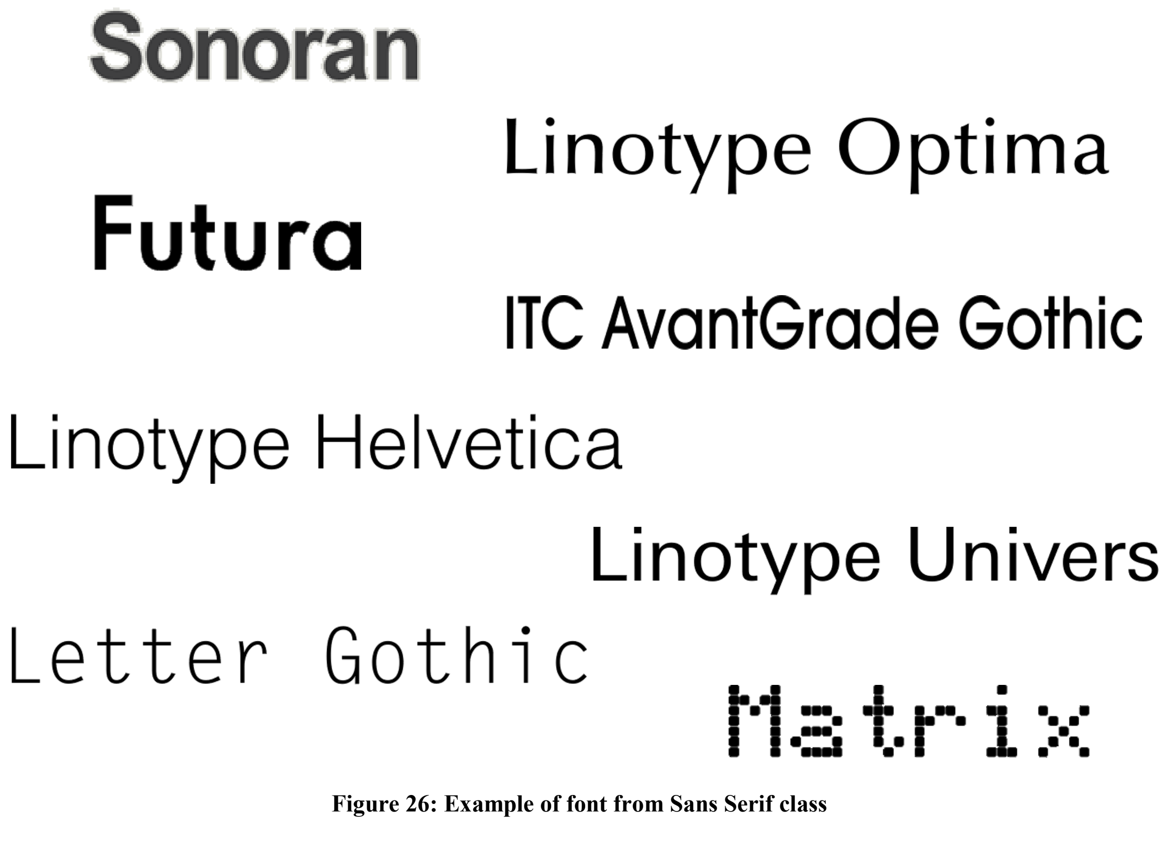


Figure 26: Example of font from Sans Serif class

Class ID=9: Ornamentals

Under this class, four sub-classes are defined while the rest are reserved for the future use by IBM.

Sub-class 1: Engraver
This is characterized by fine lines or lines engraved on stems. An example of this sub-class is the Copper plate family. A sample is shown in Figure 27.

Sub-class 2: Black Letter
Fonts' design within this sub-class is based on the style used within German monasteries during the 12th to 15th century. An example of this sub-class is the Old English family. A sample is shown in Figure 27.

Sub-class 3: Decorative
In this sub-class the ornamental design is inspired from nature. Fonts can have leaves, flowers or animals formed into the strokes of the characters. An example of this sub-class is the Saphire family. A sample is shown in Figure 27.

Sub-class 4: Three Dimensional
Fonts' design within this sub-class is characterized by a 3-D appearance created by using shading or geometric effects. An example of this sub-class is the Thorne Shaded family. A sample is shown in Figure 27.

Sub-classes (5-14):
These sub-classes are reserved for future use to be defined by IBM.

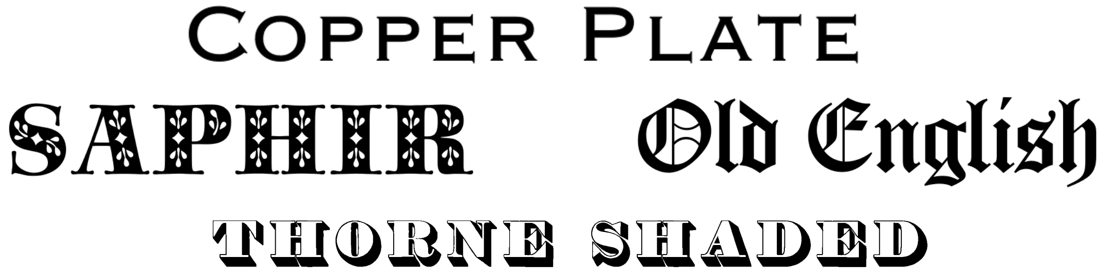


Figure 27: Example of font from Ornamentals class

Class ID=10: Script

Sub-class 1: Unical

Fonts of this sub-class are characterized by having non-connected characters that are drawn in the handwriting style used in Europe during the 6th to 9th century. An example of this sub-class is the Libra font. A sample is shown in Figure 28.

Sub-class 2: Brush Joined

Fonts of this sub-class are characterized by having connected characters with moderate contrast. An example of this sub-class is the Mistral family. A sample is shown in Figure 28.

Sub-class 3: Formal Joined

Fonts of this sub-class are characterized by having connected characters that look like they are drawn with a stiff brush. An example of this sub-class is the Commercial Script font. A sample is shown in Figure 28.

Sub-class 4: Monotone Joined

Fonts of this class are characterized by having connected characters with no or little contrast. An example of this sub-class is the Kaufman family. A sample is shown in Figure 28.

Sub-class 5: Calligraphic

Fonts of this sub-class are characterized by having non-connected characters that look like they are drawn with a broad edge pen. As an example of this sub-class is the Thompson Quillscript family. A sample is shown in Figure 28.

Sub-class 6: Brush Unjoined

Fonts of this sub-class are characterized by having non-connected characters with moderate contrast that look like they are drawn with a brush. An example of this sub-class is the Soltino family. A sample is shown in Figure 28.

Sub-class 7: Formal Unjoined

Fonts of this sub-class are characterized by having non-connected characters with extreme contrast. They look like they are drawn with a stiff brush. An example of this sub-class is the Virtuosa family. A sample is shown in Figure 28.

Sub-class 8: Monotone Unjoined

Fonts of this sub-class are characterized by having non-connected characters with little or no contrast. An example of this class is the Gilles Gothic family. A sample is shown in Figure 28.

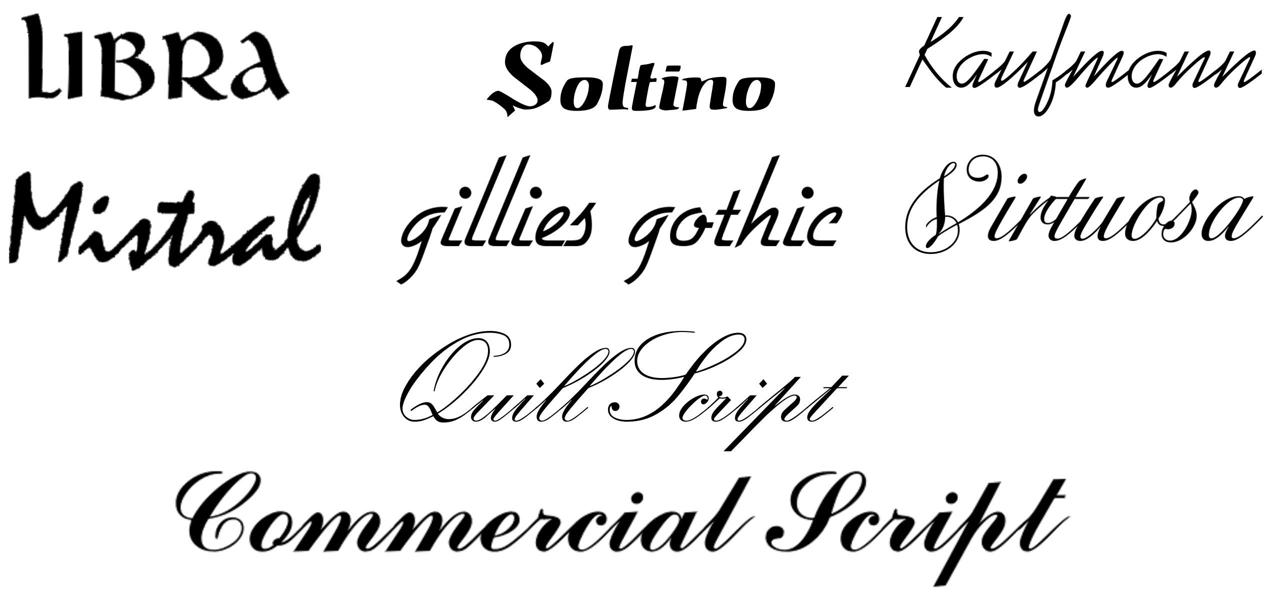


Figure 28: Example of fonts from Script class

Class ID=11:

This class is reserved for future use by IBM.

Class ID=12: Symbolic

This class focuses on special characters in a font. Under this class, three sub-classes are defined while the rest are reserved for the future use by IBM.

Sub-class 3: Mixed Serif

Fonts of this sub-class are characterized by having a combination of serif and sans serif characters such as superscript, subscript, numbers and symbols. An example of this sub-class is the IBM Symbol family.

Sub-classes 4-5:

These sub-classes are reserved for future use by IBM.

Sub-class 6: Old Style Serif

Fonts of this sub-class are characterized by having an old style serif. An example of this sub-class is the IBM Sonoron Pi Serif family.

Sub-class 7: Neo-grotesque Sans Serif

Fonts of this sub-class are characterized by similar features of Neo-grotesque Sans Serif class.

Sub-classes 8-14:

These sub-classes are reserved for future use by IBM.

Class ID=13-14:

These classes are reserved for future use by IBM.

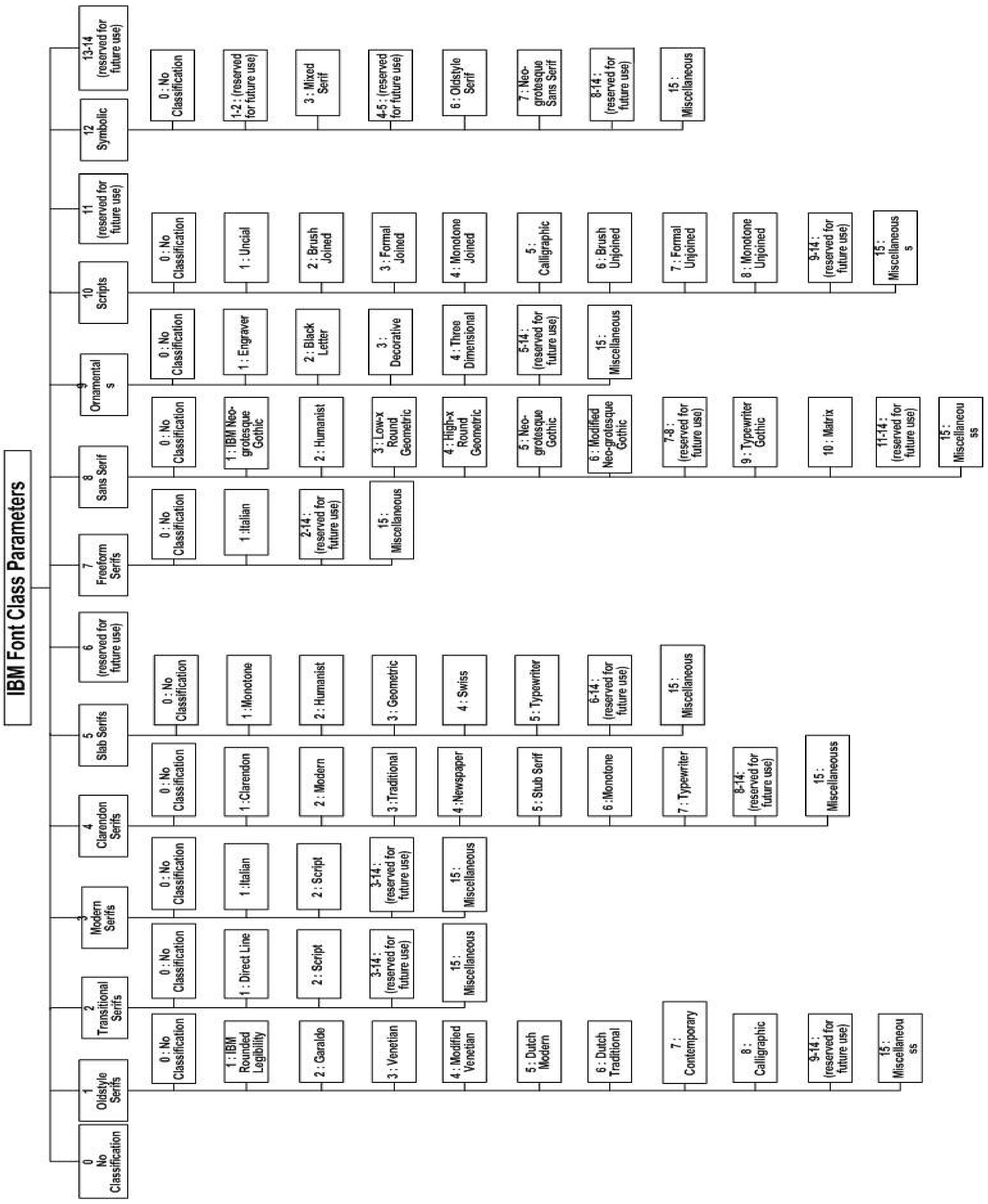


Figure 29: I BM font classification parameters

3.2. PANOSE-1 Classification System

3.2.1. What is PANOSE?

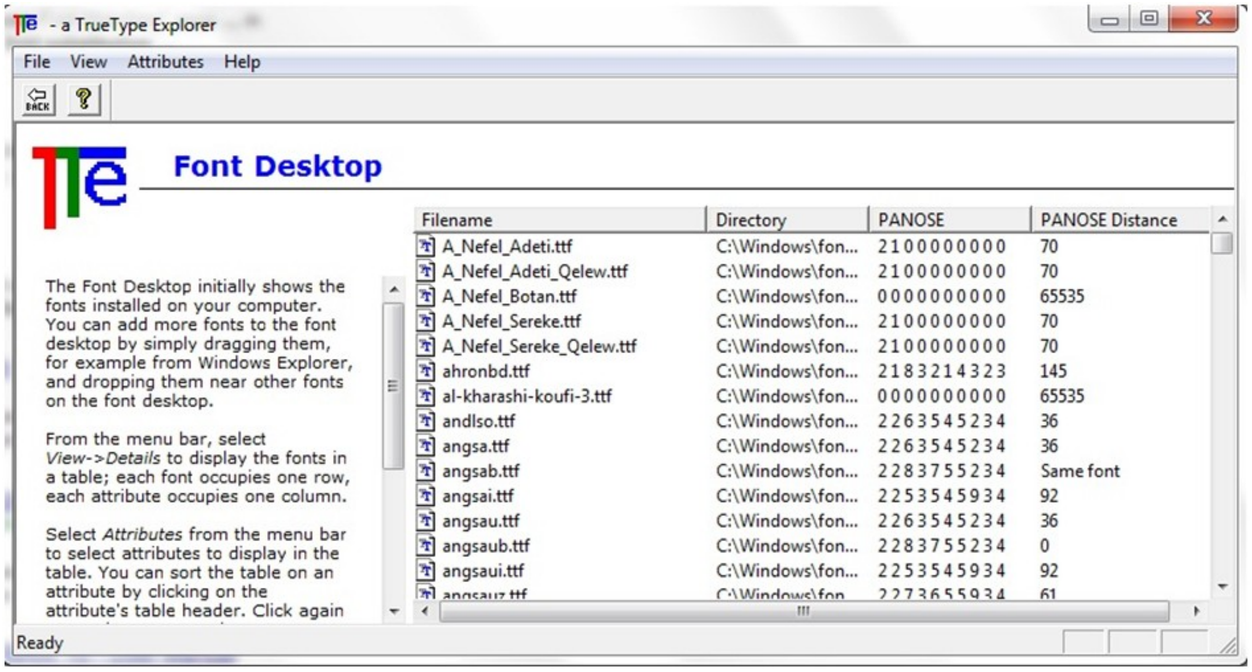
PANOSE-1 is a system for classifying and matching typefaces based on their visual and design characteristics. The author of PANOSE, Ben Bauermeister, divided the Latin alphabet into six letter groups: diagonal (A, V, W, Z), square (H, N, X, K, M), half-square (E, F, T, L, Y), round (O, C, Q, G), half-round (S, U) and quarter-round (D, B, P, R, J). The name "PANOSE" came from selecting one letter of each group [15] [22].

The system encodes the description of characteristics into a number called the PANOSE number. The original version of PANOSE is composed of seven digits [22]. Then, in 1991, a PANOSE-1 version was defined with ten digits. Each digit represents a visual or design characteristic and takes a value between 0 and 15, or (0-9, A-F) if a hexadecimal number is used. Each value has a specific meaning related to its represented feature. In the font design process, the PANOSE number is inserted into the OS/2 Table of the Rich Font Description (RFD) [22].

A PANOSE-1 number is used in several applications. It can be used for font sorting, comparing, searching and grouping based on visual characteristics. For example, Typograph is a font management tool that compares fonts' similarities based on their PANOSE-1 numbers. Another font tool, called True Type Explorer (TTE), shown in Figure 30, uses the PANOSE-1 number to find the best font match for a specific font. In addition, it sorts fonts based on their PANOSE-1 numbers. It is also used to classify fonts in font management software, such as Fontmatrix.

Another important use of the PANOSE-1 number is in the process of font substitution. When an application or operating system detects a missing font in a foreign document or website, the PANOSE-1 number is used to find the best matching substitution font. Finding font substitution by comparing fonts' names may not provide accurate results as it is possible to have the same font without the same exact name. For example, Adobe provides Goudy Old Style for Windows, while it is called GoudyOldStyle (without space) for Mac [23].

By using a PANOSE-1 number, the substitution algorithm calculates the match value for each available font. The match value represents the distance between the missing font and the available fonts. The smaller the distance value means the more common the visual characteristics are between fonts. The match value can be calculated by the Pythagorean Theorem $\sqrt{\sum_{i=0}^{n}(R_i - A_i)^2 \times W_i}$ where R_i represents a digit of the PANOSE number of missing fonts, A_i is a digit of available fonts and W_i is the weight for each digit. The weight value provides bias to some of the characteristics to have more or less effect on the match value [24]. Moreover, the PANOSE-1 number has been used to generate new fonts [25].



The Font Desktop initially shows the fonts installed on your computer. You can add more fonts to the font desktop by simply dragging them, for example from Windows Explorer, and dropping them near other fonts on the font desktop.

From the menu bar, select *View->Details* to display the fonts in a table; each font occupies one row, each attribute occupies one column.

Select *Attributes* from the menu bar to select attributes to display in the table. You can sort the table on an attribute by clicking on the attribute's table header. Click again

Filename	Directory	PANOSE	PANOSE Distance
A_Nefel_Adeti.ttf	C:\Windows\fon...	2 1 0 0 0 0 0 0 0 0	70
A_Nefel_Adeti_Qelew.ttf	C:\Windows\fon...	2 1 0 0 0 0 0 0 0 0	70
A_Nefel_Botan.ttf	C:\Windows\fon...	0 0 0 0 0 0 0 0 0 0	65535
A_Nefel_Sereke.ttf	C:\Windows\fon...	2 1 0 0 0 0 0 0 0 0	70
A_Nefel_Sereke_Qelew.ttf	C:\Windows\fon...	2 1 0 0 0 0 0 0 0 0	70
ahronbd.ttf	C:\Windows\fon...	2 1 8 3 2 1 4 3 2 3	145
al-kharashi-koufi-3.ttf	C:\Windows\fon...	0 0 0 0 0 0 0 0 0 0	65535
andlso.ttf	C:\Windows\fon...	2 2 6 3 5 4 5 2 3 4	36
angsa.ttf	C:\Windows\fon...	2 2 6 3 5 4 5 2 3 4	36
angsab.ttf	C:\Windows\fon...	2 2 8 3 7 5 5 2 3 4	Same font
angsai.ttf	C:\Windows\fon...	2 2 5 3 5 4 5 9 3 4	92
angsau.ttf	C:\Windows\fon...	2 2 6 3 5 4 5 2 3 4	36
angsaub.ttf	C:\Windows\fon...	2 2 8 3 7 5 5 2 3 4	0
angsaui.ttf	C:\Windows\fon...	2 2 5 3 5 4 5 9 3 4	92
angsauz.ttf	C:\Windows\fon	2 2 7 3 6 5 5 9 3 4	61

Figure 30: Use of PANOSE-1 number to sort and compare fonts in TTE.

3.2.2. Meaning and assigning PANOSE-1 digits

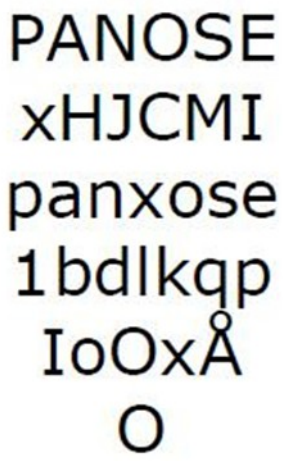
The PANOSE-1 number consists of ten digits. The first digit defines what type of font is being classified. The remaining nine digits provide classification within that type. For example, if the first digit has a value 2, its type is the Latin Text family. The digits that follow are then serif, weight, proportion, contrast, stroke variation, arm style, letterform, midline, and x-height. For the Latin Script family, with a value 3, meaning of the digits that follow are tool kind, weight, monospace, aspect ratio, contrast, topology, form, finals, and x-ascent [26]. The values of the first digit that represent family type and the meaning of the remaining nine digits are summarized in Table 6. Up until now, the PANOSE-1 system considers only four types of Latin typefaces, which are Text, Handwritten, Decorative and Pictorial. No work has been done for Arabic typefaces which is the focus of this research.

Each digit will have 0 and 1 as possible values. Value 0 means 'Any' which is meaningful for the matching process to allow matching with any value. While 1 means 'No fit', which indicates that characteristics are not applicable to typeface [26]. Assigning a PANOSE-1 number to a typeface starts with calculating several measurements related to its visual characteristics [24].

Table 6: The meaning of the digits in a PANOSE-1 number

Digit 1 Family type	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9	Digit 10
2=Latin Text	Serif Style	Weight	Proportion	Contrast	Stroke Variation	Arm Style	Letterform	Midline	X-height
3=Latin Hand Written	Tool kind	Weight	Spacing	Aspect Ratio	Contrast	Topology	Form	Finials	X-Ascent
4=Latin Decorative	Class	Weight	Aspect	Contrast	Serif Variant	Treatment	Lining	Topology	
5=Latin Pictorial	Kind	Weight	Spacing	Aspect ratio of & contrast	Aspect ratio of character 94	Aspect ratio of character 119	Aspect ratio of character 157	Aspect ratio of character 163	Aspect ratio of character 211

There are several steps for assigning values to PANOSE-1 digits. It starts by calculating several measurements of visual characteristics such as serif, weight, contrast etc. Measurements are taken on samples of characters at any size, like the ones shown in Figure 31. An explanation and an illustration of all the measurements used for PANOSE-1 are grouped into tables 12-19 in the appendix. The ratios are then computed using those measurements. The assigned values to PANOSE-1 digits are based on those ratios [26]. The value of this digit identifies two of the font properties, which are the script kind and the genre kind. The values defined in PANOSE-1 are listed in Table 6. The values from 2 to 5, indicates that the font script is Latin and the genre sequentially is Text, Hand Written, Decorative and Pictorial. The rest of the values are open to cover any undefined extension. The value of this digit determines the meaning of the remaining nine digits as indicated in Table 6. The details of each family digit are presented in the rest of this chapter.



PANOSE
xHJCMI
panxose
1bdlkqp
IoOxÅ
O

Figure 31: Character set that is needed to produce a PANOSE-1 number.

3.2.2.1. Latin Text

Digit 1: Family Type

Assigning values for this digit does not require any measurements. The characters of fonts in this family have standard topology and are suitable to compose a paragraph of text. Fonts of this family usually have versions in italic style.

Digit 2: Serif Style

This digit describes the appearance of serif used in a font design. In this model, fourteen general serif shapes are described. Sixteen measurements are required to determine the value of this digit. The measurements are measured on three letters, ‘I’, ‘E’ and ‘H’. Within one family, serif style may change according to the change of the weight. Therefore, measurements should be calculated using medium weight.

The classification is done in several steps. Initially, it starts by separating Serif from Sans Serif classes. Then, each of the two main classes is narrowed down into sub-classes. For the Serif class, sub-classes include Cove, Square, Thin Line, Exaggerated and Triangle. For the Sans Serif class, there are Normal Square, Perpendicular Square, Flared and Rounded ends sub-classes. Based on this classification, the values for all sub-classes that could be assigned to this digit are listed below. The form of each class is shown in Figure 32.

0: Any
1: Not Fit
2: Cove
3: Obtuse Cove
4: Square Cove
5: Obtuse Square Core
6: Square
7: Thin
8: Oval
9: Exaggerated
10: Triangle
11: Normal Sans Serif
12: Obtuse Sans Serif
13: Perpendiculaire Sans Serif
14: Flared Sans Serif
15: Rounded Sans Serif

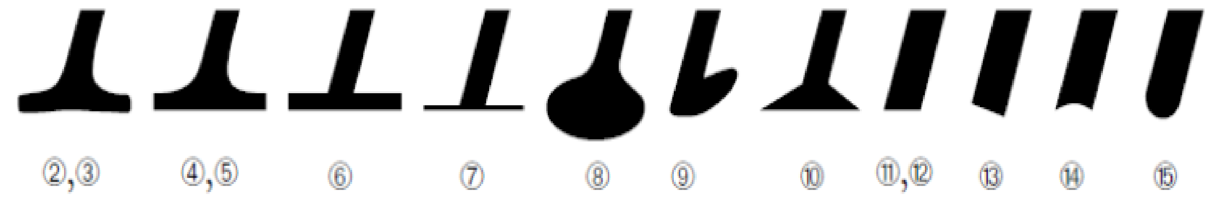
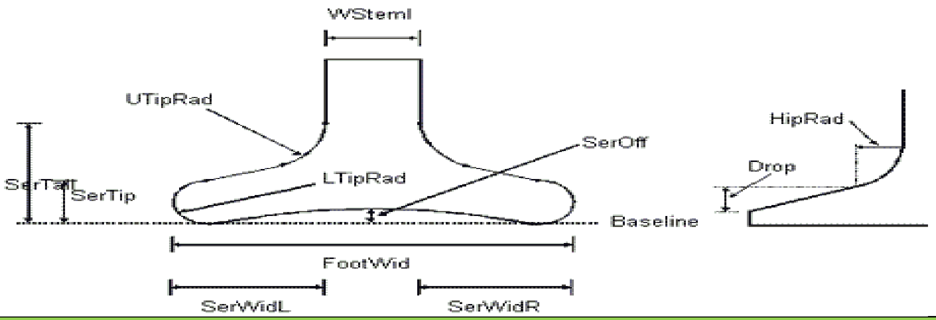


Figure 32: Serif style variations [26]

The distinction between Serif and Sans Serif classes is done based on the value of the *FootRat* variable that is calculated as in the equation (1). This variable indicates the size of a serif at the end of glyph stem as a ratio between *FootWid* and *WStem* measurements. These measurements are calculated on the uppercase ‘I’, as described in Table 7. The classification is obtained using the Serif /Sans Serif classification algorithm.

$$FootRat = FootWid / WStem(I) \quad (1)$$

Table 7: Measurements taken on the letter ‘I’.

 WSteml UTipRad SerTall SerTip LTipRad SerOff Baseline FootWid SerWidL SerWidR HipRad Drop	
Universal serif measurements	
SerTall	Vertical measurement from the point that the serif departs from the vertical stem to the baseline.
SerTip	From the highest extent to the lowest extent of the serif (to the bottom of the contour, not the baseline).
HipRad	Distance from the theoretical left edge of the stem on the lower left serif to either the left edge of the serif or the point where the curve becomes tangent with a line extending to the left edge of the serif.
Drop	Cannot be measured on a serif whose *HipRad* value is equal to the *SerWidL* value. Vertical distance from the top of the serif tip to the point of tangency with the bottom of the cove curve.
Measurements used to calculate overall symmetry of the serif	
SerWidL	Horizontal measurement from the left-most extent of the serif at the base, to the left edge of the vertical stem at the point of serif departure.
SerWidR	From the right side of the vertical stem, at the point of serif departure, to the right-most X-extent of the serif.
FootWid	Horizontal measurement from the left-most extent to the right-most extent of the lower serif.
Measurements that apply to curved, rounded or stylized serifs	
UTipRad	Vertical measurement defines the radius of the largest possible circle drawn within the upper portion of the serif tip while retaining the maximum points of tangency. Square serifs, thin line serifs, and triangle serifs will often have zero UTipRad.
LTipRad	Reflects the lower left hand corner of the serif tip.
SerOff	Vertical distance measured along the theoretical mid-point of the vertical stem from the intersection of that line with the edge of the glyph to the lowest extent of the serif. *SerOff* is zero for glyphs that rest fully on the baseline.
Measurements taken on the Upper I	
WStem(I)	Measured horizontally at the midpoint of the vertical stem. It is the width of the vertical stem of the character, and is taken perpendicular to the stem. In the case of an oblique letter, the horizontal axis is positioned perpendicular to the stem. For the serif designs, this measurement applied to 400 point.

Serif /Sans Serif Style classification algorithm:

IF *FootRat* $\leq$ 1.6 THEN go to Sans Serif Classification

ELSE IF *FootRat* > 1.6 THEN go to Serif Classification

Classifying Sans Serif fonts begins by distinguishing flattened from pointed and rounded serifs. The Flared design is characterized by slightly wider stems, which can be indicated from the value of *FootRat*. Furthermore, the *SerProp* variable is examined to avoid the widening that is an attribute of a concave stem. *SerProp* is the general proportion between serif tall *SerTall* and *CapH* as in the equation (3). *SerTall* is measured on 'I' while *CapH* is measured on 'H', as described in Table 8. The other types of Sans Serif require more analysis. The Rounded Sans Serif class can be determined based on the *RonRat* variable. This variable indicates the ratio between the radius of the lower left corner *StemCor* and the width of the vertical stem *WStem* of uppercase 'I', as in equation (4). *StemCor* and *WStem* measurements are described in Table 7 and Table 9. The Perpendicular Sans Serif class is determined by the value of the *FootPitch* measurement that is measured on uppercase 'A', as described in Table 10. The last two classes of the Sans Serif classification are determined by examining the *SerOb* variable. It is the ratio between two width measurements, *EWid* and *EOut*. These two measurements are measured on 'E' as described in Table 9. *SerOb* is used to determine whether a shape of the end without serif is obtuse or non-obtuse. The classification is obtained by the Sans Serif classification algorithm.

$$TipRat = SerTip / WStem(I) \tag{2}$$

$$SerProp = SerTall / CapH \tag{3}$$

$$RonRat = StemCor / WStem(I) \tag{4}$$

Sans Serif classification algorithm:

```
IF E, A and N are serifed AND $TipRat \geq 0.1$ THEN go to Serif Classification
IF $FootRat > 1.05$ AND $SerProp < 0.35$ THEN **Flared**
ELSE IF $RonRat \geq 0.2$ THEN **Rounded**
ELSE IF $FootPitch > 0$ THEN **Perpendicular Sans Serif**
ELSE IF $SerOb \leq 0.97$ OR $SerOb \geq 1.03$ THEN **Obtuse Sans**
ELSE IF $0.97 < SerOb < 1.03$ THEN **Normal Sans**
```

Classifying Serif fonts begins by separating flat from non-flat serifs. The non-flat design can be distinguished by *TipRat* or by *FlatRat* variables. *FlatRat* is the ratio between *TipSum* and the height of serif's tip *SerTip* as in equation (15). *TipSum* is calculated using equation (10). It is the sum of two serifs measurements, *UTipRad* and *LTipRad*, measured on serif rounded corners.

The measurements are measured on ‘I’, as described in Table 7. *FlatRat* is used to indicate a completely rounded tip from a simple one that just corner-rounded. The classification algorithm is below.

Serif classification algorithm:

```
    IF $FlatRat > 0.8$ OR $TipRat \leq 0.25$ THEN go to Non Flat
    ELSE go to Flat Sided
```

Table 8: Measurements taken on letters "H, M, and J"

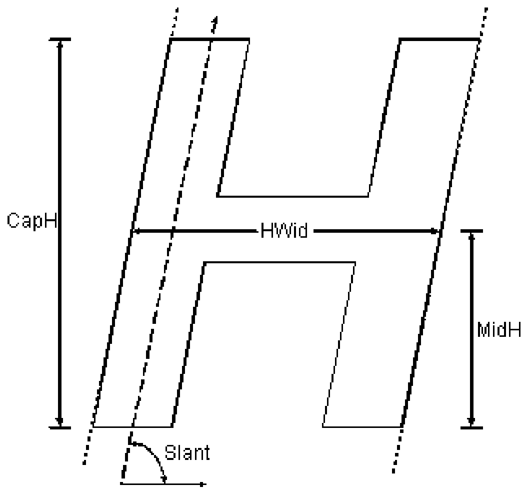
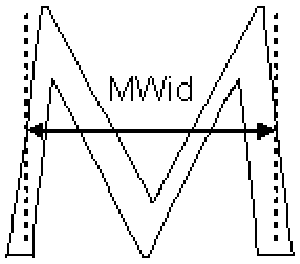
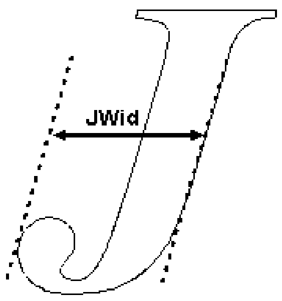
Measurements Taken on the Upper H		
CapH	Vertical measurement from the top-most Y-extent to the bottom-most Y-extent along the theoretical midline of the left vertical stroke.	
Hwid	Horizontal measurement from the left theoretical stem edge of the left stem to the right theoretical stem edge of the right stem. The horizontal location is the average of the horizontal crossbar.	
Slant	Measured from the center of the left vertical stem, with respect to the Baseline.	
MidH	Vertical measurement can be used in place of *MidE* if *MidE* is out of character with the face.	
Measurements taken on the Upper M		
MWid	Horizontal measurement gauged at the exact mid-height of the glyph from the left-most edge of the stroke on the left stem to the right-most edge of the stroke on the right stem.	
Measurements taken on the Upper J		
Jwid	Horizontal measurement from the right theoretical edge of the stem to the left-most extent of the bowl or tail of the glyph.	

Table 9: Measurements taken on letters "E, S, stem of I"

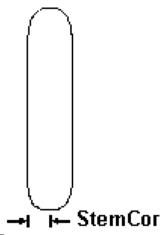
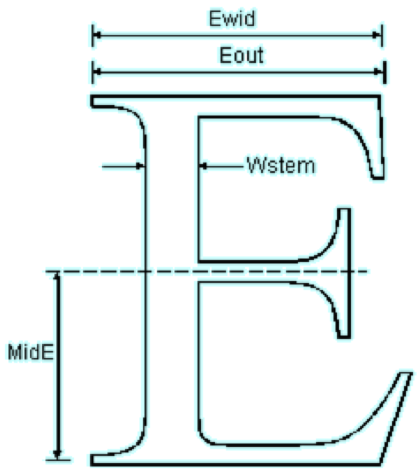
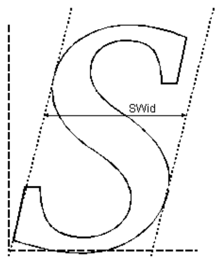
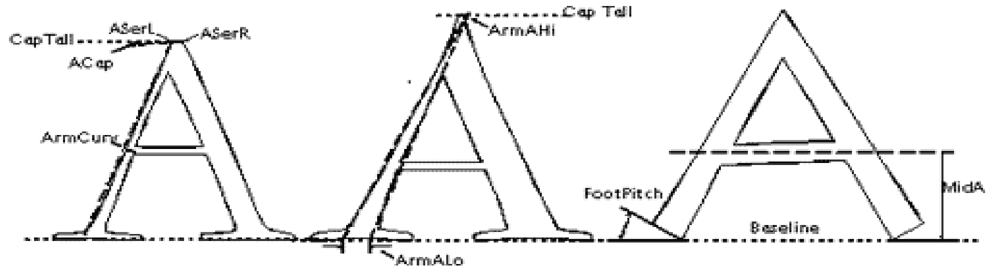
Rare visual traits to identify more unusual san serif designs		
StemCor	Sans serif glyph's stems are rounded instead of sharp, *StemCor* measures the horizontal radius of the lower left corner of the uppercase I. A fully rounded sans serif design would have a *StemCor* value equal to half the stem width.	
Measurements taken on the Upper E		
Ewid	Taken at the cap height line from the left-most extent of the theoretical stem edge, discounting the serif, to the right-most extent of the serif. For fonts whose uppercase E stem is bowed or curved, the x-position of the left edge of the stem is placed average to the right and left extremes of the stem discounting the protrusions of serifs. The right extent of the upper arm of the uppercase E is taken from the closest vertical point on the tip of the arm to the cap height line.	
Eout	taken horizontally from the left-most X-extent of the theoretical backbone (i.e. excluding the upper left serif) to the right-most X-extent of the serif on the upper-most arm of the uppercase E.	
Wstem	Measured horizontally at the x-height of the uppercase E, half way between the upper two arms taken perpendicular to the stem. In the case of an oblique letter, horizontal axis is positioned perpendicular to the stem.	
MidE	Distance of the center of the middle stem of the uppercase E from the baseline. Measurement is also taken from the midpoint on the stem to avoid curvature or stem slanting that may be incorporated into the fonts design. The *MidE* measurement is used to determine Midline.	
Measurements taken on the Upper S		
Swid	Horizontal measurement, taken from the left-most extent of the upper bowl to the right-most extent of the lower bowl. The skewing angle used for this measurement should be the same as that derived in the skew measurement taken on the uppercase H.	

Table 10: Measurements on Upper 'A'.



Measurements taken on the Upper A	
FootPitch	Records the angle at which the stem on a sans serif is terminated.
ArmAHi *ArmALo*	Thickness of the left diagonal stem. The *ArmAHi* is determined by measuring the thickness of the stroke at the capline. The *ArmALo* is measured at the baseline. Both measurements are taken perpendicular to a line that depicts the middle of the stem.
ArmCurv	Taken at the midpoint of an imaginary line which is drawn from the point where the left theoretical edge intersects the baseline and capline. In the case of a concave stem this will result in a negative number, in the case of a bowed stem this will result in a positive number.
MidA	The distance from the center of the horizontal arm to the baseline, taken at the horizontal midpoint of the glyph. Strictly a vertical measurement.
ACap	Measures the amount of flatness at the apex.
CapPitch	Measuring the angle of the theoretical tip. Generally it is between 0 and 90 degrees. Taken on any design where the upper extent of the right diagonal arm creates a surface that is not a rounding point for the apex.
ASerL *ASerR*	Point-count variable in the Arm Style digit. In this case the corners, rounded or square, of the left and right sides of the apex are counted. If the apex forms a clean point, both *ASerL* and *ASerR* equal one.

If a serif is determined as flat, then its symmetry is analysed to determine if it is of an Exaggerated class. To do so *SymRat* is calculated, as in equation (5). It indicates the ratio between the width of the left side *SerWidL* and the right side *SerWidR* of the serif. The description of these measurements is in Table 7. Further classification to distinguish between cove or non-cove flat serifs is done by using the *HipRat* variable. This variable describes the proportion of the upper connection of a serif to the stem. It is calculated using three measurements as in equation (6). These measurements are measured on ‘I’ as described in Table 7. Further analysis is performed for cove serifs by using *DropRat* to distinguish between steep and shallow serifs. *DropRat* indicates the slope of the top edge of a serif. For shallow serifs, the classification is directly based on the several variables. *SerOb* determines the Square Cove and Obtuse Square Cove classes. For steep serifs, *HipRat* is used to determine Triangular class. For non-cove serifs, *StepRat* is used to classify the rounded serifs as heavy or light. Hence, it is used to distinguish Triangle serif class. *StepRat* is the ratio between the serif tip height *SerTip* and the left side serif width *SerWidL* , as indicated in equation (9). Finally, to distinguish between Square and Thin serif classes, the variable *TipRat* is used. The full classification algorithm for flat sided serifs is indicated below.

$$SymRat = SerWidL \,/\, SerWidR \tag{5}$$

$$HipRat = SerWidL\text{-}UTipRad/HipRad \tag{6}$$

$$SerOb = EWid \,/\, EOut \tag{7}$$

$$DropRat = Drop \,/\, (SerWidL\text{-}HipRad) \tag{8}$$

$$StepRat = SerTip \,/\, SerTall \tag{9}$$

Pointed Serif / Flat Sided classification algorithm:

```
    IF 1.2 < $SymRat$ < 0.85 THEN **Exaggerated**
    ELSE IF $HipRat$ > 0.1 THEN go to Coved
    ELSE go to Non Cove

        Cove:
        IF $DropRat$ > 0.2 THEN go to Steep
        ELSE go to Shallow

            Shallow:
            IF $SerOb$ > 0.93 THEN **Square Cove**
            ELSE **Obtuse Square Cove**

            Steep:
            IF $HipRat \leq 0.35$ THEN **Triangle**
```

Non Cove:

IF *StepRat* $\leq$ 0.85 THEN **Triangle**

ELSE IF *TipRat* > 0.35 THEN **Square**

ELSE **Thin**

For non-flat serifs, initially, the exclusion of Exaggerated serifs is determined by *CuspRat* or *Sersize* variables. *CuspRat* indicates the proportion between a serif's cusp to the overall width of the stem, as in equation (11). *SerSize* is the ratio between the width of the left side of a serif *SerWidL* and *CapH,* as in equation (13). *SerWidL* is measured on 'I' as described in Table 7. *SerSize* defines whether serifs are normal or over-sized. If a serif is not Exaggerated, then it will either be rounded or pointed. The *TipRad* variable is used to distinguish between them. *TipRat* is the ratio between the height of a serif tip *SerTip* and the width of the vertical main stem *WStem* of letter 'I'. The measurements are described in Table 7. If a serif is rounded, further classification is required. Thin class is determined by *HipRat*. The *SerRat* variable distinguishes Oval from and Obtuse Cove classes, as indicated in classification algorithm below. *StepRat* is the ratio between the height of the tip and height of the serif. It is used to determine if the serif top edge is parallel to the baseline, which helps to identify a triangular serif design.

$$TipSum = \mathrm{UTi}p\mathrm{Rad} + LTipRad \tag{10}$$

$$CuspRat = SerOff \,/\, WStem(\mathrm{I}) \tag{11}$$

$$SerRat = SerTip \,/\, SerWidL \tag{12}$$

$$SerSize = SerWidL \,/\, CapH \tag{13}$$

$$TRadAv = (UTipRad + LTipRad)/\,2 \tag{14}$$

$$FlatRat = TipSum \,/\, SerTip \tag{15}$$

Rounded Serif / Non Flat classification algorithm:

IF *CuspRat* > 0.15 OR *SerSize* > 0.19 THEN **Exaggerated**

ELSE IF *TipRad* > 0 THEN go to Rounded

ELSE go to Pointed

Pointed:

IF *SerSize* $\leq$ 0.09 THEN **Flared**

ELSE IF *HipRat* < 0.3 THEN **Triangle**

```
ELSE IF $SerOb > 0.93$ THEN **Cove**
ELSE **Obtuse Cove**
```

```
Rounded:
IF $HipRat \leq 0.15$ THEN **Thin**
ELSE IF $SerRat \geq 0.55$ THEN **Oval**
```

Digit 3: Weight

Assigning the value for this digit is determined by calculating the *WeightRat* variable. It represents the ratio between *CapH*, height of the upper case 'H', and *WStem,* which is the thickness of the vertical stroke between the upper and middle arms of letter 'E'. *CapH* and *WStem* are described in Table 8 and Table 9. The values and weight classes attached to them are listed below. A visual representation for each class is shown in Figure 33. A value is assigned by using weight classification algorithm.

0: Any
1: Not Fit
2: Very Light
3: Light
4: Thin
5: Book
6: Medium
7: Demi
8: Bold
9: Heavy
10: Black
11: Extra Black

$$WeightRat = CapH / WStem(E) \tag{16}$$

Weight classification algorithm:

```
IF $WeightRat \geq 35$ THEN **Very Light**
ELSE IF $18 \leq WeightRat < 35$ THEN **Light**
```

ELSE IF 10 $\leq$ *WeightRat* < 18 THEN **Thin**

ELSE IF 7.5 $\leq$ *WeightRat* < 10 THEN **Book**

ELSE IF 5.5 $\leq$ *WeightRat* < 7.5 THEN **Medium**

ELSE IF 4.5 $\leq$ *WeightRat* < 5.5 THEN **Demi**

ELSE IF 3.5 $\leq$ *WeightRat* < 4.5 THEN **Bold**

ELSE IF 2.5 $\leq$ *WeightRat* < 3.5 THEN **Heavy**

ELSE IF 2.0 $\leq$ *WeightRat* < 2.5 THEN **Black**

ELSE IF *WeightRat* < 2.0 THEN **Extra Black**

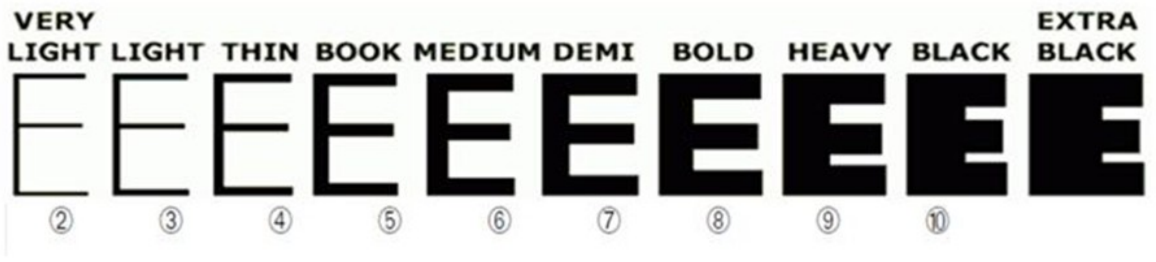


Figure 33: Weight variations [26]

Digit 4: Proportion

This digit describes the ratio of a glyph's general dimensions. In addition, it describes the width of some specific characters, which are often designed in a specific way to distinguish a typeface with a specific appearance. These characters are 'S', 'E', 'J', 'M', 'O' and 'H'. Generally, three proportions classes are defined, which include normal, distorted and monospaced classes. Under the normal class, three variants are defined, namely Old Style, Modern, and Even Width. Under the distorted class, four variants are defined which are Extended, Condensed, Very Extended and Very Condensed. Figure 34 illustrates proportion variations for various variants. As a result, eight different values and classes can be assigned to this digit, as listed below.

0: Any

1: No fit

2: Old Style

3: Modern

4: Even Width

5: Extended

6: Condensed

7: Very Extended

8: Very Condensed

9: Monospaced

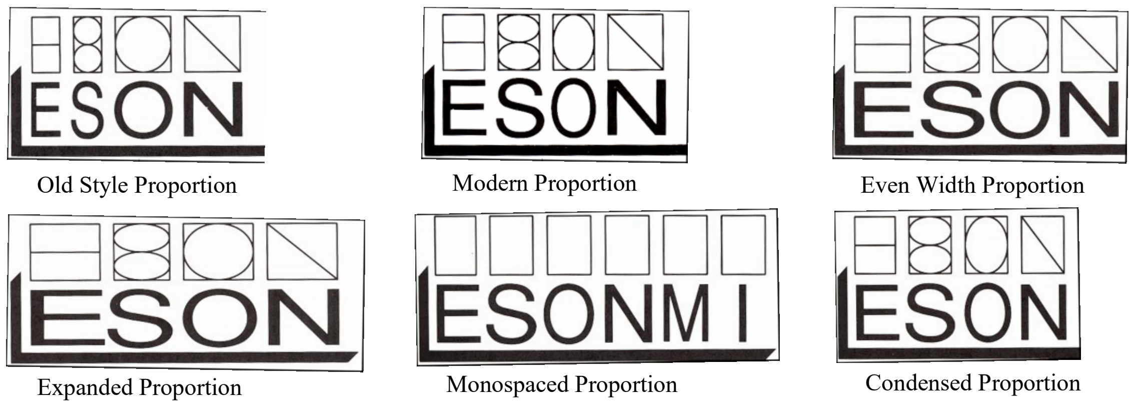


Figure 34: Proportion variations [26]

The value of this proportion digit is determined by using eight measurements to calculate eight variables. The variable *JMRat*, as indicated in the equation (23), represents the ratio between the width of uppercase letters 'J' and 'M' , *JWid* and *MWid* respectively. The *ORat* variable represents the ratio between the height and width of uppercase 'O'. It is calculated using equation (24) based on *OTall* and *OWid* measurements that are described in Table 11. The rest of the calculated variables, *ThinAv*, *WideAv*, *CalcEm*, *ThinRat* and *WidRat*, are used to calculate the general proportion ratio *PropRat*. *ThinAv* represents the average between the width of uppercase letters 'E' and 'S', *Ewid* and *SWid*. The variable *WideAv* represents the average between the width of uppercase letters 'O' and 'H', *OWid* and *HWid* . The ratio between *CalcEm* and *WideAv* results in the value of *ThinRat*, which indicates the width ratio of thin letterforms. The ratio between *CalcEm* and *WideAv* results in producing the value of *WideRat,* which indicates the width ratio of wide letterforms. Finally, the overall proportional ratio *PropRat* is calculated as equation (22) using *ThinRat* and *WidRat* variables. The classification is obtained using *PropRat*, *JMRat* and *ORat* variables, according to the proportion classification algorithm below.

$$ThinAv = (EWid + SWid) / 2 \tag{17}$$

$$WideAv = (OWid + HWid) / 2 \tag{18}$$

$$CalcEm = CapH * 1.5 \tag{19}$$

$$ThinRat = CalcEm / ThinAv \tag{20}$$

$$WideRat = CalcEm / WideAv \tag{21}$$

$$PropRat = WideRat / ThinRat \tag{22}$$

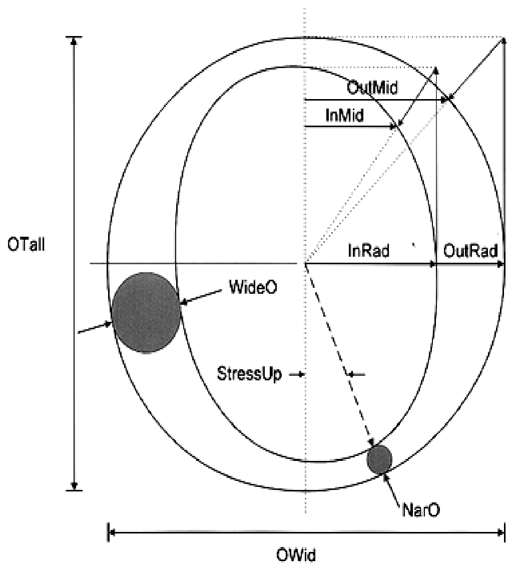
$$JMRat = JWid / MWid \tag{23}$$

$$ORat = OTall / OWid \tag{24}$$

Proportion classification algorithm:

```
IF $JMRat < 0.78$ AND $0.92 \leq ORat < 1.27$ THEN Check PropRat
        Check PropRat:
        IF $PropRat < 0.70$ THEN Old Style
        ELSE IF $0.70 \leq PropRat < 0.83$ THEN Modern
        ELSE IF $0.83 \leq PropRat \leq 0.91$ THEN Even Width
ELSE IF $JMRat < 0.78$ AND $0.90 \leq ORat \leq 0.927$   THEN Extended
ELSE IF $JMRat < 0.78$ AND $1.27 \leq ORat < 2.1$ THEN Condensed
ELSE IF $JMRat < 0.78$ AND $0.85 \leq ORat < 0.90$ THEN Very Extended
ELSE IF $JMRat < 0.78$ AND $2.1 \leq ORat < 2.6$ THEN Very Condensed
ELSE IF $JMRat \geq 0.78$ THEN Monospaced
```

Table 11: Measurements taken on the letter ‘O’.

Measurements taken on the Upper O		
OWid	Horizontal measurement, from the left-most extent of the left side of the stroke to the right-most extent of the right side of the stroke.	 OutMid InMid OTall InRad OutRad WideO StressUp NarO OWid
OTall	A strictly vertical measurement from the outside edge of the stroke at the top-most extent to the outside edge of the stroke at the bottom-most extent of the glyph.	
WideO	Thickest of the glyph stem. Often this will be at the right or left-most extent of the letter-form, measured in a horizontal line.	
NarO	Narrowest point of the glyph stem. Usually the top most extent of the letter-form and is measured vertically.	
OutRad	Horizontal measurement, from the center of the glyph to the right-most extent of the glyph shape.	
OutMid	The upper right corner is used to determine the *OutMid*. It is a horizontal measurement that extends from the middle of the character to the character edge. The vertical placement specified by the intersection of a diagonal bisecting line referred to as the Inter-edge line.	
InRad	Horizontal measurement is taken on the same line used for *OutRad*,is taken from the center of the glyph to the inside edge of the right stroke of the character.	
InMid	Similar to the *OutMid*. in this case the Inter-edge line is drawn from the upper and right-most extents of the inner ellipse	
StressUp	Locating the point at which the outer ellipse and inner ellipse are closest together and measuring that point’s angle to the center of the glyph.	
StressLo	*StressLo* variable is similar to the *StressUp* variable, the measurements are taken on the lowercase o.	
CentDist	The vertical distance from baseline to the point of right-most extent of glyph edge. It is used to determine the place of visual center of fully rounded letterforms off original center.	

Digit 5: Contrast

This digit defines the degree of variation between thick and thin parts of a letterform. The degree of variation is determined based on the *ConRat* variable. It is calculated by equation (25) using *NarO* and *WideO* measurements. Both measurements are described in Table 11. The values and contrast classes attached to them are listed below. A visual representation for each class is shown in Figure 35. A value is assigned by using the weight classification algorithm.

0: Any
1: No fit
2: None
3: Very Low
4: Low
5: Medium Low
6: Medium
7: Medium High
8: High
9: Very High

$$ConRat = NarO \,/\, WideO \tag{25}$$

Contrast classification algorithm:

```
IF $0.80 < ConRat$ THEN **None**
ELSE IF $0.65 < ConRat \leq 0.80$ THEN **Very Low**
ELSE IF $0.48 < ConRat \leq 0.65$ THEN **Low**
ELSE IF $0.30 < ConRat \leq 0.48$ THEN **Medium Low**
ELSE IF $0.20 < ConRat \leq 0.30$ THEN **Medium**
ELSE IF $0.15 < ConRat \leq 0.20$ THEN **Medium High**
ELSE IF $0.08 < ConRat \leq 0.15$ THEN **High**
ELSE IF $ConRat \leq 0.08$ THEN **Very High**
```

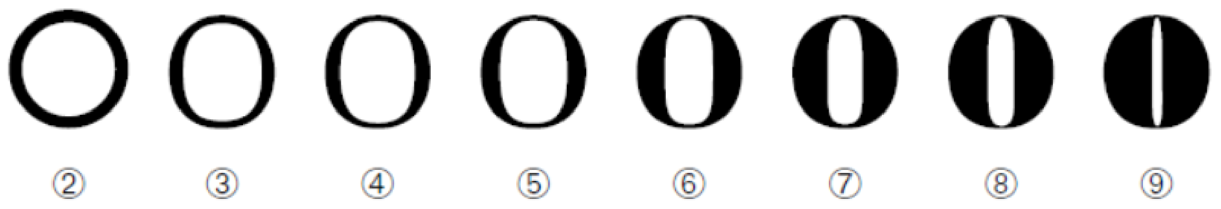


Figure 35: Contrast variation [26]

Digit 6: Stroke Variation

This digit describes the kind of transition in stem weight in rounded characters, such as uppercase ‘O’. The transition is defined through two attributes, the speed and the angle of the transition. The speed of transition can be Gradual, Rapid or Instance. For the angle of the transition, four variations are defined, including Diagonal, Transitional, Vertical and Horizontal variations. The values and stroke variation classes attached to them, which are based on the possible combination of the two attributes, are listed below. A visual representation for each class is shown in Figure 36.

0: Any
1: Not Fit
2: No Variation
3: Gradual/Diagonal
4: Gradual/Transitional
5: Gradual/Vertical
6: Gradual/Horizontal
7: Rapid/Vertical
8: Rapid/Horizontal
9: Instant/Vertical
10: Instant/Horizontal

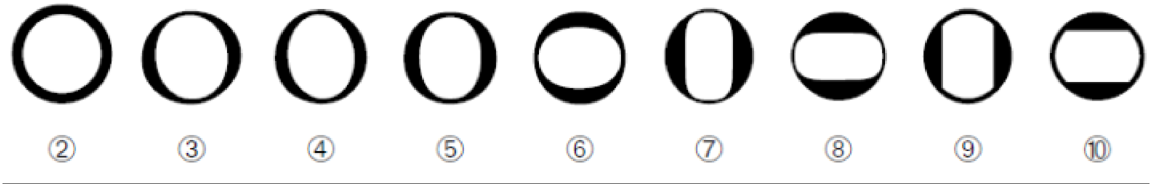


Figure 36: Stroke variations [26]

To determine the class of transition speed, three variables are calculated. These variables are calculated using several measurements on uppercase ‘O’. The variable *OutCurv* represents the curvature of the outer ellipse. It is calculated using equation (26) where *OutMid* and *OutRad* are measurements that are described in Table 11. The other variable is *InCurv.* It represents the curvature of the inner ellipse of the letterform. It is calculated using equation (27), using *InMid* and *InRad* measurements that are described in Table 11. The ratio between *OutCurv* and *InCurv* represents the transition speed as in equation (28). Then, the classification is obtained using the transition speed classification algorithm as follows.

$$OutCurv = OutMid \,/\, OutRad \tag{26}$$

$$InCurv = InMid \,/\, InRad \tag{27}$$

$$Speed = OutCurv \,/\, InCurv \tag{28}$$

Transition Speed classification algorithm:

```
IF $Speed \geq 0.96$ THEN **Gradual**
IF $85 < Speed < 0.96$ THEN **Rapid**
IF $Speed \leq 0.85$ THEN **Instant**
```

The angle of transition is also known as the angle of stress. It can be determined by C*apStress* and *LowerStress* variables. *CapStress* represents the transition angle on uppercase ‘O’. It is calculating using equation (29). *Slant* and *StressUp* measurements are described in Table 8 and Table 11 respectively. *LowerStress* is similar to *CapStress* but it is for lowercase ‘o’. It is calculated using equation (30). *StressLo* is described in Table 11. The classification is obtained using the stress classification algorithm below.

$$CapStress = StressUp - Slant \tag{29}$$

$$LowerStress = StressLo - Slant \tag{30}$$

Stress classification algorithm:

```
IF $82^\circ < StressUp < 98^\circ$ AND $82^\circ < StressLo < 98^\circ$ THEN **Vertical**
IF $98^\circ < StressLo$ OR $StressLo < 82^\circ$ THEN **Transitional**
IF $98^\circ < StressUp \leq 172^\circ$ THEN **Diagonal**
```

IF 172º ≤ *StressUp* ≤ 188º THEN **Horizontal**

Digit 7: Arm Style

This digit classifies two design characteristics, the treatment of the diagonal stem, as in letter ‘A’, and the shape of termination of open rounded letterforms, as in letter ‘C’. The diagonal stem can be Straight or Non-Straight while the termination can have five variants, Horizontal, Wedge, Vertical, Single Serif or Double Serif. This is shown in Figure 37. The values and classes this digit may take, based on the combination of the two characteristics, are listed below.

0: Any
1: Not Fit
2: Straight Arms/Horizontal
3: Straight Arms/Wedge
4: Straight Arms/Vertical
6: Straight Arms/Double Serif
8: Non-Straight/Wedge
9: Non-Straight/Vertical
10: Non-Straight/Single Serif
11: Non-Straight/Double Serif

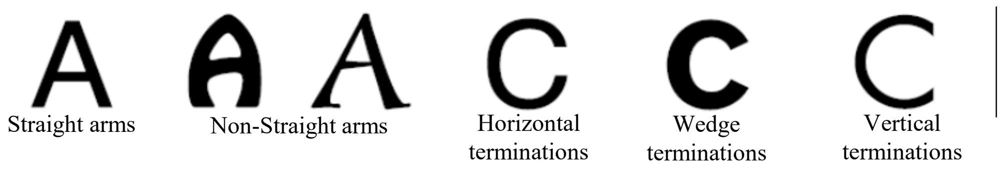


Figure 37: Arm Style and Termination of open curve [26]

Either of the two calculated variables, *TpaerRat* or *CurvRat*, is used to determine whether the treatment of the diagonal stem is Straight or Non-Straight. These variables are calculated using equations (31) and (32). The classification rules are listed below the equations.

$$TaperRat = ArmAHi \,/\, ArmALo \tag{31}$$

$$CurvRat = ArmCurv \,/\, CapH \tag{32}$$

Arm Style classification algorithm:

```
IF $TaperRat < 0.6$ THEN **Non-Straight**
ELSE IF $TaperRat \geq 0.6$ THEN **Straight**

IF $CurvRat \geq 0.02$ THEN **Non-Straight**
ELSE IF $CurvRat < 0.02$ THEN **Straight**
```

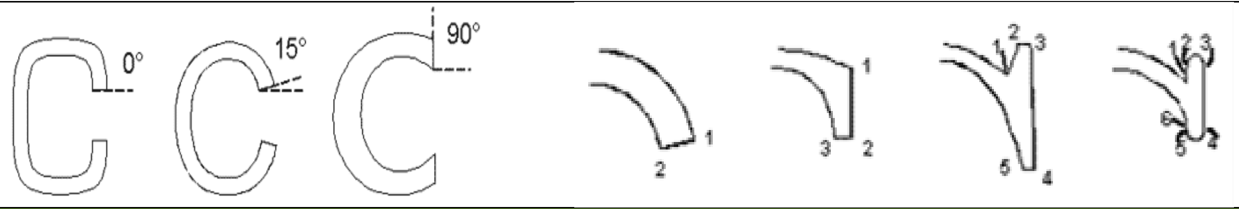
Two different values are used to distinguish between the five variations of the termination of opened round letterforms. The *CutPitch* measurement is measured on uppercase ‘C’, as described in Table 12. It is used with Sans Serif typeface to distinguish between Horizontal, Wedge and Vertical, as in the algorithm below. For typefaces with serif, the *CutRat* variable is used to distinguish between Single or Double Serif variations. *CutRat* is the ratio between the count of corner points at both terminations of uppercase ‘C’. It is calculated by equation (33) using the *CutCountLo* and *CutCountHi* that are described in Table 12.

$$CutRat = CutCountLo \,/\, CutCountHi \tag{33}$$

C termination classification algorithm:

```
IF $CutPitch \leq 7^{\circ}$ THEN **Horizontal**
ELSE IF $7^{\circ} < CutPitch \leq 83^{\circ}$ THEN **Wedge**
ELSE IF $83^{\circ} < CutPitch \leq 112^{\circ}$ THEN **Vertical**
IF $CutRat \leq 0.75$ THEN **Single Serif**
ELSE IF $CutRat > 0.75$ THEN **Double Serif**
```

Table 12: Measurements taken on the letter 'C'

 0° 15° 90° 1 2 1 2 3 1 2 3 4 5 1 2 3 6 5 4	
Measurements taken on the Upper C	
CutCountHi *CutCountLo*	Depict the number of corners at both terminations. Only applied to serif letter forms and indicates the amount of serif detailing at the ends of the stroke.
CutPitch	Angle of termination. The left-most of the two points is used as the fulcrum of the angle. In the case of a highly rounded corner style at the termination of the stroke, the theoretical edge of the stroke must be determined and its angle recorded. The treatment of the lower termination of the stroke is not factored into this attribute.

Digit 8: Letter Form

This digit describes the skewing and overall roundness of character forms. There are two variants of the character skewing, which are Normal and Oblique. Seven variations of the letterform roundness are defined, which are Contact, Weighted, Boxed, Flattened, Rounded, Off Center and Square. A visual representation for each class of roundness is illustrated in Figure 38. Based on the combination of skewing and roundness characteristic variations, the values that this digit may take include the following.

0: Any
1: No Fit
2: Normal/Contact
3: Normal/Weighted
4: Normal/Boxed
5: Normal/Flattened
6: Normal/Rounded
7: Normal/Off Center
8: Normal/Square
9: Oblique/Contact
10: Oblique/Weighted
11: Oblique/Boxed
12: Oblique/Flattened
13: Oblique/Rounded

14: Oblique/Off Center
15: Oblique/Square

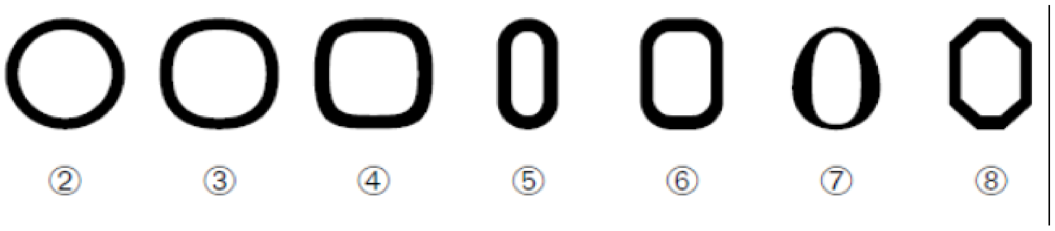


Figure 38: Letterform variation

Character skewing is determined based on the *Slant* measurement that is taken on letter 'H' as indicated in Table 8. The classification is obtained based on the algorithm below

Skew classification algorithm:

```
    IF $Slant < 85$ THEN **Oblique**
    ELSE **Normal**
```

The roundness of a letterform is determined based on *OutCurv* and *CentProp* variables. *OutCurv* represents the curvature of the outer ellipse of the uppercase O glyph. It is calculated by equation (34) using *OutMid* and *OutRad* measurements, which are described in Table 11. *CentProp* represents the proportion between the vertical distance of the center of the glyph *CentDist* and the tall of the glyph *OTall* as in equation (35). The measurements *CentDist* and *OTall* are taken on the uppercase of 'O', which is also described in Table 11. The classification is obtained using *CentProp* and *OutCurv* variables as follow:

$$OutCurv = OutMid / OutRad \tag{34}$$

$$CentProp = CentDist / OTall \tag{35}$$

Roundness classification algorithm:

```
    IF $CentProp > 0.56$ OR $CentProp < 0.44$ THEN **Off Center**
    ELSE check $OutCurv$
        IF $OutCurv > 0.95$ THEN Square
        ELSE IF $0.95 \geq OutCurv > 0.83$ THEN **Rounded**
        ELSE IF $0.83 \geq OutCurv > 0.80$ THEN **Flattened**
        ELSE IF $0.80 \geq OutCurv > 0.77$ THEN **Boxed**
        ELSE IF $0.77 \geq OutCurv \geq 0.74$ THEN **Weighted**
        ELSE IF $OutCurv < 0.74$ THEN **Contact**
```

Digit 9: Midline

This digit represents two independent characteristics of a font design. One is the placement of the midline across uppercase letters, ‘A’ and ‘E’, while the other is the shape of the diagonal stem apexes in letter ‘A’. For this model, four midline placement variations are defined, which are Standard, Constant, High and Low. For the apex shape, three variants are defined, which are Trimmed, Pointed, and Serifed. Midline variations and apex shapes are illustrated in Figure 39. The values this digit may take, based on the combination of the two characteristics, are listed below.

0: Any
1: No Fit
2: Standard/ Trimmed
3: Standard/ Pointed
4: Standard/ Serifed
5: High/ Trimmed
6: High/ Pointed
7: High/ Serifed
8: Constant/ Trimmed
9: Constant/ Pointed
10: Constant/ Serifed
11: Low/ Trimmed
12: Low/ Pointed
13: Low/ Serifed

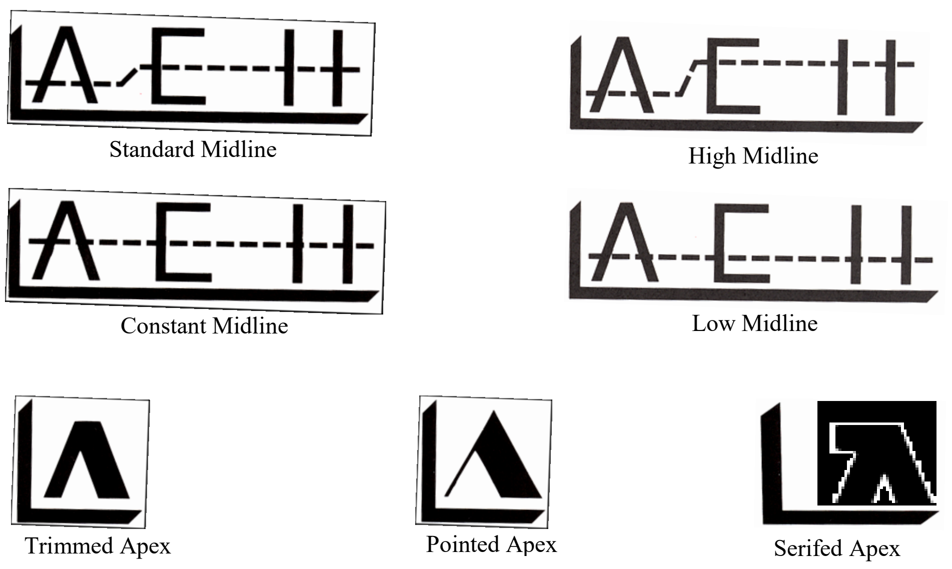


Figure 39: Midline variations and shapes of Apex [22]

To determine the value of this digit, five variables are calculated. Three of them are used to determine the placement of the midline, *EArm*, *AArm* and *ArmDif*. *EArm* is the ratio of the midline height in ‘E’ to the cap height of ‘H’, as in equation (36). *AArm* is the ratio of the midline height in ‘A’ to the cap height. *EArm* and *AArm* are used to calculate the difference ratio *ArmDif,* as in equation (38). The classification is then obtained using the midline position classification algorithm below.

$$EArm = MidE \,/\, CapH \tag{36}$$

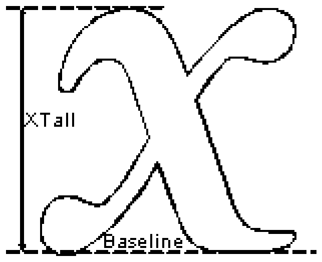
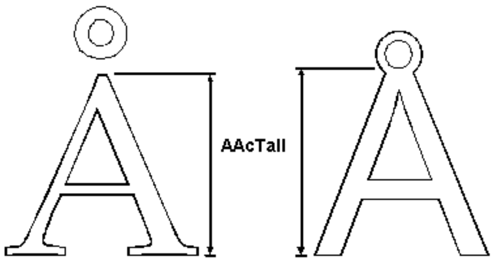
$$AArm = MidA \,/\, CapH \tag{37}$$

$$ArmDif = EArm - Aarm \tag{38}$$

$$TrimRat = ACap \,/\, WStem(E) \tag{39}$$

$$ASer = (ASerL + ASerR) \,/\, 2 \tag{40}$$

Table 13: Measurements taken on lowercase ‘x’ and ‘ARing’.

Measurements taken on the Lower x		
Xtall	Vertical measurement from the baseline vertically to the upper extent of the upper left stem.	
Measurements taken on the ARing character, “Å”		
AAcTall	Vertical measurement from the baseline to the upper extent of the uppercase A portion of the ARing glyph. In the cases where the ring is joined, the measurement is taken from the baseline to the lower extent of the inner white-space of the ring.	

Midline position classification algorithm:

```
IF $ArmDif < 0.08$ THEN **Constant**
ELSE IF $ArmDif \geq 0.08$ THEN
    Check EArm:
    IF $EArm > 0.58$ THEN **High**
    ELSE IF $EArm < 0.45$ THEN **Low**
    ELSE **Standard**
```

The apex shape is determined by using *ASer* and *TrimRat* variables. *ASer*, as in equation (40), is the average of the points counted on the left and right sides of the apex, which are represented as *ASerL* and *ASerR* respectively. *TrimRat* is ratio of *ACap* and *WStem* of letter ‘E’, as defined in Table 10 and Table 9. The classification is then obtained by using the apex shape classification algorithm.

Apex shape classification algorithm:

```
IF $ASer > 2$ THEN Serifed
ELSE IF $ASer \leq 2$ THEN
    Check TrimRat:
    IF $TrimRat \geq 0.6$ THEN Trimmed
    ELSE IF $TrimRat < 0.6$ THEN Pointed
```

Digit 10: X-height

This digit represents two independent characteristics of a font design. One is the x-height while the other is the treatment of uppercase glyphs with diacritical marks. For this model, three variations of the x-height are defined, which are Small, Standard and Large. These are shown in Figure 40. Regarding the uppercase treatment with diacritics, the definers of PANOSE believed that letters with accents or diacritic marks can be shorter than the regular uppercase letters, as shown in the figure that describes *AAcTall* in Table 13. Therefore, a design can have two variations. The first variation is Constant, when the height is the same for all uppercase letters, with or without diacritics. The second variation is Ducking, when letters with diacritics are shortened. The variations of x-height are illustrated in Figure 40. The values this digit may take based on the two characteristics are listed below.

0: Any

1: No Fit

2: Constant/Small

3: Constant/Standard

4: Constant/Large

5: Ducking/Small

6: Ducking/Standard

7: Ducking/Large

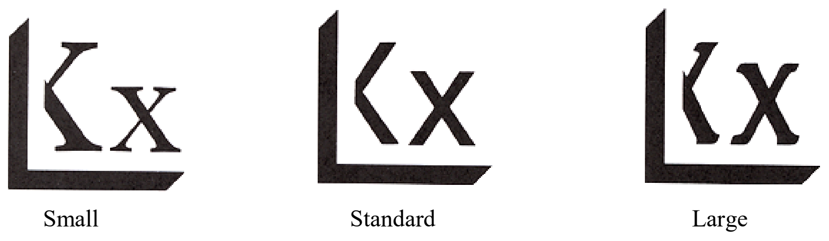


Figure 40: X-height variations [22]

To determine the value of this digit, two variables are calculated, which are the x-height ratio *XRat* and the ducking ratio *DuckRat*. The variable *XRat* is calculated using equation (41), using two measurements, *XTall* and *CapH*. *XTall* is defined in Table 13. The x-height classification is then obtained as follows:

$$XRat = XTall \,/\, CapH \tag{41}$$

$$DuckRat = AAcTall \,/\, CapH \tag{42}$$

X-height classification algorithm:

```
IF $XRat \leq 0.50$ THEN **Small**
IF $0.50 < XRat \leq 0.66$ THEN **Standard**
IF $0.66 < XRat$ THEN **Large**
```

The ducking characteristic is determined based on *DuckRat*. It is calculated using equation (42). The *AAcTall* measurement used in equation (42) is defined in Table 13. The classification is then obtained with the algorithm below.

Uppercase treatment with diacritical marks classification algorithm:

```
IF $DuckRat \leq 0.93$ THEN **Ducking**
ELSE **Constant**
```

3.2.2.2. Latin Handwritten

Digit 1: Family

Fonts of this family have a handwritten script appearance and are not related to any book typefaces. The value 3 is assigned to this digit.

Digit 2: Tool Kind

The value of this digit indicates the kind of tool used to produce letterforms. Values that can be assigned for this digit are listed below. An example of each tool kind is shown in Figure 41.

0: Any
1: No Fit
2: Flat Nib
3: Pressure Point
4: Engraved
5: Ball (Round Cap)
6: Brush
7: Rough
8: Felt Pen/Brush Tip
9: Wild Brush - Drips a lot

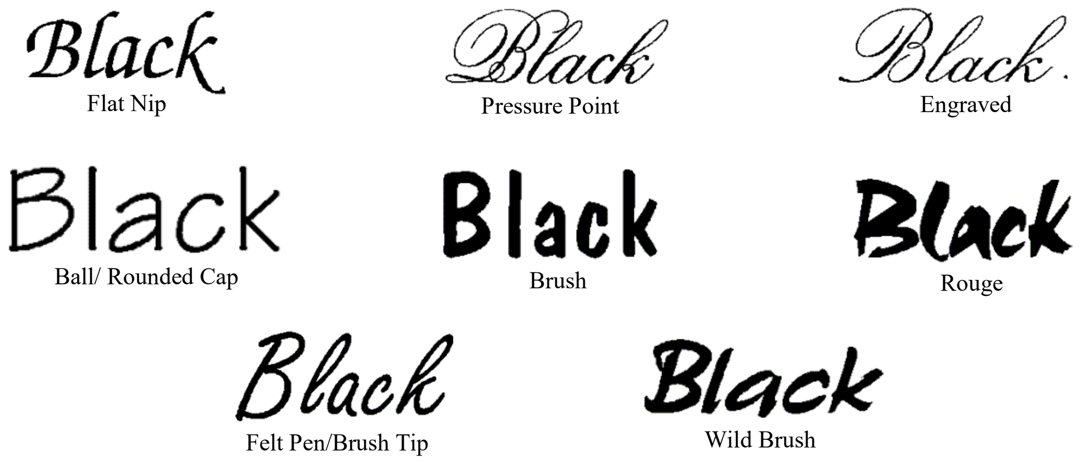


Figure 41: Tool kind classes

Digit 3: Weight

This digit is similar to the digit 3 as defined for Latin text family.

Digit 4: Spacing

The value of this digit indicates whether the font is monotone or proportionally spaced. The values it may take are as follows:

0: Any
1: No Fit
2: Proportional
3: Monospaced

Digit 5: Aspect Ratio

The value of this digit indicates the aspect ratio between the width and height of uppercase letter 'O'. Assigning the value for this digit is based on the *ORat* variable, which is calculated by equation (24). The values this digit may take and the classification algorithm are listed below.

0: Any
1: No Fit
2: Very Condensed
3: Condensed
4: Normal
5: Expanded
6: Very Expanded

Aspect ratio Classification:

```
IF $ORat \geq 2.1$ THEN **Very Condensed**
ELSE IF $1.27 \leq ORat < 2.1$ THEN **Condensed**
ELSE IF $0.92 \leq ORat < 1.27$ THEN **Normal**
ELSE IF $0.90 \leq ORat < 0.92$ THEN **Expanded**
ELSE IF $ORat < 0.90$ THEN **Very Expanded**
```

Digit 6: Contrast

This digit is similar to the digit 5 that is defined for Latin text family.

Digit 7: Topology

This digit indicates two general characteristics of a face appearance. First, it classifies a font into three styles which are Roman, Cursive and Blackletter. In addition, the value of the digit classifies a font based on letterform connection into three classes, which are Disconnected, Trailing and Connected. Roman means that the letterforms are similar to the text family but slanted to look like handwritten letterforms. Cursive means that the letterforms appear as flowing handwritten ones. Blackletter class means that a font looks very black and condensed. A major design modification is done to its letterforms in such a way give them an angry or aggressive feel. A font is classified as Disconnected if its letterforms are distinct and there is no connection between them. If the trailing serifs of a face letterform are extended in such a way that the letters overlap, then a face is classified as Trailing. In a case where a face's letterforms are explicitly connected, a face is classified as Connected. A sample of each topology class is shown in Figure 42. The set of values and meanings that can be assigned to this digit are listed below.

0: Any
1: No Fit
2: Roman Disconnected
3: Roman Trailing
4: Roman Connected
5: Cursive Disconnected
6: Cursive Trailing
7: Cursive Connected
8: Blackletter Disconnected
9: Blackletter Trailing
10: Blackletter Connected

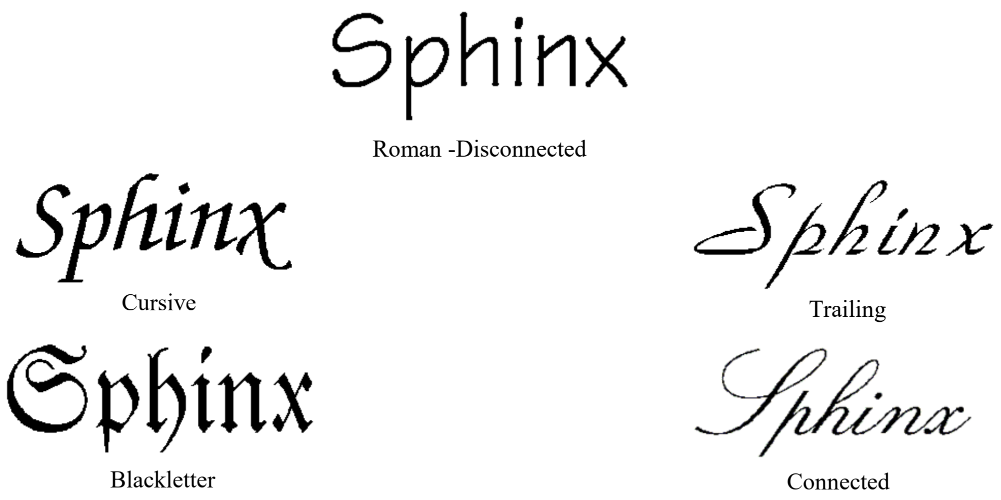


Figure 42: Example of each topology class

Digit 8: Form

This digit describes the general look of a face through two attributes, the slope of the vertical stems and the wrap of the tail of connecting strokes. The slope classes are Upright, Oblique or Exaggerated. Four degrees of the tail wrapping are defined, which are No wrapping, Some wrapping, More wrapping and Extreme wrapping. An example of each tail wrapping class is shown in Figure 43. Values this digit may take based on the combination of the two attributes are listed below.

0: Any
1: No Fit
2: Upright \ No wrapping
3: Upright \ Some wrapping
4: Upright \ More wrapping
5: Upright \ Extreme wrapping
6: Oblique \ No wrapping
7: Oblique \ Some wrapping
8: Oblique \ More wrapping
9: Oblique \ Extreme wrapping
10: Exaggerated \ No wrapping
11: Exaggerated \ Some wrapping

12: Exaggerated \ More wrapping

13: Exaggerated \ Extreme wrapping

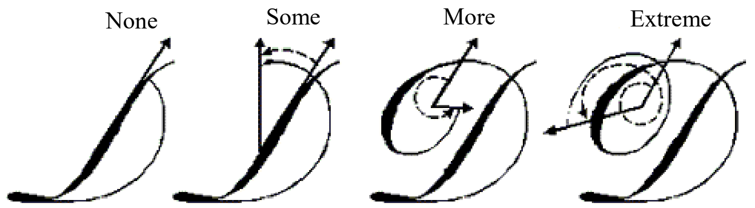


Figure 43: Tail wrapping classes

The class of the slope is determined based on the *Slant* that measured on uppercase 'H', as described in Table 8. The classification is obtained by the algorithm below.

Slope classification algorithm:

```
IF 0º ≤ Slant < 5º THEN Upright
ELSE IF 5º ≤Slant < 15º THEN Oblique
ELSE IF 15º ≤Slant THEN Exaggerated
```

The class of the tail wrapping is determined based on uppercase 'D'. The classification is obtained as follow:

Wrapping classification algorithm:

```
IF bowl stroke meets vertical stem THEN No Wrapping
IF bowl stroke passes vertical stem AND bowl stroke curve less than 90º
    THEN Some Wrapping
IF bowl stroke passes vertical stem AND bowl stroke curves less than 360º
    THEN More Wrapping
IF bowl stroke passes vertical stem AND bowl stroke curves more than 360º
    THEN Extreme Wrapping
```

Digit 9: Finials

This digit describes the general look of a face through two attributes, the treatment of the end of uppercase 'A' and the treatment of the ascender of lowercase characters such as 'l'. The treatment of the end of 'A' can be classified into four sub-classes. When the end has extra treatment, more than what a nip naturally does, it is classified as None. If the end treatment appears chopped at an angle, it is classified as Sharp. When the end width gradually becomes less than the width of nip, the end is classified as Tapered. When the end is bulbous, it is classified as Rounded. The treatment of the ascender has three variants, which are No loop, Closed loop and Open loop. All four classes of the end treatment of uppercase ‘A’ and the three classes of the ascender treatment in letter ‘l’ are shown in Figure 44. The values this digit may take based on the combination of the two attributes are listed below.

0: Any
1: No Fit
2: None / No loops
3: None / Closed loops
4: None / Open loops
5: Sharp / No loops
6: Sharp / Closed loops
7: Sharp / Open loops
8: Tapered / No loops
9: Tapered / Closed loops
10: Tapered / Open loops
11: Round / No loops
12: Round / Closed loops
13: Round / Open loops

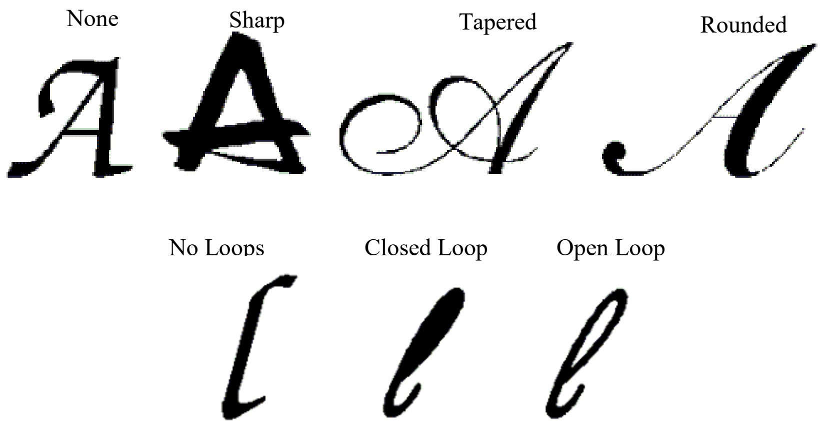


Figure 44: Classes of end of 'A' treatment (Top), Classes of ascender treatment (Below).

Digit 10: X-Ascent

The value of this digit represents the ratio aspect of lowercase ‘x’. Assigning a class to this digit is determined based on the value of the calculated variable *XRat,* as in the equation (41). The values this digit may take and the classification algorithm are listed below.

0: Any
1: No Fit
2: Very Low
3: Low
4: Medium
5: High
6: Very High

X-Ascent classification algorithm:

```
    IF $XRat \leq 0.40$ THEN Very Low
    ELSE IF $0.4 < XRat \leq 0.50$ THEN Low
    ELSE IF $0.50 < XRat \leq 0.66$ THEN Medium
    ELSE IF $0.66 < XRat \leq 0.75$ THEN High
    ELSE IF $0.75 < XRat$ THEN Very High
```

3.2.2.3. Latin Decorative:

Digit 1: Family Kind

Fonts of this family are usually designed for use in displays or are used for special purposes. Small caps fonts are also included in this family. The number 4 is the value assigned to this digit.

Digit 2: Class

The value of this digit indicates the general look of a face. Eleven classes are defined for this model. If a face appearance is close to the standard text form, the Derivative class is assigned. If a face has an unusual letterform but still uses basic stems, the Non-Standard Topology is assigned. If a face has usual letterforms but unusual treatments in some of their parts, the Non-Standard Elements class is assigned. If a face has usual letterforms but unusual proportions, the Non-Standard Aspect class is assigned. If a face is highly ornamented or has majuscule characters, the Initials class is assigned. If a face's letterforms are formed from single pictures then the Cartoon class is assigned. In the case pictures are used to form the stems that make up the letterforms, and then the Picture Stem class is assigned. If a face has additional details and flourishes, then the Ornamented class is assigned. When face characters are displayed on patterned or solid backgrounds, they are classified as Text and Background. If a face characters are constructed from repeating non standard elements, they are classified as Collage. When there is no repetition, a face is classified as Montage. Examples of each class are shown in Figure 45. All classes and the values attached to them are listed below.

0: Any
1: No Fit
2: Derivative
3: Non-Standard Topology
4: Non-Standard Elements
5: Non-Standard Aspect
6: Initials
7: Cartoon
8: Picture Stem
9: Ornamented
10: Text and Background
11: Collage
12: Montage

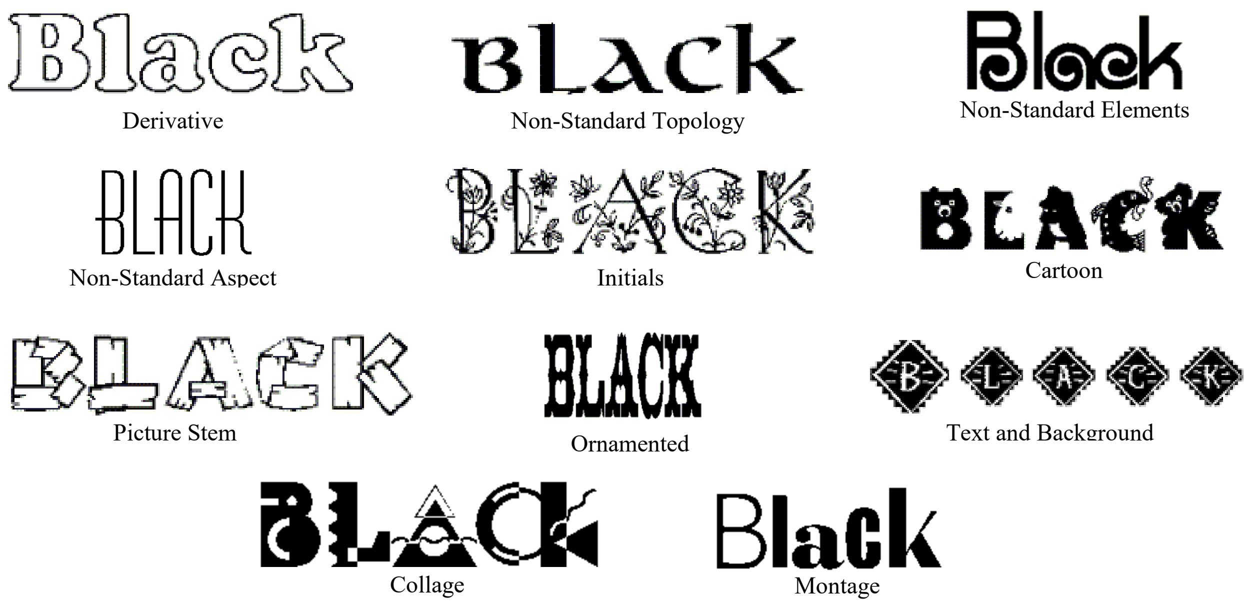


Figure 45: An example for each class of Latin Decorative fonts defined by a value of digit 2.

Digit 3: Weight

This digit is similar to digit 3 for Latin Text and Latin Handwritten families.

Digit 4: Aspect

This digit is similar to digit 5 for the Latin Handwritten family.

Digit 5: Contrast

This digit is similar to digit 5 for the Latin Text family. Four additional values are added for decorative family fonts, which are listed below.

10: Horizontal Low
11: Horizontal Medium
12: Horizontal High
13: Broken.

Digit 6: Serif Variant

The serif classes defined for this digit are similar to classes defined for digit 2 within the Latin text family. One additional class is added, called Script with number 16 as a value attached to it. This additional class covers the serifs that do not fit in to any other classes.

Digit 7: Treatment

Fonts are classified based on fills and outlines used to construct the letterforms. This digit focuses on the fill treatments while the outline treatment is covered by digit 8. If the letterforms use simple line with solid fill as regular text, the None/Standard Solid Fill class is assigned. When the letterforms only have outlines with white or no fill, the White/No Fill class is assigned. If the letterforms are filled with pattern, then the Pattern Fill class can be assigned. When different letterforms are filled with different patterns, it becomes classified under the Complex Fill class. If a font is filled by a pattern is formed from other contexts, the Shaped Fill class is assigned. Last defined class is the Drawn/Distorted class, which is assigned to this digit if the fill of each letter is unique and individual. Each fill class is illustrated in Figure 46. All classes and the values attached to them are listed below.

0: Any
1: No Fit
2: None/Standard Solid Fill
3: White/No Fill
4: Pattern Fill
5: Complex Fill
6: Shaped Fill
7: Drawn/Distorted

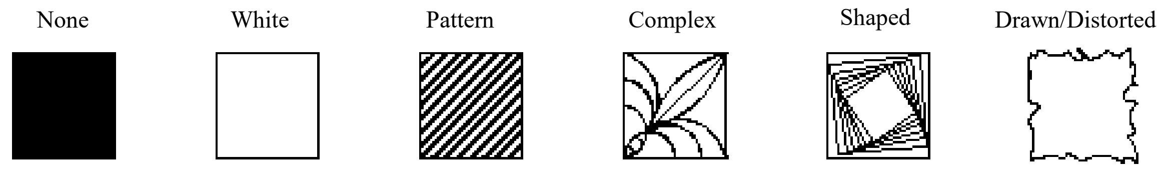


Figure 46: Fill classes

Digit 8: Lining

This digit classifies fonts based on the letterforms' outline treatments. If outlines are just simple lines, then the None class is assigned. When outlines are shaded on the inside, the Inline class is assigned. If outlines are shaded from the outside, the font is classified under the Outline class. If outlines are multiplied, the Engraved/Multiple Lines class is assigned. If a copy of the outline is offset to one side for each letterform, then the face is classified as Shadow. When a letterform has a 3-D look created by a copy of its outline being attached to the letterform itself, the font is classified as Relief. When letters look as though they are floating above a background, the Backdrop class is assigned. Each outline class is illustrated in Figure 47. All classes and the values attached to them are listed below.

0: Any
1: No Fit
2: None
3: Inline
4: Outline
5: Engraved/Multiple Lines
6: Shadow
7: Relief
8: Backdrop

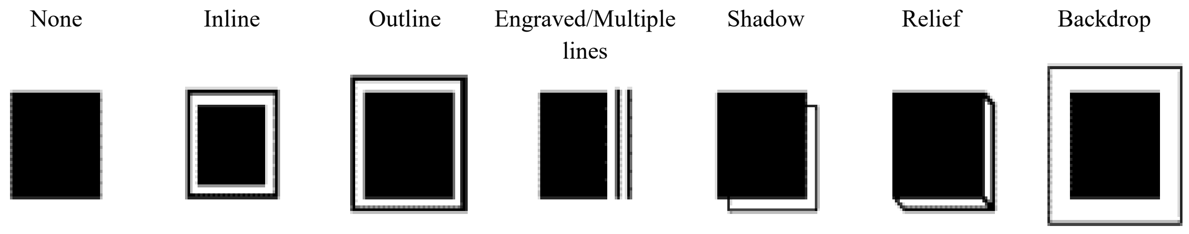


Figure 47: Outline classes

Digit 9: Topology

This digit classifies fonts based on unusual characteristics in a font topology. If a face is normal looking, it is classified as Standard. If a face's characters have exaggerated square and angular parts, then it is classified as Square. If strokes that build up the letterforms are broken into several pieces, the Multi Segment class is assigned. If a face has an Art Deco style where midlines are very low or very high, then it is classified as Deco. If a face's different letterforms have different weights, then it is classified under the Uneven Weighting class. If arms on different letters are different, the face is classified as Diverse Arm. If a face has dissimilar letterforms that are usually similar, such as d, p, q, it is classified as Diverse Forms. When a face has exaggerated stems, it is classified as Lombardic. If a face has variant caps or small caps instead of lowercase letters in the characters map, the Uppercase in Lowercase class is assigned. When a face's characters are missing some parts in their letterforms, such as ascenders, the face is classified as Implied Topology. When the most angular characters in a face, such as 'E' and 'A', have rounded letterforms, the face is classified as Horseshoe E and A. If a face's letterforms follow cursive models, then it is classified as Cursive. If a face's letterforms design follows the German fraktur model then the face is classified as Blackletter. Finally, if a face has multiple variant swashes, it is classified as Swash Variance. An example of each class is shown in Figure 48. All classes and the values attached to them are listed below.

0: Any
1: No Fit
2: Standard
3: Square
4: Multi Segment
5: Deco
6: Uneven Weighting
7: Diverse Arm
8: Diverse Forms
9: Lombardic
10: Uppercase in Lowercase
11: Implied Topology

12: Horseshoe E and A

13: Cursive

14: Blackletter

15: Swash Variance

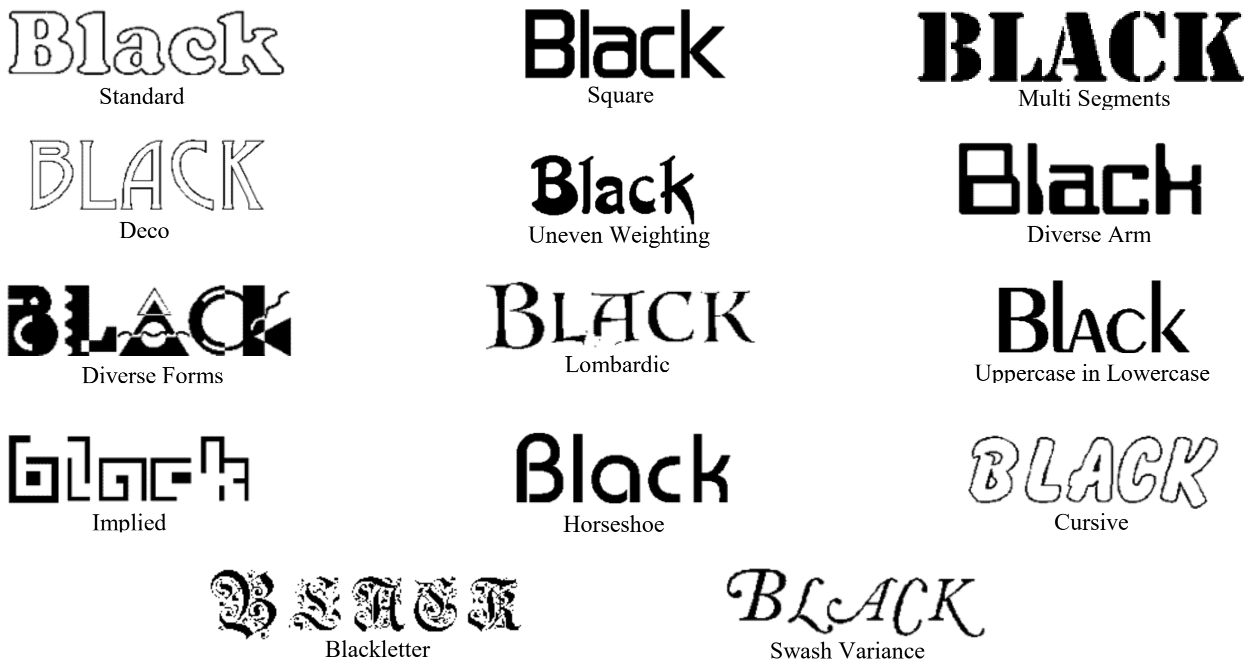


Figure 48: Examples for Latin Decorative fonts of each defined topology class

Digit 10: Range of Characters

Four classes are defined for this digit. If a font contains a full font range, it is classified as an Extended collection. If a font has only contains alphanumeric characters, it is classified as Literals. When a font only has uppercase characters, the No-Lowercase class is assigned. If a font only has small cap characters and no upper or lower cases, it is classified as Small Caps. All classes and the values attached to them are listed below.

0: Any

1: No Fit

2: Extended

3: Literals

4: No-Lowercase

5: Small Caps

3.2.2.4. Latin Pictorial:

Digit 1: Family Kind

This family kind is for fonts that contain specialized symbols or dingbats and do not contain readable characters. It is assigned a value of 5.

Digit 2: Kind

The value of this digit indicates the type of characters in fonts. Eleven types are defined for this model. The most common class is the Montages class. This class contains sets of symbols, such as arrow, math operators, math equations, zodiac, etc., where no single type is dominant. If a font contains a set of symbols that are constructed from pictures, the Picture class is assigned. If a font contains a set of abstract shapes, such as square, dot, star, etc., the Shape class is assigned. If a font contains specialized scientific characters, the Scientific class is assigned. When a font contains notes and specialized music symbols, the Music class is assigned. The Expert class value indicates that a font contains an extension set that includes ligatures or small caps glyphs. If a font contains set of symbols filled with lines or texture, the Pattern class is assigned. If a font contains a set of fancy borders or decorative symbols, the Boarder class is assigned. A font that contains a set of symbolic shapes will be classified under Icon class. The Logo class indicates that a font contains a set of copyrighted and registered work symbols. If a set contains certain field specific symbols, the Industry Specific class is assigned to this digit. All classes and the values attached to them are listed below.

0: Any

1: No Fit

2: Montages

3: Picture

4: Shape

5: Scientific

6: Music

7: Expert

8: Pattern

9: Boarder
10: Icon
11: Logo
12: Industry

Digit 3: Weight

This digit is required by the PANOSE engine but it is not applicable for this family type. Therefore this digit takes only one value which is 1: Not Fit.

Digit 4: Spacing

This digit is similar to the digit 4 defined for Latin Handwritten family.

Digit 5: Aspect Ratio & Contrast

This digit is also not applicable to a pictorial family type. It is only assigned 1: Not Fit.

Digit 6-10: Aspect Ratio & Character

Each digit, from 6 to 10, indicates the aspect ratio of special a selected character's height to its black width, as in the equation (43). Digit 6 is for character 95 and digit 7 is for character 119 in the lower ASCII range. Digit 8 is for character 157of the standard ASCII range. The last two digits, 9 and 10, are for character 163 and character 211 respectively from the upper ASCII range. The values and the meaning that can be assigned to each digit are listed below. The classification is obtained using the algorithm below.

0: Any
1: No Fit
2: No Width
3: Exceptionally Wide
4: Super Wide
5: Very Wide
6: Wide
7: Normal

8: Narrow

9: Very Narrow

$$Ratio = Height/BlackWidth \tag{43}$$

Aspect ratio classification algorithm:

```
IF $0 < Ratio \leq 0.40$ THEN Exceptionally Wide
ELSE IF $0.40 < Ratio \leq 0.60$ THEN Super Wide
ELSE IF $0.60 < Ratio \leq 0.80$ THEN Very Wide
ELSE IF $0.80 < Ratio \leq 0.93$ THEN Wide
ELSE IF $0.93 < Ratio \leq 1.10$ THEN Normal
ELSE IF $1.10 < Ratio \leq 1.40$ THEN Narrow
ELSE IF $1.40 > Ratio$ THEN Very Narrow
```

Chapter 4 : Arabic Typeface Classification

IBM and PANOSE-1 are font classification systems, which are defined for Latin. They encode font's design characteristics into numbers composed of digits. The structure, the values and the meaning of each digit are reviewed in detail in the previous chapter. IBM system has some digits reserved for future use by IBM if more or different characteristics are needed. Unlike PANOSE-1 system, where for the first digit that indicate the font family, values from 2-5 are assigned to define four font families for Latin as listed in Table 6. The rest of values can be used to define any other font families for any scripts [26]. Since there is no such previous work has been done to classify Arabic fonts, the focus of this chapter is about a proposed extension for PANOSE-1 to cover Arabic fonts. The proposed model was named PANOSE-A.

4.1. PANOSE-A for Arabic

Each of the six common known styles of Arabic, mentioned in section "Calligraphy and Typography" of Chapter 1, has special visual characteristics that make it suitable for specific uses. Kufi style is an inscriptional style. It has evolved into a more geometrical and decorative direction. It is used for display and decorative purposes unlike Thuluth style, which is used as a text face, mostly for short texts and titles. Rukaa style has a heavier and compact script. It is mainly used in informal texts. Nastaliq style is mostly used as a text face in eastern Muslim countries such as Iran and Pakistan. Naskh style has clear and simple letterforms, which make it the most commonly used style and it is an industry standard for long texts [3]. For PANOSE-A, the digits are defined to describe Naskh style fonts for text. The proposed model includes eight digits after the family type digit, which is the first digit in PANOSE-1

PANOSE-A digits are defined based on basic guidelines and measurements used in any Arabic font design process mentioned in [3], [4], [19], [27] and [28]. In addition, some design concepts were inspired from calligraphy guidelines mentioned in [11] and [12]. Some of the measurements used to define the digits, have been discussed and adjusted with a group of four Arabic typeface designers. Several initial models have been presented to the four designers and several modifications have been made based on their suggestions and recommendations. They suggested replacing some features that are not relevant to distinguish font from one another. They had some suggestions about the selected characters for the measurements. Some modifications were related to the type of the information described by each digit. The rest of this section describes all eight digits in detail.

Digit 1: Weight Classification

The weight describes the overall stroke thickness of a font [3]. Since Arabic letters are curvy by nature, normally stroke thickness varies in different parts of one glyph. Therefore, to estimate overall font weight, the average stroke thickness *AvgThick* is calculated for 18 basic glyphs, as shown in Figure 50, using equation (44). Then, the variable *WeightRate,* which is the proportion between *AvgThick* and *Ascender*, is evaluated using equation (45). *Ascender* is measured on letters Kaaf 'ك' or Alef 'ا'. In the case that letters Kaaf and Alef have different ascenders, the average value between them is considered in the *WeightRate* calculation. For this model, only three weight variations are defined. The weight variants example is shown in Figure 49. The values and the attached weight variants for each value are listed below. Assigning a value for this digit is obtained by using the weight classification algorithm:

0: Any
1: No Fit
2: Light
3: Regular
4: Bold

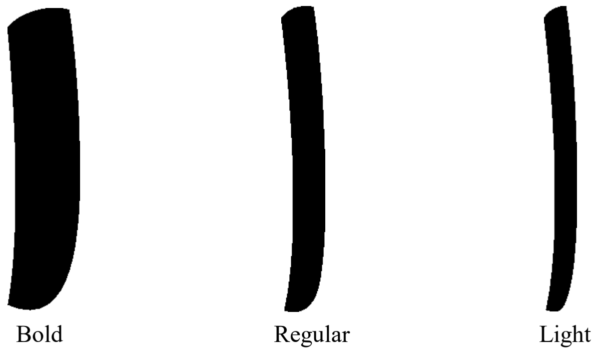


Figure 49: Three weight variants for Alef glyph

$$AvgThick = \sum glyphThick / glyphsCount \quad (44)$$

$$WeightRate = Ascender / AvgThick \quad (45)$$

Weight classification algorithm:

IF *WeightRate* > 11 THEN **Light**

ELSE IF 8 ≤ *WeightRate* ≤ 11 THEN **Regular**

ELSE IF *WeightRate* < 8 THEN **Bold**

The thresholds for the weight classification algorithm were determined by testing a set of 23 fonts. Each font has three weights, which are light, regular and bold, giving a total of 69 fonts. These three weights are more common than other weight variants for text. For 23 fonts of the same weight, the weight ratio for each font is calculated. The line graph plot for all fonts' weights is shown in Figure 51. For each weight, the minimum and maximum values of weight ratio are used as boundaries to select the threshold. For each candidate threshold value, classification perception is evaluated. The candidates' values with the highest classification perception are selected as the weight classes' threshold.

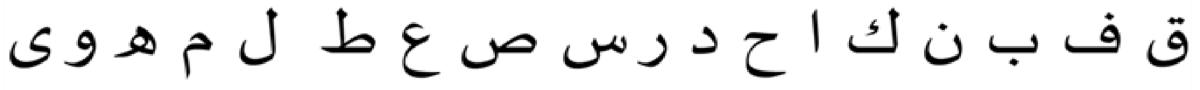


Figure 50: Sample image of 18 glyphs

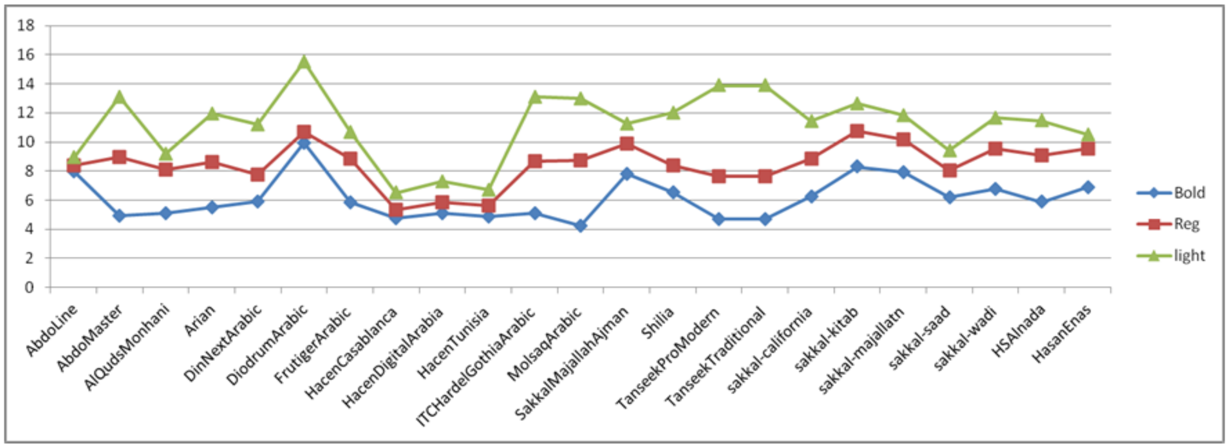


Figure 51: Three weight values for 23 fonts

Digit 2: End Style Classification

The End Style digit describes the shape of the terminal stroke's end in letters Baa 'ب', Aien 'ع' or Noon 'ن'. Six variant shapes are defined for this model. An illustration of the variant shapes is shown in Figure 52. The variable *CornerNo* is used to classify different styles.

CornerNo is the number of corner points at the end of a curved tail stroke. When the tail end has no corner points, it means that the tail end is smoothly rounded. If the tail end has one corner point, it means that it has a sharp pointy end. If the tail end has two corner points, it means that its end is trimmed. The trimmed ends have three variants shapes based on the cut slop. It can be horizontal, vertical or diagonal. If the tail end has more than two corner points, then the end has a serifed style class, which is not a common class for Naskh style. The classification is obtained by using the algorithm below.

0: Any
1: No Fit
2: Rounded
3: Pointed
4: Trimmed Horizontally
5: Trimmed Vertically
6: Trimmed Diagonally
7: Serifed

End style classification algorithm:

```
IF CornerNo = 0 THEN Rounded
ELSE IF CornerNo = 1 THEN Pointed
ELSE IF CornerNo = 2 THEN Check the slope
    Check the slope:
    IF slope =0 THEN Trimmed Horizontally
    ELS IF slope<0 OR slope >0 THEN Trimmed Diagonally
    ELSE Trimmed Vertically
ELSE IF CornerNo > 2THEN Serifed
```

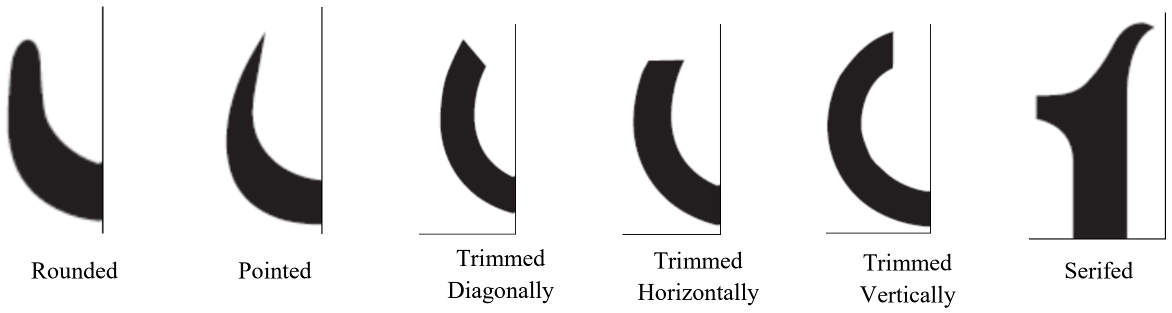


Figure 52: End styles of a curved tail stroke

Digit 3: Contrast Classification

The contrast of a glyph is expressed as the ratio between the thickest and the thinnest parts of the glyph as in (46) [19] [27]. After calculating the contrast of each of the18 glyphs, that are shown in Figure 50, the overall font contrast is then evaluated as the average of the 18 glyphs' contrast using equation (47). Four contrast variations are defined for this model, as shown in Figure 53. The values that can be assigned to contrast classes are listed below. The classification is obtained by the contrast classification algorithm.

0: Any
1: No Fit
2: None
3: High
4: Medium
5: Low

$$GlyphContrast = thickest/thinnest \tag{46}$$

$$ConRat = \sum_{i=1}^{18} GlyphContrast_i/18 \tag{47}$$

Contrast classification algorithm:

IF *ConRat* = 1 THEN **None**

ELSE IF *ConRat* < 0.3 THEN **High**

ELSE IF 0.3 ≤ *ConRat* ≤ 0.6 THEN **Medium**

ELSE IF *ConRat* > 0.6 THEN **Low**

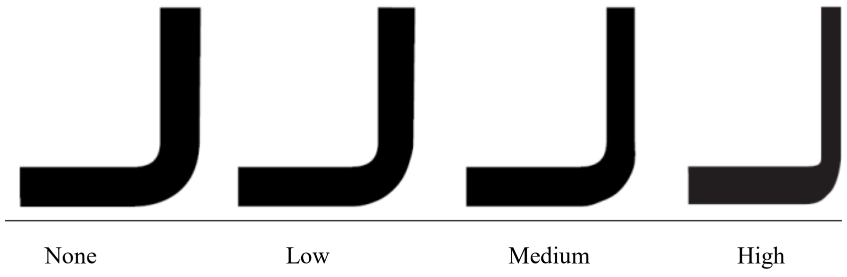


Figure 53: Contrast variations

Digit 4: Bowl Form Classification

The Bowl Form describes the shape of the bowl stroke in letter Aien 'ع' based on its vertical breadth, horizontal breadth and the symmetry of the bowl open counter. The shape of bowl stroke is described by two variables. The first variable *BowlBProp* represents the proportion between the vertical and horizontal breadth B_V and B_H respectively, as in equation (48). The vertical breadth is the vertical distance between the two bowl ends. Hence, the horizontal breadth is the horizontal distance between the middle of the imaginary vertical breadth line and the bowl stroke, as illustrated in Figure 54. In the Naskh style calligraphy guideline, the proportion between the vertical and horizontal breadth is around 5/3 [11] [12]. Using that as our guide, three variants bowl breadth proportions are defined, which are Stretched, Normal and Squeezed. When the vertical breadth is at least twice the horizontal breadth, the bowl breadth is classified as Stretched. In the case where the vertical breadth is less than or equal to the horizontal breadth, the bowl breadth is classified as Squeezed. Otherwise, the bowl breadth is a Normal class. An illustration of the three bowl breadth proportion classes are shown in Figure 55. The breadth class is determined by the breadth classification algorithm below.

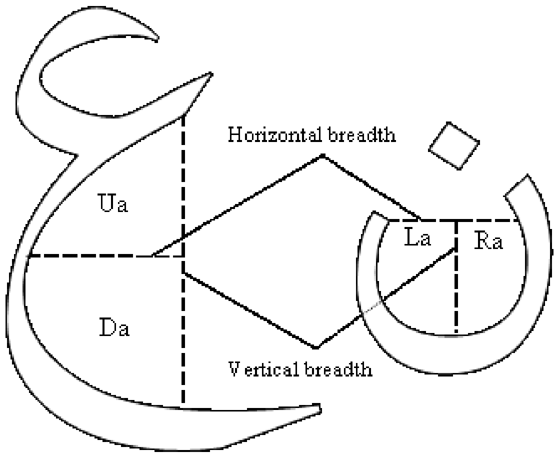


Figure 54: Measurements for Bowl and Curved Tail strokes

The second variable *BowlOC* describes the shape of the open counter of the bowl stroke. It indicates the proportion between the two areas, above and under the horizontal breadth imaginary line, which are denoted as *Ua* and *Da* in equation (49), and are shown in Figure 54. If the two areas are equal, the open counter is symmetric. When the upper area is larger than the lower one, the deepest point of the open counter is oblique upward. In contrast, if the lower area is larger than the upper one, it means that the deepest point of the open counter is oblique downward. An illustration of the three open counter shape classes is shown in Figure 55. The open counter class is determined by the open counter classification algorithm below. The combination between the two classes defines nine different sub-classes, which are each attached with a value. The list of all values and sub-classes combination is listed below.

0: Any
1: No Fit
2: Normal / Oblique Upward
3: Normal / Oblique Downward
4: Normal / Symmetric
5: Squeezed / Oblique Upward
6: Squeezed / Oblique Downward
7: Squeezed / Symmetric
8: Stretched / Oblique Upward
9: Stretched / Oblique Downward
10: Stretched / Symmetric

$$BowlBProp=B_V/B_H \tag{48}$$

$$BowlOC=Ua/Da \tag{49}$$

Bowl breadth proportion classification algorithm:

```
IF $BowlBProp \leq 1$ THEN **Squeezed**
ELSE IF $1 < BowlBProp \leq 2$ THEN **Normal**
ELSE IF $BowlBProp > 2$ THEN **Stretched**
```

Bowl open counter classification algorithm:

```
IF $BowlOC \geq 1.3$ THEN **Oblique Upward**
ELSE IF $BowlOC \leq 0.7$ THEN **Oblique Downward**
ELSE **Symmetric**
```

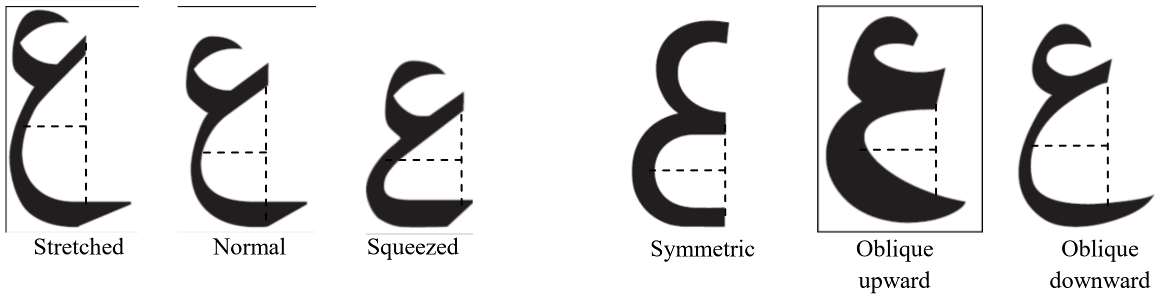


Figure 55: Bowl stroke shape classes

Digit 5: Curve Tail Form Classification

A curve tail form is described with variables similar to ones used to describe the bowl but the measurements are taken on the letter Noon 'ن'. It is separate from the bowl digit due to the different description and proportion each stroke can have within the same font. The variable *CurvTBProp* describes the breadth proportion between the horizontal B_H and vertical B_V breadths. It is calculated using equation (50). B_H is the distance between the two tail stroke ends while B_V is the distance between the middle of the horizontal breadth line and the tail stroke, as in Figure 54. The proportion between B_H and B_V for a curve tail in normal forms of Naskh style is different from the bowl, hence it is around 3/2 [11] [12]. Therefore, similar breadth proportion classes are defined for the curve tail stroke but the classification algorithm is different. An illustration of the three breadth proportion classes are shown in Figure 56. The classification is obtained by using the curve tail breadth proportion algorithm. The curve tail open counter shape classes are similar to the ones defined for the bowl but are based on the *CurvTOC* variable. It is calculated using equation (51), where *Ra* is the area between the vertical breadth imaginary line and the stroke while *La* is the area on the left side of the vertical breadth imaginary line, as shown in Figure 54. The values and sub-classes that can be assigned to this digit are listed below.

0: Any
1: No Fit
2: Normal / Oblique Right
3: Normal / Oblique Left
4: Normal / Symmetric
5: Stretched Vertically / Oblique Right

6: Stretched Vertically / Oblique Left

7: Stretched Vertically / Symmetric

8: Stretched Horizontally/ Oblique Right

9: Stretched Horizontally / Oblique Left

10: Stretched Horizontally / Symmetric

$$CurvTBProp=B_H/B_V \tag{50}$$

$$CurvTOC=Ra/La \tag{51}$$

Curve tail breadth proportion classification algorithm:

IF *CurvTBProp* < 1 THEN **Stretched Vertically**

ELSE IF 1 ≤ *CurvTBProp* ≤ 1.5 THEN **Normal**

ELSE IF *CurvTBProp* > 1.5 THEN **Stretched Horizontally**

Curve tail open counter classification algorithm:

IF *CurvTOC* ≥ 1.3 THEN **Oblique Right**

ELSE IF *CurvTOC* ≤ 0.7 THEN **Oblique Left**

ELSE **Symmetric**

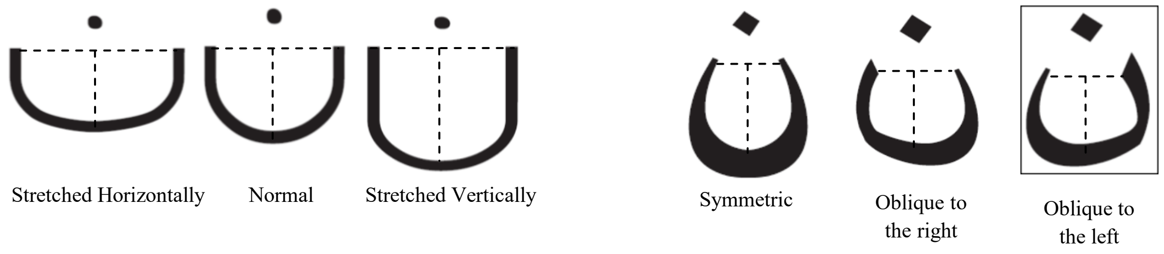


Figure 56: Curve tail stroke shape classes

Digit 6: Tooth Height Classification

This digit describes the two teeth strokes at the beginning of the letter Siin 'س'. They are described based on two attributes, which are the height variation and the proportion between the first tooth to the ascender. The height variation is determined by measuring the heights of both teeth, T_1 for the first one and T_2 for the second one. T_1 and T_2 represent the distance between the baseline and their upper extensions [3] [12].The nature of Naskh style is to have the first tooth higher than the following one. For this case, the Standard class is assigned. If the two teeth have the same height, the Constant class is assigned. The proportion between the first tooth height and the ascender is represented by variable *TeethProp* that is calculated using equation (52). Three tooth proportion classes, which are Small, Middle and Large, are defined for this model. If the tooth's height is equal to 1/3 or less than the ascender's height, then it is considered as Small. If the tooth's height is higher than 1/3 and less than 2/3 of the ascender, then it is considered as Middle. The tooth height is considered as Large if its height is larger than 2/3 of the ascender height. The thresholds values that distinguish between the three tooth proportion classes are used in algorithm below. An illustration of the height variation and tooth proportion classes are shown in Figure 57. The combination of the two attributes classes produces six sub-classes; each is attached to a value as follows:

0: Any
1: No Fit
2: Standard / Small
3: Standard / Middle
4: Standard / Large
5: Constant / Small
6: Constant / Middle
7: Constant / Large

$$TeethProp = T_1 / Ascender \tag{52}$$

Tooth proportion classification algorithm:

IF $TeethProp \leq 0.33$ THEN **Small**
ELSE IF $0.33 < TeethProp < 0.66$ THEN **Middle**
ELSE IF $TeethProp \geq 0.66$ THEN **Large**

Height variation classification algorithm:

IF $T_1 = T_2$. THEN **Constant**
ELSE **Standard**

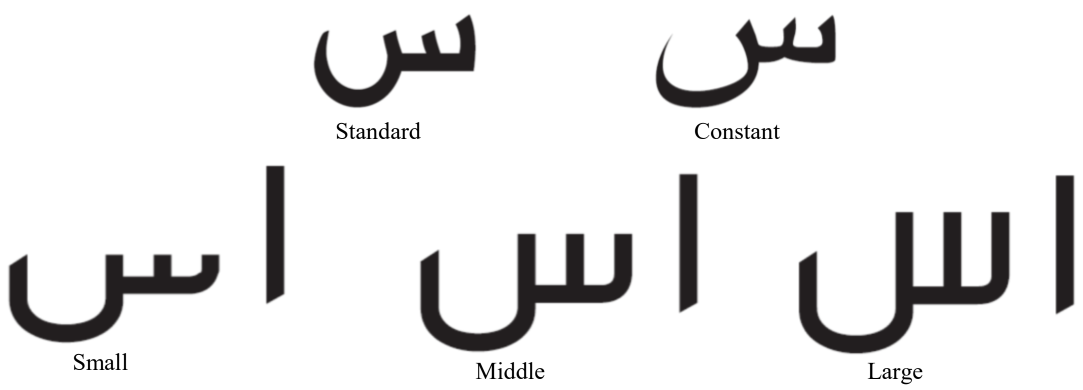


Figure 57: Height variation and tooth proportion classes

Digit 7: Loop Height Classification

The Loop Height class names, class values and the thresholds that classify them are similar to Tooth Height digits except that the measurements are based on the letters Faa ‘ف’ to get the first loop height $L_{1,}$ and Waaw ‘و’ for the second loop height, $L_{2.}$ The variable *LoopProp* represents the proportion between loop height of Faa and *Ascender* as in the equation (53). An illustration of the loop height variation and loop proportion classes are shown in Figure 58.

$$LoopProp = L_1 / Ascender \tag{53}$$

Loop proportion classification algorithm:

IF *LoopProp* $\leq$ 0.33 THEN **Small**

ELSE IF 0.33 < *LoopProp* < 0.66 THEN **Middle**

ELSE IF *LoopProp* $\geq$ 0.66 THEN **Large**

Loop height variation classification algorithm:

IF $L_1 = L_{2.}$ THEN **Constant**

ELSE **Standard**

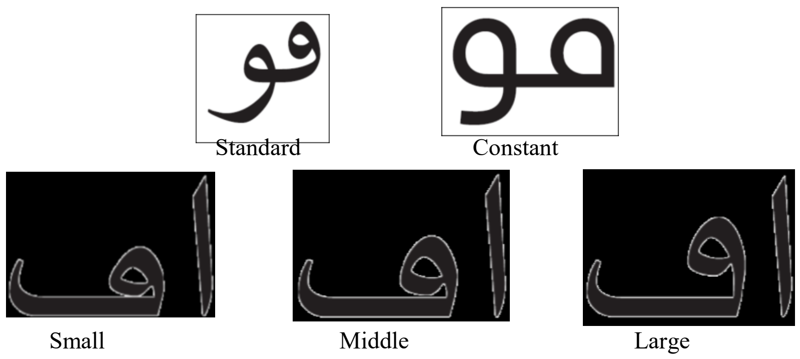


Figure 58: Loop height variations and loop proportion classes

Digit 8: Round Stroke Classification

Round strokes shape are described by the shape of its counter. One attribute in this digit was used to classify the shape of the counter. This attribute indicates how much the counter's shape is far from any of the five geometric shapes, which are triangle, circle, oval, square or rectangle. The counter's shape is determined by a variable called *Extent* as in (54). The variable indicates the ratio between its area and the area of the bounding box [29].

$$Extent = ShapeArea / BBoxArea \tag{54}$$

$$BBDim = BBwidth / BBheight \tag{55}$$

The Bounding box is an imaginary box that encloses the whole counter. Its sides are parallel to the main axes. The *Extent* variable separates between three major classes of a counter shape. In [29], if a shape is a perfect circle, the value of the *Extent* is constant, which is 0.78 regardless of its radius. If the *Extent* is 1, this indicates the shape is square or rectangle. For the triangle, based on geometrical rules, the diagonal axes of a square divide the square into two equal triangles. This indicates that the *Extent* for a triangle is equal to 0.5.

Shape classification thresholds in [29] are used with modifications to classify the shape of the round stroke counter. Naskh style does not have a geometrical appearance. Round strokes could have a counter look similar to a triangle, square or a circle but not exact ones. It is not common to have squares or triangles with acute angles and pointy corners. They are more likely to have them with curvy corners, as in Figure 59. Even a circle counter cannot be a perfect circle. It may have a corner that makes it look like drop, as in Figure 59. Therefore, taking the previous variations in the consideration, new thresholds are defined to classify counter shapes. The thresholds based on the *Extent* variable are defined to be in the middle of the difference between thresholds in [29]. These thresholds are to separate between the main three shape classes, which are triangle, circle and square. Then for the circle and square classes, oval and rectangle classes are defined based on the proportion of bounding box dimensions *BBDim* as in (55). The values attached to each counter shape class and the algorithm to obtain them, are as follow:

0: Any
1: No Fit
2: Triangle
3: Oval
4: Circle
5: Rectangle
6: Square

Round Stroke classification algorithm:

```
IF $0.25 < Extent < 0.64$ THEN **Triangle**
ELSE IF $0.64 \leq Extent < 0.89$ THEN Check BBDim
        Check BBDim:
        IF $BBDim = 1$ THEN **Circle**
        ELSE **Oval**
ELSE IF $0.89 \leq Extent \leq 1$ THEN Check BBDim
        Check BBDim:
        IF $BBDim = 1$ THEN **Square**
        ELSE **Rectangle**
```

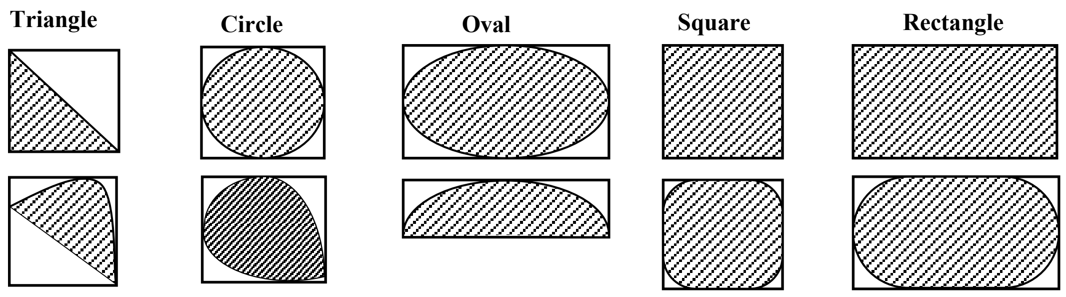


Figure 59: Round stroke counter classes

Chapter 5 : Evaluating PANOSE-A

The previous chapter describes the digits of PANOSE-A model for classifying Arabic fonts. To evaluate the efficiency of the proposed model, two similarity clustering processes of a set of 30 fonts were conducted. The fonts in the set are listed in Table 19. The two clustering processes used two similarity matrixes, which are constructed differently. One similarity matrix was constructed using a font matching tool to calculate the similarity between each pair of fonts. The other similarity matrix was constructed based on the Euclidian distance between PANOSE-A numbers that were assigned by the proposed model. The similarity between the results of the two clustering processes was evaluated by the measure B_k that represents the number of matching fonts in the output clusters.

5.1. Data and tools

For this research, the *Arabic Font Specimen Book* [4] was used as an essential reference to get a general perception of existing digital fonts to select the initial set for this research. The book has a non exhaustive list containing 615 fonts and includes pages showing text samples in heading, introduction and caption sizes. The book also displays font Arabic characters with one character enlarged to show the quality of the outline. Moreover, font information is mentioned, such as style classification, designer name, format, date of issue, and font family.

Generally, the list of fonts in the book includes five main categories; fonts in each category are organized differently. One category includes fonts from cultural foundations and non-profit organizations, which have research centers that design and produce fonts, such as Khatt collections. Fonts in another category are organized according to fonts' designers. This category includes 16 groups of well known designers such as Pascal Zoghbi, Mamoun Sakkal and Sultan Maktari. Fonts in another category are organized based on fonts' vendors such as Decotype, Linotype, Monotype and Adobe while another category includes fonts of Arabic publishing software and fonts' developers such as Diwan, Winsoft and Layout Company. The last category includes fonts that are often available for free. The fonts in this category are either offered by their designers for free or are created though collaboration on an open source network, such as an Arabic community, Arabeyes, that simulates a Linux open source. Also, this category includes fonts created by activities and projects usually done by universities which have semigovernment support in countries that use Arabic scripts. Some examples of groups that belong in this category are the KACST font collection of King Abdulaziz City for Science and Technology in Saudi Arabia and the Farsi Web font collection, which is supported by the high counsel Informatics of Iran and Sharif University of technology.

Several criteria have been applied to the font selection process. Since the proposed model is inspired from Naskh style design characteristics, the selected fonts are mainly from Naskh style. Also there are some modern design fonts for text that merge Naskh with other styles. Among these fonts, ones that are designed exclusively for specific organizations and companies and fonts that have decorative appearances, such as outline or shaded fonts have been excluded. The selected fonts have been accessed through Monotype Library Subscription. Monotype provides monthly subscription services for user to able to access more than 2000 font's family. It allows users to install fonts through their management system skyfont and therefore use them in any document designing. Other Seven fonts from the selected set, which are Diodrum Arabic, Abdo Line, Abdo Master, AlQuds Monhani, Molsaq Arabic, HS Alnada have been bought at a total cost of 958$. Each font is bought with a set of three different weights which are light, regular and bold. Some other fonts and sample images of fonts are denoted from their designers. Sakkal Design donated the Sakkal Majllah font family in addition to a set of image samples of other three font families with different weights to estimate weight thresholds that are defined in the weight classification digit. Hasan Enas font family is also donated by Hiba Studio. The remaining fonts were selected randomly to have some fonts from each collection with respect to the available budget. Most fonts used in this research were chosen from the book and were purchased from an online font store https://www.myfonts.com/. In addition, several fonts, which are not listed in the book, have been taken from common websites for free Arabic fonts [30] [31]. The selecting process ended up with a set of 30 fonts for the purpose of evaluating the efficiency of the proposed model of PANOSE-A.

To facilitate dealing with font files, the *NexusFonts2.5.8* font management system was used. It is a font management that has a simple interface that displays font file attributes such as font name, file name, file size, file format and vendor. It allows for working with fonts in folders even fonts that are not installed. In addition to folder organization, it allows for organizing fonts into user defined sets. The font files remain in their original folders and can be included in many different sets and groups. Moreover, it displays any text sample and allows changing the font size and style (italic, bold and underlined) as well as displays a character map of a font with characters grouped according to their Unicode. It gives an option to export text samples and font character maps as images, which can be saved into separate files. In addition, it is able to detect duplicated fonts even if they have different file names.

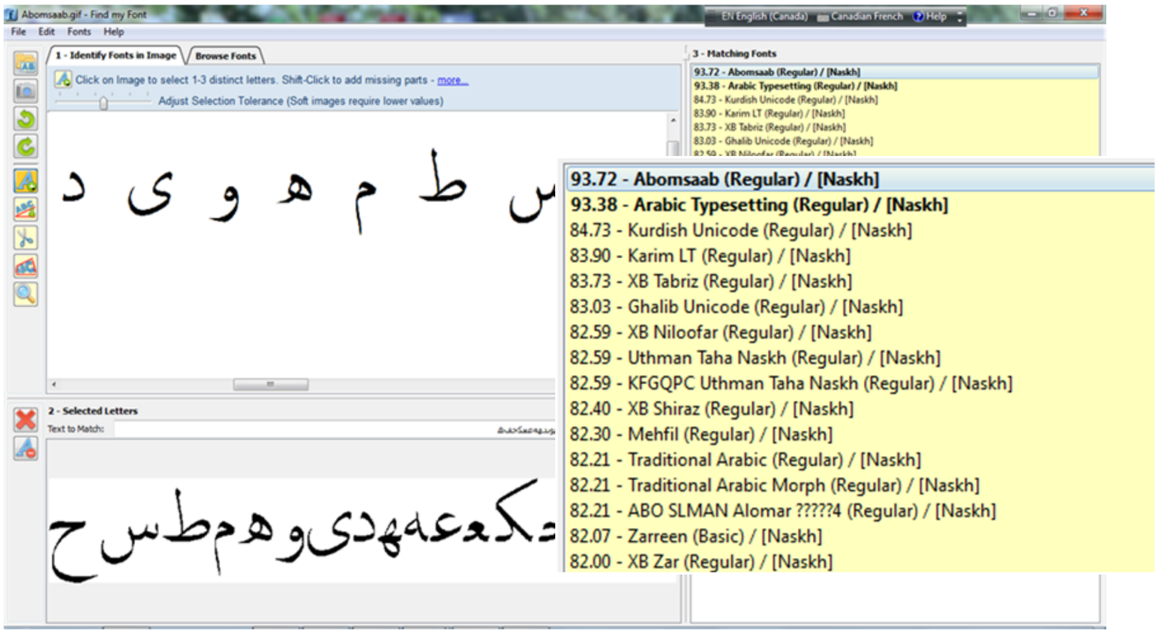
For each font in the selected collection for this research, an image was created using *NexusFonts2.5.8*. Each image contains a sample test of the basic 18 glyphs separated by spaces, as in the example shown in Figure 50. In order to have glyphs' details clear for comparison, the size of fonts in the samples were set to 480 points. In total, 30 images were used to calculate the similarity between fonts.

5.2. Similarity Matrixes

Two 30×30 similarity matrixes were constructed. Each element in a similarity matrix is the similarity evaluation or distance between each pair of fonts. One similarity matrix was constructed by using a font matching tool to estimate the similarity between fonts. Elements in the other similarity matrix indicate the similarity between fonts based on the PANOSE-A numbers assigned to represent each font in the set. Details of how the two similarity matrixes were constructed will be outlined in this section.

5.2.1. Similarity matrix by font matching tool:

To calculate the similarity between fonts, *Find My Font 3.1* was used. It is a font matching tool that takes an image of text or characters and finds the most similar fonts to those used in the image. A user can determine where the tool should look for similar fonts. The search can be done online, among system fonts, or in specific folders even if fonts are not uninstalled. As a result, it displays a list of the most similar fonts as well as their similarity percentages. The list was sorted in descended order according to the similarity as shown in Figure 60. The accuracy level of matching can be adjusted from 50% up to 100%. This will determine which fonts should be considered and displayed in the results [32].



Click on Image to select 1-3 distinct letters. Shift-Click to add missing parts - more...

Adjust Selection Tolerance (Soft images require lower values)

```
93.72 - Abomsaab (Regular) / [Naskh]
93.38 - Arabic Typesetting (Regular) / [Naskh]
84.73 - Kurdish Unicode (Regular) / [Naskh]
83.90 - Karim LT (Regular) / [Naskh]
83.73 - XB Tabriz (Regular) / [Naskh]
83.03 - Ghalib Unicode (Regular) / [Naskh]
82.59 - XB Niloofar (Regular) / [Naskh]
82.59 - Uthman Taha Naskh (Regular) / [Naskh]
82.59 - KFGQPC Uthman Taha Naskh (Regular) / [Naskh]
82.40 - XB Shiraz (Regular) / [Naskh]
82.30 - Mehfil (Regular) / [Naskh]
82.21 - Traditional Arabic (Regular) / [Naskh]
82.21 - Traditional Arabic Morph (Regular) / [Naskh]
82.21 - ABO SLMAN Alomar ?????4 (Regular) / [Naskh]
82.07 - Zarreen (Basic) / [Naskh]
82.00 - XB Zar (Regular) / [Naskh]
```

Figure 60: Find My Font screen shot

The *Find My Font* matching process needs to select one or more letters from an input image. Therefore, *Find My Font* has beneficial image editing features to ease letters selection of the input image. It has a baseline rotation and image deformation correction tool to fix slanted and crooked text. In addition, it has a cut tool to separate connected characters to select them individually. The selection process is done by clicking on a connected component with the same color using the selection tool. By clicking on a letter, the program selects an entire area of same color and determines the boundary of the letter where a big change in color has occurred. The tolerance of the selecting process can be adjusted to allow selecting the maximum range of surrounding colors that are slightly different around the selected one. A low value selects few colors very similar to the clicked color. A higher value selects a broader range of colors. This allows selecting the whole character if the boundary may have different colors. After selecting the character from image, the program requires inserting text that matches the selected letters to speed up the search and the matching process [32]. The tolerance level was adjusted to the lowest value since the images used for analysis have black and white colors with slightly gray levels for an anti aliasing view. This allows selecting the whole contour of the glyph.

The accuracy of matching was set to 50% so it could detect any possible similarity. For each font, all 18 glyphs were selected to contribute in the similarity estimation during the matching process. The results are stored in a 30×30 matrix. Pairs of fonts with no similarity have a zero value in their cells. The reliability of the tool has been estimated on all collected fonts. For each matching process, the correctness of matching was considered when the tool gave the highest similarity value to the font used in the input image. Based on that, the confusion matrix for 30 fonts was calculated and produced a matching accuracy of 92.9%. A sample of the result similarity matrix is shown in Table 14.

Table 14: A sample of a similarity matrix constructed by Find My Font

	Lotus Linotype Light	**Maged LT**	**Mitra LT Light**	**PakType Tehreer**	**Qadi Linotype**	**Sakkal Majalla**	**Simplified Arabic**	**Simplified Arabic Fixed**	**Tahoma**	**Tanseek Modern Pro Arabic Medium**	**Times New Roman**	**Traditional Arabic**	**XB Tabriz**	**Yakout Linotype Light**
Lotus Linotype Light	95.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	89.7	91.6	0.0
Maged LT	0.0	95.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Mitra LT Light	0.0	0.0	95.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
PakType Tehreer	0.0	0.0	0.0	96.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Qadi Linotype	0.0	0.0	0.0	0.0	96.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Sakkal Majalla	0.0	0.0	0.0	0.0	0.0	95.1	0.0	77.0	0.0	0.0	0.0	0.0	0.0	0.0
Simplified Arabic	0.0	0.0	0.0	0.0	0.0	0.0	83.2	0.0	0.0	0.0	91.8	0.0	0.0	0.0
Simplified Arabic Fixed	0.0	0.0	0.0	0.0	0.0	77.0	0.0	78.2	0.0	0.0	86.6	0.0	0.0	75.4
Tahoma	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	95.8	0.0	0.0	0.0	0.0	0.0
Tanseek Modern Pro Arabic Medium	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	94.7	0.0	0.0	0.0	0.0
Times New Roman	0.0	0.0	0.0	0.0	0.0	0.0	91.8	86.6	0.0	0.0	95.3	0.0	0.0	0.0
Traditional Arabic	89.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	94.5	93.2	0.0
XB Tabriz	91.6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	93.2	95.4	0.0
Yakout Linotype Light	0.0	0.0	0.0	0.0	0.0	0.0	0.0	75.4	0.0	0.0	0.0	0.0	0.0	95.5

5.2.2. Similarity Matrix by PANOSE-A

Constructing the similarity matrix based on PANOSE-A requires assigning PANOSE-A numbers to represent each font. The similarity for each pair of fonts is then calculated by using the Euclidian distance formula.

5.2.2.1. *Assigning PANOSE digits*

The digits of PANOSE-A were assigned by implementing the required measurements and calculated variables through the use of several image analysis techniques. All the sample images of fonts were transformed into black and white images. No size normalization preprocessing was done to avoid affecting or changing the fonts' aspect proportions. For individual character measurements the images of characters glyphs were cropped by extracting the connected components methods. Dots were removed during preprocessing to reduce their effect in the projection files.

For each font sample image, the horizontal projection profiles were calculated to detect baseline position. In a horizontal projection profile of an image, the 2-D array of pixels was transformed into 1-D array. The size of the array is equal to the height the image or number of rows of 2-D array. Each element of 1-D array indicates the number of pixels in a row of the image [33]. By analyzing the projection profile, the baseline position is where the greatest value was found. It was shown within the projection profile plot where the greatest peak indicated the baseline position. An example projection profile plot of a font sample image and baseline are shown in Figure 61.

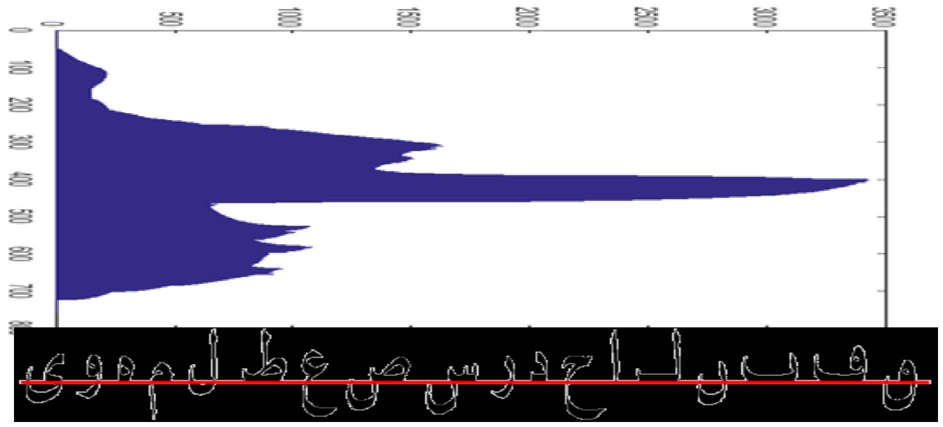


Figure 61: Horizontal profile to detect baseline

Stroke weight and contrast were calculated by applying Euclidean distance transform for all 18 glyphs. Distance transform produces an image that has the same size as an input image. For each pixel of the output image, the distance between the stroke pixel and the nearest background pixel is assigned [34]. A sample of input and output images are shown in Figure 62.

From the output image, the thickest, thinnest and average stroke thickness was calculated. The pixel with the highest value indicates the thickest stroke thickness while the pixel with lowest value indicates the thinnest thickness. The average thickness was the average of the sum of all non zero pixels.

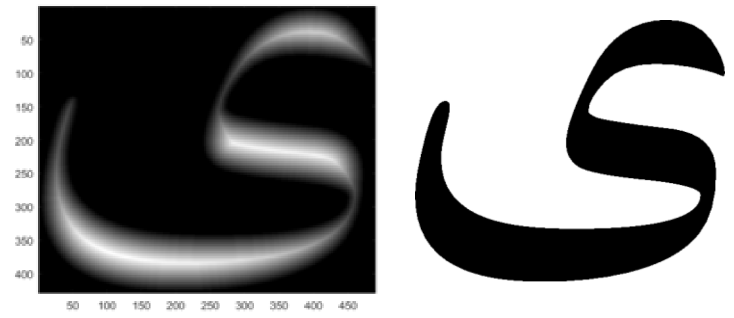


Figure 62: Distance transform image (left) and original image (right)

To assign a value to the end style digit of a terminal stroke, a corner points detecting algorithm in [35] was implemented. In this algorithm, a point on shape contour is a corner point if a triangle with a specific opening angle and size can be formed as shown in Figure 63. The corner points detecting algorithm in [35] is a two-pass algorithm that detects the corner points on an image contour. The contour's curves are represented by a sequence of points p_i, densely sampled along the curve with no required regular spacing between them. In the first pass, the algorithm detects candidate high curvature points by inscribing a variable triangle (p^-, p, p^+) on the curve. The triangle should satisfy three rules in (56). Where $|a|=|p-p^+|$ is the distance between p and p^+, $|b|=|p-p^-|$ is the distance between p and p^- and c is the distance between p^+ and p^-. The opening angle of the triangle α is calculated as in (57). From the first-pass of the algorithm, it is possible to have several consecutive points as candidate points. Therefore, in the second-pass of the algorithm, some of the candidate points are discarded. It is a post-processing technique to select the strongest candidate points and discard the non-maxima ones. A candidate point p is discarded if it has neighbor candidate point p_v with a smaller opening angle, which means if $\alpha(p)>\alpha(p_v)$.

$$
\begin{aligned}
&d_{min}^2 \leq \ |a|^2 \leq \ d_{max}^2 \\
&d_{min}^2 \leq \ |b|^2 \leq \ d_{max}^2 \\
&\alpha \leq \alpha_{max}
\end{aligned}
\tag{56}
$$

$$\alpha \ = \arccos \frac{a^2+b^2-c^2}{2ab} \tag{57}$$

The parameters d_{min} , d_{max} and α_{max} are thresholds that are used to control the algorithm. The d_{min} is the minimum distance between the triangle points while the d_{max} is the maximum distance that helps avoid false sharpness formed by far points. The α_{max} is the limit of the angle that is accepted as high curvature. For this research, the α_{max} is set to 120^{o}, d_{min} to 2 and d_{max} *to* 4.

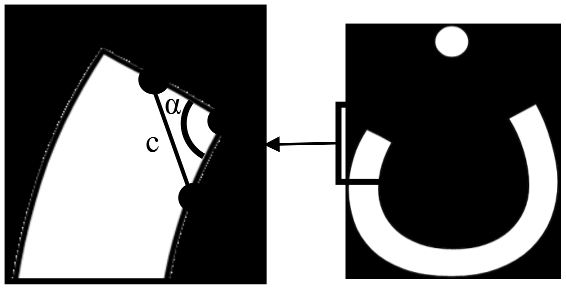


Figure 63 : Illustration of the corner points detection technique

Tooth and loop heights were detected by analyzing peaks of vertical projection profiles of the concerned letters. For the tooth heights, since our concern is the height of the first and second tooth of letter Siin, only the two greatest peaks from the right side of the profile were considered, if they satisfied two thresholds as shown in Figure 64. Their height should not be less than 7/4 of the font's weight which is the minimum peak height. In addition, the distance between the two peaks should not be less than 5/4 of the font's weight, which is the minimum peak distance. The loop heights were also detected from the vertical projection profiles of letters Faa and Waaw but after filling the counter inside the loop. Thus, the loop heights were the highest peak of each letter profile. The output assigned numbers for the 30 fonts are in Table 15.

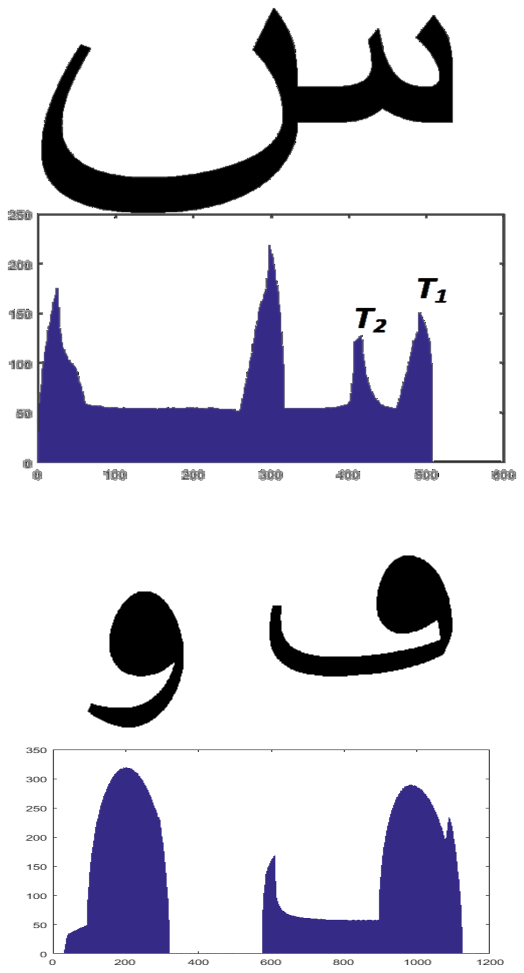


Figure 64: Vertical projection profile to detect teeth and loop heights

Table 15: Fonts and their assigned PANOSE numbers

Font Name	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
AA Sameer Qamri	3	2	3	8	10	7	6	2
ASVCodarLTLight	2	3	3	10	7	6	6	2
ArabicTypeseting	2	3	3	8	7	6	6	3
Arial	3	6	3	5	7	6	6	3
BadrLTLight	2	3	3	5	7	6	5	3
BuridahUnicode	2	4	5	10	6	6	5	3
Courier New	2	3	4	7	3	7	6	3
Droid Arabic Naskh	3	6	4	5	10	7	6	3
Dubai Unicode	4	2	3	2	8	7	6	3
Ghadeer	4	2	3	2	8	7	6	3
Hacen Newspaper	4	3	3	7	5	7	6	2
HacenTypographyHeavy	4	3	3	7	7	7	6	2
Hasan Enas	3	6	4	5	6	7	6	3
JalalLTRegular	3	2	3	5	4	6	6	2
JannaLTRegular	3	4	5	7	7	3	6	3
KacstNaskh	3	3	3	5	7	7	6	3
Lotus Linotype Light	3	3	3	5	7	7	5	3
MagedLT	3	3	3	7	7	7	6	3
MitraLTLight	2	6	4	7	6	6	6	3
PakType Tehreer	3	3	3	8	5	6	5	2
Qadi Linotype	4	3	3	7	7	7	6	2
Sakkal Majalla	3	6	3	5	7	6	6	3
Simplified Arabic Fixed	3	3	3	5	6	6	5	3
SimplifiedArabic	3	6	3	5	7	6	6	3
Tahoma	3	4	5	5	7	7	6	3
Tanseek Modern Pro Arabic Medium	4	6	5	4	4	4	4	5
TimesNewRoman	3	6	3	5	7	6	6	3
TraditionalArabic	3	6	3	5	7	7	5	3
XB Tabriz	2	6	3	5	7	7	5	3
Yakout Linotype Light	3	3	3	5	7	6	6	2

5.2.2.2. PANOSE-A similarity matrix

The similarity matrix of all fonts was constructed by using the eight digits assigned as their PANOSE-A numbers. The Euclidian distance of the PANOSE-A number between each pair of fonts was calculated by using equation (58)$d(F,S) = \sqrt{\sum_{i=1}^{7}(\mathrm{F_i} - \mathrm{S_i})^2}$, where $F1_i$ is the value of digit *i* from the first font and $F2_i$ is the value of the same digit of the second font.

$$d(F1,F2) = \sqrt{w_i(F1_i - F2_i)^2} \tag{58}$$

Table 16: A sample of similarity matrix constructed based on fonts' PANOSE-A number

	Lotus Linotype Light	**MagedLT**	**MitraLTLight**	**PakType Tehreer**	**Qadi Linotype**	**Sakkal Majalla**	**Simplified Arabic Fixed**	**SimplifiedArabic**	**Tahoma**	**Tanseek Modern Pro Arabic Medium**	**TimesNewRoman**	**TraditionalArabic**	**XB Tabriz**	**Yakout Linotype Light**
Lotus Linotype Light	100	95.9	94.8	99.0	96.7	95.0	98.6	96.8	93.4	95.2	96.8	96.8	96.7	97.0
MagedLT	95.9	100	96.5	95.8	92.9	96.8	96.7	95.6	92.9	92.8	95.6	95.0	94.7	92.9
MitraLTLight	94.8	96.5	100	94.7	92.0	98.6	95.4	96.7	92.8	93.8	96.7	95.6	95.8	92.3
PakType Tehreer	99.0	95.8	94.7	100	96.5	94.9	99.0	96.7	93.4	95.5	96.7	96.7	96.8	96.8
Qadi Linotype	96.7	92.9	92.0	96.5	100	92.3	95.6	94.8	92.4	94.5	94.8	95.6	95.1	98.6
Sakkal Majalla	95.0	96.8	98.6	94.9	92.3	100	95.6	97.0	92.7	93.8	97.0	95.9	95.8	92.4
Simplified Arabic Fixed	98.6	96.7	95.4	99.0	95.6	95.6	100	96.8	93.4	95.2	96.8	96.5	96.7	95.9
SimplifiedArabic	96.8	95.6	96.7	96.7	94.8	97.0	96.8	100	93.3	95.9	100	98.6	98.3	95.0
Tahoma	93.4	92.9	92.8	93.4	92.4	92.7	93.4	93.3	100	92.4	93.3	93.3	93.1	92.4
Tanseek Modern Pro Arabic Medium	95.2	92.8	93.8	95.5	94.5	93.8	95.2	95.9	92.4	100	95.9	95.6	96.3	94.9
TimesNewRoman	96.8	95.6	96.7	96.7	94.8	97.0	96.8	100	93.3	95.9	100	98.6	98.3	95.0
TraditionalArabic	96.8	95.0	95.6	96.7	95.6	95.9	96.5	98.6	93.3	95.6	98.6	100	98.3	95.6
XB Tabriz	96.7	94.7	95.8	96.8	95.1	95.8	96.7	98.3	93.1	96.3	98.3	98.3	100	95.5
Yakout Linotype Light	97.0	92.9	92.3	96.8	98.6	92.4	95.9	95.0	92.4	94.9	95.0	95.6	95.5	100

5.3. Clustering

Clustering is the process of grouping a set of objects where the objects in the same group are more similar to each other than objects in other groups. All clustering algorithms require distance measure function between two objects or clusters. Some of the clustering algorithms require determining the number of desirable clusters in advance. Since, in our analysis, there was no indication of how many clusters we could group the fonts into, the hierarchical agglomerative clustering algorithm was applied. Agglomerative means that the algorithm will start by having clusters equal to the number of objects, which means that each object forms a singleton cluster. The major process in hierarchical clustering is merging the two nearest clusters. The algorithm will proceed until all objects become one cluster [36] [37].

A variation in the hierarchical clustering algorithm is how to measure the distance between clusters. With Single-Link, the distance measure is the smallest distance between any two objects, each belonging to a cluster. Whereas with Complete-Link the distance is the largest distance between any two objects, each belonging to a cluster. The Average Link distance is the average distance between each pair of objects between the clusters [36] [37] [38]. The clustering process was conducted as follow:

1. Assigning each font into a cluster.
2. Defining the distance between each pair of clusters by referring to the similarity matrix.
3. Merging the most similar pair of clusters into a single cluster.
4. Updating the similarity matrix and computing the similarity between the new cluster and each old cluster (Single-Link).
5. Repeating steps 3 and 4 until all fonts are clustered into a single cluster.

The clustering was done twice using both similarity matrices; clustering processes are denoted as C_1 and C_2. C_1 is the clustering process that used similarity matrixes of *Find My Font* whereas C_2 is the clustering process that used similarity matrixes constructed based on the PANOSE-A numbers. The clustering stages were recorded in agglomeration schedule tables where each row represents a stage. For each stage, the two clusters to be merged were recorded with the similarity value between them. The clustering process can also be illustrated by a dendogram. It is a tree diagram that represents the arrangement and the merging of clusters through the clustering process.

The result clusters can be selected at any stage of the clustering process depending on the desired characteristics in the results. This selection is represented on the dendogram as a cut of the clustering tree diagram. Since we did not have the ground truth class assignment for the fonts, two measurements were used for analyzing the clustering results in order to decide at which stage the clustering should stop. The first measurement is the Silhouette coefficient s_f is calculated by using (59) for each font f of the whole set of F fonts. The variable, a, is the mean distance between a font f and all other fonts in the same cluster, while the variable, b, is the mean distance between a font f and all other fonts in the nearest cluster. It was used to evaluate the validity and the consistency of the cluster results. Silhouette coefficient values range from 1 to -1, where 1 means that the font is well matched within its cluster and -1 means that the font is badly matched when compared to the nearest cluster. Then the Silhouette coefficient for all fonts S is the mean of Silhouette coefficient s_f of each font f as in (60). When most of the fonts have a high value of s_f, then the clustering configuration is appropriate [39].

$$s_f = b - a/\max(a, b) \tag{59}$$

$$S = \frac{\sum_{f=1}^{F} s_f}{F} \tag{60}$$

The second measurement used to analysis clustering results is called B_k as in [40]. If there are two clustering, denoted as C_1 and C_2, for the same set of objects n, then B_k evaluate the similarity between clustering results who have the same number of clusters k. B_k is derived from a matching matrix M where the value in each cell M_{ij} is the number of common objects between i^{th} and j^{th} clusters from C_1 and C_2 respectively. B_k is '0' when m_{ij} is '0' meaning that every pair of objects that are in the same cluster in C_1, are assigned to different clusters in C_2. When B_k is '1' if M has k nonempty m_{ij} has '1', it means that all objects in the i^{th} cluster of C_1 are also assigned to the j^{th} cluster of C_2 [40]. The B_k is calculated as in (61), where its parameters are calculated by the equations in (62).

$$B_k = T_k/\sqrt{P_k Q_k} \tag{61}$$

$$\left(\begin{array}{c} T_k = \sum_{i=1}^{k} \sum_{j=1}^{k} m_{ij}^2 - m_{..} \\ m_{i.} = \sum_{j=1}^{k} m_{ij} \\ m_{.j} = \sum_{j=1}^{k} m_{ij} \\ m_{..} = \sum_{i=1}^{k} \sum_{j=1}^{k} m_{ij} \\ P_k = \sum_{i=1}^{k} m_{i.}^2 - m_{..} \\ Q_k = \sum_{i=1}^{k} m_{.j}^2 - m_{..} \end{array}\right) \tag{62}$$

5.4. Weighting PANOSE-A digits

Not all digits should participate equally in estimating the similarity between fonts. Some digits are more distinctive than others. Therefore it would be beneficial to attach a weight to each digit to decrease or to increase its participation to evaluate fonts' similarity. From the discussions with consultant designers, it was not an intuitive decision to assign a precise weight for each digit. They came up with general suggestions that favor some digits over other but it was not easy to provide and agree on specific values of weights. Therefore, to assign a weight to each digit, we applied the optimal variable weighting method in [41].

Giving a rectangular data matrix *Y*, consists of *n* objects as rows and *m* variables as columns, the dissimilarity matrix *D* between all pairs of objects was computed using the Euclidean distance as in (63). This formula defines an element of *D,* which is d_{ij}, that indicates the distance between object *i* and *j.* The two variables y_{ip} and y_{jp} are the values of a variable *p* of object *i* and object *j.* The w_p is the weight associated with variable *p.* The method in [41] estimates the value of the $w_{1...}$ w_m in such a way that the computed dissimilarity metric is optimal for hierarchical clustering. The calculated weights should satisfy the two conditions in (64)

The hierarchical clustering was represented graphically by an ultrametric tree which is a labeled rooted tree where a nonnegative weight is attached to each node. The distance between *i* and *j* is defined as the largest weight associated with nodes on the path connecting *i* and *j*. This distance satisfies the ultrametric inequality in (65).

$$d_{ij} = \left[\sum_{p=1}^{m} w_p \left(y_{ip} - y_{jp}\right)^2\right]^{1/2} \quad (63)$$

$$\left.\begin{matrix} w_1, \ldots, w_m \geq 0 \\ w_1 + \cdots + w_m = 1 \end{matrix}\right\} \quad (64)$$

$$d_{ij} \leq \min(d_{ik}, d_{ik}) \quad (65)$$

Determining the optimal weight values for formula (63) is an optimization problem aimed to minimize the normalized sum of square deviation from ultermetric equality as in formula (66) where Ω_U is the set of ordered triple objects (*i,j* and *k*) that satisfy the ultrametric inequality as defined in (67).

$$min_{w_1,\ldots,w_m}\left[L_U(w_1, \ldots, w_m) = \frac{\sum_{\Omega_U}\left(d_{ik} - d_{jk}\right)^2}{\sum_{i<j}\sum d_{ij}^2}\right] \quad (66)$$

$$\Omega_U = \left\{(i,j,k)\middle| d_{ij} \leq \min(d_{ik}, d_{ik})\ and\ d_{ik} \neq d_{jk}\right\} \quad (67)$$

In [41], to remove constrains in (64), a variable transformation was applied. The process transformed the formula of (66) into the formula in (68) where d_{ij} is defined as in (69). The original weights, $w_{1\ldots}\ w_{m,}$ are obtained from $v_1, \ldots,\ v_{m-1}$ using equations in (70).

$$min_{v_1,\ldots,v_{m-1}}\left[L_U(v_1, \ldots, v_{m-1}) = \frac{\sum_{\Omega_U}\left(d_{ik} - d_{jk}\right)^2}{\sum_{i<j}\sum d_{ij}^2}\right] \quad (68)$$

$$d_{ij}^2 = \sum_{p=1}^{m-1} \frac{v_p^2}{1+\sum_{q=1}^{m-1} v_p^2}\left(y_{ip} - y_{jp}\right)^2 + \frac{1}{1+\sum_{q=1}^{m-1} v_q^2}\left(y_{im} - y_{jm}\right)^2 \quad (69)$$

$$\left.\begin{matrix} w_p = \dfrac{v_p^2}{1+\sum_{q=1}^{m-1} v_q^2} \quad for\ p = 1, m-1 \\ w_m = \dfrac{1}{1+\sum_{q=1}^{m-1} v_q^2} \end{matrix}\right\} \quad (70)$$

The minimization problem in (68) was solved by using the conjugate gradient method in [42]. The solution requires the first order derivative of L_U with respect to *v*. The result of the derivative of formula (68) is in (71). Initially all v_p are set equal to one then iteratively these initial estimates improved until the norm of the gradient becomes smaller than some pre-specified constant.

$$\begin{aligned}
&\frac{\partial L_U^*}{\partial v_p} = 2\left[\Sigma_{\Omega_U}(d_{ik} - d_{ji})\left(\frac{\partial d_{ik}}{\partial v_p} - \frac{\partial d_{jk}}{\partial v_p}\right)\Sigma\Sigma_{i<j} d_{ij}^2 - \Sigma\Sigma_{i<j} d_{ij}\frac{\partial d_{ij}}{\partial v_p}\Sigma_{\Omega}(d_{ij} - d_{jk})^2\right]\Big/\left[\Sigma\Sigma_{i<j} d_{ij}^2\right]^2 \\
&where \quad \frac{\partial d_{ij}}{\partial v_p} = \left\{-\left[\Sigma_{q=1}^{m-1}\frac{v_p v_q^2 (y_{iq} - y_{jq})^2}{c^2}\right] + \frac{v_p (y_{ip} - y_{jp})^2}{c} - \frac{v_p (y_{im} - y_{jm})^2}{c^2}\right\}\Big/ d_{ij} \qquad (71) \\
&and \quad c = 1 + \Sigma_{q=1}^{m-1} v_q^2
\end{aligned}$$

5.5. Results and Analysis

In the conducted clustering processes C_1 and C_2, the number of clusters was equal in all stages from the same level. The similarity matrix, for C_2, in Table 16 was evaluated using the equation (58) while the weights, w_i, for i=1,..,8, were set to 1. In this case, all eight digits were contributed equally to estimate the similarity of each pair of fonts. To decide at which clustering stage the cut should be to get the result clusters, the two measurements *S* and *B*, which were explained in previous section, were evaluated.

The similarity measurement *B*, between clustering processes C_1 and C_2, was calculated for all clustering process stages, except for the two extreme cases, where each font was in a cluster and all fonts were in one cluster. A plot of the number of clusters *k* and the measure *B* is shown in Figure 65-a. The highest clustering similarity was B_{22}=73% at stage 9, with 22 clusters. To emphasize the result, the Silhouette coefficient *S* was used to evaluate the validity and the consistency of the result clusters of both clustering processes. For clustering using *Find My Font*, S_{C1}= 97.04 while for clustering using PANOSE-A, S_{C2}= 98.24. This indicated that result clusters from C_1 and C_2 were coherent. The dendograms of C_1, C_2 and where the trees are cut are shown in Figure 66 and Figure 68.

From the results, feature, used to describe the fonts in PANOSE-A, were partially successful to imitate the similarity clustering that resulted when using the *Find My Font* tool to build the similarity matrix. It succeed in detecting fonts with high similarity due to their distinctive features, such as Ghadeer and Dubai Unicode fonts. But some pairs of fonts with less similarity were clustered differently in C_1 and C_2. In order to improve the result, the optimal weighting method in [41] was applied to the results of PANOSE-A numbers in Table 6. We obtained the result shown in Table 17.

Table 17: Optimal weights attached to each digit of PANOSE-A number

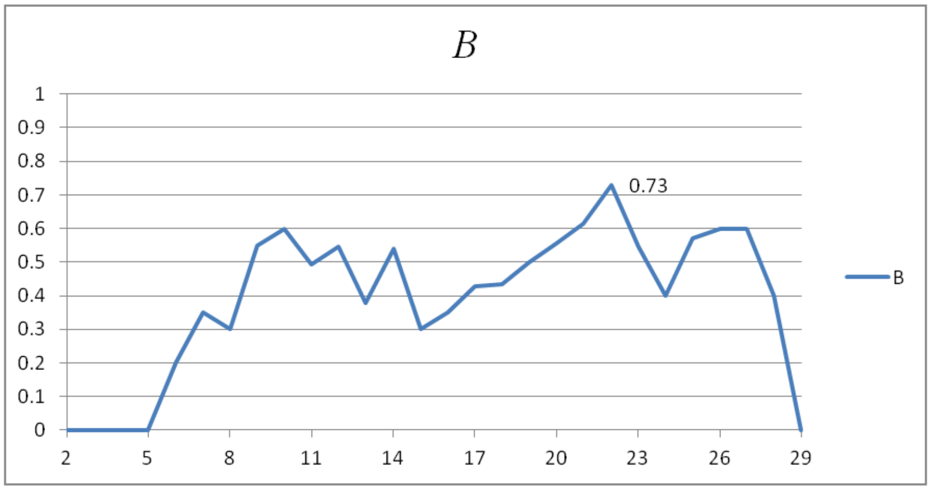
Digit ID	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8
Weight	0.201941	0.063160	0.203197	0.009275	0.025625	0.175746	0.219490	0.101566

The similarity matrix then was recalculated by using (58) with the weights in Table 17. A sample of the result similarity matrix is listed in Table 18. Then a third clustering process was conducted, denoted as C_3, using the new similarity matrix. The measurement B was recalculated between clusters of C_1 and C_3. The plot of B for all stages is shown in Figure 65-b.

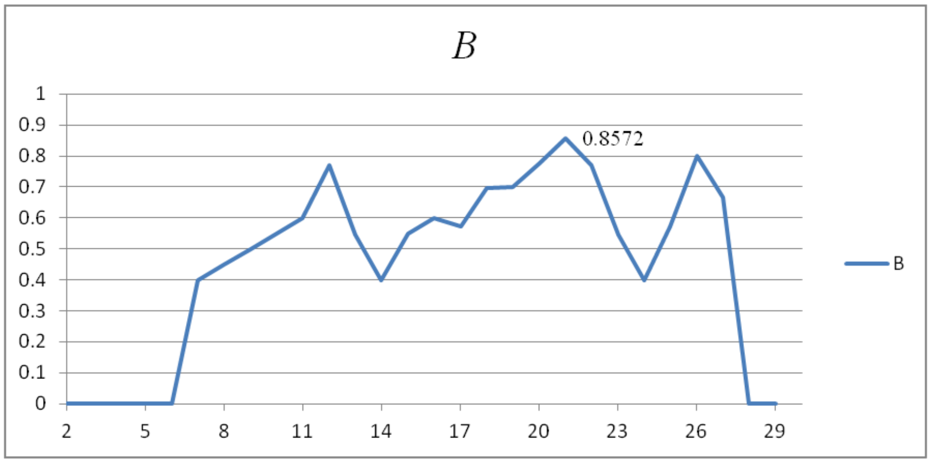
Table 18: Sample of similarity matrix after weighting digits

	Lotus Linotype Light	**MagedLT**	**MitraLTLight**	**PakType Tehreer**	**Qadi Linotype**	**Sakkal Majalla**	**Simplified Arabic Fixed**	**SimplifiedArabic**	**Tahoma**	**Tanseek Modern Pro Arabic Medium**	**TimesNewRoman**	**TraditionalArabic**	**XB Tabriz**	**Yakout Linotype Light**
Lotus Linotype Light	100	99.5	98.8	99.3	99.3	99.0	99.6	99.0	99.0	98.0	99.0	99.2	99.1	99.3
MagedLT	99.5	100	98.9	99.2	99.4	99.1	99.3	99.1	99.0	97.8	99.1	99.1	99.0	99.4
MitraLTLight	98.8	98.9	100	98.8	98.6	99.3	98.9	99.3	99.1	98.2	99.3	99.1	99.2	98.9
PakType Tehreer	99.3	99.2	98.8	100	99.2	99.0	99.5	99.0	98.8	98.1	99.0	99.0	98.9	99.4
Qadi Linotype	99.3	99.4	98.6	99.2	100	99.0	99.1	99.0	98.9	97.7	99.0	98.9	98.7	99.4
Sakkal Majalla	99.0	99.1	99.3	99.0	99.0	100	99.1	100	98.9	98.2	100	99.4	99.2	99.2
Simplified Arabic Fixed	99.6	99.3	98.9	99.5	99.1	99.1	100	99.1	98.9	98.3	99.1	99.1	99.0	99.4
SimplifiedArabic	99.0	99.1	99.3	99.0	99.0	100	99.1	100	98.9	98.2	100	99.4	99.2	99.2
Tahoma	99.0	99.0	99.1	98.8	98.9	98.9	98.9	98.9	100.	98.1	98.9	98.9	98.8	98.9
Tanseek Modern Pro Arabic Medium	98.0	97.8	98.2	98.1	97.7	98.2	98.3	98.2	98.1	100	98.2	98.1	98.0	97.9
TimesNewRoman	99.0	99.1	99.3	99.0	99.0	100	99.1	100	98.9	98.2	100	99.4	99.2	99.2
TraditionalArabic	99.2	99.1	99.1	99.0	98.9	99.4	99.1	99.4	98.9	98.1	99.4	100	99.6	99.0
XB Tabriz	99.1	99.0	99.2	98.9	98.7	99.2	99.0	99.2	98.8	98.0	99.2	99.6	100	98.9
Yakout Linotype Light	99.3	99.4	98.9	99.4	99.4	99.2	99.4	99.2	98.9	97.9	99.2	99.0	98.9	100

The highest value of $B_{21}=0.85$ was found in stage 10. The Silhouette coefficients S also were evaluated to check the coherence of the clusters at this stage for C_1 and C_3. The result were S_{C1}=93.11 while S_{C3}=96.27. The dendograms of C_3 is shown in Figure 68. The result of B with C_3 were better than C_2 although there is drop in S values. Again, pairs of font with very high similarity were grouped in the same manner but fonts with lower similarity were clustered differently. For example, the font PakType Tehreer, was clustered differently in C_1, C_2 and C_3 although it has a distinctive appearance. The features defined in PANOSE-A are not adequate to distinguish that difference. The system still has room for improvement. More features are needed to give a font a more distinctive number.



(a) Without digits weights



(b) With weighted digits

Figure 65: B measure for all clustering stages

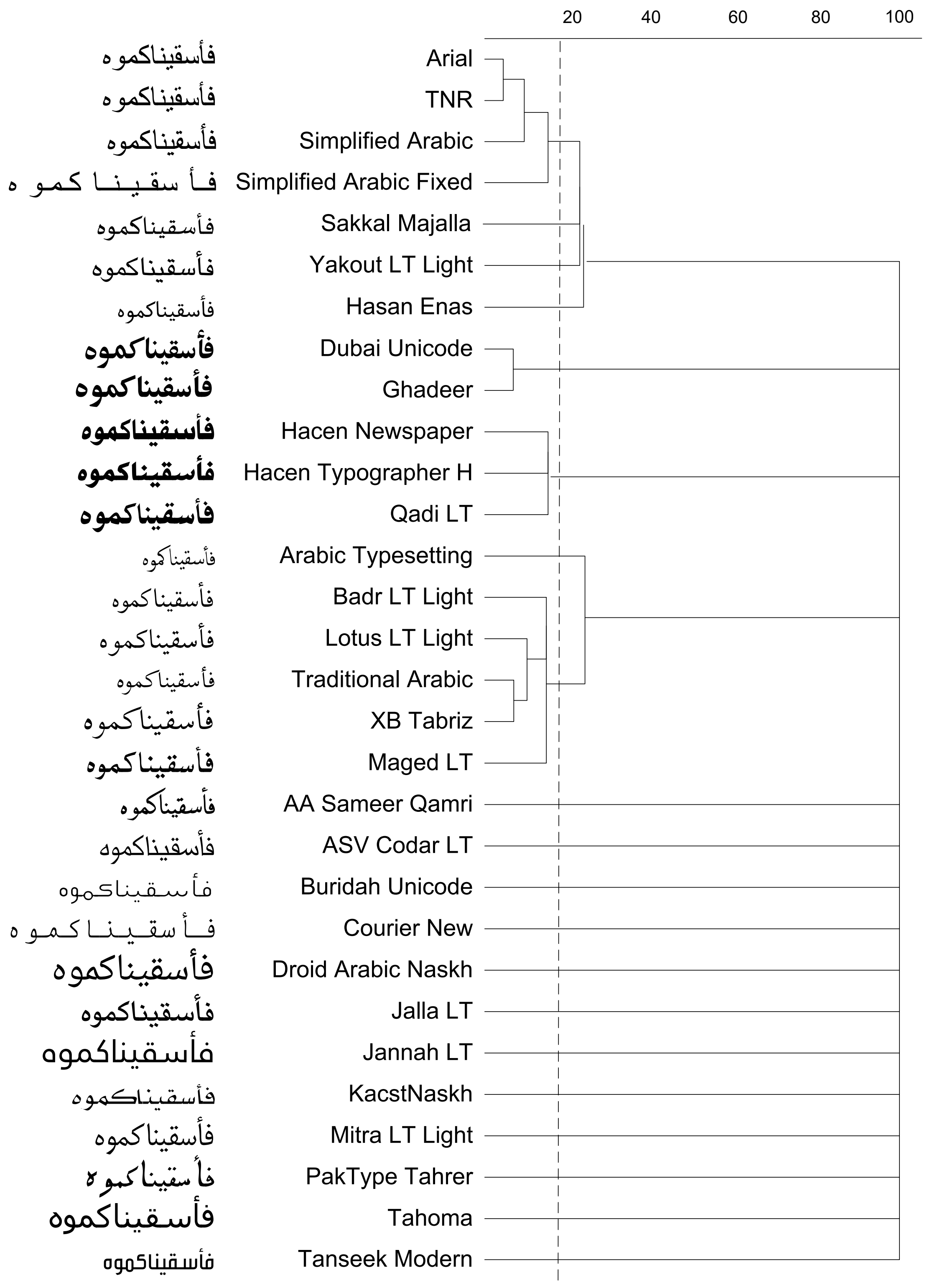

20
40
60
80
100
فأسقيناكموه Arial
فأسقيناكموه TNR
فأسقيناكموه Simplified Arabic
فـأ سقـيـنـا كـمـو ه Simplified Arabic Fixed
فأسقيناكموه Sakkal Majalla
فأسقيناكموه Yakout LT Light
فأسقيناكموه Hasan Enas
فأسقيناكموه Dubai Unicode
فأسقيناكموه Ghadeer
فأسقيناكموه Hacen Newspaper
فأسقيناكموه Hacen Typographer H
فأسقيناكموه Qadi LT
فأسقيناكموه Arabic Typesetting
فأسقيناكموه Badr LT Light
فأسقيناكموه Lotus LT Light
فأسقيناكموه Traditional Arabic
فأسقيناكموه XB Tabriz
فأسقيناكموه Maged LT
فأسقيناكموه AA Sameer Qamri
فأسقيناكموه ASV Codar LT
فأسقيناكموه Buridah Unicode
فـأ سقـيـنـا كـمـو ه Courier New
فأسقيناكموه Droid Arabic Naskh
فأسقيناكموه Jalla LT
فأسقيناكموه Jannah LT
فأسقيناكموه KacstNaskh
فأسقيناكموه Mitra LT Light
فأسقيناكموه PakType Tahrer
فأسقيناكموه Tahoma
فأسقيناكموه Tanseek Modern

Figure 66: Dendrogram of clustering with Find My Font similarity

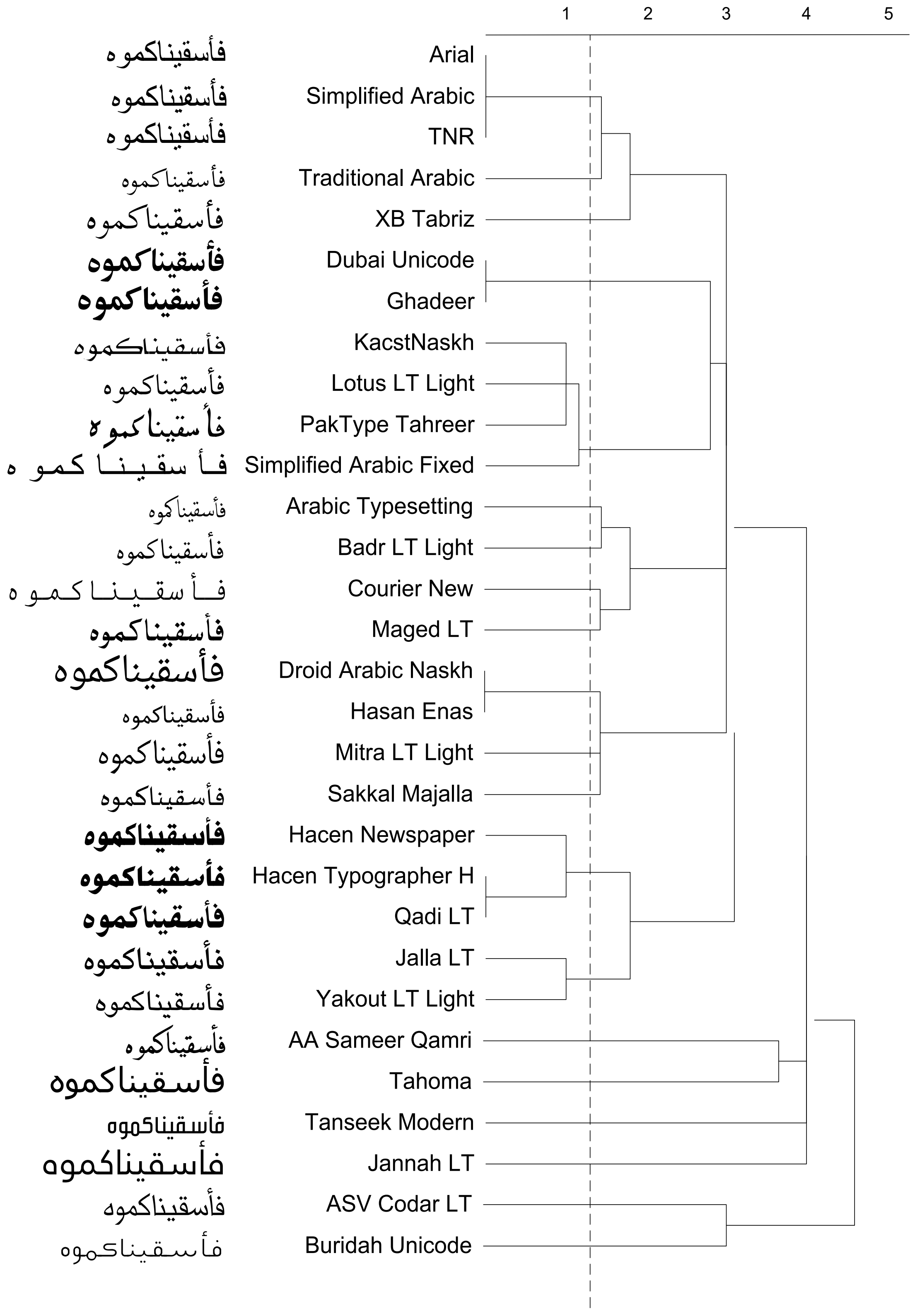


Figure 67: Dendrogram of clustering with PANOSE-A similarity matrix without weighting

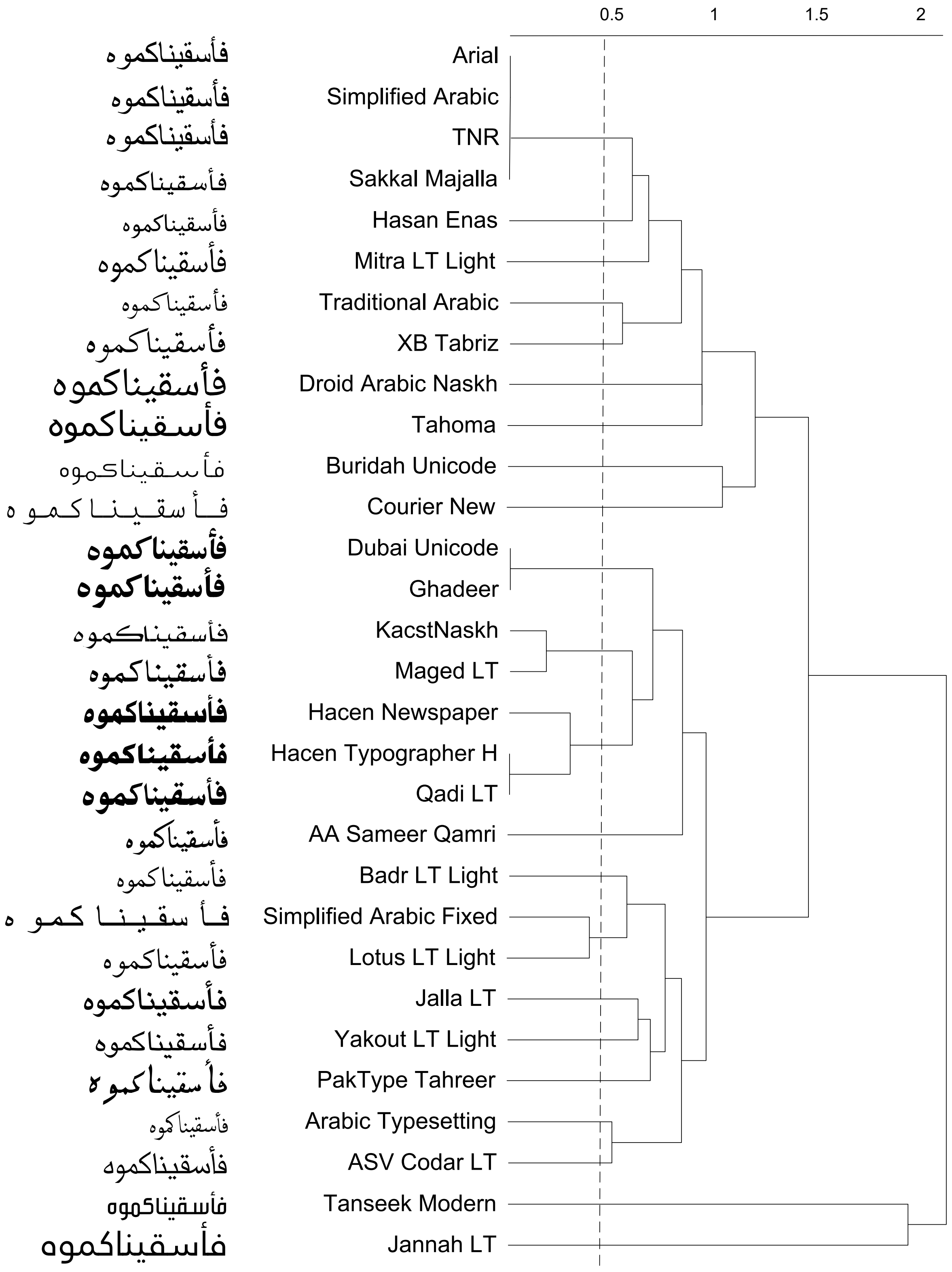


Figure 68: Dendrogram of clustering with PANOSE-A similarity matrix after digits weighting

Chapter 6 : Conclusion

In this research, a classification model for Arabic digital fonts is defined. The classification is obtained from Arabic font design characteristics. The model was designed as an extension of the existing PANOSE-1 which is a computerized classification system for Latin script. Hence, the proposed model has been denoted as PANOSE-A. The PANOSE system in general encodes a font's design characteristics into a number composed of several digits. Each digit represents a design characteristic where each value a digit can take indicates a variation of this characteristic. Assigning a specific value for a digit is based on specific measurements that should describe the represented characteristics by this digit.

Representing a font's design characteristics within a comparable data structure, like the PANOSE-1 number, is beneficial in many ways. When designing a document, searching and selecting an appropriate font are mostly based on the font's look. Having a description of a font's appearance as a number facilitates the searching process. Another important use of encoding fonts' design characteristics into a number is the ability to find the most suitable replacements for missing fonts in foreign documents. This helps avoid losing information or changing the document layout, which could happen as a result of bad font substitution. Documents that use an Arabic script font could be exposed to these kinds of problems as no work has been done before to encode Arabic fonts' design characteristics.

PANOSE-A defines eight design characteristics for Arabic Naskh style and text fonts. Some of these characteristics are taken from Arabic font design characteristics used by most Arabic font designers while some other characteristics are inspired from calligraphy rules for Naskh style. The selection of these characteristics and the measurements that describe them were based on several discussions with four Arabic font designers about emphasizing or excluding these characteristics. The basic design characteristics, described in PANOSE-A, are the weight, the contrast and the loop and tooth heights. These characteristics are essential for any Arabic font design process as mentioned in [3], [4] and [19]. The description of bowl and curved tail strokes are defined based on common rules for Naskh calligraphy as in [11] and [12]. Two other characteristics, which describe the shape of terminal parts of the terminal stroke and the shape of closed round strokes, are defined based on the possible variation it may take in text fonts. Some of the measurements and attributes are used to describe these two characteristics are mentioned in [19] [27], and [29].

Since there is no ground truth data of how Arabic fonts can be classified, the efficiency of the model was tested in different ways. First, PANOSE-A digits were evaluated for a set of 30 fonts. Then, two hierarchical clustering processes were conducted to cluster that set of fonts into groups based on their similarity. The two clustering processes used different similarity matrixes. One matrix was constructed based on the similarity result from the font matching tool, called *Find My Font*. The second similarity matrix was constructed based on the Euclidian distance between PANOSE-A numbers assigned to represent each font in the set. The validity of the resulting clusters, from each clustering process, was evaluated using the Silhouette coefficient as described in [39]. The similarity between the two clustering processes was evaluated by the method in [40].

The PANOSE-A number is composed of eight digits. At first, the similarity matrix based on PANOSE-A numbers was constructed, where each digit contributes equally to estimate the similarity between each pair of fonts. The clustering similarity evaluation*,* between clustering using *Find My Font* and clustering using un-weighted PANOSE digits -A, factor B_{22} had a value of 73%. The pairs of fonts that had high similarity were identically classified. These pairs of fonts also had distinctive variation of characteristics that made them more prominent than others. On the other hand, pairs of fonts with lower similarity were clustered differently.

In order to improve the efficiency of PANOSE-A, another clustering process for the same set of fonts was conducted after weighting PANOSE-A digits optimally using the method in [41]. By this way, the most distinctive characteristics contribute more to evaluating the similarity between fonts. When the resulting clusters of this clustering process was compared to those resulting from clustering based on the *Find My Font* similarity matrix, the highest value of B was 85 with 21 clusters. Although weighting the digits enhanced PANOSE-A efficiency, there is a need for further improvement.

PANOSE-A is new model that can be improved in several ways. There is still one more digit that is undefined, so the whole PANOSE-A number will be composed of ten digits as in PANOSE-1. More variations for each digit can be defined to get more specific classes. For example, weight classes can be expanded to describe more variation of weights such as, Thin, Extra Thin, Black and Extra Black. In addition, to improve the model efficiency the least distinctive characteristics can be replaced with more distinctive ones. During the discussions with the group of designers, there were many other individually suggested characteristics, such as the shape of a point and its position with regard to the letter glyphs. Other suggested features are related to the raise or slope, of the horizontal stroke in a glyph body, with regards to the baseline. Another way of development can be done through analyzing and testing a larger set of fonts. PANOSE-A digits were defined based on the possible available variations. It is possible to create additional variations using synthetic font samples. These samples can be generated using morphing techniques.

BIBLIOGRAPHY

[1] F. Oweis, Pocket Guide to Arabic Script, New York: Hippocrene Books INC, 2007.

[2] S. Hussain, "Reading and Writing the Arabic Script," Arabic-Studi.com, 2009.

[3] H. S. Abifares, Arabic Typography: A Comprehensive Sourcebook, Beirut: Saqi Books, 2000.

[4] E. Smitshuijzen, Arabic Font Specimen Book, Amsterdam: De Buitenkant, 2009.

[5] S. Ager, "omniglot, online encyclopedia for writing systems & languages," Kualo, [Online]. Available: http://www.omniglot.com/. [Accessed 5 10 2012].

[6] M. Hssini and A. Lazrek, "Design of Arabic Diacritical Marks," *IJCSI International Journal of Computer Science,* vol. 8, no. 3, pp. 1-10, 2011.

[7] C. Y. Suen, S. Nikfal, B. Zhang, J. Janbi and N. Dumont, "Personality Traits of Typefaces for English, Chinese, and Arabic," in *Proceedings of ATypeI Conference*, Hong Kong, October 2012.

[8] M. J. E. Benatia, M. Elyaakoubi and A. Lazrek, "Arabic text justification," in *Proceedings of the 2006 Annual Meeting,TUGboat*, Volume 27, No 2, 2006.

[9] A. Azmi and A. Alsaiari, "Arabic Typography: A Survey," *International Journal of Electrical & Computer Sciences IJECS-IJENS,* vol. 9, no. 10, pp. 5-10, 2009.

[10] R. William, The Non-Designer's Design and Type Books, Deluxe Edition, Berkeley: Peachip Press, 2008.

[11] M. Mahmoud, علم نفسك الخطوط العربية, Cairo: Ibn Sinaa, 2005.

[12] M. Ja'far and V. Porter, Arabic Calligraphy: Naskh Script for Beginners, London: British Museum Press, 2002.

[13] M. Gelderman, "A short introduction to font characteristics," *TUGboat,* vol. 20, no. No.2, 1999.

[14] K. C.-h. Tam, "Digital typography : a primer," 6 2003. [Online]. Available: http://keithtam.net/documents/keithtam_digital_typr_primer.pdf. [Accessed 12 5 2011].

[15] Y. Haralambous, Fonts & Encodings, O'Reilly Media, 2007.

[16] R. Rubinstein, Digital Typography: An Introduction to Type and Composition for Computer System Design, USA & Canada: Addison Welsley , 1988.

[17] M. C. Dyson and C. Y. Suen, Digital Fonts and Reading, Singapore-London-New Jersey: World Scientific Publishing, 2016.

[18] P. Zoghbi, "29 Letters Blog," 30 7 2015. [Online]. Available: https://blog.29lt.com/2015/07/30/arabic-type-anatomy-typographic-terms/. [Accessed 23 10 2015].

[19] H. S. Abifares, Arabic Type Design for Beginners, an Illustrated Guidebook, Amsterdam & Dubai: Khatt Books & Tashkeel, 2013.

[20] S. Boeuf, Arabic Font Production Tutorial: Part I Typographic Fonts, Amsterdam: Khatt Books, 2011.

[21] M. Group, "IBM Font Class Parameters," Microsoft, 20 February 2008. [Online]. Available: http://www.microsoft.com/typography/otspec/ibmfc.htm. [Accessed 20 12 2012].

[22] B. Bauermeister, A manual of comparative typography: the PANOSE system, New York: Van Nostrand Reinhold, 1988.

[23] "TrueType Explorer User's Guide," [Online]. Available: http://www3.sympatico.ca/chris.lamoureux2/tte.html. [Accessed 20 Jan 2013].

[24] M. S. D. Laurentis, B. P. Bauermeister, R. G. Beausoleil, N. Gregg Brown and C. D. McQueen III, "PANOSE 2.0 White Paper," 16 December 1993. [Online]. Available: http://www.w3.org/Fonts/Panose/pan2.html. [Accessed 20 11 2012].

[25] C. D. MCQUEEN III and R. G. BEAUSOLEIL, "Infinifont: a parametric font generation system," *ELECTRONIC PUBLISHING-CHICHESTER- ,* vol. 6, no. 3, pp. 117-117 , 1993.

[26] H.-P. Corporation, "PANOSE Classification MEtrics Guide," Monotype Imaging, 14 2 1997. [Online]. Available: http://www.monotypeimaging.com/ProductsServices/pan1.aspx. [Accessed 2 10 2012].

[27] H. S. Abifares, Typographic matchmaking, Amsterdam: BIS Publishers & Khatt Foundation, 2007.

[28] J. Janbi, M. Almuhajri and C. Y. Suen, "The Effect of Fonts Design Features on OCR for

Latin and Arabic," *Journal of Multimedia Theory and Application,* vol. 1, pp. 50-59, 2014.

[29] S. Rege, R. Memane, M. Phatak and P. Agarwal, "2D Geometric Shape and Color Recognition Using Digital Image Processing," *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering IJAREEIE,* vol. 2, no. 6, pp. 2479-2487, June 2013.

[30] Arafonts, "الموسوعة المجانية للخط العربي," Microteque, [Online]. Available: http://www.arafonts.com/default.aspx?fontCategory=Naskh. [Accessed 12 4 2013].

[31] "UrduLog Font Showcase," Jameel Khat Numa, [Online]. Available: http://font.urdulog.com/. [Accessed 15 9 2013].

[32] S. Developer, "Find My Font," Softonium, 2009-2014. [Online]. Available: http://www.findmyfont.com/. [Accessed 2014].

[33] A. AL-Shatnawi and K. Omar, "Methods of Arabic Language Baseline Detection – The State of Art," *IJCSNS International Journal of Computer Science and Network Security,* vol. 8, no. 10, pp. 137-143, 2008.

[34] C. R. Maurer, R. Qi and V. Raghavan, "A Linear Time Algorithm for Computing Exact Euclidean Distance Transforms of Binary Images in Arbitrary Dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 25, no. 2, pp. 265-270, 2003.

[35] D. Chetverikov and Z. Szabo, "A simple and efficient algorithm for detection of high curvature points in planar curves," in *Lecture Note in Computer Science*, Berlin, Germany, Springer-Verlag Berlin Heidelberg, 2003, pp. 746-753.

[36] O. Maimon and L. Rokach, "Clustering Methods," in *Data Mining and Knowledge Discovery Handbook*, Springer US, 2005, pp. 321-352.

[37] P.-N. Tan, M. Steinbach and V. Kumar, "Cluster Analysis: Basic Concepts and Algorithms," in *Introduction to Data Mining*, Addison-Wesley, 2006, pp. 487-559.

[38] J. D. Ullman and A. Rajaraman, "Clustering," in *Mining of Massive Datasets*, New York, Cambridge University Press, 2012, pp. 241-280.

[39] R. J. Peter, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics,* vol. 20, pp. 53-65, 1987.

[40] E. B. Fowlkes and C. L. Mallows, "A Method for Comparing Two Hierarchical Clustrings,"

JSTOR with American Statistical Association, vol. 78, no. 383, pp. 553-569, 1983.

[41] G. De Doete, "Optimal variable weighting for ultrametric and additive tree clustering," *Quality and Quantity,* vol. 20, no. 2-3, pp. 169-180, 1986.

[42] M. Powell, "Restart Procesures for the Conjugate Gradient Method," *Mathematical Programming,* vol. 12, no. 1, pp. 241-254, 1977.

[43] G. Brown and K. Woods, "Born Broken: Fonts and Information Loss in Legacy Digital Documents," *The International Journal of Digital Curation,* vol. 6, no. 1, pp. 5-19, 2011.

APPENDIX

List of sample of Arabic Fonts

Table 19: Sample of 30 fonts used to evaluate PANOSE-A

Font Name	Sample
AA Sameer Qamri	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
ASVCodarLTLight	ق ف ب ن ک ا ح د ر س ص ع ط ل م ه و ى
ArabicTypeseting	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Arial	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
BadrLTLight	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
BuridahUnicode	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Courier New	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Droid Arabic Naskh	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Dubai Unicode	**ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى**
Ghadeer	**ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى**
Hacen Newspaper	**ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى**
HacenTypographyHeavy	**ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى**
Hasan Enas	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
JalalLTRegular	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
JannaLTRegular	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
KacstNaskh	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Lotus Linotype Light	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
MagedLT	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
MitraLTLight	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
PakType Tehreer	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Qadi Linotype	**ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى**
Sakkal Majalla	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Simplified Arabic Fixed	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
SimplifiedArabic	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Tahoma	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Tanseek Modern Pro Arabic Medium	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
TimesNewRoman	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
TraditionalArabic	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
XB Tabriz	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى
Yakout Linotype Light	ق ف ب ن ك ا ح د ر س ص ع ط ل م ه و ى