جامعة السودان للعلوم والتكنولوجيا

كلية علوم الحاسوب وتقانة المعلومات

تمييز الكلام العربي باستخدام الشبكات العصوبيه الإصطناعيه

سبتمبر/2006

مشروع مقدم كأحد متطلبات الحصول على بكالوريوس الشرف في علوم الحاسوب.

بسم الله الرحمن الرحيم

جامعة السودان للعلوم والتكنولوجيا

كلية علوم الحاسوب وتقانة المعلومات

تمييز الكلام العربي بإستخدام الشبكات العصوبيه الإصطناعيه

سبتمبر/2006

إعداد الطلاب :

1-بركات إبراهيم أحمد

2-مصطفى الزين حسن

3-طلال بهاء الدين سيد أحمد

مشروع مقدم كأحد متطلبات الحصول على بكالوريوس الشرف في علوم الحاسوب

توقيع الأستاذ المشرف ..... التاريخ

... د.إيمان أبو المعالي عبد الرحمن.. ..... /سبتمبر/2006

الحمد لله

الحمد لله الذي أحلنا محلة الفهم وحلانا حلية العلم وملكنا عقال العقل وزيننا بنطق المنطق

ونعوذ به من كدر صفاء الفكر وعكر ذهن الذهن و

الحمد لله خالق الألسن واللغات واضع الألفاظ للمعاني بحسب ما اقتضته حكمه البالغات

الذي علم آدم الأسماء كلها وأظهر بذلك شرف اللغة وفضلها

الإهداء

إلى أمهاتنا
لبنى بركات الحبر
هدى محمد الخواض
إرشاد محمد البشير

و آبائنا
إبراهيم أحمد الشيخ
الزين حسن الأمين
بهاء الدين سيد أحمد الحسن

شكر واعتراف

نتقدم بالشكر لكل من مد لنا يد العون فى إخراج هذا البحث ، فالشكر أجزله للأساتذة الكرام :
د.ايمان أبو المعالي عبدالرحمن- قسم الهندسة الكهربائية والألكترونية- جامعة الخرطوم الذي كان المشرف القدير على نجاح المشروع .
أ.محمد جمال الدين- أستاذ مبادئ الادارة والاتصالات البشرية- الذي كان له الدور الكبير في اخراج هذا البحث .
و لأسرة مدرسة المقرن أساس للبنات لتمكيننا من تسجيل عينات صوتية يحتاجها البحث .
و لأسرة مدرسة علي عبد اللطيف بنين لتمكيننا من تسجيل عينات صوتية يحتاجها البحث .

المستخلص

يحرم الغالبية ممن فقدوا أيديهم او بصرهم من استخدام جهاز الحاسوب ، و بإضافة وسيلة إدخال صوتية تتطلب إضافة ميكروفون لجهاز الحاسوب نكون قد ساعدناهم و سهلنا على غيرهم استخدامه .

وبجهد متواضع أنجزت مكتبة برمجية بسيطة تتيح للمبرمج إضافة خاصية تمييز الكلام لبرنامجه بسهولة ، و التمتع ببرنامج سهل و جذاب للمستخدمين النهائيين ، أتت المكتبة مراعية التجريد في التصميم قدر المستطاع ، وذلك بعدم إجبار المبرمج على إعدادات صوتية ثابتة أو تمييز لكلام محدد . وقد وصفت المكتبة بمخططات (UML) القياسية لكي لا يكون على المبرمج أدنى شك في فئة أو دالة . كما أن التحليل الذي يسبق كل مخطط للفئات سييسر على المبرمج التعديل مما يوسع المكتبة و يزيدها قوة وهذا ما نأمله ، إستخدام المكتبة و القضايا التنفيذية المتعلقة بها كان إهتمامنا لكي لا تهجر بسبب فقر التوثيق .

اعتمدت المكتبة على لغة (Visual C++ 2005) في تطبيقها ، و على مكتبة (PortAudio) في التعامل مع الصوت و مكتبة (FANN) في تطبيق الشبكات العصوبية ، وكلى المكتبتين مفتوحتى المصدر ، لكي نضمن نظاماً خالياً من الصناديق السوداء .

أعطت المكتبة دقة في تمييز الكلام وهي (88%) عندما عرّضت لقاموس حجمه 10 مفردات عربية .

ABSTRSCT

The most of there loss our hands or eyes deprived from using the computer , with add acoustic input way for computer we can help our and make using of computer easy .

With humble work we complete simple programming library to make programmer able to add speech recognition feature for his programme , and enjoy with easy and attractive for end-user , the library provided the abstract in it design , that done with make sound setting changeable and don't specific to const speech , the description of library done with (UML) standard because to make sure to programmer for all classes and function . before any class diagram programmer will found analyse to make modification easy to extending the library , use of library and implementation issues it take our care because programmers leave the library if their founded less in documentation .

The library based on (Visual C++ 2005) for implementation , and (PortAudio) library for sound programming , and (FANN) library for neural network implementation , the two library its open source to provide black boxes .

In testing library with dictionary have 10 arabic words , the accuracy of result (88%) .

شرح المصطلحات

ASR	**Automatic Speech Recognition(نظم تمييز الكلام)**	**تقنية تتيح إدخال الأوامر للحاسوب صوتياً (مجال البحث والمشروع).**
phonetic	.....	**خواص جميع الأصوات الواردة في الكلمات .**
phonological	.....	**القواعد التي تحكم اختلاف نطق الأصوات حسب السياق .**
morphemic	.....	**القواعد الصرفية مثل تصريف الأفعال .**
Prosodic	.....	**تصف الاختلاف في النبر و التنغيم .**
Syntactic	.....	**القواعد التي تحكم تكوين العبارات والجمل .**
semantic	.....	**القواعد التي تحكم استبعاد الكلمات أو الجمل التي قد تكون صحيحة نحوياً ولكن غير محتملة الورود .**
pragmatic	.....	**القواعد التي تمكن السامع من استنتاج نوايا المتكلم .**
Utterance(النطق)	.....	**الكلمة أو الكلمات التي تُمَثِّل كمعنى وحيد للحاسوب، ويمكن أن يكون كلمة أو عدة كلمات أو جملة أو حتى عدة جمل .**
Speaker Dependance(اعتمادية المتكلم)	.....	**هل نظام تمييز الكلام معتمد على متكلم واحد ام عدة متكلميين .**
Vocabularies(المفردات)	.....	**قوائم الكلمات أو النطق التي سوف يميزها النظام .**
LPC	**Linear Predictive Coding (التشفير التنبؤية الخطية)**	**تقنية تستخدم في تشفير الكلام .**
HMM	**Hidden Markov Models (نماذج ماركوف الخفية)**	**تقنية تستخدم في تمييز الكلام ، تعتمد على نماذج إحصائية .**
(A/D)	.....	**محول موجود في بطاقة الصوت يحول الإشارة التماثلية إلى رقمية.**

Sample (العينة)	.....	كلمة من البتات (8،16،24،32) تأخذ من الإشارة الصوتية . يمثل أصغر تمثيل حوسبي لموجة إشارة الصوت
sample rate (تردد العينة)	.....	عدد العينات في الثانية .
bits per sample(عدد البتات في العية)	.....	عدد البتات بالعينة .
DTW	Dynamic Time Warping (تطويع الزمن الديناميكي)	تقنية مستخدمة في تمييز الكلام ، تعتمد على مبدأ معالجة التغيرات في محور الزمن المسببة بإختلاف معدل النطق .
VQ	Vector Quantization (تكمية المتجه)	تقنية لضغط الكلام تعتمد على مبدأ البحث عن شفرة بديلة للمتجه تمثله في أبسط صوره .
use case diagram (مخطط حالة الإستخدام)	.....	مخطط يصف النظام او النظام الجزئي في صورة فعل و فاعل .
الفئة (class)	.....	شكل يصف الكائن بسرد صفاته و سلوكه .
class diagram (مخطط الفئات)	.....	يصف الفئات في صورة قياسية .
Frame (إطار)	.....	مخزن تتجمع فيه العينات .
Buffer (إرجاء)	.....	مكان تخزن فيه البيانات إلى أن يتم معالجتها .
sound buffer (إرجاء الصوت)	.....	مخزن تتجمع فيه الإطارات .
API	Application Programming Interface(واجهة برمجة التطبيق)	مجموعة من الدوال وضعت في شكل مكتبة لتسهيل بناء تطبيق ما .
Multimedia (الوسائل المتعددة)	.....	الوسائل التي تهيئ للتعامل مع الصوت و الصورة و الفيديو ، من ملفات و مشغلات و غيرها .
chunk	.....	مجموعة من البيانات توصف بنوع محدد في بنية ملفات الوسائل المتعددة .
ANN	Artificial Neural Networks (الشبكات العصوبيه الإصطناعية)	تقنيات تحاول محاكاة عمل الجهاز العصبي المركزي عند الإنسان .

FANN	Fast Artificial Neural Network	مكتبة برمجية مكتوبة بلغة (C) تختص بتطبيق الشبكات العصوبيه الإصطناعية .
Wikpedia	http://ar.Wikpedia.org/wiki	موسعة علمية ضخمة متعددة اللغات .
mat	http://www.mat.ucsb.edu	دروس في برمجة الصوت الرقمي .
lightlink	http://www.lightlink.com/tjweber/StripWav/WAVE.html	واصف لهيئة ملفات (.wav) .
number of channels (عدد القنوات السمعية)	.....	عدد القنوات التي تتعامل بها بطاقة الصوت في إدخاله و إخراجه .
db	decibel	مقياس لعلو الصوت .
Microsoft's RIFF	.....	مواصفات شركة مايكروسوفت في ما يتعلق بالوسائل المتعددة .
UML	Unified Markup Language	لغة قياسية تستخدم في وصف الكائنات و علاقاتها .
end point detection	.....	تحديد النطق وذلك بقص أطرافه .
dspexperts	http://www.dspexperts./com/dsp/projects/lpc/	واصف لتقنية (lpc) .
DataCompression	http://www.data-compression.com/vq.shtml	واصف لتقنية (VQ) .

فهرس الأشكال

فهرس الجداول

فهرس المحتويات

المقدمة

سهولة الاستخدام هاجس يؤرق مصممي البرامج و المستخدمين على حدا سواء ففي الوقت الذي يبذل المبرمج جل طاقه في تطوير برامجه يبتعد الكثير عن إستخدامها بسبب جمود النظام و قلة تفاعله مع المستخدم ، إن تذويد البرامج بإمكانية التعرف على الكلام سيسهل بالتأكيد استخدامها و يجذب عدد أكبر من المستخدمين .

تدخل نظم تمييز الكلام كمؤثر فعال في تطوير البرامج التعليمية و الترفيهية ، و من المتوقع ظهور أنظمة قوية تعتمد اعتماد أساسي عيها مثل أنظمة الترجمة الفورية .

أضفت نظم تمييز الكلام الكثير على البرمجيات فماذا عن العتاديات و الأجهزة الحوسبية ، تدخل مثل هذه النظم في تطوير الإنسان الآلي (الروبوت الذكي) و تساعد في تطوير بطاقات الصوت و صناعة الميكرفون .

سلطت تلك النظم الضوء على جوانب شتى من العلوم كانت مهملة لوقت قريب و من المتوقع أن تساعد دراستها على فهم بعض الخفايا المتعلقة بعلوم اللغة و الطب و الفيزياء إلى آخره .

من جانب آخر تلقي أنظمة تمييز الكلام ظلالها على نواحي إنسانية وتحاول مساعدة ذوي الاحتياجات الخاصة في مواكبة العصر وذلك بتوفير الوسائل و الأدوات ذات طبيعة الإدخال الصوتي في تشغيل البرامج و تصفح الإنترنت و العديد من التطبيقات الأخرى .

مما سبق و أكثر يتضح أن الحاجة لمثل هذه النظم ملحه تستحق منا بذل الجهد في سبيل الوصول إلى حل مقبول يمكن الاعتماد عليه في الإدخال كما نعتمد على لوحة المفاتيح الآن . لذا سعي لتنفيذ برنامج حاسوبي يستطيع التعرف على كلمات و أصوات عربية ، وقد تحقق هذا بنسبة (88%) وهي دقة النظام ، وقد إستخدمت الأدوات التالية في تحقيقه :

- الحاسوب وهو ذا معالج بقوة (1.8GHz) وذاكرة سعة (512MB) .
- بطاقة صوت تصل دقتها إلى (24bit) .
- ميكرفون من النوع (USB) .
- لغة البرمجة و هي (Microsoft Visual C++ 2005) .
- مكتبة برمجية للتعامل مع بطاقة الصوت و هي (PortAudio) .
- مكتبة برمجية في تطبيق الشبكات العصوبية الإصطناعية وهي (FANN) .
- أداة لتحيل الكائنات و رسم المخططات وهي (Visual Paradigm for UML 5.3) .

و الأبواب التالية في وصفه :

الباب الأول يوضح ما هي نظم تمييز الكلام و أنواعها و التقنيات المستخدمة فيها ... ، كما يوضح طريقة تحليل النظام .

الباب الثاني يعطي نظرة عامة عن كيفية برمجة الصوت الرقمي ، وكيفية تزويد الدخل و الخرج للنظام .

الباب الثالث يصف تقنيات استخدمت في النظام بغرض تصغير كمية البيانات التي يتعامل معها .

الباب الرابع يوضح ما هي الشبكات العصوبية و أنواعها و خصائصها ... ، طريقة دمجها في النظام .

الباب الخامس يبين كيف تم تصميم و تنفيذ و إختبار النظام .

الباب الأول(3-13)

أساسيات تمييز الكلام

1.1 مقدمة

إستمر برنامج مموّل من قبل وكالة مشروع الأبحاث المتقدمة بوزارة الدفاع في الولايات المتحدة الأمريكية لمدة خمس سنوات من (1971-1976) للبحث في التعرف على الكلام المتصل ، كان هدف البحث هو إنتاج برنامج حوسبي له القدرة على تحليل الجمل صحيحة البناء من واقع قاموس لغوي يضم حوالى ألف كلمة وتنحصر الجمل في مجالات محدودة وعلى الآ تتعدى نسبة الخطأ عشرة بالمائة. وقد نتج عن هذا المشروع عدد من البرامج مثل (speechlis) لشركة (Bolt Beranek) ، كما قادت جهود مماثلة في فرنسا إلى بناء مجموعة أخرى *(Alain Bonnet 1985)*.

ساهمت هذه البحوث بشكل أو بآخر في تحديد أساسيات أو قواعد لمطوري برامج تمييز الكلام بمختلف أنواعه . سيناقش في هذا الباب العديد من المفاهيم المتعلقة بهذه الأساسيات مع التبسيط قدر المستطاع. وبما أن نظم تمييز الكلام تحتك بشكل كبير مع علوم أخرى فقد يكون من الصعب تحديد قواعدها الحقيقية التي من شأنها أن تؤدي إلي نتائج مرضية لذلك أكتفي بما توصلت إليه بعض البحوث في مجالي معالجة الإشارة وعلم الحاسوب و القليل من علم اللغة .

القائمة التالية تمثل كل العلوم المرتبطة بالتعرف على الكلام :

1. معالجة الإشارة.
2. الإلكترونيات.
3. علم الحاسوب .
4. الرياضيات و الإحصاء.
5. علم النفس.
6. علم اللغة.
7. علم الأصوات.

قد تختلف هذه القائمة من بحث لآخر إختلافاً طفيفا *(Dr Philip Jackson 2005)* .

من الناحية اللغوية توجد مستويات عدة للتعرف على الكلام ، عند الوصول إلى المستوى النهائي نكون قد وصلنا إلى غاية إربنا ، من التحقق الكامل لفهم الكلام من الناحية النظرية وهي كما أوردها *(Alain Bonnet 1985)* .

2.1 مستويات تمييز الكلام

إذا اعتبرنا الإشارة الصوتية نقطة البداية ، فإنه يتوقف تحقيق فهم تام لما قيل على توافر أنواع المعرفة التالية :

1. (phonetic): هي خواص جميع الأصوات الواردة في الكلمات.
2. (phonological): القواعد التي تحكم إختلاف نطق الأصوات حسب السياق مثل التفخيم و الإدغام و التنوين وغيرها.
3. (morphemic): القواعد الصرفية مثل تصريف الأفعال.
4. (prosodic): القواعد التي تصف الإختلاف في النبر و التنغيم مثل النبر المرتفع نهاية السؤال.
5. (syntactic): القواعد التي تحكم تكوين العبارات والجمل.
6. (semantic): طرق إستبعاد الكلمات أو الجمل التي قد تكون صحيحة نحوياً ولكن غير محتملة الورود.
7. (pragmatic): القواعد التي تمكن السامع من إستنتاج نوايا المتكلم و ذلك بأن يكون تفسيره للرسالة أكثر من مجرد التفسير السطحي للرسالة اللغوية (*Alain Bonnet 1985*).

3.1 مشاكل تمييز الكلام

تنشأ الصعوبة الكبرى في فهم الكلام من وجود مصدرين للخطأ ، يرجع أحد المصدرين إلي المتكلم و الآخر إلى السامع . وتحدث كثير من الأخطاء أثناء ترجمة المتكلم أفكاره إلى أصوات ، مثل إختيار الكلمات الخطأ ونطقها خطأ أو بوضوح غير كافي ، أو تكرار كلمات حين لا يكون ضرورة لذلك ، و إصدار أصوات غريبة لا معني لها مثل تسليك الحنجرة . وعلى السامع أن يقوم بعكس العملية التي قام بها المتكلم فهو يبدأ من الرسالة المشوهة إلى نوايا المتكلم ، ويرتكب أخطاء في الحكم ، لأنه لا توجد قواعد دقيقة تحكم الفهم ، وفيما يلي بعض مؤثرات الرسالة الصوتية *(Alain Bonnet 1985)* .

1. سيكولوجية المتكلم.
2. الدلالة و السياق.
3. التراكيب.
4. إعتبارات معجمية.
5. جهاز النطق عد المتكلم.
6. الضوضاء المحيطة.
7. الميكروفون.

والنتيجة هي أن المدخل لبرنامج تحليل الكلام هو نسيج من العناصر الصوتية تعبر عن الجملة المنطوقة في الأصل . وسيحتوي هذا النسيج على أخطاء لا تقل نسبتها عن 30% على أحسن الفروض ، وهو ما يؤدي إلى درجة كبيرة من عدم التحديد في المعالجة الآلية . يمكن الاستخلاص من هذا النسيج نسيج من الكلمات بحساب جميع التوافقات الممكنة بين الأصوات في هذا السياق وهنا يمكن أن تنحصر مهمة برنامج التحليل في أن يجد مسارا مترابطا منطقيا مستخدما جميع المعلومات الممكنة . وقد ثبت أن هذه العملية من أسفل إلى أعلى وافية للغات التي تزيد درجة تعقيدها عن الحد الأدنى و يجب أن تكمل بعملية معالجة من أعلى لأسفل ترشدها معلومات دلالية و تركيبية ومعلومات أخرى .

إن الأجهزة المستخدمة حالياً تعالج مشكلة تمييز الكلمات المنفردة . وهذه المشكلة أسهل بكثير من تمييز الكلام المستمر ، لكن حتى بالنسبة لتمييز الكلمات المنفردة و التي تحقق أكثر البرامج تقدما فيها نجاحا يصل إلى 99.5% ، فإن هذا لا يكون إلا لمحصول من الكلمات لا يزيد عن 120 كلمة و تتطلب أن ينطقها متكلم واحد و مدرب .

لتوضيح أكبر للمشكلة سنقوم بمقارنة بسيطة بينها وبين نظيرتها اللغة المكتوبة فاللغة المكتوبة تتميز بالخصائص التالية :

- تفصل الكلمات بفراغات .
- في الغالب تكون صحيحة الإملاء و كاملة .
- لدينا المقدرة على تجاهل أو إستخلاص أو إعادة القراءة لكلمة ما *(Dr Philip Jackson 2005)*.

إن كل العوامل السابقة غير متوفرة في اللغة المنطوقة فالكلمة قد تكرر أو يفقد جزء منها أو تشوه تماما نتيجة للأسباب التالية :

1. إحتواء الرسالة المنطوقة على ضجيج قد لا يكون له اي تفسير منطقي.
2. نطق الكلام نادرا ما يكون مضبوطا (إختلاف نطق نفس العبارة من شخص إلى آخر).
3. إختلاف نطق المتحدث الواحد لنفس العبارة (حسب الحالة النفسية و الفسيولوجية).
4. تأثر الصوت الواحد في حالة ينطق منفردا او مع كلمات أخرى .
5. ليست هنالك حدود واضحة في الإشارة الصوتية بين الكلمات المتتالية.
6. بعض الكلمات قد تتفق في النطق و تختلف في الإملاء .

يقول دكتور فيلب جاكسون أن سبب صعوبة تمييز الكلام ناتجة عن جهلنا بأمور جذرية كثيرة في عملية الكلام بين البشر و البحوث الحالية تحاول التحايل على هذا الجهل قدر الإمكان فتارة نستخدم تقنيات مقارنة الأنماط وتارة أخرى الشبكات العصوبية، ويرى أن للصعوبة تتفاوت حسب النظام و تقاس بالتالي:

1. هل النظام غير معتمد على متكلم واحد.
2. ما هو حجم الذخيرة اللغوية.
3. هل الكلمات منفردة أم متصلة.

4. ما مدى تعقد اللغة وما حجم المعرفة المتاحة حولها.
5. درجة الغموض الصوتي.
6. غلوظة التشويش.

لفهم أكثر لأساسيات تمييز الكلام سنتعرض في الفقرة القادمة إلى مصطلحات ذات صلة *(Stephen Cook 2005).*

4.1 مصطلحات أساسية

1. النطق (Utterance)

في نظم تمييز الكلام نعني بالنطق الكلمة أو الكلمات التي تُمَثّل كمعنى وحيد للحاسوب، ويمكن أن يكون كلمة أو عدة كلمات أو جملة أو حتى عدة جمل .

2. إعتمادية المتكلم (Speaker Dependance)

بعض نظم تمييز الكلام مصممة لمتكلم بعينه و هي أكثر دقة من الأنظمة المصممة للتكيف مع عدة مستخدمين ، تفترض الأنظمة المعتمدة على المتكلم أنه يتكلم بصوة ثابتة ومعدل إيقاف منتظم أما نظيرتها فتستخدم تقنيات التدريب .

3. المفردات (Vocabularies)

المفردات أو القاموس هي قوائم الكلمات أو النطق التي سوف يميزها النظام ، عموماً كلما ذاد حجم القاموس ذات صعوبة تمييز الكلمات . وعلى خلاف القواميس الطبيعية ليس بالضرورة أن يكون لكل مدخل كلمة مقابلة وحيدة فقد تكون جملة أو إثنتين .

4. قياس الدقة (Accuract)

قابلية التمييز يمكن أن تفحص بقياس دقتها ليس فقط تمييز النطق الصحيح ولكن معرفة ما إذا كان المنطوق من القاموس أو لا . أنظمة تمييز الكلام الجيدة لها دقة تصل إلى 98% أو أكثر ! . تعتمد الدقة المقولة في الغالب على التطبيق .

5. التدريب (Training)

بعض أنظمة تمييز الكلام لها القدرة على التكيف مع المستخدم ، متى ما وجدت هذه القابلية يبدأ ظهور التدريب . يكون التدريب بإمتلاك تكرارات قياسية من المتكلم أو بالعبارات الشائعة و تعدل خوارزميات المقارنة لمجارات متكلم معين .

نستنتج مما سبق أن نظم تمييز الكلام ليس مسمى لنظام بعينة و إنما قاسم مشترك بين عدد من الأنظمة قد تختلف كثيرا من ناحية التصميم و التطبيق، فيما يلي أنواعها الشائعة .

5.1 أنواع أنظمة تمييز الكلام

من وجهة نظر المصممين ، قسمت إلى فئات بإعتبار نوع النطق الذي يحدد قابلية التمييز ، هذه الفئات تعتمد على أساس واحد و هو تحديد متى يبدأ المتكلم النطق و متى ينتهي :

1. الكلمات المتفرقة (Isolated Words)

أنظمة الكلمات المعزولة هذه في العادة تتطلب أن يكون كل نطق هادئ ، نعنى بهذا قلة الإشارة السمعية على كلا جانبي النافذة المعنية (النافذة هي نتاج تقسيم الإشارة إلى كلمات) . هذا لا يعني أن النظام يستقبل كلمات وحيدة لكن يتطلب النطق الوحيد في الزمن المعين ، في غالب الأحيان هذه الأنظمة لها حالتي السمع و عدمه ، تطلب هذه النظم من المتكلم الإنتظار بين النطق و الذي يليه في هذه الفترة تجري عمليات المعالجة . قال إستيفن كوك أن الإسم الأفضل لهذه الفئة هو النطق المنفصل .

2. الكلمات المتصلة (Connected Words)

تشبه إلى حد كبير الكلمات المتفرقة لكن تسمح بأكثر من نطق للعمل سوية بفراغات أقل بينها .

3. الكلام المستمر (Continuous Speech)

الفئة الأكثر صعوبة لان على المصممين إستخدام طرق خاصة في تحديد حدود النطق ، تقرباً يجعل المستخدمين يتكلمون بطبيعتهم بينما يقوم الحاسوب بتقرير لفظ الكلام ، اي إملاء للحاسوب .

4. الكلام التلقائي(Spontaneous Speech)

بدون تدريب يقوم النظام بتمييز الكلام الطبيعي، نظم التمييز من هذه الفئة يجب أن تكون لها القدرة تلمس التنوع في خصائص الكلام الطبيعي.

5. البصمة الصوتية(Voice Verification/Identification)

بعض نظم تمييز الكلام لها قادرة على إكتشاف من يستخدمها، هذا التوثق لا يعتبر نظاماً أمنياً .

أما من وجهة نظر المستخدم، الذي يهتمم بالتطبيق فإن كل مهام الحاسوب يمكن لنظم تمييز الكلام أدائها ، فالتطبيقات التالية تمثل فئات تلك النظم الموجودة حالياً *(2005 Stephen Cook)* :

1. الإملاء (Dictation)

الأكثر شيوعاً اليوم في البيئة الأعمال، حيث لتلك الفئة القدرة على معالجة الكلمات ، وفي بعض الحالات تزاد الدقة تلقائياً بمفردات خاصة .

2. الأوامر و التحكم (Command and Control)

هي تلك الأنظمة المصممة لأداء الوظائف و الأعمال في نظام يكون تمييز الكلام إضافة ثانوية مثل "افتح الملف" .

3. الإرسال الهاتفي (Telephony)

تسمح بعض أنظمة البريد الصوتي للأشخاص المتصلين بإصدار الأوامر بدل كبس الأزرار .

4. تطبيقات أخرى

العديد من التطبيقات لنظم تمييز الكلام التي نشهد بداياتها اليوم مثل التعرف على الكلام في الأجهزة المحمولة ، أو بالأحرى على حد قول إستفن كود لماذا لا أستطيع التحدث إلى تلفازي رغم ذلك ؟!.

6.1 مراحل تمييز الكلام

كيف تعمل أنظمة تمييز الكلام ؟ مبدأ عمل تلك النظم لا يخرج من نوعين أساسيين ، النوع الأول أنظمة التعرف على الأنماط وهي إما أن تقارن أو تدرب لمعرفة الكلمة . و النوع الثاني هو الأنظمة اللفظية الصوتية (Acoustic Phonetic systems) وهي التي تحاول محاكاة طريقة التعرف على الكلام عند البشر بمعرفة خصائصه مثل أصوات حروف العلة . أغلب الأنظمة اليوم تعتمد على مبدأ التعرف على الأنماط لأنه الحاصد على قدر أعلى من النتائج بالحواسيب الحالية . تمر أغلب أنظمة تمييز الكلام بالخطوات التالية *(Stephen Cook 2005)* :

1. تسجيل الصوت و تحديد النطق

يمكن إنجازها بعدد من الطرق بحث يمكن إيجاد نقاط بداية النطق بمقارنة المستويات الصوتية بالعينة المسجلة ، أما نهاية النطق فهو أكثر صعوبة لأن المتكلم يميل إلى ترك ما ليس له معنى مثل إصطكاك الأسنان و التنفس و التنهد و غيره .

2. مرحلة ما قبل التنقية

كذلك يمكن إنجازها بطرق منوعة إعتماداً على خصائص أخرى في النظام ، فالطريقة الأكثر شوعاً تسمى ببنك المرشحات (Bank–of–Filters) حيث تستخدم سلسلة من المرشحات السمعية لإعداد العينة ، و طريقة التشفير التنبؤية الخطية Linear Predictive Coding (LPC) التي تستخدم دالة تنبؤية لحساب الإختلافات (الأخطاء) .

3. مرحلة الترشيح بالتقسيم إلى إطارات (Framing/Windowing)

تقسم في هذه المرحلة العينة إلى إحجام محددة ، في أغلب الأحيان تدرج هذه الخطوة إما في الثانية أو الرابعة ، كما يتم فيها تقليم أطراف العينة لتهيئتها للمرحلة القادمة .

4. ترشيحات إضافية

قد لا توجد هذه المرحلة و هو آخر تهيئة لكل نافذة قبل المقارنة و الربط ، وفيها يتم حياز و تطبيع للزمن .

5. المقارنة و الربط

هنالك عدد ضخم من التقنيات المستخدمة، التي سيتم التطرق إليها في فقرات قادمة مثل نماذج ماركوف الخفية (Hidden Markov Models) (HMM) ، التحليل الترددي ، التحليل التفاضلي ، الجبر الخطي ، وطريق وقت التشويش ، كل هذه الطرق و أكثر تتبارى للوصول للدقة الأعلى .

6. التطبيق

هنا يضع المبرمج إي شئ يريد أن يفعله .

7.1 أساسيات الصوت الرقمي

يتم تسجيل الصوت بتحويل الإشارة التماثلية من الميكرفون إلى الإشارة الرقمية عبر محول موجود في بطاقة الصوت يسمى (A/D) . عندما يبدأ الميكرفون بالعمل فإن فيض من الموجات المغنطيسية تتذبذب مسببة تياراً كهربائياً يتوجه هذا التيار نحو بطاقة الصوت ، يقوم ال(A/D) بتسجيل قيم الفولطيات الكهربائية في مقاطع محددة ، هنالك عاملان مهمان في هذه العملية ، الأول هو تردد العينة (sample rate) أو كم سجلنا من الفولطيات ، و العامل الثاني هو معدل البتات للعينة (bits per sample) أو ما هي دقة العينة المسجلة ، هنالك عوامل أخرى أقل أهمية مثل عدد القنوات الصوتية ثنائية (stereo) أو أحادية (mono) ، لكن في أغلب أنظمة تمييز الكلام تكون القناة الأحادية كافية .

أغلب التطبيقات تحضر مسبقاً مجموعة من القيم لا تتغير طالما لم يتغير التوثيق للنظام ، كما على مطوري النظم تجريب عدد من القيم لتحديد الأنسب إلى خوارزمياتهم ، و السؤال الذي يطرح نفسه هنا ماهي الإعدادات المثالية للنظام ؟ أولاً ما هو تردد العينة المناسب ؟ لأن مدى تردد الترددات في الكلام البشري منخفض نسبياً فإن من المستحسن أن يكون مدى العينة بين (100Hz-8kHz)، في الغالب تكون (8kHz) كافية ، ثانياً ما هو معدل البتات للعينة ؟ 8 bits per sample ستسجل قيم مابين (0-255) هذا يعني أن عناصر الميكرفون واقعة في مدى (256) ، أما 16 bits per sample فستعطي إحتمالية وجود العنصر في مدى (65536) ، إذا كنت تمتلك القوة المعالجة و الذاكرة الكافيتين فلا غضاضة حتى بإستعمال مدى عينة أكبر .

صيغة التشفير المستعملة يجب أن تكون بسيطة مثل خوارزميات U-Law/A-Law ، في العادة مخططات الضغط الأخرى تستهلك من قوة المعالج و لا تكسبك الكثير*(Stephen Cook 2005)* .

8.1 تقنيات تمييز الكلام

علمنا فيما سبق أن أي نظام لتمييز الكلام لابد و أن يمر بعدة مراحل ، وتعتبر مرحلة المقارنة و الربط هي المرحلة الرئيسية ، حيث يتحقق فيها معنى التمييز للنطق المراد للآلة فهمه . يستخدم أي نظام لتمييز الكلام

تقنية ما في هذه المرحلة ، فيما يلي بعض هذه التقنيات المستخدمة حالياً ، ولأننا نتوخى التبسيط في هذا البحث قمنا بتقسيم تلك التقنيات إلى فئات و اخترنا مثالاً واحد لكل فئة .

1. تقنيات إحصائية

حققت نماذج ماركوف الخفية (HMM) رواجاً في الفترة السابقة ، نتيجة للنجاح الكبير الذي حققته في تمييز الكلام ، يرجع ذلك النجاح إلى الأساس الإحتمالي الذي بنيت عليه ، فالتعامل مع الكلام تعتريه مشاكل النقص وعدم المصداقية كما ذكرنا آنفا .

كانت البداية في الخمسينات حيث إجتمع خبراء إحصائيون لدراسة خصائص العينة العشوائية ، جاءت خوارزمياتهم لتكون محايدة في التخمين و أساس ليبدأ به بوم (Baum) و من معه في طريقهم إلى (HMM) الذي دخل حقل تمييز الكلام على يد بيكر (Baker) في 1975 *(د.إيمان أبو المعالي1993)*.

تعمل هذه التقنية عن طريق دالة إحتمالية لسلسلة ماركوف الخفية ، التي تتكون من عدد محدد من الحالات و مجموعة من الدوال العشوائية المرتبطة بكل حالة ، تتغير الحالات نسبة لمصفوفة إحتمالات إنتقالية ، حتى الوصول للتوقع النهائي للنطق .

2. تقنيات المقارنة

التقنية الأكثر شيوعاً في هذا المجال هي تقنية تطويع الزمن الديناميكي Dynamic Time Warping (DTW) ، تعمل هذه التقنية على مبدأ معالجة التغيرات في محور الزمن المسببة بإختلاف معدل النطق . لأن الكلمات التي تمثل بقيم صوتية متشابهة تقارن على أساس أنها متساوية الطول و أزمنتها متطابقة و هذا عملياً غير صحيح ، فالعملية الكلامية معتمدة على الزمن فالكلمات المتشابه النطق قد تختلف في طولها الزمني نتيجة لتغيير سرعة النطق .

فالطرق التقنية في حالة تطوير بحيث أنها أصبحت قادرة على إحداث تناسب كلمة مع أخرى بتحريف مقياس الزمن غير الخطي ليعطي تناسبا في كل نقاط العينة.

تستعمل ال(DTW) الإنحلاف المتري (Distortion Metric) لمقارنة متجهات الخصائص في كل من النطق و النطق المقارن و تحذف الفروقات بين العينتين بتطويع محور الزمن بالنسبة لعينة التي تحقق أكبر توافق مع الأخرى *(مجدي محمد الطيب 2001)* .

3. تقنيات الربط

أثناء السنوات الثلاثون الأخيرة ، إزدهر إتجاه حوسبي جديد مستند على العلوم العصوبية سموه الشبكات العصوبية ، قياساً على العمل الداخلي للإنسان . إن الفكرة الرئيسية وراء الشبكات العصوبية هو محاكاة سلوك الربط العصبي في الدماغ *(Pablo Zegers 1998)*.

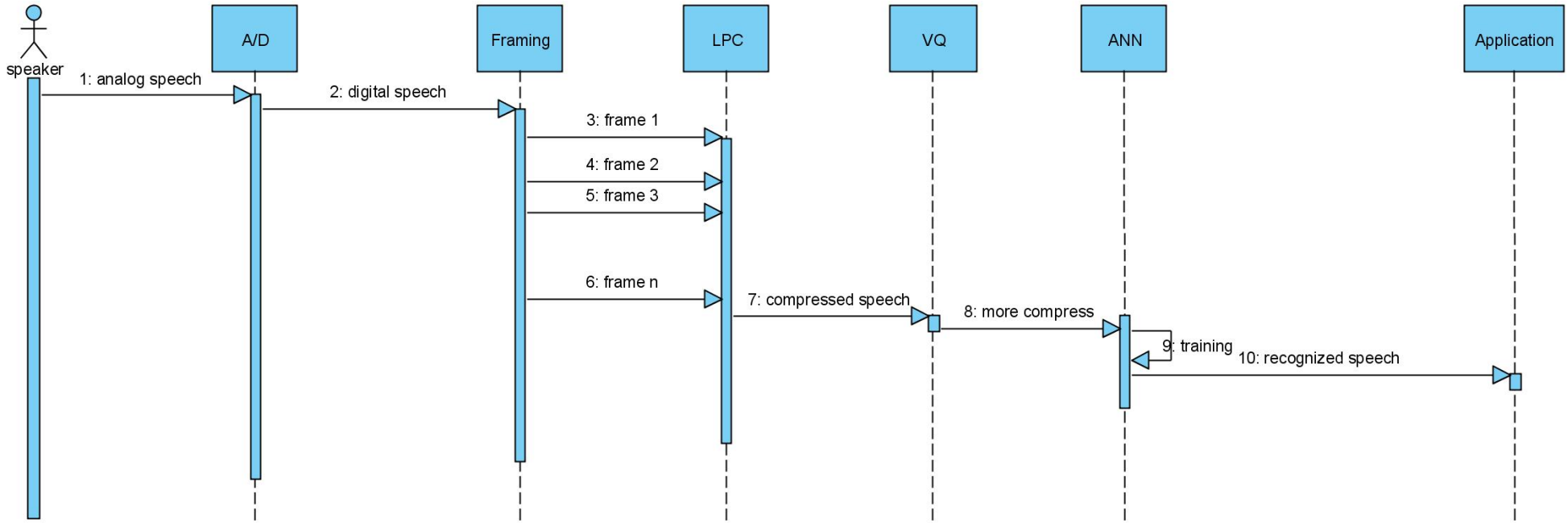
وهي ممتازة في تمييز الكلام لقدرتها على التكيف مع البيانات الناقصة أو المبهمة ، فالكلام قلما يسلم من هذه العيوب ، كما تتميز بالسرعة بالمقارنة مع نماذج ماركوف الخفية .

بدأ الباحثون بالبحث عن أساليب تعلم للشبكة تمكنها آلياً من إعطاء قيم الأوزان التي تجعلها تحسب أي دالة تكلف بها ، إكتشف روزينبلات (Rosenblatt) (1962) طريقة تعليمية تكرارية لنوع معين من الشبكات

أحادي الطبقة ، و يقول طالما أن الدالة يمكن حسابها فإن شبكته ستعطي الخرج المطلوب ، هذا الإكتشاف ولد موجة من الحماس نبأت الباحثين بإمكانية خلق آلة ذكية *(Joe Tebelskis 1995)*. سيناقش هذا الموضوع بإستفاضة في الباب الرابع .

و يوضح المخطط التسلسلي التالي التقنيات المختارة لهذا البحث ، و يحتوي على المختصرات التالية :

1. Linear Predictive Coding : (LPC) (الشفرة التنبؤية الخطية).
2. Vector Quantization : (VQ) (تكمية المتجه) .
3. Artificial Neural Networks : (ANN) (الشبكات العصوبية الإصطناعية).



شكل (1.1) المخطط التسلسلي للنظام

9.1 خطة تحليل النظام

بالنظر إلي المخطط التسلسلي السابق يمكن وصف تسلسل النظام بالآلات التالية :

- مدخل للصوت : من ال(A/D) وليكن (آلة إدخال وإخراج الكلام) .
- مشفر للكلام : (LPC)(VQ) وليكن (آلة تشفير الكلام) .
- مقارنة الكلام و ربطه : (ANN) وليكن (آلة ربط و مقارنة الكلام) .

تحتوي كل آلة على مايلي :

- جمع المتطلبات .
- مخطط حالة الإستخدام (use case) الذي يعيد وصف المتطلبات في صورة فاعل و فعل .
- جدول للكائنات الموجودة في الآلة يمكن إستنتاجه من مخطط حالة الإستخدام .
- مخطط فئات (class diagram) يصف الآلة في شكلها النهائي .

و ضعت كل آلة مع الباب المناسب لها ، ثم جمعت في آلة تمييز الكلام في الباب الخامس .

الباب الثاني(15-22)

أساسيات برمجة الصوت الرقمي

1.2 مقدمة

2.2 مفاهيم أساسية للتعامل مع الصوت الرقمي

3.2 تحليل آلة إدخال و إخراج الكلام

4.2 القضايا التصميمية و التطبيقية لآلة إدخال و إخراج الكلام

1.2 مقدمة

علم من الباب السابق أن نقطة البداية لأي نظام لتمييز الكلام ايا كان نوعه هي تسجيل الصوت ، الذي يتطلب قدرات برمجة خاصة مثله مثل الرسم بالحاسوب (Computer Graphics) . هذا لباب سيناقش القضايا البرمجية المتعلقة بالتعامل مع الصوت في الحاسوب مثل كيفية قرأته و تخزينه و كتابيه .

إن التعامل مع الصوت الرقمي هو من أنشط مجلات الحاسوب بحثاً بدئاً من تصميم بطاقات الصوت إنتهائاً بطرق الضغط المعقدة ، لذا كان التركيز في هذا البحث على ما يخدم القضية الأساسية فقط ، بمعنى أن كل الدول و الفئات (Classes) المخصصة للتعامل مع الصوت صممت خصيصاً للكلام ، و بالتالي هي غير فعالة مع غيره .

ذكر في الباب السابق مقتطفات عن الصوت الرقمي ، يناقش لباب القادم المفاهيم الأساسية المتعلقة بالتعامل مع الصوت الرقمي بنظرة برمجية .

2.2 مفاهيم أساسية للتعامل مع الصوت الرقمي

إن التعامل مع الصوت الرقمي قريب جداً من التعامل مع الصورة الرقمية و التي يلم بها أكثر دارسي الحاسوب نسبة لوجود مادة الرسم بالحاسوب في أغلب المقررات ، لذا سنحاول على قدر المستطاع التبسيط بذكر أوجه الشبه .

إن أصغر تمثيل حوسبي لموجة إشارة الصوت يسمى العينة (sample) وهي إما 8 ، 16 ، 24 ، 32 ، bit التي يقابلها نقطة (pixel) في الصورة ، تجتمع العينات مكونة إطار (frame) ، ثم تجتمع الإطارات مكونة إرجاء (buffer) يسمى في الصوت إرجاء الصوت (sound buffer) وفي الصوة إرجاء الإطار (frame buffer) ، عدد الإطارات المعروض في الثانية يسمى بتردد الشاشة ، اما في الصوت فعدد العينات المسموعة في الثانية تسمى تردد العينة (sample rate) ، القياس بالعينة بدلاً من للإطار قياس بديهي لأن في الصورة تعرض كل النقاط في نفس اللحظة لكن في الصوت لابد من التسلسل .

يستخلص مما سبق أن هنالك عاملان أسياسيان يحددان دقة الصوت وهما عدد البتات في العينة (bit per sample) و تردد العينة ، لكن و على خلاف الصورة يمكن أن تسمع العينة في أكثر من قناة إبتداءً من قناة واحدة إلى ما يسمى اليوم 5.1 او 7.1 أي الخمس او حتى السبع قنوات في الحظة الواحدة حينها نصل إلى دقة عالية جداً تستخدم في المسارح المنزلية و غيرها ، وهي مكلفة مادياً ، إذاً عدد القنوات السمعية (number of channels) عامل مهم في دقة الصوت .

أين تردد الصوت أو حدته من كل هذا ؟ لا يشكل التردد أهمية في تحديد دقته بعد أن تحول إلى صورته الرقمية .

لا يفوتنا أن نذكر عامل مهم في التعامل مع الصوت و هو مقياس علوه decibel (db) يبدأ تأثير هذا العامل بالظهور عند التعامل مع عدد من البتات كبير ، ويجب وضعه في الإعتبار حينها لأنه قد يخفض من دقة النظام .

أثبتت البحوث السابقة في التعرف على الكلام أن دقة الصوت لا تشكل نسبة كبيرة في دقة تمييز النظام بقدر ما تجره من مشاكل في التعامل مع الذاكرة و السرعة ، بإختصار الجهد المبذول أكبر من الفائدة المكتسبة . لذا قناة واحدة و 8bit لكل عينة و 8kHz لتردد العينة كافية لإنشاء نظام تمييز جيد .

إن التعامل مع بطاقة الصوت مباشرة معقد جداً و يتطلب معرفة قوية بطرق تصميم البطاقات و نظام التشغيل الذي يقود البطاقة ، لذا وجدت ما يسمى (Sound I/O APIs) (API Application Programming Interface) لتنوب عن المبرمج في عنونة إرجاء الصوت ، في مجال الرسوميات توجد أيضاً مثل هذه المكتبات مثل (OpenGL , DirectX) ، في الفقرة القادمة سنناقش أشهرها بإختصار و أيها إخترنا لنظامنا و لماذا *(2006mat)* .

1.2.2 مكتبات إدخال و إخراج الصوت (Sound I/O APIs)

تختلف هذه المكتبات بناءً على عدة عوامل مثل نظام التشغيل و التطبيقات المستخدمة بها و طريقتها في التعامل مع بطاقة الصوت وغيرها الكثير و تتفق في الغالب على أنها مكتوبة بلغة ال(C) .

1. Sndlib

طورت في 1990 من أجل دعم إمكانيات الصوت المحترفة في أغلب أنظمة التشغيل مثل (Linux ، Mac ، Windows) و جعلها أكثر سهولة في يد المبرمجين ، تعتمد على نموذج إدخال و إخراج كتلة كاملة (Blocking I/O model) تبدأ دوالها عادة ب(mus) مثل (mus_sound_open_input ، mus_audio_initialize) .

2. PortAudio

وكما يتضح من إسمها أنها تستطيع التماشي مع كل أنظمة التشغيل ، طورت عام 2000 وهي مفتوحة المصدر و سهلة التعامل ، و صممت خصيصاً من أجل تطبيقات الموسيقى و معالجة الإشارة الرقمية ، مما شجعنا لإختيارها في نظامنا ، تبدأ دوالها ب(Pa) ، سيأتي إن شاء الله تفصيل أكثر عنها و كيفية إستخدامها في فصل القضايا التصميمية و التطبيقية .

3. DirectX, DirectSound

من أشهر المكتبات في مجالي الصوت و الرسوميات ثلاثية الأبعاد و هي مخصصة فقط لنظام التشغيل (Windows) ، تمتاز بقوة في مجال الألعاب و الموسيقى و معالجة الإشارة الرقمية .

4. CoreAudio

من أقدم المكتبات الداعمة لنظام التشغيل (Mac) ، وهي الآلة الخفية وراء البرنامج الشهير (QuickTime) أشهر ما أنتجته (Apple) في مجال الوسائل المتعددة .

5. JavaSound

تتميز الجافا بسهولة إستخدامها الذي إنعكس على (JMF) Java Media Framework المسؤول عن الوسائل المتعددة ، مما ميزها عن سابقاتها ببساطتها و انتظامها .

6. أخر

يوجد العيد و العديد من المكتبات المعنية بالتعامل مع الصوت المنتشرة على شبكة الإنترنت مثل (ASIO ، OSS) وغيرها ، وعلى المبرمج الفطن قراءة مستندات المكتبة جيداً قبل الشروع بإستخدامها .

يتبقى إعطاء نبذة عن تخزين الصوت في مخزنه الأخير (وسائل التخزين الثانوية) و الهيئات المستخدمة لفعل ذلك ، وهي هيئات كثير جداً إلا أن المستخدم في تطبيقات الكلام و الإشارة الرقمية قليل ، نسبة لإهتمامها بسرعة القراءة دون كفاءة التخزين . يتعامل نظامنا بهيئتين الأولى معروفة جداً و مشهورة وهي (Wave) و الثانية تشبهها كثير وتسمى (Raw) في أغلب البحوث لاتعد من هيئات التخزين لأنها فاقدة معلومات أساسية إن لم تعلم بشكل أو بآخر فلن تستطيع تشغيل الصوت بطريقة التي خزن بها *(2006mat)* .

2.2.2 هيئة ملف (WAV) WAVE File Format

هيئة ملف (Wav) هيئة فرعية من مواصفات (Microsoft's RIFF) التي تتضمن عدد كبير من البيانات المختلفة ، وهي بيانات الوسائل المتعددة . بنية على أساس و جود قطعة (chunk) و هو مجموعة من البيانات توصف بنوع محدد ، و لهيئة (Wav) إثنين من القطع الأساسين :

الأولى تسمى بقطعة الهيئة و هى التي تصف عدد البتات في العينة و ترددها وغيرها.

أما الثانية فهي قطعة البيانات وهي التي تحتوي على البيانات الحقيقية .

كما من المسموح إضافة أنواع اخرى من القطع مثل بيانات حقوق النشر و المؤلف و غيرها. كما يمكن لل(Chunks) أن تظهر بأي ترتيب.

إن الشكل السابق لملف (Wav) شرع قانونيا بعد ظهور مكتبة (Win32) إلى جانب هيئات سابقة تشترط و جود القطع الأساسية فقط و بنفس الترتيب ، وفيها تكون طول قطعة الهيئة ثابت ب (44 byte) .

تحمل قطعة الهيئة معلومات عن خوارزمية التشفير المستخدمة في أحد حقوله ، فإذا كانت قيمته واحد فهذا يشير إلى عدم التشفير *(2006lightlink)* .

ملف (raw) يحمل قطعة البيانات فقط.

الفصل القادم هو الجزء التحليلي من النظام الذي يهتم بإدخال وإخراج الكلام من وإلى النظام ، الذي يتطلب معرفة بأساسيات البرمجة الموجة بالكائنات و كيفية وصفها بال(UML) Unified Markup Language

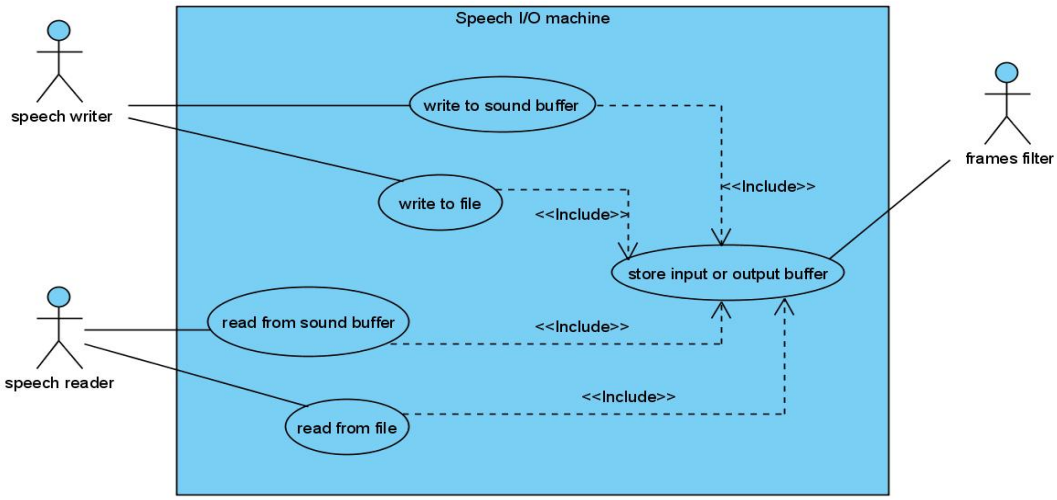
3.2 تحليل آلة إدخال و إخراج الكلام

ماذا ينبغي أن تفعل هذه الآلة ؟ سؤال يقود لأول خطوة في تحليل أي نظام وهي جمع المتطلبات وهي كما يلي :

1. القدرة على تسجيل الصوت .
2. القدرة على حفظ الصوت في ملفات.
3. إمكانية إدخال الصوت عن طريق الملفات و خصوصا (wav).
4. تشغيل الصوت (playing).
5. إمكانية إضافة أو تعديل أو حذف أي عينة أو إطارأو حتي إرجاء .
6. القدرة على إعادة تقسيم الإرجاء إلي إطارات بإدخال طول الإطار.
7. فتح المجال أمام دوال معالجة الإشارة الرقمية لكي تضاف في مكانها.
8. أن يكون النظام شفافاً لا يعتمد على قيم لتردد العينة ثابتة و لا لعدد البتات في العينة و..
9. أن يتعامل مع الصوت المدخل ببطاقة الصوت و المدخل بالملفات دون تمييز .
10. أن يعرض موجة الصوت على الشاشة.
11. أن يكون سريعاً في إدخال و إخراج الصوت.
12. أن تحول كل الإرجائات لإطار واحد في أي وقت.
13. أن تستطيع تحديد النطق وذلك بقطع أطراف الموجة .

تجمع المتطلبات و تدرس على مهل

لتحديد نطاق النظام و الكائنات و العلاقة التي تربط بعضها البعض، مخطط حالة الإستخدام (use case) التالي يوضح ذلك.



شكل (1.2) مخطط حالة الاستخدام لآلة ادخال واخراج الكلام

يوضح المخطط السابق ثلاثة عناصر رئيسية في النظام وهي قارئ الكلام (speech reader) و كاتب الكلام (speech writer) و مرشح الإطارات (frames filter) و كما هو مبين في الشكل يقوم كلاً من القارئ و الكاتب بوظيفتهما سواء كانت من الملف أو من بطاقة الصوت ، بينما يقوم مرشح الإطارات بتهيئة الخرج أو إستقبال الدخل .

يتضح أن مرشح الإطارات مكون من إطارات ! أي كأن هنالك عنصر خفي يقوم بتهيئة الخرج أو الدخل لإطار واحد فقط يقوم مرشح الإطارات بندائه ، دعنا نطلق عليه مرشح الإطار ، و يتضح أن مرشح الإطار مكون من عينات ، كل عينة تهيئ للدخل أو للخرج بواسطة مرشح العينة .

يجدر القول أن الشكل السابق ليس بالبساطة الظاهرة فهو يحمل في داخله كائنات كثيرة وهي التي ستنقلنا للخطوة التحليلية التالية و هي تحليل الفئات ، القائمة التالية تشمل كل الكائنات الظاهر و الضمنية في المخطط السابق .

جدول (1.2) تحليل كائنات آلة إدخال و إخراج الكلام

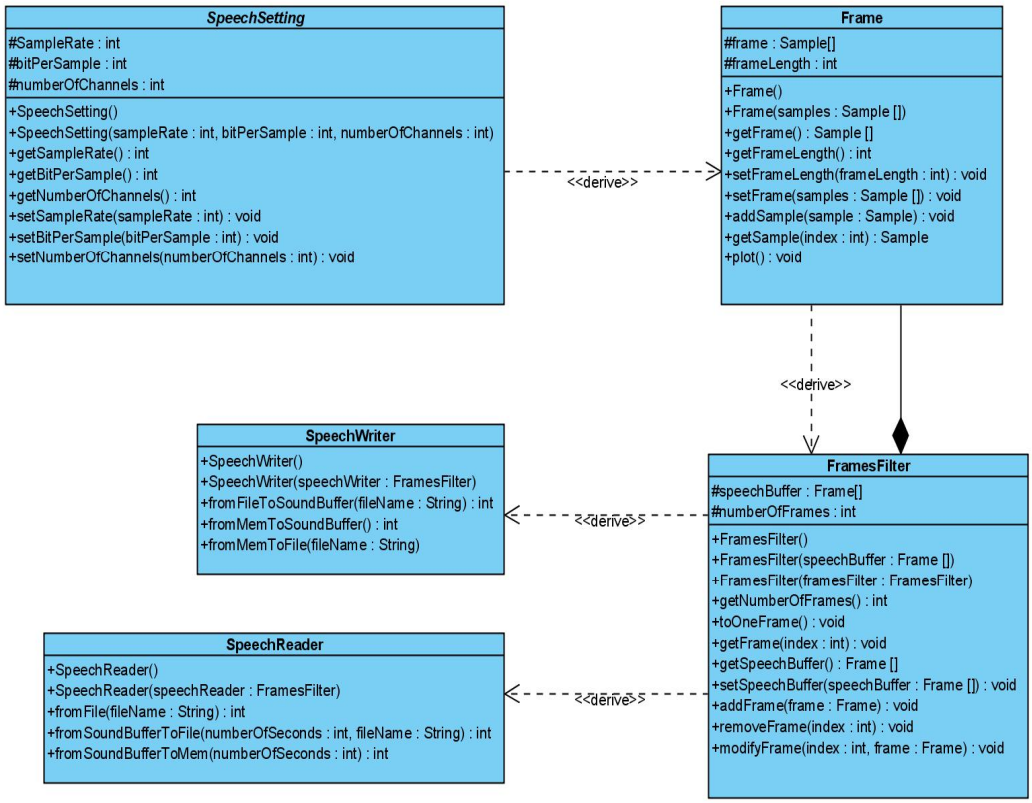
الكائن	وظيفته
Frames filter	تهيئة الدخل و الخرج لإرجاء واحد .
Frame	تهيئة الدخل و الخرج لإطار واحد .
Sample	تهيئة الدخل و الخرج لعينة واحدة .
Speech reader	قراءته إرجاء واحد من ملف أو بطاقة الصوت .
Speech writer	كتابة إرجاء واحد في ملف أو بطاقة الصوت .

بقي لنا دراسة العلاقة بين الكائنات و كيف تتفاعل مع بعضها ؟ من الجدول السابق نستخلص الجواب ، من الواضح أن كلا من قارئ الكلام و كاتبه يحتاجان للإرجاء المراد قراءته او كتابته و هو إختصاص مرشح

الإطارات دعنا نقول مبدئياً أن كل قارئ أو كاتب يحتاج لكائن و احد فقط من مرشح الإطارات و هذه العلاقة هي (composition) وهي نفس العلاقة بين مرشح الإطارات و الإطار وبين الإطار و العينة ، لكن بالعودة لمتطلبات النظام نجد أن المطلب (12) يجبرنا على قلب كل العلاقات إلى وراثة لأن النظام يراد به في أحد مراحلة معامل كل من القارئ و الكاتب كأنما الذي قرأوه أو كتبوه هو إطار واحد فقط ، ما عدا علاقة و احدة هي التي ستظل في مكانها وهي التي بين الإطار و العينة ، وهذا بديهي لأن الإطار لابد أن يكون الأب الأكبر تحقيقاً للمطلب (12) .

وبالعودة إلى المتطلبات نجد المطلب (8) الذي يجبرنا على خلق كائن جديد مسؤوليته ضبط إعدادات القراءة و الكتابة ، من الواضح أن هذا الكائن تحتاجه كل الكائنات ، مما يجبرنا على جعله الأب الأكبر بدلاً من الإطار ، سميناه (speech setting) .

المخطط التالي يمثل المخطط النهائي لآلة إدخال و إخراج الكلام بعد إضافة كل الدوال المطلوبة ، و كل التعديلات التي ذكرت ، و به تكون إنتهت مرحلة التحليل للجزء الأول من النظام و الحمد لله ، يأخذ لباب القادم هذا المخطط و يحاول سرد القضايا التصميمية و التطبيقية المتعلقة به و الله الموفق .



شكل (2.2) مخطط الفئات لآلة إدخال و إخراج الكلام

4.2 القضايا التصميمية و التطبيقية لآلة إدخال و إخراج الكلام

ذكرنا آنفاً أن المكتبة التي إخترناها للتعامل مع الصوت هي (**PortAudio**) ، الفقرة القادمة تبين بإختصار كيفية إستخدامها .

1.4.2 إستخدام (PortAudio)

في أغلب المكتبات المتعاملة مع الصوت يوجد دالتان لا غنى عنهما تستخدمان طريقة النداء (Callback) وهما (recordCallback ، playCallback) وكما هو معروف في طريقة النداء هذه أن الدالة تكون ذات شكل محدد (تستقبل عدد ونوع من البيانات محدد بواسطة المكتبة) تنادى الدالة من هذا النوع بإرسال إسمها كمتغير عبر دالة أخرى مبنية مسبقاً لتتماشى مع طريقة النداء المذكورة ، و لا سبيل لندئها إلا بها . ينتشر مثل هذا النوع من الدوال في التعامل مع الرسوميات أيظاً ، وظيفة كل من الدالتين ربط المكتبة أثناء زمن التنفيذ ببرنامج قيادة بطاقة الصوت في نظام التشغيل ، يمكن ان تسمى الدالة أي اسم و أو تعاد كتابتها لتناسب النظام دون لمس المتغيرات التي تستقبلها .

إن الإصدارة من (PortAudio) المستخدمة في هذا النظام مكتوبة بلغة ال(C) و كلها مفتوحة المصدر.

2.4.2 كيفية تسجيل و تشغيل الصوت

نبدأ أولاً بإعداد المتغيرات و المتغيرات المطلوبة هي :

- stream من النوع (PortAudioStream).
- متغير من النوع (PaError) سمه مثلاً (err) . (مهم جداً لأن الأخطاء كثير عند التعامل مع بطاقة الصوت) .
- data من النوع (PaTestData) . (struct) يحمل ثلاثة متغيرات عداد للإطارات و أقصى عدد من الإطارات و مؤشر من نوع (SAMPLE) يكون ال(SAMPLE) (unsigned char) عند التعامل ب8bit ، هذا المؤشر يشير إلى البيانات المراد تشغيلها أو إلى مكان محجوز في الذاكرة للبيانات القادمة من عملية التسجيل ، يلاحظ أن عدد الخانات المحجوزة يتضاعف على حسب عدد القنوات المستخدمة) .

ثانياً تهيئة الجهاز (بطاقة الصوت) : جعل المتغير (err) يساوي خرج الدالة (Pa_Initialize) إذا كان ال(err) يساوي (paNoError) فالعملية تمت بنجاح .

ثالثاً القيام بالتخاطب مع بطاقة الصوت بفتح منفذ لذلك بإستخدام الدالة (Pa_OpenStream) التي تستقبل المتغير (stream) و مجموعة من المتغيرات الأخرى من ضمنها ال(data)، و عند إرادت التسجيل ففي متغير دالة ال(Callback) يكتب إسم الدالة المعنية بالتسجيل والتكن مثلاً (recordCallback) او تريد التشغيل ف(playCallback) ، يعيد تنفيذ دالة فتح المنفذ (Pa_OpenStream) كذلك خطاء من النوع (PaError).

رابعاً التسجيل او التشغيل الحقيقى، إذا تم كل ما سبق بصورة سليمة فإن المتغير (stream) بعد تمريره للدالة (Pa_OpenStream) يكون جاهزاً للتنشيط بإستخدام الدالة (Pa_StreamActive) التي لابد من إدخاها في حلقة (while) لتنشط كل ال(stream)، التنشيط يعني تشغيل أو تسجل الصوت حسب الدالة ال(Callback) المرسلة للدالة (Pa_OpenStream) .

3.4.2 المشاكل التي واجهت تنفيذ مخطط الفئات

مشكلة تحديد النطق المعروفة باسم (end point detection) ، تم حلها بإختيا رقم يمثل أقل قيمة للعينة يمكن أن يبدأ او ينتهي عنده النطق ثم مسح كل الإطارات قبل الإطار الذي يحتوي على أول وجود للعينة و مسح كل العينات قبلها ، ثم إزاحة العينة لتكون تلك العينة أول عينة في الإطار وهذا يؤدي لإزاحة كل الإرجاء ، بعدها نقوم بمسح كل الإطارات بعد آخر تواجد للعينة تم مسح كل العينات التي بعدها .

الباب الثالث(24-26)

تشفير الكلام

1.3 مقدمة

2.3 التشفير التنبؤية الخطية Linear Predictive Coding(LPC)

3.3 تكمية المتجه Vector Quantization(VQ)

4.3 تحليل آلة تشفير الكلام

1.3 مقدمة

بعد أن قامة آلة إدخال و إخراج الكلام في الباب بجلب النطق المراد التعرف عليه ، و مع عدد من النطوق (جمع نطق) كل نطق له أكثر من تكرار تجهيزاً لتمريرها إلى الشبكة العصبية ، يبدو أن الأمر سيخرج عن السيطرة مع هذا الحجم الهائل من البيانات ، فلابد من وسيلة لتقليصه مع الإحتفاظ بخصائصه كما هي ، هذه الوسيلة هي تشفير الكلام الذي يعتبر مجال بحث منفصل .

في هذا الباب سيطرق نوعين من التشفير ، دون الخوض في تفاصيلهما المعقدة ، والتي تتطلب معرفة قوية بعلم معالجة الإشارة الرقمية . وبالتالي فإن النظام يتعامل مع آلة تشفير الكلام كما و كأنها صندوق أسود .

بالعودة لمخطط تسلسل النظام في الباب الأول نجد المربعين LPC و VQ وهما الجزء من النظام المختص بضغط الكلام ، فيما يلي نظرة عامة عن طريقة عمل كلا منهما .

2.3 التشفير التنبؤية الخطية Linear Predictive Coding (LPC)

هي تقنية تحليل و تركيب (synthesizing) الكلام البشري ، وهي مستخدمة منذ زمن طويل ، ففي أحد معاهد تكساس عام 1987 خلفت أول آلة لنطق الكلام (speech synthesizer) في دائرة متكاملة (integrated circuit) ، كانت هذه الإشارة الأولى لإمكانية صنع آلة تحاكي عمل الأوتار الصوتية البشرية (human vocal tract) بإستخدام تقنية التشفير التنبؤي الخطي . للتشفير التنبؤي الخطي إمكانية خفض تردد العينة تسمي (low bit rates) . و بالرغم من الصوت الصناعي الذي تنتجه شقة التقنية طريقها في التطبيقات العسكرية حيث عدم الإهتمام بالدقة مقابل خفض حجم البيانات و سريتها .

فكرتها الأساسية هي أن العينات الموزعة في زمنياً في الإطار يمكن أن يتوقع بديل خطي لها ، هذا البديل يمثل خصائص الصوت الأساسية في الإطار . فمثلاً إطار مكون من 256 عينة يمثل 20 جزء من الثانية يمكن تمثيل هذه الفترة الزمنية ب13 عينة فقط تمثل خصائص الكلام البشري كما لو كانت 256 عينة .

تستخدم شفرة التنبؤ الخطي عموما في ضغط الصوت في شركات الهاتف علي سبيل المثال GSM ، وتستخدم في تأمين الإشارات اللاسلكية بحيث كل صوت يشفر ويرسل في قناة صوتية. *(2006dspexperts)* .

3.3 تكمية المتجه Vector Quantization (VQ)

الخوارزمية الأكثر شيوعاً في ضغط الصور الرقمية ، المستندة في طريقة عملها على مبدأ تشفير (block) و تسمي خوارزميتها الأساسية ب(fixed-to-fixed length) ، في السابق كان تركيز الخوارزمية على مبدأ الأبعاد المتعددة فقط إلا أن جاء (LBG) (Linde, Buzo, Gray) بخوارزميتم المعتمدة على مبدأ التدريب المتسلسل عام 1980م . ومن حينها إرتبط إسمهم بمطلح ال(Vector Quantization) .

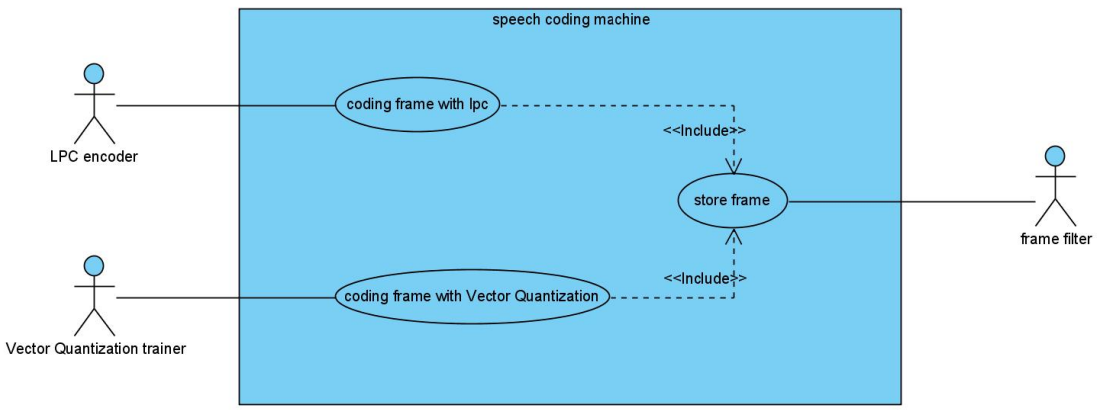
فكرة عمل الخوارزمية هو البحث عن ال(codebook) وهو الشفرة المظغوطة النهائية للمتجه (Vector) ، حيث أن عملية التدريب في الخوارزمية تقوم بتطبيق عدد من المعادلات الرياضية العقدة ، وتجربة بعد القيم التي عساها تمثل المتجه في أبسط صوره *(2006DataCompression)* .

4.3 تحليل آلة تشفير الكلام

متطلباتها كما يلي :

1. أن تقوم خوارزمية تشفير التنبؤ الخطي بتحويل إطار مكون 256 عينة (sample) إلى 13 فقط.
2. أن تأخذ خوارزمية (VQ) الإطار المضغوط ثم تضغطة أكثر إلى أن يصل إلى 4(sample) .

الشكل التالي يمثل حالة الإستخدام لآلة تشفير الكلام في صورنتها البسيطة



شكل (1.3) حالة الإستخدام لآلة تشفير الكلام

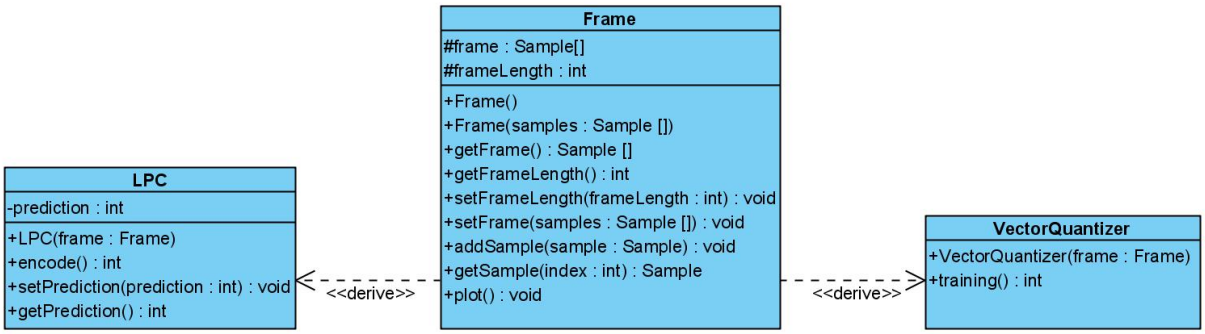
إن مرشح الإطار تم تناوله في آلة إدخال و إخراج الكلام ، وفيما يلي جدول الكائنات الواردة في المخطط .

جدول (1.3) تحليل كائنات آلة تشفير الكلام

الكائن	الوظيفة
Frame filter	تخزين إطار .
LPC	تشفير إطار واحد .
Vector quantization	تشفير إطار واحد .

من الواضح ان كلاً من ال(LPC) و ال(vector quantization) عبارة عن إطار لكنه صغير ، إذاً العلاقة بينهما و الإطار علاقة وراثة ، فيما يلي المخطط النهائي لآلة تشفير الكلام .

من الواضح ان التحيل سطحي و يخفي كثيراً من التفاصيل ، وهذا نتيجة للتعامل مع كلاً من ال(LPC) و ال(VQ) كصندوق أسود وظيفته فقط ضغط الكلام ، وبالتالي لن تكون هنالك فقرة لمناقشة القضايا التصميمية و التطبيقية لآلة تشفير الكلام .



شكل (2.3) مخطط الفئات لآلة تشفير الكلام

الباب الرابع (28–39)

الشبكات العصوبيه الإصطناعية

1.4 مقدمة

بعد قيام آلة تشفير الكلام في الباب السابق بدوها تكون البيانات جاهزة لدخول آلة مقارنة و ربط الكلام ، التي تعتمد على الشبكات العصوبيه الإصطناعية في عملها ، سيناقش هذا الباب المفاهيم المتعلقة بها و كيفية دمجها في النظام تحليلياً و تصميمياً و القضايا التطبيقية المتعلقة بذلك .

تتوزع في جسم الإنسان عدد لا يحصي من الخلايا العصوبيه والتي تتفرع إلى الملايين من الزوائد العصوبيه والتي تقوم بوظيفة حيوية من نقل ردود الفعل إلي العقل البشري عبر الحبل الشوكي وعبر هذه الخلايا العصوبيه يتم حفظ المعرفة عن العالم الخارجي في العقل البشري وتتم هذه العملية بواسطة ضبط الأوزان داخل هذه الخلايا، لو أخذنا مثال في طريقة تعرف الأطفال إلى صور الحيوانات حيث عندما نمرر صور حيوانات للطفل ولنفرض أنها صور لبقره وقط ودجاجة وتم تمرير هذه الصور للطفل مع ذكر اسم كل حيوان أمامه وتم تكرير هذه العملية لعدة مرات، بعد ذلك نأتي لمرحلة الاختبار ويتم فيها عرض الصور السابقة مع إدخال صور جديدة ولتكن صورة عصفور حيث نطلب من الطفل التعرف علي الصور سنلاحظ انه سيتعرف علي الصور الحيوانات التي تعلمها أثناء مرحلة التعليم ولكن عند عرض صورة العصفور سيتعرف على الصورة على أنها صورة دجاجة وذلك لتشابه الخصائص بين العصفور و الدجاجة ولكن مع تكرار تمرير الصور سيتعلم الطفل ومن هذا المنطلق فكر العلماء بطريقة تحاكي هذه العملية التي تتم في العقل البشري وتوصلوا إلي الشبكات العصوبيه التي تندرج تحت علوم الذكاء الاصطناعي بحيث يجعلون من أجهزة الحاسوب أجهزة ذكية، بإمكانها أن تكتسب المعرفة بنفس الطريقة التي يكتسب بها الإنسان المعرفة، وذلك عن طريق ضبط الأوزان أثناء التعلم.

وترجع قوة الدماغ البشري في تمكنه من معالجة المعطيات بأكثر من مجموعة من الخلايا العصوبيه داخله بنفس اللحظة بشكل متوازي، وهذه التقنية لها سرعة عالية إلا أنها تفتقر إلى القدرة على الاستقلال بحل المشكلة، لان النظام عاجز عن حل المشكلة بدون معرفة أسلوب حل المشكلة، الخلايا العصوبيه الموجودة في الدماغ البشري أبطأ بكثير من البوابات المنطقية المستخدمة في الحاسب الرقمي والتي تقاس سرعتها با ns ولكن يظل العقل البشري متفوق على الحاسب في كثير من العمليات من التعرف على الأصوات و الأشكال وهذا يبين أن الدماغ البشري يعمل بآلية مختلفة عن الخلايا العصوبيه الاصطناعية.

أشار دونالد هيب أن المشابك العصوبيه بين العصوب تقوي كلما تم استخدمها بكثرة تصبح عمليات المعالجة أكثر سرعة، وكانت هذه الخطوة الأولي لتفكير بالمعالجات العصوبيه التي كانت مطروحة في وقتها على صورة خلايا و ليس شبكات مترابطة، في الخمسينات كانت أول محاولة لمحاكاة الخلايا العصوبيه بواسطة شركة (IBM)ونجحت هذه المحاولة بعد عدة محاولات فاشلة ولكن علوم الحاسب كانت تتجه نحو الحساب المتسلسل مما أدي إلى تعليق الموضوع.

في نهاية الخمسينات بدأ فرانك روزنبلات العمل في(Perceptron)الذي له القدرة فصل النقاط القابلة للفصل خطيّا دون النقاط غير القابلة للفصل خطيا. وهذه من العيوب قي(Perceptron) ، في عام 1959م قام

برنارد فيدرو و ماركيان هووف ببناء نموذجي عنصر تكيفي خطي (ADAptive LINear Element)و مجموعة عناصر تكيفية خطية (Many ADALINE)كان هذا أول ظهور للشبكات العصوبيه حيث كانت تستخدم كفلا تر (Adaptive Filter) لإلغاء صدي الهاتف ومازالت تستخدم حتى الآن .

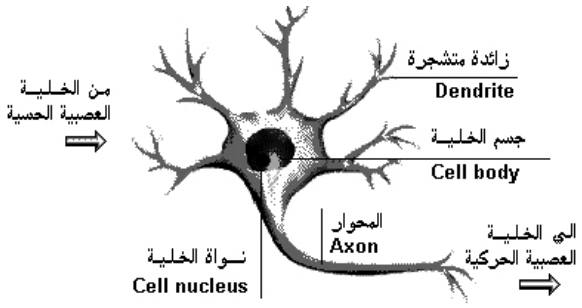
إن المحولات في بناء شبكات عصوبيه عن طريق البرمجة عبارة عن برامج كمبيوتر يتم وضع حد حدا لعدد النورونات التي نريد استعمالها لحل مشكلة معينة بهذه الطريقة يكن استعمال بضعة مئات من العصوبات وذلك لصعوبة تعليم الشبكة، أما الشبكات المبنية على أسس عتادية (أي أن الشبكة النورونية هي عبارة عن وصلات كهر بائية أو شيب) فإن عدد العصوباتات المستعملة يصل إلى 40.000 وللمقارنة فإن الحلزون يمتلك 20.000 خلية، كما أن بعض العلماء نجح في تصنيع بعض الخلايا العصوبيه من بعض خلايا دماغ الفئران ثم استعمال الخلايا العصوبيه البيولوجية في تسير في تسيير برنامج لمحاكات الطيران وهذه خطوة يمكن الاستفادة منها في حل المسائل باستخدام العصوب البيولوجية *(2006Wikpedia)*.

2.4 تعريف الشبكات العصوبيه الاصطناعية

الشبكات العصوبيه الاصطناعية (artificial neural network ANN) أو ما يدعى أيضا بالشبكات العصوبيه المحاكية (simulated neural network SNN)

1. هي تقنيات تحاول محاكاة عمل الجهاز العصبي المركزي عند الإنسان وهي عبارة عن وحدات حسابية صغيرة تربط بينها شبكة من التوصيلات وهذه الوحدات تسمى عصوبات أو عقد (Nodes, Neurons) والتي لها خاصية عصبية حيث تقوم بخزن المعلومات التجريبية لتكون متاحة المستخدم (*د.أحمد عبدا لله إمام 2004)*.
2. هو جهاز مصمم لمحاكاة الطريقة التي يؤدي بها العقل البشري مهمة معينة، وهو يتكون من معالج ضخم موزع على التوازي، ويتكون من وحدات معالجة بسيطة، بحيث يقوم بتخزين المعرفة العملية ليجعلها متاحة للمستخدم وذلك عن طريق ضبط الأوزان.
3. مجموعة مترابطة من عصوبات افتراضية تنشئها برامج حاسوبية لمحاكة عمل الخلايا الطبيعية أو بنى الكترونية (شيبات الكترونية مصممة لمحاكاة عمل العصوبات) *(2006Wikpedia)*.

الخلية العصوبيه (العصوب) يمكن شرحها بالاستعانة بالشكل التالي:



شكل (1.4) الخلية العصوبيه البشرية

يتكون العصوب(Neurn)من الآتي:

1. نواة الخلية (Cell nucleus) .
2. جسم الخلية (Cell body) .
3. زائدة متشجرة (Dendrite).
4. المحور :(Axon) وهو بمثابة خط نقل للجهد بين الخلايا العصوبيه و الخلايا العصوبيه الأخرى عبر خطوط النهايات والألياف.

إذا نلاحظ أن الشبكات العصوبيه الاصطناعية تتشابه مع العقل البشري حيث تكتسب المعرفة بالتدريب وتخزن هذه المعرفة باستخدام قوى وصل داخل العصوبات تسمى الأوزان التشابكية.

تحتاج الشبكات العصوبيه الاصطناعية لوحدات إدخال و معالجة تتم فيها عمليات حسابية لنحصل من خلالها على ردة الفعل المناسبة لكل دخل من مدخلات الشبكة .

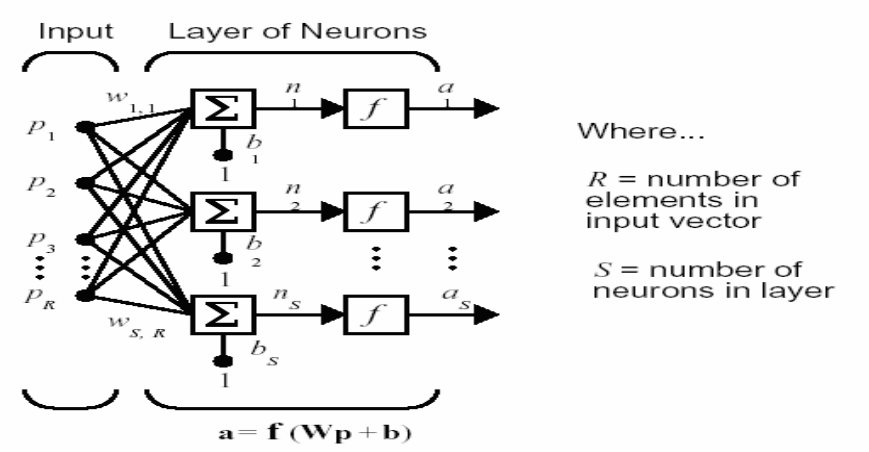
فوحدات الإدخال تكون طبقة الإدخال ووحدات المعالجة تكون طبقة المعالجة وهي التي تخرج نواتج الشبكة وبين كل طبقة بين هذه الطبقات هناك طبقة من الوصلات البينية التي تربط كل طبقة بالطبقة التي تليها وتحتوي الشبكة على طبقة إدخال واحدة فقط ولكنها قد تحتوي على أكثر من طبقة من طبقات المعالجة *(2006Wikpedia)*.

3.4 البنية المعمارية للشبكات العصوبيه الاصطناعية

تعتبر الشبكات العصوبيه مجموعة متوازية من وحدات المعالجة الصغيرة التي تسمي بالعقد أو العصوبات بحيث تكسب الاتصالات البينية بين الوحدات أهمية لأنها تكون ذكاء الشبكة، وعلى الرغم أن آلية عمل الشبكات العصوبيه تحاول محاكاة عمل الخلايا الطبيعية ولكن الخلايا العصوبيه أصغر و أكثر بساطة من الخلايا البيولوجية وتكون عبارة عن وحدات افتراضية تتكون بواسطة برامج الحاسب، كما أن عمل الخلايا العصوبيه يحاول تقليد الميزات الخلايا العصوبيه وفي المقابل أضيفت العديد من المميزات و التقنيات للخلايا الاصطناعية بعض هذه الإضافات مقتبس من الإحصاء أو من نظرية المعلومات لا ترتبط بالخلايا البيولوجية ولكن استطاعت

أن تتفرد بميزة مهمة كان العقل البشري يتفرد بها وهى التعلم التي اضافت للخلايا العصوبيه أهميه خاصة،تم اقتباس عمل العصوب الاصطناعي من آلية عمل الخلايا العصوبيه البشرية ، يمكن أن ننسب لكل مشبك اتصال قادم (incoming synapse) أي مشابك من التفرعات العصوبيه (dendrite) قيمة تسمى قيمة المشبك(weight)تساهم هذه القيمة في نمذجة المشبك بواسطة تحديد قيمته و أهميته بواسطة الوزن نستطيع تحديد يحدد قوة هذا المشبك و أثره في العصوب يضرب وزن كل مشبك بالدخل القادم ومن ثم تجميع نواتج الضرب لكل المشابك القادمة في العادة ما تكون العصوبات البيولوجية تابعة لقاعدة قيمة العتبة ('threshold value') فإذا كان المجموع الموزون (weighted Sum) لقيم الدخل اكبر من قيمة معينة تدعى العتبة (threshold) يتفعّل العصوب بإرسال إشارات كهر بائية على طول المحور العصبي(Axon)ومن ثم تصل الإشارة إلى كل المشابك الخارجة (outgoing synapses) التي ترتبط بعصوبات أخري في العقل، الشبكات العصوبيه النموذجية تحاول محاكاة هذا السلوك بحيث كل عقدة تستلم مجموعة من المدخلات من الاتصالات بالعصوبات القبلية و كل عقدة لها تابع تفعيل (activation function) أو تابع تحويل (transfer function) يحدد للعقدة السلوك الذي يجب أن تسلكه من تحديد العمل في أي لحظة و قيمة الخرج التي يجب أن تعطيها مشابه للعصوب البيولوجي *(2006Wikpedia)*.

معمارية الشبكة العصوبيه هي الطريقة التي ترتبط بها العصوبات مع بعضها لتكوين الشبكة وهذا له صلة مباشرة بخوارزميات التدريب. في الشكل التالي يرتبط الدخل P P بكل عصبون من خلال مصفوفة الوزن W .



شكل (2.4) بنية المعمارية للشبكات العصوبيه

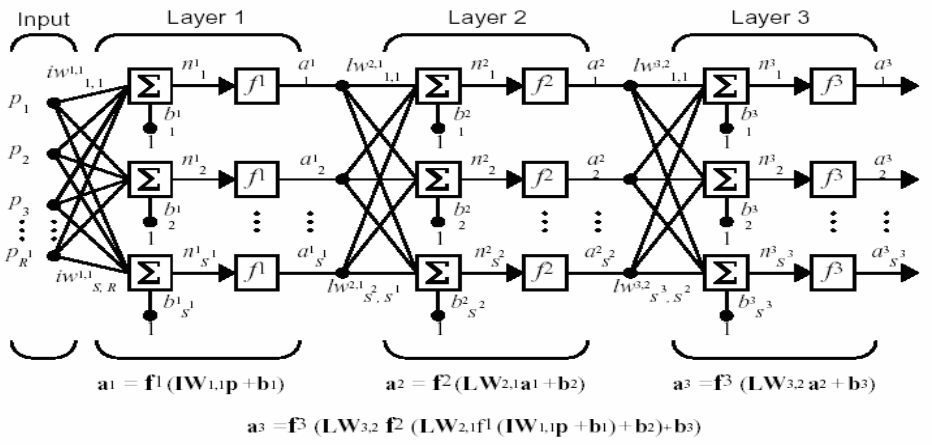
كل عصبون يحوي وصلة جامع بحيث تقوم كل وصلة تقوم بجمع الدخل الموزون مع الإزاحة لتشكيل الخرج العددي للعصوب. وفي النهاية تشكل مركبات خرج طبقة العصوبات تشكل شعاع الخرج مصفوفة من عمود واحد كما هو موضح أدناه :

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ & & & \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

4.4 أنواع الشبكات العصوبيه الاصطناعية

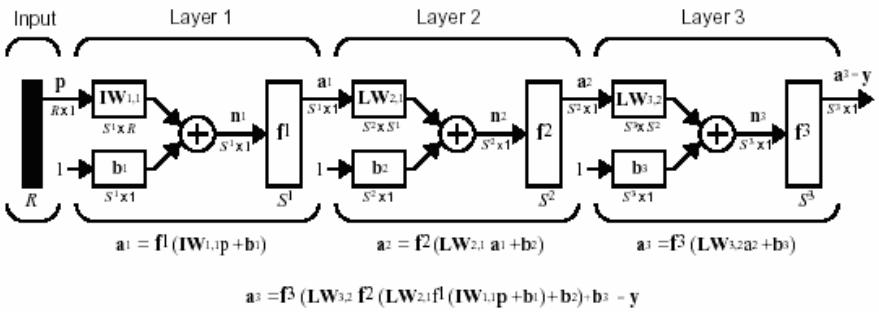
وللشبكات العصوبيه الاصطناعية عدة أنواع ومنها:

1. الشبكة ذات الطبقات المتعددة الأمامية :



شكل (3.4) الشبكة ذات الطبقات المتعددة الأمامية

الشبكة ذات الطبقات المتعددة الأمامية تجمع الخلايا في طبقات. بحيث تنتقل الإشارات من طبقة الدخل (Input layer) في اتجاه طبقة الخرج(Output layer) خلال توصيلات أحادية الاتجاه تكون التوصيلات من خلايا طبقة إلي خلايا الطبقة التالية لها وليس من نفس الطبقة . يمكن أن نرسم الشبكة الثلاثية المبينة في الشكل السابق باستخدام الرسم المختصر التالي :



شكل (4.4) الشبكة ذات الطبقات المتعددة الأمامية باستخدام الرسم المختصر

الشبكات متعددة الطبقات فعالة وخاصة الشبكات المكونة من طبقتين فهي منتشرة بشكل واسع لأنها تستطيع حل المشاكل المعقدة المعقدة ولكن تدريبها يستغرق وقتا أطول.

2. الشبكات ذات التغذية الخلفية:

هذا النوع من الشبكات يتكون من من طبقة واحدة من النيرونات وكل عصوب يعود خرجه إلى دخل كل العصوبات المتبقية. بحيث وقد يكون هناك تغذية خلفية ذاتية أي أن خرج العصوب يعود إلى دخله وهذا غير منتشر لعدم تحقيق الأهداف الحيوية من خلال شبكات أمامية *(2006Wikpedia)*.

5.4 خصائص وإمكانات الشبكات العصوبيه الاصطناعية

وسنوضح في الفقرة القادمة خصائص وإمكانات الشبكات العصوبيه الاصطناعية:

1. التعميم (Generalization):

المقصود بهذا المصطلح قدرة الشبكة على التعرف على أشياء لم تتدرب عليها وذلك عن طريق الاستعانة بأشياء شبيه تعلمت منها.

وهذه الخاصية تتيح من تمثيل الدوال بشكل مرتب وإكمال الباقي من خلال التنبؤ بتصرفات سابقة.

1. التكيف (Adaptivity):

يكمن للشبكات تغير الأوزان في الوصلات للتكيف مع البيئة المحيطة وإعادة تدريبها لتعامل مع الاختلافات الطارئة في محيط العمل .

2. تجاوز الأعطال (Fault Tolerance):

الشبكات العصوبيه لها القدرة على تجاوز الأخطاء في بعض الأجزاء وذلك في حالة تقطعت بعض التوصيلات بين الخلايا عندئذ لا تتأثر الشبكة بالكامل لان المعرفة تكون موزعة في الشبكة المكونة من عدد كبير من المعالجات المتوازنة. يمكن تحقيق هذه التقنية باستخدام (VLSI) *(د.أحمد عبدا لله إمام 2004)*

6.4 تطبيقات الشبكات العصوبية الاصطناعية

الشبكات العصوبيه انتشرت بشكل واسع ودخلت في عدت مجالات منها:

1. معالجة الإشارة (Signal processing):

من أهم التطبيقات في هذا المجال حيث سمح هذا التطبيق بإزالة الضوضاء عن الخطوط الهاتفية.

2. ضغط البيانات (Data compression):

هي عملية تحويل المتجهات الدخل ذات البعد n إلى متجهات ذات خرج ذات بعد m حيث $m<n$ وهذه العملية توفر في وسائط تخزين البيانات .

3. التشخيص:

هو تطبيق منتشر بشكل واسع في مجالات الطب والهندسة والأنظمة الخبيرة .

4. التحكم:

تستخدم الشبكات العصوبيه في مجالات التحكم ومنها التحكم بالمركبات وبالانسان الآلي وغيرها من التطبيقات المهمة .

5. التنبؤ (Forecasting) :

هذه التطبيقات تستخدم في التنبؤ بالأحوال الجوية وتستخدم في سوق العمل بالتنبؤ بالأحوال الاقتصادية .

(د.أحمد عبدا لله إمام2004).

7.4 طرق تعليم الشبكة العصوبيه الاصطناعية

تتعلم الشبكة بإعطائها مجموعة من الأمثلة التي تكون مختارة بشكل دقيق مما يسهم في تعلم الشبكة بصورة أسرع وتسمي الأمثلة بفئة التدريب.

تقسم طرق التدريب إلي قسمين حسب فئة التدريب:

1. التعليم المراقب بواسطة معلم (Supervised Learning of ANN's):

فكرة هذه الطريقة تنبني على عرض البيانات التدريبية على الشبكة هيئة زوج من الأشكال وهما الشكل المدخل (input)والشكل المستهدف(target).

2. التعليم غير المراقب بدون معلم (Unsupervised learning)

تعتمد هذه الطريقة على متجه المدخلات فقط من دوت اللجوء لعرض الهدف على الشبكة ويطلق عليها طريقة التعليم الذاتي حيث تبني الشبكات العصوبيه الاصطناعية طرق التعليم على أساس قدرتها على اكتشاف الصفات المميزة لما يعرض عليها من أشكال وأنساق وقدرتها على تطوير تمثيل داخلي لهذه الأشكال وذلك دون معرفة مسبقة وبدون عرض أمثلة لما يجب عليها أن تنتجه *(2006Wikpedia)*.

8.4 خوارزميات تعليم الشبكة العصوبيه الاصطناعية

في مرحلة التعليم لابد من مراعاة الأوزان لأنها بمثابة المعلومات الأولية ومن هذا المنطلق تستخدم الخوارزميات حسب نوع الشبكة، من أهم هذه الخوارزميات خوارزمية الانتشار العكسي (Back Propagation Algorithm)التي تستخدم في تدريب الشبكات العصوبيه كاملة الارتباط وذات التغذية الأمامية ومتعددة الطبقات وغير الخطية.

ويتم تنفيذ هذه الخوارزمية من خلال مرحلتين:

1. مرحلة الانتشار الأمامي (Feed forward Back Propagation)
2. مرحلة الانتشار العكسي (Back Propagation)

أولاً : مرحلة الانتشار الامامي

تبدأ هذه المرحلة بعرض الشكل المدخل للشبكة من دون التغير في الأوزان وذلك بتخصص كل عنصر معالجة من طبقة عناصر الإدخال لأحد مكونات الشعاع الذي يمثل الدخل ما يؤدي لاستثارة لوحدات طبقة الإدخال مما يؤدي لانتشار أمامي لتلك الاستثارة عبر بقية طبقات الشبكة.

ثانيا : مرحلة الانتشار العكسي

في هذه المرحلة يتم ضبط أوزان الشبكة، إن خوارزمية الانتشار العكسي القياسية هي خوارزمية الانحدار التدريجي (gradient descent algorithm)التي تتيح لأوزان الشبكة أن تتحرك على الجانب السلبي من تابع الأداء. إن مهمة الانتشار العكسي تعود إلى الطريقة التي يتم بها حساب الميل لطبقات الشبكة المتعددة اللاخطية، وتتم في أحد المراحل إعادة انتشار الإشارة من الخرج إلى الدخل بشكل عكسي، ويتم خلالها ضبط أوزان الشبكة.

توجد طريقتان لحساب الانحدار التدريجي :

أولاً : النظام التزايدي (Incremental mode) :

في هذه الطريقة يتم حساب الميل ومن ثم تعدل الأوزان بعد كل دخل يعطى للشبكة.

ثانياً : نظام الدفعة الواحدة (Batch mode) :

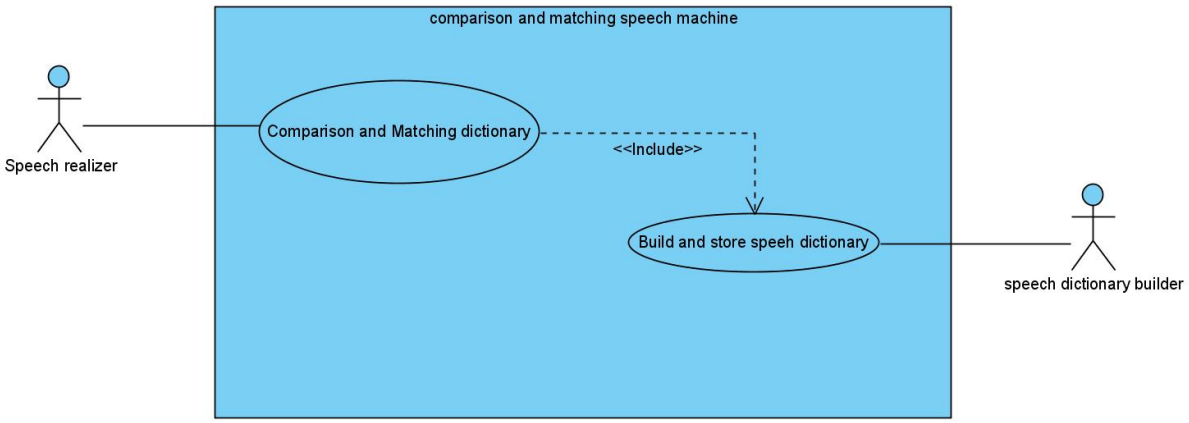
في هذا النمط تزود الشبكة بكل أشعة الدخل قبل القيام بعملية تحديث الأوزان ويمكن القول أن الأوزان والانحيازات في هذه الطريقة تعدل بعد تزويد الشبكة بكامل مجموعة التدريب *(Wikpedia2006)*.

9.4 تحليل آلة مقارنة و ربط الكلام

متطلباتها :

1. أن تأخذ الآلة دخلها من قاموس يصف كل نطق ، و تكراراته التي هي خرج آلة تشفير الكلام.
2. أن لا يشترط القاموس عدد محدد من التكرارات لكل نطق .
3. معرفة حجم القاموس .
4. إضافة ، حذف ، تعديل ، أي نطق في القاموس و أي تكرار لنطق .
5. أن يكون تدريب الشبكة على القاموس معقول الوقت .
6. أن تكون إعداد الشبكة قابلة للتغيير .
7. أن لا تختلط القواميس إذا إستخدمنا أكثر من قاموس .
8. أن تخرج نتيجة المقارنة بإسم النطق و النسبة التي أدت إلى إختياه .
9. و إذا أمكن إضافة تكرار لنطق بعد التدريب .
10. و إذا أمكن إضافة نطق جديد بعد التدريب .

مخطط حالة الإستخدام التالي يوضح الآلة في صورتها البسيطة بعد أن درست المتطلبات جيداً



شكل (5.4) مخطط حالة الاستخدام لآلة مقارنة و ربط الكلام

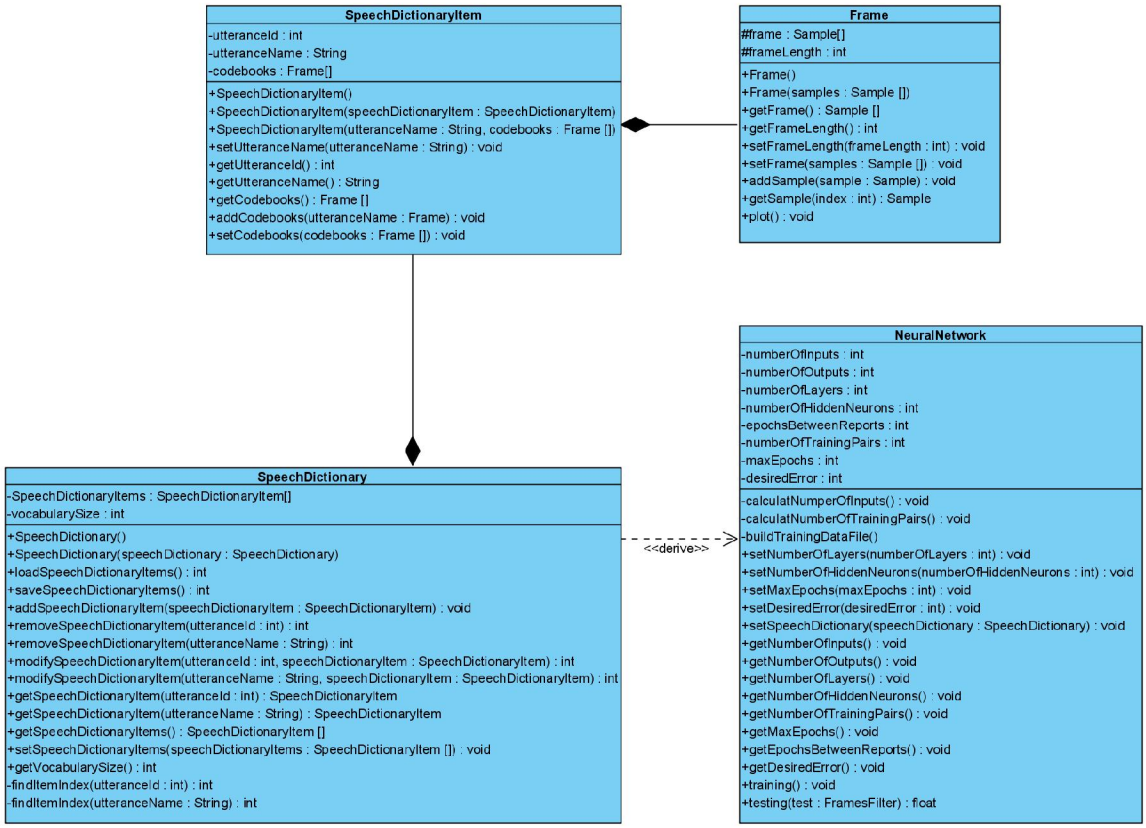
وعند تحليل الكائنات الواردة في المخطط و علاقاتها ، يتضح أن مدرك الكلام (speech realizer) هو الذي يقوم بعمليات الربط و المقارنة ، إذاً لابد و أن يحمل بداخله كائن الشبكة العصوبيه (Neural Network) ، و باني قاموس الكلام (speech dictionary builder) لكي يبني قاموسه عليه أولا أن يبني كل نطق بالقاموس الذي يتطلب بناء كل تكرار لنطق ، وعليه فإنه يوجد عدد من الكائنات داخل باني القاموس . فالجدول التالي يصف كل الكائنات المستنتجة من المخطط .

جدول (1.4) تحليل كائنات آلة ربط ومقارنة الكلام

الكائن	وظيفته
(speech dictionary builder) Speech dictionary	بناء و تخزين قاموس الكلام .
speech dictionary item	بناء و تخزين نطق واحد (النطق يتكون من عدة تكرارات) .
utterance	بناء و تخزين أحد تكرارات النطق .
(speech realizer) neural network	ربط و مقارنة قاموس الكلام .

يحتاج كائن الشبكة العصوبيه لكائن قاموس الكلام لكي يمده بالبيانات المراد ربطها و مقارنتها فيما بعد ، هذا يعني أن العلاقة (composition) ، لكن المطلب (9) و (10) يهدمان العلاقة و يوصلانها من جديد على أنها وراثة هذا لأن إضافة نطق من إختصاص القاموس ، فتصبح حينها الشبكة العصوبيه كالقاموس قابلة للتوسع . أما العلاقة بين القاموس و النطق و النطق و تكراراته هي (composition) ، سؤال مهم جداً يطرح نفسه هنا لماذا لم تتغير العلاقات بسبب المطلب (9) ؟ ، لأن معالجته تمت ضمنياً عند معالجة المطلب (10) ، كيف ؟ عند إرادة إضافة تكرار ينشأ نطق جديد بنفس بيانات النطق القديم المراد زيادة تكراراته لكن بإضافة التكرار الجديد ، وهذا يدمج المطلب (9) في المطلب (10) .

لماذا لم يجبر المطلب (6) على إنشاء كائن جديد كما فعل مع آلة إدخال و إخراج الكلام ؟ لأن إعدادات الشبكة لا تؤثر على القاموس كما تفعل إعدادات القراءة و الكتابة ، و فيما يلي مخطط الفئات النهائي لآلة مقارنة و ربط الكلام



شكل (6.4) مخطط فئات آلة ربط ومقارنة الكلام

10.4 القضايا التصميمية و التطبيقية لآلة ربط و مقارنة الكلام

يوجد الكثير من المكتبات المتخصصة في الشبكات العصوبيه المنتشرة على الإنترنت ، التي تتباين بين السعر و السرعة و التطبيق المستخدم و المعمارية و ... ، قمنا بإختيار مكتبة (FANN) (Fast Artificial Neural Network) لزعم أنها سريعة حسب إسمها وهذا ما تبين فعلا ، الفقرة القادمة تصف طريقة إستخدامها بشكل مختصر .

1.10.4 إستخدام (FANN)

بنية هذه المكتبة بلغة (C) و هي مفتوحة المصدر ، تتمتع بطريقة سهلة للإستخدام و واضحة التسلسل ، وذات مرونة ملموسة ، فيما يلي طريقة التدريب و الإختبار بها .

1.1.10.4 تدريب و إختبار الشبكة بإستخدام (FANN)

تعمل الشبكة على نمطين مختلفين ، نمط التدريب و نمط التشغيل ، ويمكن أن يعمل النمطين على نفس البرنامج ، لكن صانعيها يوصوا بفصلهما .

نبدأ أولاً بضبط إعدادات الشبكة و هذه خطوة حساسة ستؤثر على سرعة التدريب و دقة النتائج ، تضبط إعدادات الشبكة بمتغيرات من النوع (const unsigned int) وهي تمثل عدد المدخلات و عدد المخرجات و عدد الطبقات إلى آخره .

ثانياً نقوم بإنشاء الشبكة بإستخدام الدالة (fann_create_standard) التي تستقبل جزء من الإعدادات السابقة ، تقوم الدالة بإنشاء شبكة من النوع (BP) (Back Propagation) تعيدها في متغير خصص لذلك من النوع (struct fann) .

ثالثاً نقوم بتنشيط الشبكة بإستخدام (fann_set_activation_function_hidden ، fann_set_activation_function_output) .

رابعاً نهيئ بيانات التدريب وهي إما من ملف وفي هذه الحالة يبنى ملف نصي اول ثلاثة أرقام به تمثل عدد أزواج التدريب و عدد المدخلات و عدد المخرجات على الترتيب ، أو من مصفوفة في من النوع (fann_type) .

خامسا تدريب الشبكة بواسطة الدالة المقابلة لنوع بيانات التدريب لديك (fann_train_on_file ، fann_train_on_data) .

سادسا نحفظ التدريب بواسة الدالة (fann_save) . ونحررها من الذاكرة بإستخدام الدالة (fann_destroy) .

سابعا تبدأ مرحلة الإختبار حيث نقوم بإنشاء الشبكة من جديد و هذه المرة من ملف التدريب المنشأ في الخطوة السابقة وذلك بواسطة الدالة (fann_create_from_file) ، ثم نقوم بتهيئة عينة الإختبار و وضعها في مصفوف بعدد المدخلات من النوع (fann_type) ، بعدها ننادي دالة الإختبار (fann_run) التي تعيد مصفوفة من النوع (fann_type) ممثلة الخرج .

عند تطبيق الخطوات السابقة تكون قد حصلت على أقل برنامج ممكن لتدريب و إختبار الشبكة ، التي تتمتع بمكتبة ضخمة من الدوال من شأنها تحسين النتائج و تسريع التدريب .

2.10.4 المشاكل التي واجهت تنفيذ مخطط الفئات

تعتمد الشبكة على عدد من المدخلات ثابت بينما يكون القاموس المراد تمريره للشبكة ذا أطول متباينة النطق (ق) لاتساوي (السباق) إذا لابد من تكمة (ق) لتساوي (السباق) فبماذا تكمل ؟ جربنا الأصفار ثم جربا الفكرة التالية فكان التحسن بنسبة 9% في دقة خرج الشبكة :

قم بإدخال صوت فارغ (كله قيم العينة به تمثل نصف الموجة مثلاً 128 عند إستخدام 8bit) في آلة تشفير الكلام (LPC ثم VQ) ، تم خذ الخرج و الذي هو عبارة عن تسلسل منتظم من الأرقام لتكمل به النطق الناقص .

الباب الخامس(41-49)

تصميم و تنفيذ و اختبار آلة تمييز الكلام

1.5 مقدمة

2.5 تحليل آلة تمييز الكلام

3.5 القضايا التصميمية و التنفيذية لآلة تمييز الكلام

4.5 طريقة إستخدام آلة تمييز الكلام

5.5 الاختبارات و النتائج لآلة تمييز الكلام

1.5 مقدمة

يأتي هذا الباب لكي يدمج كل الآلات في الأبواب السابقة في آلة واحدة وهي آلة تمييز الكلام ، وكما هو الحال في تركيب الآلات الفيزيائية مع بعضها نحتاج لبعض التعديلات لتتوافق كلها لخدمة الغرض الأساسي . سيوضح هذا الباب كيفية هذا التوافق و القضايا التطبيقية للآلة النهائية مع طريقة الاستخدام ، و الاختبارات التي أجريت عليها مع نتائجها ، الفقرة القادمة ستوضح النظرة التحليلية لآلة تمييز الكلام .

2.5 تحليل آلة تمييز الكلام

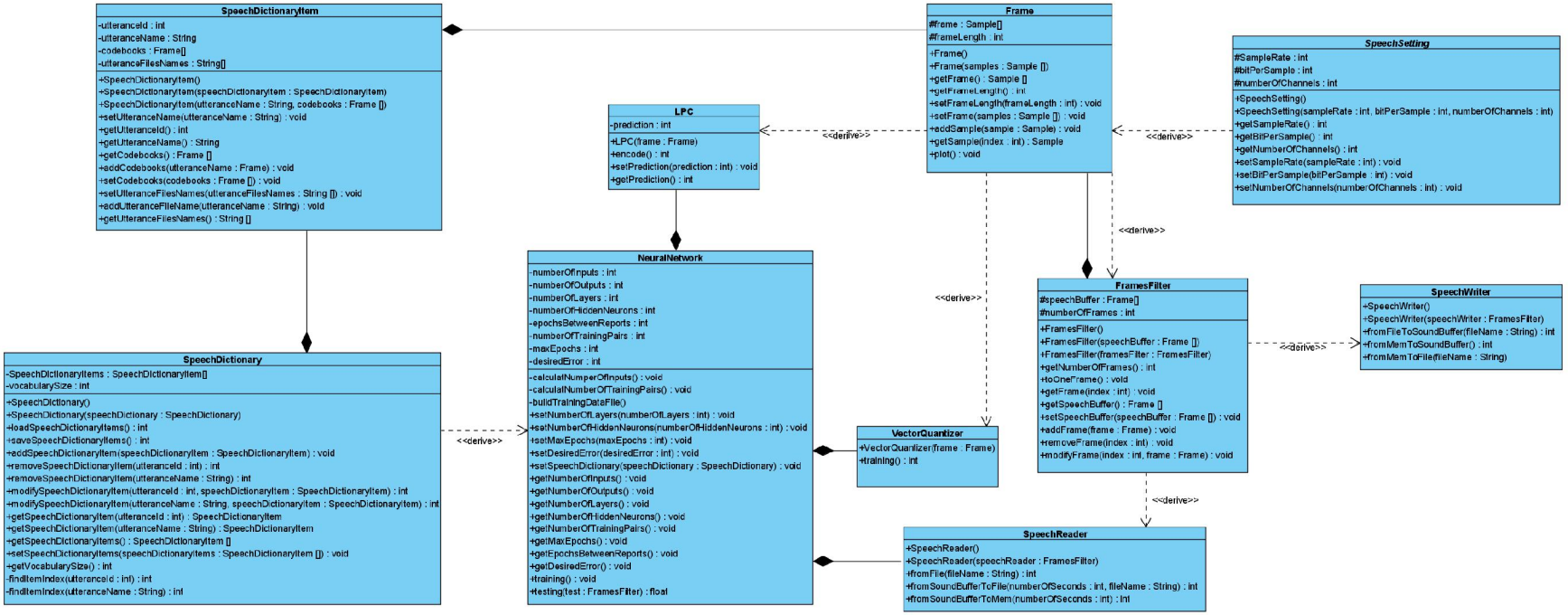
المتطلبات لهذه الآلة هي أهداف النظام و هي كما يلي :

1. أن تكون قادرة على تمييز النطق حسب القاموس .
2. أن تخفي تفاصيل تقسيم الكلام و ضغطه عن المستخدم (شفافة) .
3. أن لا تكون مبنية من أجل تطبيق بعينه .
4. أن تكون هناك أكثر من طريقة لمعرفة النطق المنطوق .

وعلى خلاف الآلات السابقة لن يتطرق لمخطط حالة الاستخدام و لا لجدول تحليل الكائنات لأن كل الكائنات معروفة ، لكن العلاقة بينهم قد تتغير بسبب المطالب وهذا ما اعتدنا عليه في التحليلات السابقة . و بالنظر لآلة الربط و المقارنة نجد أنها يقع عليها عبء تمييز الكلام لكنها تعتمد في دخلها على شفرة مضغوطة و المطلب (2) يجبرنا لتعديل ذلك إلى ملفات صوتية لكن التعامل مع الملفات الصوتية من اختصاص آلة إدخال وإخراج الكلام ، الفقرة القادمة ستوضح كيف ستتوافق الآلات مع بعضها لحل هذه المشكلة .

1.2.5 توافقية الآلات

المطلوب هو القيام بإرسال ملفات لآلة الربط و المقارنة بدلاً من إرسال شفرات ، تأخذ آلة الربط و المقارنة الملفات و تنادي آلة إدخال و إخراج الكلام التي تقسم الملف لإطارات يرسل كل إطار لألة تشفير الكلام ثم تدمج الإطارات مرة أخرى بعد ضغطها وهو الدخل القديم لآلة الربط و المقانة ، يعني ذلك أن لكل تكرار لنطق في القاموس ما يقابله من ملف و ما يقابله من شفرة . هذا يعني أن العلاقة بين آلة الربط و المقارنة و باقي الآلات هي علاقة (composition) لأنها تناديهم فقط لإتمام و ظيفة ما ، فيما يلي المخطط النهائي لآلة الربط و المقارنة المعدلة و هي آلة تمييز الكلام .



شكل (1.5) مخطط فئات آلة تمييز الكلام

3.5 القضايا التصميمية و التنفيذية لآلة تمييز الكلام

من الواضح أن الآلة تأخذ زمن في الإقلاع و هو زمن ملء الشفرات المقابلة لكل نطق ، فإذا فرضنا أن القاموس به 10 مفردات كل مفردة تمثل نطق كل نطق له 40 تكرار مثلاً فقد تأخذ الآلة دقيقة للمرور على كل الملفات التي عددها 400 و تشفيرها إطار إطار .

4.5 طريقة إستخدام آلة تمييز الكلام

كل العناء السابق كان من أجل تبسيط طريقة الأستخدام ، وهي كمايلي :

1. إنشاء قاموس من النوع (SpeechDictionary) .
2. إنشاء عنصر لقاموس من النوع (SpeechDictionaryItem) ثم القيام بملأ البيانات .
3. القيام بإضافة العنصر للقاموس بإستخدام الدالة (addSpeechDictionaryItem) و تكرار العملية إلي أن تتم كل المفردات المراد تمييزها .
4. القيام بإنشاء شبكة عصبية من النوع (NeuralNetwork) و إرسال القاموس في دالة البناء ، ستأخذ هذه العملية دقائق على حسب حجم القاموس .
5. تدريب الشبكة بإستخدام الدالة (trainingWithFANN) .
6. تجهيز عينة الإختبار و ذلك بإنشاء قارئ من النوع (SpeechReader) ثم القيام بالقراءة من الميكرفون بإستخدام الدالة (fromSoundBufferToMem).
7. القيام بإرسال القارئ لدالة إختبار الشبكة بإستخدام الدالة(testingWithFANN) ، خرج هذه الدالة نتيجة التمييز وهم الرقم الفريد لكل نطق المدخل عند ملء بيانات عنصر القاموس في الخطوة (2).

5.5 الاختبارات و النتائج لآلة تمييز الكلام

أجريت الاختبارات على قاموس به 10 مفردات ، الجدول التالي يوضح كل مفردة و عدد تكرارات نطقها و رقمها الفريد .

جدول (1.5) قاموس إختبار آلة تمييز الكلام .

رقم المفردة	المفردة	تكرارات نطقها
1	أصغر	38
2	السباق	42
3	صمغ	40
4	غ	38
5	غرس	39
6	غين	34
7	ق	43
8	قاف	36
9	قال	39
10	وقف	37

نلاحظ أن عدد تكرارات نطق المفردات متباينة ، وهذا حدث نتيجة لاستبعاد كل نطق رديء التسجيل ، الفقرة التالية تصف ظروف التسجيل .

سجلت العينات من مدرستي المقرن أساس للبنات و علي عبد اللطيف للبنين ، من تلاميذ الصف الثاني ، في مكتب تقليدي غير مهيأ لعملية التسجيل ، سجلت العينات بإعدادات ثابت و هي مبينة في الفصل التالي .

1.5.5 إعدادات إختبار آلة تمييز الكلام

اولاً : إعدادات التسجيل :

- عدد البتات في العينة 8bit .
- تردد العينة 8kHz .
- عدد القنوات (mono) (واحدة) .
- تقنية تحديد النطق (end point detection) (تعمل) .

ثانياً : إعدادات الشبكة العصوبية :

- عدد المدخلات يحدده النظام بناءاً على أطول تكرار لنطق .
- مخرج واحد هو رقم المفردة الفريد بعد تعديله ليناسب الشبكة .
- عدد الطبقات ثلاثة .
- عدد العصوبيات الخفية جربت له أكثر من قيمة ما بين (3 - 100) .
- عدد أزواج التدريب (386) يحدده النظام بجمع كل التكرارات للنطق .

2.5.5 اختبار أساليب تدريب الشبكة العصوبية

1. تدريب كل القاموس دفعة واحدة

هذه الأسلوب ذو دقة عالية خصوصا عند استخدم عدد كبير من العصوبيات الخفية مثلاً (100) ، لكن قد يأخذ التدريب أياماً خصوصاً إذا كان حجم القاموس كبير (10 او اكبر) ، و عندها لا تستطيع التفكير بإعادة التدريب إذا حدث تعديل في القاموس .

بعد تدريب الشبكة على القاموس الموضح في الجدول (1.5) الذي استغرق تدريبه 29 ساعة بعدد من العصوبيات الخفية يساوي (3) فقط و توقف التدريب عند عدد من الدورات يساوي (65250000) ، وبأخذ نطق لم يدرب من كل مفردة و مقارنته مع تسجيل مباشر من الميكرفون كانت النتائج كما يلي :

جدول (2.5) نتائج تدريب كل القاموس دفعة واحدة .

المفردة	الدقة باستخدام الملفات	الدقة باستخدام الميكروفون
أصغر	%90.5	%73.2
السباق	%95.3	%77.5
صمغ	%91.7	%71.3
غ	%79.2	%50.8
غرس	%88	%62.4
غين	%81.3	%56
ق	%82	%59.4
قاف	%90	%66.7
قال	%92	%70.1
وقف	%93	%74.4
دقة الشبكة	%88.3	%66.1

لوحظ الفرق في دقة الميكرفون عن الملفات التي فاجأتنا للوهلة الأولى ، لكن سرعان ما تذكرنا أن تسجيلنا للعينات كان قبل إنشاء نظامنا ، وعزونا ذلك لإختلاف طرق التسجيل من برنامج لآخر ، لوجود دوال مطبقة على الصوت المسجل مثل التطبيع و إزالة التشويش و غيرها .

وبهذا الزمن الطويل في التدريب لا نستطيع تجربة عدد أكبر من العصوبيات في الطبقة ، الذي من شأنه زيادة الزمن .

2. تدريب كل نطق مع نطق مولد عشوائية

النطق المولد عشوائياً ليس عشوائياً بالمرة فهو يعتمد على محكاة التكرارات للعينات (samples) في أي نطق آخر موجود بالقاموس . هذا الأسلوب سريع جداً و ذا دقة منخضة جداً ، نعزي السبب لجهلنا بطبيعة النطق البشري الذي لابد و أن يؤثر في طريقة توليد النطق العشوائي .

بعد تدريب الشبكات على القاموس ، استغرق التدريب 3 دقائق بعدد من العصوبيات الخفية يساوي (3) فقط ، كانت النتائج كما يلي :

جدول (3.5) نتائج تدريب كل نطق مع نطق مولد عشوائية

المفردة	الدقة باستخدام الملفات	الدقة باستخدام الميكروفون
أصغر	%55.4	%49.1
السباق	%70.1	%58.5
صمغ	%40.9	%43.3
غ	%44.2	%36.8
غرس	%59.6	%40.4
غين	%38.3	%20
ق	%47	%30.7
قاف	%60	%39.7
قال	%57	%46.2
وقف	%53	%19.4

3. تدريب كل نطق مع النطق الذي يليه

هنا لابد من مراعاة الترتيب مثلاً لديك قاموس يحتوي الكلمات التالية (ق ، وقف ، غ ، السباق) إذا درب هذا القاموس بهذا الشكل ستنتج مشاكل في التمييز بين (ق ، غ) ، فلابد من إعادة ترتب القاموس يدوياً على أساس قرب النطق ، تبدأ المشكلة بالتفاقم إذا كانت هنالك ثلاثة كلمات شبيه (ق ، غ ، ع) ، هنا لابد من تدريب كل نطق مع الكلمتين اللتين تليه و هكذا وصولاً للإسلوب الأول ، هذا الأسلوب سريع ، ويعطي نتائج مقبولة إلى أن التدخل اليدوي و حجم القاموس الكبير يكبلانه .

هنا سنعيد ترتيب القاموس الموضح بالجدول (4.5) ليصبح بالشكل التالي :

جدول (4.5) قاموس الإختبار المعدل

رقم المفردة	المفردة
1	ق
2	غ
3	غين
4	قاف
5	السباق
6	غرس
7	أصغر
8	وقف
9	قال
10	صمغ

بعد تدريب الشبكات (لكل مفردة مع التي تليها) على القاموس ، استغرق التدريب 12 دقيقة بعدد من العصوبيات الخفية يساوي (3) فقط ، كانت النتائج كما يلي :

جدول (5.5) نتائج تدريب كل نطق مع النطق الذي يليه

المفردة	الدقة باستخدام الملفات	الدقة باستخدام الميكروفون
ق	%77.7	%50.6
غ	%70.3	%49
غين	%79	%72.2
قاف	%61	%53.1
السباق	%91.6	%80.8
غرس	%.74	%66.9
أصغر	%60.3	%58.5
وقف	%67.8	%48.9
قال	%78.6	%70
صمغ	%87	%67.2

4. تدريب كل نطق مع كل نطق

عدا نفسه طبعاً ، عدد الشبكات المولدة أمام القاموس تساوي ((حجم القاموس * حجم القاموس) – حجم القاموس) ، ذا سرعة مقبولة و دقة مقبولة ، إلا انه ينهك الذاكرة ويعقد خوارزمية الإختبار ، هذا الأسلوب أجبرنا على إختياه نتيجة لميزاته التوسعية التي تناسب مشكلتنا .

بعد تدريب الشبكات على القاموس ، استغرق التدريب 42 دقيقة بعدد من العصوبيات الخفية يساوي (3) فقط ، كانت النتائج كما يلي :

(6.5) نتائج تدريب كل نطق مع كل نطق

المفردة	الدقة باستخدام الملفات	الدقة باستخدام الميكروفون
أصغر	%69	%63.1
السباق	%90.4	%72.5
صمغ	%61.3	%73.3
غ	%69	%60.8
غرس	%86.7	%69.4
غين	%71.6	%63.5
ق	%60.8	%59.4
قاف	%60	%66.7
قال	%75.3	%50.1
وقف	%68	%44.4

1.2.5.5 خلاصة نتائج اختبار أساليب تدريب الشبكة العصوبية

الجدول التالي يقارن بين الأساليب الأربعة في سرعة التدريب و دقة النتائج .

جدول (7.5) خلاصة نتائج اختبار أساليب تدريب الشبكة العصوبية

الأسلوب	زمن التدريب	متوسط دقة النتائج
تدريب كل القاموس دفعة واحدة	29 ساعة	%88
تدريب كل نطق مع نطق مولد عشوائية	3 دقائق	%52
تدريب كل نطق مع النطق الذي يليه	12 دقيقة	%74
تدريب كل نطق مع كل نطق	42 دقيقة	%71

ويتضح من الجدول أن أفضل نتيجة تمييز في التجارب السابقة هي تدريب كل القاموس دفعة واحدة و التي اعطت نتيجة 88% و لكن زمن التدريب كان عالياً 29 ساعة مقارنة بالأساليب الأخرى .

تقنية الشبكات العصوبية هي معالجة متوازية و تم تطبيقها على جهاز حاسوب شخصي و هو يعتمد على المعالجة المركزية ، فإذا تم تدريب الشبكات العصوبية على اجهزة معالجة متوازية ستعطي نتائج افضل في زمن التدريب .

التوصيات

الاسم الأدق لآلة تمييز الكلام المنتجة في هذا البحث هو آلة تمييز المفردات المنفصلة ذلك لأنها لا تستطيع تمييز المفردات إذا كانت في وسط جملة ، نوصي بإضافة آلة تقوم بتقطيع الجملة إلى مفردات والتكن آلة تقطيع الكلام ، ثم دمجها مع آلة تمييز المفردات المنفصلة لتوليد آلة تمييز المفردات المتصلة .

الملاحق

ملحق أ أجزاء من الشفرة المصدرية

جزء من الشفرة المصدرية لآلة إدخال و إخراج الكلام

```
#include <stdio.h>
#include <stdlib.h>
#include "..\AudioInterface\portaudio.h"

#include <stdlib.h>
#include <stdio.h>
#include <string.h>


#include <engine.h>//matlab engine interface to showWithMatlab method

using namespace System;
using namespace IO;
using namespace Collections;


//default sample recording setting
#define NUM_CHANNELS    (1)
typedef float SAMPLE;
typedef unsigned char SAMPLE_B;
#define PA_SAMPLE_TYPE  paUInt8
#define SAMPLE_SILENCE (64)

//sample buffer declaration
typedef struct{
    int          frameIndex;  /* Index into sample array. */
    int          maxFrameIndex;
      SAMPLE_B      *recordedSamples;
}PaTestData;


ref class SpeechSetting{
protected:
      //sample recording setting
      int SAMPLE_RATE;

      //const default sample recording setting
      /* #define DITHER_FLAG     (paDitherOff)  */
      int DITHER_FLAG;
      !SpeechSetting(void);

public:
      SpeechSetting(void);
      ~SpeechSetting(void);
      //gets methods
      int getSampleRate(void);
      //int getBitPerSample(void);
      //int getNumberOfChannels(void);

      //sets methods
      void setSampleRate(int);
      //void setBitPerSample(int);
      //void setNumberOfChannels(int);
};
```

```
ref class Frame : public SpeechSetting{
protected:
	ArrayList ^frame;
	int samplesCounter;//current sample number
	int frameLength;//max number of sample per frame

	!Frame(void);//Finalizers

public:
	//Constructors
	Frame(void);
	Frame(ArrayList ^);
	Frame(array<SAMPLE> ^);
	Frame(Frame ^);

	~Frame(void);//Destructors

	//gets methods
	ArrayList ^getFrame(void);
	array<SAMPLE> ^toSamples();
	int getFrameLength(void);
	int getSamplesCounter(void);
	SAMPLE getSample(int);

	//sets methods
	int setFrame(ArrayList ^);
	int setFrame(array<SAMPLE> ^);
	void setFrame(Frame ^frame);
	int addSample(SAMPLE);
	void setFrameLength(int);
	void setMaxFrameLength(void);

	//display methods
	int showWithMatlab(void);

	//remove methods
	void removeSample(int);

	//modify methods
	void modifySample(int,SAMPLE);
};
```

```
ref class FramesFilter : public Frame{
private:
	//pre-filtering methods its help utteranceDetection method
	array<int> ^startPointDetection(SAMPLE);//take max value to start ,
return index of begin sample and frame index [fameIndex,sampleIndex] if not
found [-1,-1]
	array<int> ^endPointDetection(SAMPLE);//take max value to end ,
return index of last sample and frame index [fameIndex,sampleIndex] if not
found [-1,-1]
protected:
	//main speech matrix
	ArrayList ^speechBuffer;//its have Frame objects
	int numberOfFrames;
	//Finalizers
	!FramesFilter(void);
```

```
public:
	//Destructors
	~FramesFilter(void);

	//Constructors
	FramesFilter(void);
	FramesFilter(FramesFilter ^);
	FramesFilter(ArrayList ^);
	FramesFilter(array<Frame^> ^);

	//sets methods
	void setSpeechBuffer(ArrayList ^);
	void setSpeechBuffer(array<Frame^> ^);
	void addFrame(Frame ^);

	//gets methods
	int getNumberOfFrames(void);
	ArrayList ^getSpeechBuffer(void);
	Frame ^getFrame(int);

	array<Frame^> ^toFrames(void);

	//remove methods
	void removeFrame(int);

	//modify methods
	void modifyFrame(int,Frame ^);

	//Collector methods
	int toOneFrame(void);

	//pre-filtering methods
	//void speechNormalization(void);
	//void speechNoiceReduction(void);
	int utteranceDetection(SAMPLE);//its trim speech wave , and reframing
};



/*
	SpeechReader class is a part of system that responsible to providing
a system with input with multiway ,
	also is included Pre-Filtering function ,
	the main output of this class is a speech buffer , it has one pure
utterance,
*/

ref class SpeechReader : public FramesFilter{
protected:
	//Finalizers
	!SpeechReader(void);

public:
	//Constructors
	SpeechReader(void);
	SpeechReader(FramesFilter ^);

	//Destructors
	~SpeechReader(void);

	//input ways methods
```

```
	int fromFile(String ^);//the input of method is file name
	int fromSoundBufferToFile(int,String ^);//.raw file
	int fromSoundBufferToMem(int);
	int fromSoundBufferToMem(int,System::Windows::Forms::ProgressBar^);
};

ref class SpeechWriter : public FramesFilter{
protected:
	//Finalizers
	!SpeechWriter(void);

public:
	//Constructors
	SpeechWriter(void);
	SpeechWriter(FramesFilter ^);

	//Destructors
	~SpeechWriter(void);

	//play sound
	int fromFileToSoundBuffer(String ^);//play .wav and .raw only
	int fromFileToSoundBufferWithCopyData(String ^);//play .wav and .raw
only
	int fromMemToSoundBuffer(void);//play
	//save sound
	int fromMemToFile(String ^);

};
```

جزء من الشفرة المصدرية لآلة تشفير الكلام

```
#include "SpeechIO.h"
#include <malloc.h>
#include <math.h>


typedef int INT16_10;
typedef long INT32_10;
typedef float real_10;
typedef double doublereal_10;
typedef INT32_10 integer_10;
//typedef long int integer;
typedef INT32_10 logical_10;
typedef INT16_10 shortint_10;
#define TRUE_ (1)
#define FALSE_ (0)

struct lpc10_encoder_state {
    /* State used only by function hp100 */
    real_10 z11;
    real_10 z21;
    real_10 z12;
    real_10 z22;

    /* State used by function analys */
    real_10 inbuf[540], pebuf[540];
    real_10 lpbuf[696], ivbuf[312];
    real_10 bias;
    integer_10 osbuf[10];  /* no initial value necessary */
    integer_10 osptr;     /* initial value 1 */
    integer_10 obound[3];
    integer_10 vwin[6]  /* was [2][3] */;   /* initial value vwin[4] = 307;
vwin[5] = 462; */
    integer_10 awin[6]  /* was [2][3] */;   /* initial value awin[4] = 307;
awin[5] = 462; */
    integer_10 voibuf[8]      /* was [2][4] */;
    real_10 rmsbuf[3];
    real_10 rcbuf[30]   /* was [10][3] */;
    real_10 zpre;


    /* State used by function onset */
    real_10 n;
    real_10 d__;   /* initial value 1.f */
    real_10 fpc;   /* no initial value necessary */
    real_10 l2buf[16];
    real_10 l2sum1;
    integer_10 l2ptr1;   /* initial value 1 */
    integer_10 l2ptr2;   /* initial value 9 */
    integer_10 lasti;    /* no initial value necessary */
    logical_10 hyst;   /* initial value FALSE_ */

    /* State used by function voicin */
    real_10 dither;   /* initial value 20.f */
    real_10 snr;
    real_10 maxmin;
```

```
    real_10 voice[6]    /* was [2][3] */;   /* initial value is probably
unnecessary */
    integer_10 lbve, lbue, fbve, fbue;
    integer_10 ofbue, sfbue;
    integer_10 olbue, slbue;
    /* Initial values:
      lbve = 3000;
      fbve = 3000;
      fbue = 187;
      ofbue = 187;
      sfbue = 187;
      lbue = 93;
      olbue = 93;
      slbue = 93;
      snr = (real) (fbve / fbue << 6);
      */

    /* State used by function dyptrk */
    real_10 s[60];
    integer_10 p[120]   /* was [60][2] */;
    integer_10 ipoint;
    real_10 alphax;

    /* State used by function chanwr */
    integer_10 isync;

};
struct lpc10_decoder_state {
    /* State used by function decode */
    integer_10 iptold;   /* initial value 60 */
    logical_10 first;   /* initial value TRUE_ */
    integer_10 ivp2h;
    integer_10 iovoic;
    integer_10 iavgp;   /* initial value 60 */
    integer_10 erate;
    integer_10 drc[30]  /* was [3][10] */;
    integer_10 dpit[3];
    integer_10 drms[3];

    /* State used by function synths */
    real_10 buf[360];
    integer_10 buflen;   /* initial value 180 */

    /* State used by function pitsyn */
    integer_10 ivoico;   /* no initial value necessary as long as
first_pitsyn is initially TRUE_ */
    integer_10 ipito;   /* no initial value necessary as long as
first_pitsyn is initially TRUE_ */
    real_10 rmso;   /* initial value 1.f */
    real_10 rco[10];   /* no initial value necessary as long as
first_pitsyn is initially TRUE_ */
    integer_10 jsamp;   /* no initial value necessary as long as
first_pitsyn is initially TRUE_ */
    logical_10 first_pitsyn;   /* initial value TRUE_ */

    /* State used by function bsynz */
    integer_10 ipo;
    real_10 exc[166];
    real_10 exc2[166];
    real_10 lpi1;
    real_10 lpi2;
```

```
    real_10 lpi3;
    real_10 hpi1;
    real_10 hpi2;
    real_10 hpi3;
    real_10 rmso_bsynz;

    /* State used by function random */
    integer_10 j;   /* initial value 2 */
    integer_10 k;   /* initial value 5 */
    shortint_10 y[5];  /* initial value { -21161,-8478,30892,-10216,16950 }
*/

    /* State used by function deemp */
    real_10 dei1;
    real_10 dei2;
    real_10 deo1;
    real_10 deo2;
    real_10 deo3;

};

ref class LPC : public Frame{
private:
      int LPC10_SAMPLES_PER_FRAME;
      int LPC10_BITS_IN_COMPRESSED_FRAME;

      int prediction;

      #define abs(x) ((x) >= 0 ? (x) : -(x))
      #define dabs(x) (doublereal)abs(x)
      #define min(a,b) ((a) <= (b) ? (a) : (b))
      #define max(a,b) ((a) >= (b) ? (a) : (b))
      #define dmin(a,b) (doublereal)min(a,b)
      #define dmax(a,b) (doublereal)max(a,b)

      int lpcini_(void);

      int encode_(integer_10 *voice, integer_10 *pitch, real_10 *rms,
real_10 *rc, integer_10 *ipitch, integer_10 *irms, integer_10 *irc);
      int decode_(integer_10 *ipitv, integer_10 *irms, integer_10 *irc,
integer_10 *voice, integer_10 *pitch, real_10 *rms, real_10 *rc, struct
lpc10_decoder_state *st);

      int chanwr_0_(int n__, integer_10 *order, integer_10
*ipitv,integer_10 *irms, integer_10 *irc, integer_10 *ibits,struct
lpc10_encoder_state *st);
      int chanwr_(integer_10 *order, integer_10 *ipitv, integer_10 *irms,
integer_10 *irc, integer_10 *ibits, struct lpc10_encoder_state *st);
      int chanrd_(integer_10 *order, integer_10 *ipitv, integer_10 *irms,
integer_10 *irc, integer_10 *ibits);

      int analys_(real_10 *speech, integer_10 *voice, integer_10 *pitch,
real_10 *rms, real_10 *rc, struct lpc10_encoder_state *st);
      int synths_(integer_10 *voice, integer_10 *pitch, real_10 *rms,
real_10 *rc, real_10 *speech, integer_10 *k, struct lpc10_decoder_state
*st);

      int prepro_(real_10 *speech, integer_10 *length,struct
lpc10_encoder_state *st);

      //int inithp100_(void);
```

```
      int hp100_(real_10 *speech, integer_10 *start, integer_10 *end,struct
lpc10_encoder_state *st);

      int tbdm_(real_10 *speech, integer_10 *lpita, integer_10 *tau,
integer_10 *ltau, real_10 *amdf, integer_10 *minptr, integer_10 *maxptr,
integer_10 *mintau);
      int rcchk_(integer_10 *order, real_10 *rc1f, real_10 *rc2f);
      int mload_(integer_10 *order, integer_10 *awins, integer_10 *awinf,
real_10 *speech, real_10 *phi, real_10 *psi);
      int onset_(real_10 *pebuf, integer_10 *osbuf, integer_10 *osptr,
integer_10 *oslen, integer_10 *sbufl, integer_10 *sbufh, integer_10
*lframe, struct lpc10_encoder_state *st);
      int dcbias_(integer_10 *len, real_10 *speech, real_10 *sigout);
      int placea_(integer_10 *ipitch, integer_10 *voibuf, integer_10
*obound, integer_10 *af, integer_10 *vwin, integer_10 *awin, integer_10
*ewin, integer_10 *lframe, integer_10 *maxwin);
      int placev_(integer_10 *osbuf, integer_10 *osptr, integer_10 *oslen,
integer_10 *obound, integer_10 *vwin, integer_10 *af, integer_10 *lframe,
integer_10 *minwin, integer_10 *maxwin, integer_10 *dvwinl, integer_10
*dvwinh);
      int preemp_(real_10 *inbuf, real_10 *pebuf, integer_10 *nsamp,
real_10 *coef, real_10 *z__);
      int voicin_(integer_10 *vwin, real_10 *inbuf, real_10 *lpbuf,
integer_10 *buflim, integer_10 *half, real_10 *minamd, real_10 *maxamd,
integer_10 *mintau, real_10 *ivrc, integer_10 *obound, integer_10 *voibuf,
integer_10 *af, struct lpc10_encoder_state *st);
      int lpfilt_(real_10 *inbuf, real_10 *lpbuf, integer_10 *len,
integer_10 *nsamp);
      int ivfilt_(real_10 *lpbuf, real_10 *ivbuf, integer_10 *len,
integer_10 *nsamp, real_10 *ivrc);
      int energy_(integer_10 *len, real_10 *speech, real_10 *rms);
      int invert_(integer_10 *order, real_10 *phi, real_10 *psi, real_10
*rc);
      int dyptrk_(real_10 *amdf, integer_10 *ltau, integer_10 *minptr,
integer_10 *voice, integer_10 *pitch, integer_10 *midx, struct
lpc10_encoder_state *st);

      int deemp_(real_10 *x, integer_10 *n, struct lpc10_decoder_state
*st);
      int bsynz_(real_10 *coef, integer_10 *ip, integer_10 *iv, real_10
*sout, real_10 *rms, real_10 *ratio, real_10 *g2pass, struct
lpc10_decoder_state *st);
      int irc2pc_(real_10 *rc, real_10 *pc, integer_10 *order, real_10
*gprime, real_10 *g2pass);
      int pitsyn_(integer_10 *order, integer_10 *voice, integer_10 *pitch,
real_10 *rms, real_10 *rc, integer_10 *lframe, integer_10 *ivuv, integer_10
*ipiti, real_10 *rmsi, real_10 *rci, integer_10 *nout, real_10 *ratio,
struct lpc10_decoder_state *st);

      int difmag_(real_10 *speech, integer_10 *lpita, integer_10 *tau,
integer_10 *ltau, integer_10 *maxlag, real_10 *amdf, integer_10 *minptr,
integer_10 *maxptr);

      int vparms_(integer_10 *vwin, real_10 *inbuf, real_10 *lpbuf,
integer_10 *buflim, integer_10 *half, real_10 *dither, integer_10 *mintau,
integer_10 *zc, integer_10 *lbe, integer_10 *fbe, real_10 *qs, real_10
*rc1, real_10 *ar_b__, real_10 *ar_f__);

      integer_10 median_(integer_10 *d1, integer_10 *d2, integer_10 *d3);
      int ham84_(integer_10 *input, integer_10 *output, integer_10
*errcnt);
```

```
	integer_10 random_(struct lpc10_decoder_state *st);

	long pow_ii(long *ap,long *bp);
	double r_sign(real_10 *a,real_10 *b);
	long i_nint(real_10 *x);


	struct lpc10_encoder_state * create_lpc10_encoder_state ();
	void init_lpc10_encoder_state (struct lpc10_encoder_state *st);
	int lpc10_encode(real_10 *speech, INT32_10 *bits, struct
lpc10_encoder_state *st);

	struct lpc10_decoder_state * create_lpc10_decoder_state ();
	void init_lpc10_decoder_state (struct lpc10_decoder_state *st);
	int lpc10_decode (INT32_10 *bits, real_10 *speech, struct
lpc10_decoder_state *st);

	real_10 *toRealArray(void);
	void toSampleArray(INT32_10 *);

protected:
	!LPC(void);
public:
	LPC(void);
	LPC(Frame ^);
	LPC(array<SAMPLE> ^);
	LPC(ArrayList ^);

	~LPC(void);

	void setFrame(Frame ^);
	int setFrame(ArrayList ^);
	int setFrame(array<SAMPLE> ^);

	void setPrediction(int);

	int encodeWithLPC10(void);
	int encodeWithMatlab(void);
};

ref class VectorQuantizer : public Frame{
private:
	//ArrayList ^codebook;
	int dimension;
	int numberOfTrainingVectors;
	float rateOfLBG;
protected:
	!VectorQuantizer(void);
public:
	VectorQuantizer(void);
	VectorQuantizer(Frame ^);
	VectorQuantizer(array<SAMPLE> ^);
	VectorQuantizer(ArrayList ^);

	~VectorQuantizer(void);
	void setFrame(Frame ^);
	int setFrame(ArrayList ^);
	int setFrame(array<SAMPLE> ^);
	int trainingWithMatlab(void);
	int trainingWithLBG(void);
};
```

جزء من الشفرة المصدرية لآلة ربط و مقارنة الكلام

```
#include "SpeechIO.h"
#include "SpeechCoding.h"

#include <stdio.h>

#include "..\src\include\fann.h"


using namespace System;
using namespace IO;
using namespace Collections;

/*
	SpeechDictionaryItem class is a part of SpeechDictionary class ,
	any object from this class to represent as item in speech dictionary
vocabulary ,
	also it to contribute to link vocabulary with physical files ,

	example:
		utteranceId = 1 , utteranceName = "ق" , utteranceFilesNames[0]
= "c:\\1.wav"
*/

ref class SpeechDictionaryItem{
protected:
	int utteranceId;
	String ^utteranceName;
	ArrayList ^utteranceFilesNames;//strings
	ArrayList ^codebooks;//frames

	int utteranceFilesNamesCounter;
	int codebooksCounter;

	!SpeechDictionaryItem(void);//Finalizers

public:

	//Constructors
	SpeechDictionaryItem(void);
	SpeechDictionaryItem(String ^,array<String ^> ^);
	SpeechDictionaryItem(SpeechDictionaryItem ^);//Copy constructor

	~SpeechDictionaryItem(void);//Destructors

	//sets methods to update data
	void setUtteranceId(int utteranceId);
	void setUtteranceName(String ^);
	void setUtteranceFilesNames(array<String ^> ^);
	void addUtteranceFileName(String ^);

	void setCodebooks(array<Frame ^> ^);
	void setCodebooks(ArrayList ^);
	void addCodebook(Frame ^);

	void setSpeechDictionaryItem(SpeechDictionaryItem ^);
```

```
	//gets methods to retrieve data
	int getUtteranceId(void);
	String ^getUtteranceName(void);
	array<String ^> ^getUtteranceFilesNames(void);
	String ^getUtteranceFileName(int);

	Frame ^getCodebook(int);
	ArrayList ^getCodebooks(void);
	array<Frame ^> ^toFrames(void);
	int getUtteranceFilesNamesCounter(void);
	int getCodebooksCounter(void);
};

/*
	SpeechDictionary class to add and remove utterance to dictionary ,

	also can know dictionary vocabulary size ,
*/
ref class SpeechDictionary{
protected:
	//dictionary items array

	int vocabularySize;
	ArrayList ^speechDictionaryItems;

	String ^name;//to crate multi dictionary


	//this method to help other methods to find speech dictionary item
index if found its return the index
	int findItemIndex(int);//its take utterance id
	int findItemIndex(String ^);//its take utterance name


	!SpeechDictionary(void);//Finalizers

public:

	//Constructors
	SpeechDictionary(void);
	SpeechDictionary(SpeechDictionary ^);

	~SpeechDictionary(void);//Destructors

	//load speech dictionary items from hard disk its save in
speechDictionary.data if loading done return 1
	//this method fill the SpeechDictionaryItems arry with items stored
in file
	int loadSpeechDictionaryItems(void);

	//add new speech dictionary item
	void addSpeechDictionaryItem(SpeechDictionaryItem ^);

	//remove speech dictionary item if done its return 1
	int removeSpeechDictionaryItem(int);//its take utterance id
	int removeSpeechDictionaryItem(String ^);//its take utterance Name

	//modify speech dictionary item if done its return 1
	int modifySpeechDictionaryItem(int ,SpeechDictionaryItem ^);//its
take utterance id and new item
```

```
	int modifySpeechDictionaryItem(String ^,SpeechDictionaryItem ^);//its
take utterance Name and new item
	void modifySpeechDictionaryItemByIndex(int ,SpeechDictionaryItem ^);

	//save speech dictionary to hard disk its save into
SpeechDictionary.data if saving done return 1
	int saveSpeechDictionaryItems(void);

	//gets methods
	SpeechDictionaryItem ^getSpeechDictionaryItem(int);//its take
utterance id
	SpeechDictionaryItem ^getSpeechDictionaryItem(String ^);//its take
utterance Name
	SpeechDictionaryItem ^getSpeechDictionaryItemByIndex(int);
	array<SpeechDictionaryItem^> ^getSpeechDictionaryItems(void);
	String ^getSpeechDictionaryName(void);
	int getVocabularySize(void);

	//sets methods
	void setSpeechDictionaryItems(array<SpeechDictionaryItem^> ^);
	void setSpeechDictionary(SpeechDictionary ^);
	void setSpeechDictionaryName(String ^);
};

ref class NeuralNetwork : public SpeechDictionary {
private:
	int numberOfInputs;
	int numberOfOutputs;
	int numberOfLayers;
	int numberOfHiddenNeurons;

	int numberOfTrainingPairs;

	__int64 maxEpochs;
	int epochsBetweenReports;

	float desiredError;

	array<NeuralNetwork ^,2> ^derivatives;

	//sets methods
	void setNumberOfInputs(int);
	void setNumberOfOutputs(int);
	void setNumberOfTrainingPairs(int);

	//helps methods
	int fillCodebooks(void);//may take long time
	int codebooksFileBuilder(void);
	int codebooksFileLoader(void);

	int calculatNumperOfInputs(void);
	int calculatNumberOfTrainingPairs(void);
	char *fromStringToCharArray(String ^);

	Frame ^completCodebook(Frame ^,SAMPLE);//make samples in codebook =
number of inputs
	int buildTrainingDataFile(void);//the output files name is an
dictionary Name
	//int buildDisarrayTrainingDataFile(void);//the output files name is
an dictionary Name
	void fillFannTrainingData(void);//initailize data to training
```

```
        //void disarrayTrainingData(void);

        void multiNeuralNetworksBuilder(void);
        void randomSamplesBuilder(void);
        int calculatMaxNumberOfUtteranceFilesNames(void);

protected:
        //Finalizers
        !NeuralNetwork(void);
public:
        //Destructors
        ~NeuralNetwork(void);

        //Constructors
        NeuralNetwork(void);
        NeuralNetwork(SpeechDictionary ^);

        //sets methode
        void setNumberOfLayers(int);
        void setNumberOfHiddenNeurons(int);

        void setMaxEpochs(int);
        void setEpochsBetweenReports(int);

        void setDesiredError(float);

        void setSpeechDictionary(SpeechDictionary ^);
        void setNeuralNetwork(SpeechDictionary ^);

        //gets methods
        int getNumberOfInputs(void);
        int getNumberOfOutputs(void);
        int getNumberOfLayers(void);
        int getNumberOfHiddenNeurons(void);

        int getNumberOfTrainingPairs(void);

        int getMaxEpochs(void);
        int getEpochsBetweenReports(void);

        float getDesiredError(float);

        //training methods
        int trainingWithFANN(void);//the output files name is an utterance
Name + .net
        int trainingMultiNeuralNetworksWithFANN(void);
        int trainingWithMatlabBP(void);
        //int trainingWithMatlabElman(void);

        //testing methods
        float testingWithFANN(Frame ^);
        String ^testingMultiNeuralNetworksWithFANN(Frame ^);
        int testingMultiNeuralNetworksWithFANNWithIndex(Frame ^);
        float testingMultiNeuralNetworksWithFANNWByOne(int,Frame ^);
        //float testingWithMatlabBp(Frame ^);
        //float testingWithMatlabElman(Frame ^);
        void addTrainingPairs(int,FramesFilter ^);

};
```

المراجع

الكتب :

- -(SPEECH RECOGNITION USING NEURAL NETWORKS) : Pablo Zegers (1998م) .
- -(SPEECH RECOGNITION USING NEURAL NETWORKS) :Joe Tebelskis (1995م) .
- (Articial Intelligence) - (1985م) : Alain Bonnet .
- إيمان أبو المعالي عبد الرحمن : تمييز الكلام العربي باستخدام نماذج ماركوف الخفية – (1993م) .
- أحمد عبدا لله أمام: الشبكات العصبية الاصطناعية– (2004م) .
- مجدي محمد الطيب: نظام متكامل للتعرف على المتكلم و الكلام العربي المنطوق– (2001م) .

الإنترنت :

- Stephen Cook ، SPEECH RECOGNITION HOW TO ، 2005-12-24م، http://www.Gear21.com/speech
- Dr Philip Jackson ، Speech Communication ، 2005-12-3م ، http://www.ee.surrey.ac.uk/Teaching/Courses/eem.ssr .
- http://www.data-compression.com/vq.shtml ، 2006-4-1م .
- http://www.dspexperts.com/dsp/projects/lpc/ ، 2006-6-11م .
- http://www.lightlink.com/tjweber/StripWav/WAVE.html ، 2006-5-20م .
- http://www.mat.ucsb.edu ، 2006-5-20م .
- http:// ar.Wikpedia.org/wiki ، 2006-6-7م .