Automatically Grading Learners' English using a Deep Gaussian Process

Sebastian Gabriel Popescu

Department of Engineering

University of Cambridge

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Machine Learning, Speech and Language Technology

Saint Edmund's College Date: 12/08/2016

I, Sebastian Gabriel Popescu of Saint Edmund's College, being a candidate for the M.Phil in Machine Learning, Speech and Language Technology, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Word count: 9155

Signed:
Date:

Table of Contents

1 Introduction

1.1 Motivation and Applications

In the past decades, our world has steadily evolved into a highly globalized, inter-connected environment where transnational conversations are becoming ever more present in our daily life. From the end of the Second World War, English has become the new “lingua franca”, most business meetings or informal conversations between people of different nationalities being pursued in English by default. Naturally, there has been a surge in English examinations on offer, either as a gateway to access English-speaking universities such as in the case of universally accepted exams like Test of English as a Foreign Language (TOEFL) or Cambridge English Advanced (CAE) or as a starting point to a career in an English-speaking country by offering niche orientated examinations such as Business Language Testing Service (BULATS).

The main aim of this project is to develop an automated assessment system of spontaneous non-native speech. One of the main motivations behind building an automated grading system is that even well-trained human graders can be inconsistent in grading, introducing human bias in terms of the weights that they attach on various components of the grading process or by various other exogenous factors. Therefore, automated graders can solve this consistency issue, additionally providing feedback at a fraction of the cost associated to human graders. Besides feedback speed and consistency, other feasible applications can emerge such as self-tutoring apps that also assess learners on quality of conversational English and not solely based on written competence, as it is currently the case.

While closed-class question based automatic assessment systems can easily provide feedback and are extremely accurate, in situations where constructed responses are needed this introduces difficulties for the automated grader. In order to provide accurate feedback it has to take into consideration not only notions of correct grammar, richness of vocabulary or complex syntactical constructions, but the system also has to take into account elements of pronunciation, prosody, coherence and topic development. An algorithm must not only attach weights to the previously enumerated components, but also to extract highly discriminative features that could aid this process.

Early research into automated graders of spontaneous non-native speech have focused mostly on elements of prosody and pronunciation, such as attempting to characterize aspects of communicative competence by Hidden Markov Model forced-alignment in order to obtain posterior probabilities for pronouncing certain phones. These scores were combined with various other features such as number of words per second or duration of phonemes to obtain the final grades (Franco et al., 2000). Zechner and Bejar(2006) have expanded the feature set by including elements of lexical sophistication and content and passing them to a Support Vector Machine to obtain predictions. Van Dalen et al. (2015) introduce for the first time the uncertainty of predictions by using a Gaussian Process on audio features related to fundamental frequency or energy and fluency features such as disfluencies or silence duration. The same research paper also introduces an innovative combination scheme which backs off to human grades in case of high uncertainty in predictions. In terms of extracting more information from audio signal, as opposed to using hand-crafted fluency and audio features as previously mentioned, Yu et al. (2015) introduce a Bidirectional Long Short-Term Memory Network framework for extracting sequential highly abstract features from audio data, further adding a Multi-linear Perceptron layer at the end to obtain predictions.

1.2 Intended goals

This work is under the umbrella of the Automated Language Teaching and Assessment (ALTA) project in collaboration with Cambridge University English for Speakers of Other Languages (ESOL) group to develop new methods for assessment of spoken English. Our contributions to the overall aim of the project can be divided in two categories. Firstly, we expand the work of Rashid (2015) and aim to extract more sophisticated linguistic features pertaining to dependency-grammars from the parse trees of our Automated Speech Recognition Transcription. We introduce a state-of-the-art Smoothed Partial Tree Kernel to assess the similarity of our ASR transcripts in comparison with manually transcribed text from the DTAL, obtaining a similarity converging to 80% for various configurations of dependency-grammar derived tree structures. Experiments show that this similarity is enough to extract simple linguistic features such as Part-of-Speech tags or Grammatical Relation tags, but still not high enough to enable us to use more complex structural features such as Skeletons or Annotated Skeletons as a substitute. The other major component of the project is the use of a Deep Gaussian Process grader to obtain better predictive means and well-calibrated propagation of uncertainty. While our experimental findings show that even using a very sparse Deep Gaussian Process with no hidden layers yields higher correlation with human graders in comparison to using a fully Gaussian Process grader on the same data, we also notice some problems with the propagation of uncertainty in estimates which affects our variance based back off model.

2 English Language Examination Format

This chapter is used as an introduction to the common framework of assessing foreign languages, while also giving brief details regarding the format of BULTAS.

2.1 Proficiency Levels

The Common European Framework of Reference for Languages (Verhelst et al., 2009) provides a shared framework under which the elaboration of language syllabuses, curriculum guidelines, examinations and textbooks are devised across Europe. It represent a comprehensive formulation of what type of knowledge and skills language learners must attain in order to use a specific foreign language in a context. It also provides clear definitions of the varying levels of proficiency, which we detail in table 1:

Level		Brief Description
Proficient User	C2	Can understand with ease virtually everything heard or read
	C1	Can understand a wide range of demanding, longer texts and recognize implicit meaning
Independent User	B2	Can understand the main ideas of complex text on both concrete and abstract topics
	B1	Can understand the main points of clear standard input on familiar matters
Basic User	A2	Can understand sentences and frequently used expressions related to areas of most immediate relevance
	A1	Can understand and use familiar everyday expressions and very basic phrases aimed at the satisfaction of needs of a concrete type

Table 1: CEFR language level details

2.2 BULTAS Examination Structure

The Business Language Testing Service is an examination targeted for business-context situations, tailored for professionals who would like to attest that their English competency is at an appropriate level to be hired at corporations. The speaking part of the examination is composed of five main parts:

- Part 1: test takers respond to eight basic questions about themselves and their work
- Part 2 : repetition of text that might be read aloud in a business situation
- Part 3 : the candidate talks about a work-related topic
- Part 4 : involves delivering a mini-presentation describing a pie chart or a bar char in a business context
- Part 5 : test takers must imagine that they are in a specific situation with a colleague and have to respond to question that may be asked in that situation

Each section is graded between 0 and 6, resulting in a maximal score for the speaking component of 30.

3 Smoothed Partial Tree Kernels

In Natural Language Processing, a central challenge is the design of complex and intricate features that can aid in research topics such as Question Classification or Sentiment Analysis. Since hard-coding such features has proven to be a daunting task, methods to automatically encode enough lexical and syntactic information into features had to be devised.

Kernel methods represent a stepping stone in the development of Machine Learning, being behind one of the most simple but effective early ML algorithms such as Perceptrons or Support Vector Machines. Kernels also hold a significant role in our project, since both of our Grader methods such as Gaussian Processes and its extension, the Deep Gaussian Process rely on kernel methods.

Since we are dealing with text, we do not have an easily definable R^d input domain. With this in mind, Tree Kernels were introduced by Duffy and Collins (2001) and represent an instance of Convolutional Kernels, which involve a recursive computation over fragments of discrete structures, such as in our case statistical parse trees.

3.1 Tree Kernels

Each parse tree is represented in a bag-of-words fashion :

$$h(T) = [h_1(T), h_2(T), \dots, h_n(T)]$$

where $h_i(T)$ counts the number of occurrences of the i-th sub-tree in parse tree T.

$$K(T_1, T_2) = h(T_1) * h(T_2)$$

where $K(T_1, T_2)$ is defined as the tree kernel between parse trees T1 and T2.

$$I_i(n) = \begin{cases} 1 \; if \; i-th \; subtree \; is \; at \; node \; number \; n \\ 0 \; otherwise \end{cases}$$

Using the above mentioned index function we arrive at:

$$h_i(T_1) = \sum_{n_1 \in N_1} I_i(n_1)$$
$$h_i(T_1) = \sum_{n_1 \in N_1} I_i(n_1)$$

It naturally follows to the final formula:

$$K(T_1, T_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) \, I_i(n_2) = \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} C(n_1, n_2)$$

It can be computed in polynomial time with the following recursive definitions:

$$C(n_1, n_2) = 0 \; if \; the \; productions \; at \; n_1 \; and \; n_2 \; are \; different$$

$$C(n_1, n_2) = 1 \; if \; the \; productions \; at \; n_1 \; and \; n_2 \; are \; similar \; and \; the \; two \; nodes \; are \; pre - terminals$$

$$C(n_1, n_2) = \prod_{j=1}^{n_c(n_1)} [1 + C(ch(n_1, j), ch(n_2, j)]$$

Where $n_c(n_1)$ is the number of children of node n_1 and $ch(n_2, j)$ is the j-th children of node n_2. In this case, we easily notice that $n_c(n_1) = n_c(n_2)$ since the definition is conditioned on having the same syntactic productions.

The above definitions represent the Syntactic Tree Kernel (STK) (Collins and Duffy, 2002), in which sub-graphs of the parse tree can be matched even though they have different leaves or surface forms, with the condition that the pre-terminal parent nodes have the same production. Nevertheless, Croce (2011) has argued that STKs leave room for improvement in terms of exploiting semantic smoothness, such as in the case of the sentences "the big beautiful apple" and "a nice large orange", where in the case of STKs just the syntactic structure of "apple" and "orange" are matched. Additionally, STKs cannot be applied to dependency structures, which can be loosely defined as being derived from dependency-grammar features where all nodes are seen as being terminal, thereby connecting words in terms of their grammatical relationships.

3.2 Partial Tree Kernels

To deal with the loss of syntactic information characteristic of STKs, we introduce a new Tree structure entitled Partial Tree (Moschitti, 2006). To better illustrate the difference between the two different spaces, we take the simple case of a constituent parse tree of the sentence "Mary brought a cat". In the case of sub-trees, just sub-graphs containing the full production of a parent node are taken into consideration, whereas for partial trees any possible combinations of incomplete or partial productions are admissible. The following figure better illustrates the difference.

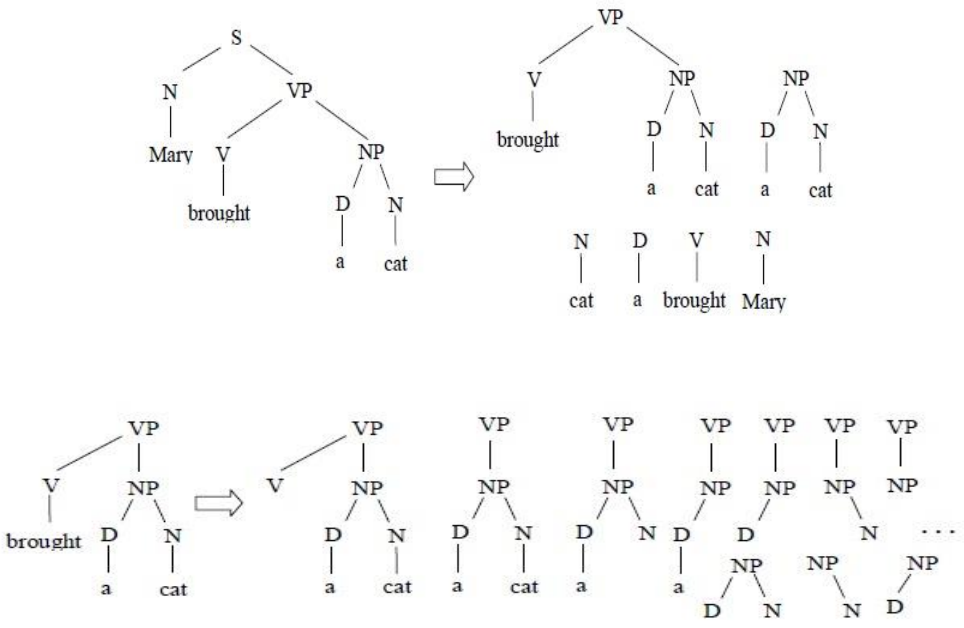

Figure 1: Upper – subtree decomposition ; Lower – partial tree decomposition [Moschitti, 2006]

We appropriately define the new Δ function for Partial Trees:

$$\Delta_{PTK}(n_1, n_2) = 0 \text{ if the labels of } n_1, n_2 \text{ are different}$$

$$\Delta_{PTK}(n_1, n_2) = \mu(\lambda^2 + \sum_{I_1, I_2} \lambda^{d(I_1)+d(I_2)} \prod_{j=1}^{l(I_1)} \Delta_{PTK}\left(c_{n1}(I_{1j}), c_{n1}(I_{1j})\right)$$

Where we have defined $d(I_1) = I_{1l(I_1)} - I_{11} + 1$ and $d(I_2) = I_{2l(I_2)} - I_{21} + 1$ and I_1, I_2 represent any type of subsequence of children for node 1, respectively node 2 of any length as long as the subsequences are equal.

μ is the vertical decay factor, which controls the amount of penalty imposed as we progress down the depth of our dependency parse tree. On the same note, λ represent the horizontal decay factor, which controls the penalty applied on children subsequences, thus enabling us to penalize children subsequences of great length or children subsequences with large gaps.

Croce et al. (2011) introduce a more general tree kernel applicable on all types of dependency parse trees, thus being able to exploit any combination of lexical and syntactic similarity.

3.3 Smoothed Partial Tree Kernel

In the case of Smoothed Partial Tree Kernels we have the following definitions of Δ:

$$\Delta_{SPTK}(n_1, n_2) = \mu\lambda\sigma(n_1, n_2) \text{ for leaf nodes}$$

$$\Delta_{SPTK}(n_1, n_2) = \mu\sigma(n_1, n_2) * \left[\lambda^2 + \sum_{I_1, I_2, l(I_1) = l(I_2)} \lambda^{d(I_1)+d(I_2)} \prod_{j=1}^{I_1} \Delta_{SPTK}\left(c_{n1}(I_{1j}), c_{n1}(I_{1j})\right)\right]$$

$\sigma(n_1, n_2)$ defines an open similarity function which can take into account any type of matches between labels of two nodes.

3.4 Efficient Implementation of SPTKs

For the implementation of SPTKs we have followed the guidelines in Croce et al (2011), starting by partitioning the computations present in the above equation with respect to the overall length of the children subsequences present there. Denoting the length of subsequences by p we arrive at the equivalent equation:

$$\Delta_{SPTK}(n_1, n_2) = \mu\sigma(n_1, n_2) * \left(\lambda^2 + \sum_{p=1}^{m} \sigma_p(c_{n_1}, c_{n_2})\right)$$

where m = $min(\ l(n_1), l(n_2)\ )$ and σ_p is evaluating the number of common sub-graphs in children subsequences of length p. If we take the two child subsequences to be $s_1 a = c_{n_1}$ and $s_2 b = c_{n_2}$ we arrive at the following updated equation for σ_p:

$$\sigma_p(s_1 a, s_2 b) = \sigma(a, b) * \sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} * \sigma_{p-1}(s_1[1:i], s_2[1:r])$$

Denoting $\sum_{i=1}^{|s_1|} \sum_{r=1}^{|s_2|} \lambda^{|s_1|-i+|s_2|-r} * \sigma_{p-1}(s_1[1:i], s_2[1:r])$ as D_p we can rewrite the set of equations as:

$$\sigma_p(s_1 a, s_2 b) = \begin{cases} \sigma(a,b) * \ D_p(|s_1|,|s_2|) \ if \ \sigma(a,b) > 0 \\ 0 \ otherwise \end{cases}$$

With this simplified notation we arrive at the following recursive relation:

$$D_p(k,l) = \ D_p(s_1[1{:}k], s_2[1{:}l]) + \ \lambda D_p(k, l-1) + \lambda D_p(k-1, l) - \ \lambda^2 \lambda D_p(k-1, l-1)$$

With the above recursive formulation, the algorithmic complexity of SPTKs is similar to the one for PTKs, respectively O($pp^2|NT_1||NT_2|$), where p is the largest number of children of a given node in the two trees. However, this is generally not very large, in our dataset being on average 3.

Choosing SPTKs over STKs or Syntactic Semantic Tree Kernels (Moschitti, 2006) is additionally motivated by the following advantages:

- STKs and SSTKs can only work on constituency trees which due to the high WER present in our ASR transcriptions make Constituency-Grammar derived trees less robust.
- Similarity between leaf nodes in SSTKs is only computed for complete matches originated from parent nodes. SPTKs allow flexibility in assessing lexical or syntactic similarity between large numbers of leaf nodes, which is a discernable advantage taking into consideration again the high WER which might result in insertions, deletions or inversions in the ASR transcripts.

3.5 Similarity between parse trees

To have a similarity score between 0 and 1, a normalization in the kernel space must be done in the following way:

$$Sim = \frac{TK(T_1, T_2)}{\sqrt{TK(T_1, T_1) TK(T_2, T_2)}}$$

3.6 Construction Process of Tailored Dependency Tree Structures

As we have previously explained in the previous subchapter, SPTKs accommodate for virtually any possible dependency-grammar derived tree structure, while still respecting Mercer's conditions of kernel validity.

Schematically, we would prefer to have dependencies attached on edges but the SPTK formulation does to cater for this. Hence, the main constructions phases are the following:

- Build central nodes which contain the information we are most interested in
- Add dependencies and additional information as children nodes to the central nodes

In this report we are to explore the following structures:

3.7 Grammatical – Relation Centered Tree

The current tree structure has the following set of classes of nodes:

- Syntactic nodes : encode dependency functions (Grammatical Relations)
- Pre-terminal nodes: represent the PoS tag of the parent Syntactic Node

- Lexical nodes : encode one lexical item in the form $< lemma_n : PoS_n >$

A GR Centered Tree of the sentence “What instrument does Hendrix play?” is presented in the following figure.

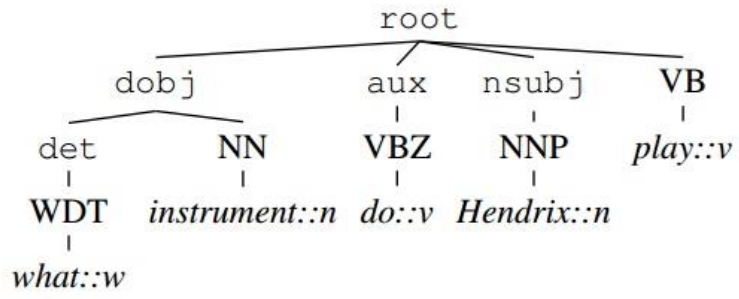

Figure 2: Grammatical Relation Centered Tree (Basili, 2014)

3.8 Lexical Centered Tree

This type is composed by the following classes of nodes:

- Lexical nodes : representing a lexical item in the form $< lemma_n : PoS_n >$
- Terminal nodes : either encode a dependency function (Grammatical Relations) or represent the PoS-tag of the parent Lexical Node

A Lexical Centered Tree of the sentence “What instrument does Hendrix play?” is present in figure 3:

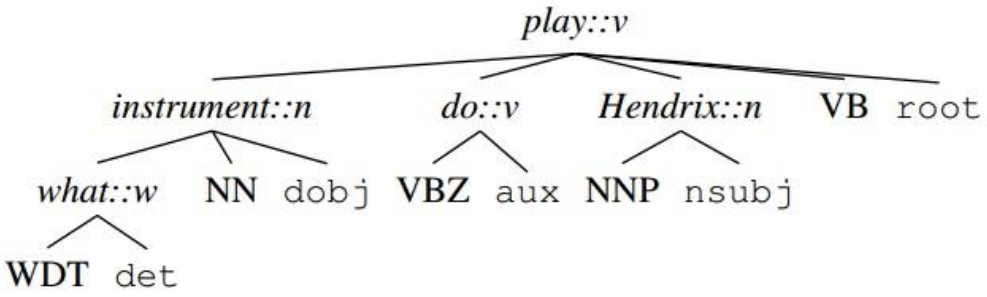

Figure 3: Lexical Centered Tree (Basili, 2014)

3.9 Compositional Dependency-Grammar Derived Structures

One of the main limitations of the previously described structures is that semantic information is assessed in a context free way, the semantic composition of the varying node types proposed being neglected in the computation of SPTKs.

An enhancement in terms of migrating our tree kernels from the sphere of Distributional Semantics to Distributional Compositional Semantics is introduced by Basili et al. (2014) by explicitly incorporating the compositionality phenomenon in the tree structure.

3.10 Compositional Lexical Centered Tree

We are to introduce compositionality by making the dependency function between heads and modifiers explicit in the structural form of the tree, alongside additional lexical information. This new compositional node can be encoded as $< Dep_{h,m}, < lemma_h : PoS_h, lemma_m : PoS_m \gg$, where $Dep_{h,m}$ represents the dependency between head and modifier. The remaining part of the structure is a 2-tuple of a Lexical node as previously described in LCTs. We better highlight the new structure in figure 4:

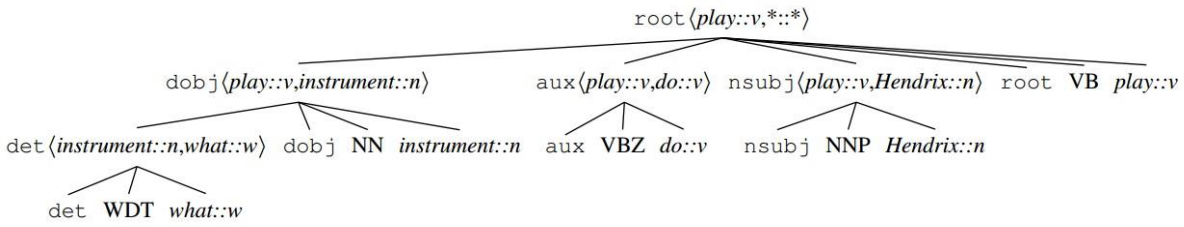

Figure 4: Compositional Lexical Centered Tree (Basili, 2014)

3.11 Compositional Grammatical-Relation Centered Tree

In the case of GCTs, we address the compositionality deficiency by expanding the previously entitled Syntactic Nodes in GCTs in the same way as we did for CLCTs. The following figure is provided to show the differences:

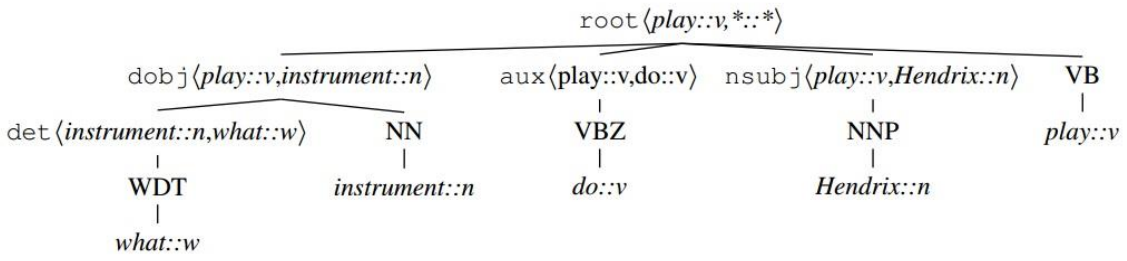

Figure 5: Compositional Grammatical Centered Tree (Basili, 2014)

4 Gaussian Processes

In this chapter our intention is to give a brief introduction to Gaussian Process Regression as this constitutes one of our methods of grading learners' English.

In terms of Supervised Learning algorithms, parametric models have been from a historical perspective more common due to their ease of interpretability. However, on complex data sets these types of models may lack expressive power. With the advent of kernel methods, Gaussian Processes have seen a huge increase in attention as they provide nonparametric modelling power and also well-calibrated uncertainty estimates of its predictions.

Intuitively, Gaussian Processes are distributions over functions, being fully specified by their mean function μ and their covariance function $k(x_1, x_2)$. We shall denote a Gaussian Process prior over a function as:

$$f \sim GP(\mu, K)$$

In the context of regression, a Gaussian Process can be used as a non-parametric prior over a function, which if combined with data will result in a posterior over the respective function.

For simplicity we shall assume our input data X to be part of R^d, with mean function μ and the covariance function K being fully specified by the hyperparameters of the chosen kernel.

In our project we are to focus on the Squared Exponential Kernel, which is defined by the following equation:

$$k(x_1, x_2) = \sigma^2 \exp\left(-\frac{|x_1 - x_2|^2}{2l^2}\right)$$

Where σ is a hyperparameter controlling the overall variance, whereas l is the characteristic lengthscale, which controls how far apart two points have to be to change the covariance function. They hyperparameter l can be a scalar or if we desire our training algorithm to automatically select the most important features we can choose a different characteristic lengthscale for each dimension of our data, the method being called Automatic Relevance Determination(ARD). This however has the drawback of requiring more hyperparameters to be optimized.

Denoting out testing data as $X_*$and the values of the function associated to the aforementioned points as f_*, we can write the joint distribution of all the components of our model as:

$$\begin{bmatrix} f \\ f_* \end{bmatrix} = N\left(\begin{bmatrix} \mu \\ \mu_* \end{bmatrix}, \begin{bmatrix} K_{X,X} & K_{X,X_*} \\ K_{X_*,X} & K_{X_*,X_*} \end{bmatrix}\right)$$

The condition of f_* given f can be derived as the following formula:

$$f_*|f \sim N\left(\mu_* + K_{X_*,X_*}{K_{X,X}}^{-1}\mu, K_{X_*,X_*} - K_{X_*,X}{K_{X,X}}^{-1}K_{X,X_*}\right)$$

Having arrived at the predictive distribution we can use the associated variance to showcase the uncertainty our Gaussian Process model has with regards to specific test input. The following figure shows depict the uncertainty by using standard confidence interval bands.

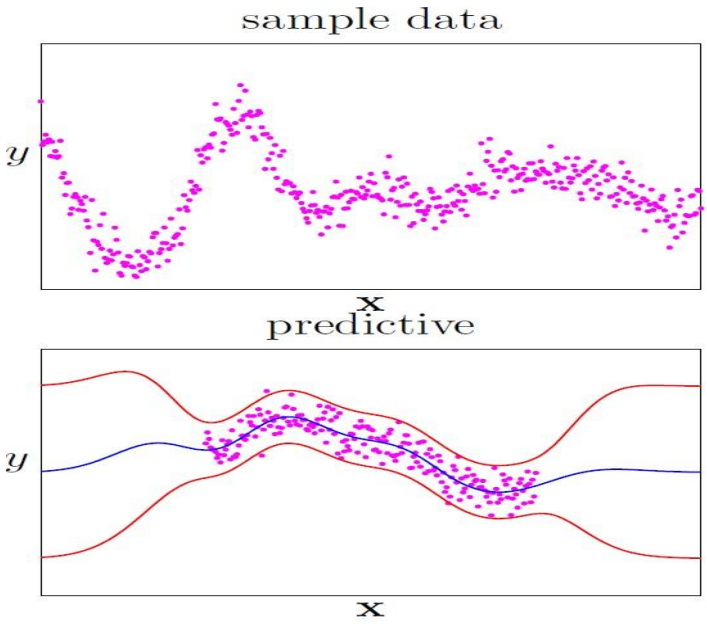

Figure 6: Above – Training data used for GP regression; Below – mean predictive estimates and variance associated to test points. (Ghahramani, 2010)

5 Deep Gaussian Processes

Probabilistic modelling with deep architectures have experiences huge success and applicability in varying research domains, however Deep Neural Network Models typically require a vast amount of data to perform learning which is not our current case. Neal (1996) noticed that Gaussian Processes can be viewed as an infinite-wide Multilayer Perceptron. However, Gaussian Processes even with sophisticated covariance functions (Durrande et al., 2011) or with complex probabilistic structures [Wilson et al.,2012] are not able to learn high-level abstractions of data such as in the case of DNNs. With this in mind, Damianou and Lawrence (2013) have introduced Deep Gaussian Processes, which can be interpreted as multi-layer hierarchical generalizations of Gaussian Processes. In contrast to DNNs, Deep Gaussian Processes have mappings between layers which are parametrised by Gaussian Processes. Therefore, they retain the nonparametric modelling power and also propagate through the hierarchy predictive uncertainties such as a regular Gaussian Process. Additionally, it can also learn layers of increasingly higher abstraction of the data due to its deep structure.

The basic Deep Gaussian Process architecture defined by Damianou (2013) consists of the following graphical model in the case of 1D output regression:

- Leaf nodes denoted as $Y \in R^n$ which is the observed output
- Intermediate nodes denoted as $h_l \in R^{n*Q_l}$, where Q_l represents the dimensionality of the l-th hidden layer
- Input nodes denoted as $X \in R^{n*d}$

In the case of multi-dimensional hidden layers we have the following set of equations which define the generative process.

$$h_{l,q} = f_{l,q}(h_{l-1}) + \varepsilon_l, q = 1, \dots, Q_l$$
$$y = f_{L+1}(h_L) + \varepsilon_{L+1}$$

Where $f_{l,q} \sim GP(0, k(h_{l-1}, h_{l-1})$ and $f_{L+1} \sim GP(0, k(h_L, h_L)$ and ε_l represents noise.

Our main goal is to infer the posterior distribution over the latent function mappings and over the intermediate hidden variables besides obtaining a marginal likelihood estimate for hyperparameter tuning and model comparison. However under the nonlinearities transmitted by the GP mappings between hidden layers, this is analytically intractable, hence approximate inference is needed. Titsias (2009) introduce a variational free energy method of approximating the posterior distribution of a set of inducing outputs **u** associated with m inducing points **z** which are introduced for sparsification of a standard Gaussian Process. This paper has created a surge in interest of treating Gaussian Process models from a variational perspective which culminated with the extension of the variational free energy method to Deep Gaussian Processes (Damianou, 2013). In that research paper, variational approximations to both latent functions and hidden variables are introduced in order to compute a variational lower bound to the true log likelihood bound of the probabilistic model that is both computationally and analytically tractable. Nevertheless, this approach has the drawback that the number of variational parameters that need to be optimized increases linearly with the number of training points, hence this method become unfeasible for medium to large scale datasets. Additionally, Turner and Sahani (2011) have argued that initialization remains a tricky task even for small models. An enhancement to the variational treatment of Deep Gaussian Processes has been developed by Hensman and Lawrence (2014), where a nested variational compression scheme is introduced that eliminates the need for optimizing over variational parameters over hidden variables.

However, none of the above mentioned methods have been tested on medium to large datasets, therefore for our project we have chosen a different approximate inference method, respectively Expectation Propagation. Bui et al. (2016) argue that their implementation is the first known instance of training DGPs on large scale datasets, mainly relying on a well-known GP sparsification method, a new Expectation Propagation scheme and a probabilistic backpropagation algorithm. All of the above mentioned methods will be detailed in depth in the upcoming subchapters.

5.1 Fully Independent Training Conditional Approximation

Full Gaussian Process Models experience cubic complexity with regards to the number of training points used, extending this complexity to a deep architecture such as DGPs would quickly make computational intractable. Hence, sparse approximation methods are of utmost importance. In our project we rely on methods that explicitly sparsify the probabilistic model. FITC (Snelson and Ghahramani, 2006) approximations are devised by choosing a smaller set M << N of function values **u** in the latent function domain space with their associated inducing points denoted by **z**. These values are optimized with respect to the marginal likelihood. Therefore, our new probabilistic model can be written in the following way:

$$p_{u_l} = N\big(u_l; 0, K_{u_l,u_l}\big) for\ l = 1, \dots, L$$
$$p(h_l|u_l, h_{l-1}) = \prod_n N(h_{l,n}; K_{h_{l,n},u_l}K_{u_l,u_l}^{-1}u_l\ , K_{h_{l,n},h_{l,n}} - K_{h_{l,n},u_l}K_{u_l,u_l}^{-1}K_{u_l,h_{l,n}})$$
$$p(y|u_{L+1}, h_L) = \prod_n N(y_n; K_{h_{L,n},u_{L+1}}K_{u_{L+1},u_{L+1}}^{-1}u_{L+1}\ , K_{h_{L,n},h_{L,n}} - K_{h_{L,n},u_{L+1}}K_{u_{L+1},u_{L+1}}^{-1}K_{u_{L+1},h_{L,n}})$$

The computational complexity of this sparse model has been reduced to O(LM^2)

5.2 Expectation Propagation

Before proceeding further with our approximate inference scheme applied to DGPs, we present a brief review of Expectation Propagation (Gelman et al., 2014).

As the variational free-energy method, Expectation Propagation is aiming to minimize the Kullback-Liebler divergence between the true posterior distribution $p(\theta|y)$ and a tractable approximation, which is usually chosen to be a multivariate normal, $q(\theta)$. Therefore , $q(\theta)$ is constructed such as to approximate the target, which can be expressed in more detail as $p(\theta|y) = p(\theta)\prod_{k=1}^{n} p(y_k|\theta)$

Expectation Propagation is a fast and parallelizable method of distributional approximation by partitioning the data and producing better posterior approximation in the following iterative way:

- Splitting the data : Y is split into K parts, each with its associated likelihood $p(y_k|\theta)$
- Initializing the posterior distribution : initial site approximation denoted $t_k(\theta)$ are chosen from tractable families od distributions, hence obtaining an initial approximation to the posterior $q(\theta) = p(\theta)\prod_{k=1}^{K} t_k(\theta)$
- EP iterations from k=1, …,K:
    - Cavity distribution : $q_{-k}(\theta) = q(\theta)/t_k(\theta)$
    - Tilted distribution : $q_{/k}(\theta) = p(y_k|\theta)q_{-k}(\theta)$
    - Update site approximation : $t_k^{new}(\theta)$ is obtained such that $t_k^{new}(\theta)q_{-k}(\theta)$ approximates well $q_{/k}(\theta)$
    - For serial models we update $t_k(\theta) = t_k^{new}(\theta)$ after each inner loop
    - For parallel models, we update the approximate posterior after the end of all K inner loop iterations

- Convergence : we repeat the previous step until convergence of $q(\theta) = p(\theta)\prod_{k=1}^{K} t_k(\theta)$

5.3 Expectation Propagation for Deep Gaussian Processes

The approximate posterior for our inducing points function values are defined as

$$q(u) \sim p(u)\prod_{n} t_n(u)$$

Where the set $t_n(u)$ represents the approximate data factors, each individual data factor encoding the contribution of the n-th data point to the posterior distribution.

The EP-like iteration in our case becomes:

- Cavity distribution : $q_{-k}(\theta) = q(\theta)/t_n(\theta)$
- Minimize $(q_{-k}(\theta)p(y_k|u, x_n)\ |q_{/k}(\theta))$, where $q_{/k}(\theta)$ represents the titled distribution as previously defined in the EP theoretical review.
- Obtain new $t_n(\theta)$ estimate and multiply with cavity distribution to obtain new approximate posterior distribution

The EP produces an approximation to the marginal likelihood:

$$\log p(y|\alpha) = \ \Phi(\theta) - \ \Phi\left(\theta_{prior}\right) + \sum_{n=1}^{N} log\widetilde{Z_n}$$

where $g\widetilde{Z_n} = log\int q_{-n}(\theta)p(y_n|u, x_n)d_u + \ \Phi(\theta_{-n}) - \ \Phi(\theta)$, $\Phi(\theta)$ is the log normalizer of q(u), $\Phi(\theta_{-n})$ is the log normalized of the cavity distribution and $\Phi\left(\theta_{prior}\right)$ is the log normalized of the prior, p(u).

A disadvantage of this methods is that the approximate data factors have to be stored in memory, which leads to a cost of $O(NLM^2)$, as the mean and covariance matrix for each factor has to be stored. An alternative method which reduces the computational load is the Stochastic Expectation Propagation [Li et al.,2015], which ties the data factors. A simple case is when all approximate data factors are tied, resulting in an average data factor entitled $g(u)$.

The new iterative procedure is the following:

- Cavity distribution: $q_{-1}(u) \sim q(u)/g(u)$
- Minimize $KL(q_{-1}(u)p(y_n|u, x_n)|\ q(u))$
- New approximate average factor $g(u)^{new}$ is multiplied with cavity to update the posterior approximation
- Explicitly update average factor : g(u) = $g(u)^{1-\beta}g(u)^{new^{\beta}}$, where β is a small learning rate

The new approximation to the log marginal likelihood is:

$$\log p(y|\alpha) = \ \Phi(\theta) - \ \Phi\left(\theta_{prior}\right) + \sum_{n=1}^{N} logZ_n + \ \Phi(\theta_{-1}) - \ \Phi\left(\theta_{prior}\right)$$

5.4 Probabilistic Backpropagation for Deep Gaussian Processes

Computing $logZ_n$ is analytically intractable for deep hierarchical structures as the likelihood given the inducing outputs *u* is nonlinear.

One of the advantages of this algorithm over the variational free energy method of Damianou and Lawrence [2013] is that the hidden variables are integrated out, hence we reduce the need to optimize over them. We now show a one hidden layer example:

$$Z = \int p(y|x,u)q(u)_{-1}d_u$$
$$= \int p(y|h_1,u_2)\, q(u_2)_{-1}d_{h_1}d_{u_2} \int p(h_1|x,u_1)q(u_2)_{-1}d_{u_1}$$

We proceed by marginalizing the inducing outputs leading us to:

$$Z = \int p(y|h_1)q(h_1)d_{h_1}$$

Where the above distributions in the integral have the following form:

$$q(h_1) = N(h_1; m_1, v_1) \text{ with:}$$
$$q(u_1)_{-1} \sim N(m_1^{\backslash 1}, V_1^{\backslash 1})$$
$$m_1 = K_{h_1,u_1}K_{u_1,u_1}^{-1}m_1^{\backslash 1}$$
$$v_1 = \sigma_1^2 + K_{h_1,h_1} - K_{h_1,u_1}K_{u_1,u_1}^{-1}K_{u_1,h_1} + K_{h_1,u_1}K_{u_1,u_1}^{-1}V_1^{\backslash 1}K_{u_1,u_1}^{-1}K_{u_1,h_1}$$

and

$$p(y|h_1) = N(y; m_{2|h_1}, v_{2|h_1}) \text{ with:}$$
$$q(u_2)_{-1} \sim N(m_2^{\backslash 1}, V_2^{\backslash 1})$$
$$m_{2|h_1} = K_{h_2,u_2}K_{u_2,u_2}^{-1}m_2^{\backslash 1}$$
$$v_2 = \sigma_2^2 + K_{h_2,h_2} - K_{h_2,u_2}K_{u_2,u_2}^{-1}K_{u_2,h_2} + K_{h_2,u_2}K_{u_2,u_2}^{-1}V_2^{\backslash 1}K_{u_2,u_2}^{-1}K_{u_2,h_2}$$

The law of iterated conditions is used to approximate the above integral, hence arriving at a Gaussian approximation of the form:

$$Z \sim N(y|m_2, v_2)$$
$$m_2 = E_{q(h_1)}K_{h_2,u_2}K_{u_2,u_2}^{-1}m_2^{\backslash 1}$$
$$v_2 = \sigma_2^2 + E_{q(h_1)}[K_{h_2,h_2}] + tr\left(BE_{q(h_1)}[K_{u_2,h_2}K_{h_2,u_2}]\right) - m_2^2$$

$$B = K_{u_2,u_2}^{-1}\left(V_2^{\backslash 1} + m_2^{\backslash 1} * T(m_2^{\backslash 1})\right)K_{u_2,u_2}^{-1} - K_{u_2,u_2}^{-1}$$

6 Parse Tree Features

Extracting features from text is a fundamental research problem in fields such as information retrieval or text categorization just to name a few. However, there is no standard method for text representation as different tasks require different features, such as in the case of functional words which are not particularly relevant for topic categorization, but useful for author classification (Massung, 2013).

6.1 Syntactic Features

Perhaps the most fundamental feature extractable from statistical parse trees are Part-of-Speech tags. Their small number has led them to be easily implemented in classifiers. Their small number has also led to the development of n-gram models. Part-of-Speech tags are present in the vast majority of research being done in Natural Language Processing as they are they capture grammar usage at its most basic level. These features have been successfully implemented in scoring non-native speech, deception detection or authorship attribution. Rashid [2015] has used these features on data belonging to the same project, leading to an increase in correlation with expert graders.

In addition, we are to also use features pertaining based on dependency grammar, such as the Grammatical Relations between heads and modifiers as they would provide a more accurate depictions of the complexity of sentence building.

6.2 Skeletons

Unlike the previously proposed methods which include syntactic categories to capture linguistic properties of the text, the next set of features diverges radically by eliminating all additional information from parse trees. Skeletons (Massung, 2013) throw away all syntactic information attached to parse tree, keeping just its structure. Jiang and Zhai (2007) have explore different parse tree features for the task of authorship detection, ranging from word sequence n-grams , grammar productions and dependency paths extracted from dependency parse trees. However, when using all of the above specified features in conjunction as opposed to separately, the researchers observe minor improvements in accuracy, Massung (2013) argues that this is due to the fact that structural properties of text are ignored, hence using this argument as motivation for the application of skeletons.

6.3 Annotated Skeletons and “1-layer deep” Annotated Skeletons

In this report we are also to explore different configurations of Annotated Skeletons, which are a compromise between obtaining pure structural features and incorporating a small set of dependency-grammar information. More specifically, using Grammatical Relation Centered Trees as our default parse tree, we also attach Grammatical Relation information to the root node of the skeleton. This is an extension of the Annotated Skeleton, more specifically also including the Grammatical Relation tags of the children of the root node.

7 Experiments Details

7.1 Data Used

The experiments pursued in this project are using speech data provided by Cambridge English, containing audio recordings of the speaking part of BULTAS alongside their corresponding grades per section. The data is summarized in the following tables:

Dataset Name	Speaker Native Language	No. Speakers
BLXXXgrd00	Gujarati	2013
BLXXXgrd01	Latin American Spanish	925
BLXXXgrd02	Polish,Vietnamese,Arabic,Dutch,French,Thai	994

Table 2: Training data

Dataset Name	Speaker Native Language	No. Speakers
BLXXXeval1	Gujarati	223
BLXXXeval2	Latin American Spanish	220
BLXXXeval3	Polish,Vietnamese,Arabic,Dutch,French,Thai	226

Table 3: Testing data

Proficiency Level	BLXXXeval1	BLXXXeval2	BLXXXeval3
C2	1	5	2
C1	44	39	42
B2	44	44	48
B1	45	44	48
A2	44	44	48
A1	33	44	38
	12	-	-

Table 4: Distribution of Proficiency Levels per testing sets

Besides these data sets, we also have gold-standard grades for BLXXXeval1 and BLXXXeval3, which have been marked by experienced human graders being of higher quality in comparison to the ones mentioned before.

Lastly, we also have manual transcriptions of sections C,D and E for 16 candidates. Sections A and B were ignored as they comprise of short question-answers and read-aloud wordlists. We can summarize the DTAL dataset with the following table:

Proficiency Level	Candidates
B1	5
B2	6
C1	5

Table 5: Proficiency distribution of DTAL speakers

The intention of this study was to keep the manual transcriptions as faithful as possible to the true sounds in the recording, including transcriptions of disfluencies, actual pronunciation rather than the intended one and also incidental non-vocalised noises. The texts are segmented both manually, entitled “Syntactic” segmentation which tries to achieve optimal clause or phrase structure and automatically, entitled “Prosodic” segmentation which relies on silent pauses of length larger than 0.3 seconds.

8 Results and Analysis

In this section we are to discuss and analyse our findings. Firstly, we will commence by investigating what are the similarity scores pertaining from different parse tree structures between the manually transcribed data from the DTAL dataset and the ASR output. The second part will consist of showcasing the results for different configurations of features extracted from the parse trees using the standard Gaussian Process Grader from the „sklearn" package. Finally, we will discuss the advantages and shortcomings of using Deep Gaussian Processes for grading.

8.1 Dependency-Grammar based Tree Kernel Similarity

Using the various Dependency-Grammar derived parse tree structures defined in chapter 3 as inputs to the Smoothed Partial Tree Kernel algorithm, we obtain the following results:

Parse Tree Structure	DTAL „as-is" Similarity	DTAL ‚cleaned-up" Similarity
Grammatical Relation	79.4%	68.8%
Compositional GR	74.2%	64.2%
Lexical	75.7%	70.2%
Compositional Lexical	76.2%	69.1%

Table 6 : Pearson Correlation results between ASR and DTAL transcriptions using various Parse Tree Structures

Taking into consideration that the ASR transcriptions come from an ASR system with high Word Error Rate (approximately 37%), we obtain results which negate our concers that the parse trees cannot be used for feature extraction due to erroneous data. Rashid [2015] has obtained correlation coefficients of 71.5% for DTAL „as-is", respectively 70.9% for DTAL „cleaned-up" using the Syntactic Tree Kernel on the standard parse tree architecture obtained from RASP.Therefore, we obtain slighly higher similarity scores using our current models. This could potentially be caused by the fact that in the case of Syntactic Tree Kernels, just complete syntactic productions are taken into consideration for matching. With transciptions that have high WER, this leads a small error introduced by the ASR system to be given a higher weight in the overall similarity score. In the case of Smoothed Partial Tree Kernels any possible subsequence of varying lengths is considered for matching, hence providing higher flexibility, where in the case that one of the children of a particular node would be originating from a ASR error, it would be ignore in the computation while still allowing the other children present on the same layer to be used for matching. This is a desirable feature for a Tree Kernel applied on high WER data,as it is still able to gather some partial similairty from a path in the parse tree, one which would otherwise be ignored by Syntactic Tree Kernels. Besides the overall increase in similarity, we also observe larger gaps in the score using the Smoothed Partial Tree Kernel between the „as-is" and „cleaned-out" data sets. The „cleaned-up" data set has all difluencies removed and all identified errors corrected. For example, a possible error that is corrected in the „cleaned-up" version is the transformation of „you must to have a look" into „you must have a look". We can argue that the Smoothed Partial Tree Kernel has a higher discriminative power, being able to show a clear difference between two different versions of transcriptions which have clear syntactical and sematical differences.

Returning to our own results, we notice that the most similar structure for parse trees is the Grammatical Relation Centered Parse Tree. This result motivates the inclusion of Grammatical Relations tags obtained from the RASP output as features for our grader as a similarity of 79.4% could be deemed as satisfactory. With regards to the similarity score obtained by the Compositional Grammatical Relation Centered Trees, it is not surprising that it has achieved the lowest score as it involves a node similarity function that require complete matches of the Grammatical Relation and

the associated Part of Speech tags coming from both head and modifier. This stringent conditioning has lead to this slighly lower score. Nevertheless, 74.2% is a satisfactory score which motivates the usage of „1-layer deep" Annotated Skeletons as described in section 6.3.

With regards to Lexical Centered Trees, it was surprising to notice that Compositional Lexical Centered Trees attain a slighly better similarity score than Lexical Centered Trees, even though in the case of CLCTs this involves a node similarity function which conditions on the Part of Speech tags from both head and mofifier to be identical as opposed to the sole condition imposed on Lexical Centered Trees.

8.2 Syntactic Parse Tree Features

In the section we are to present results gained by devising different configurations of features for the standard Gaussian Process grader on top of the standard F1 features. Following the epxerimental result of Rahsid (2015) we have opted to use the raw counts of either Part of Speech tag or a Grammatical Relation as features, which we have entitled „PoS unigram", respectively „GR unigram" . The other options included binary presence and Term Frequency Inverse Document Frequency as expressing the features but were found to give worse results.

Our results are summarized in the following table:

Feature [dimensionality]	BLXXXgrd00 / BLXXXeval1		BLXXXgrd02/ BLXXXeval3	
	Correlation(%)	RMSE	Correlation(%)	RMSE
F1 features [29]	82.67	5.047	82.37	3.564
F1 + PoS unigram [158]	85.75	4.593	84.43	3.437
PoS unigram [129]	85.19	4.638	83.55	3.525
GR unigram [21]	82.74	4.797	78.50	3.802
F1 +GR unigram [50]	83.65	4.793	74.69	6.151
F1 + PoS + GR unigram [179]	85.27	4.622	84.18	3.442

Table 7: Gaussian Process Regression results using Syntactic features

In terms of performance of the pure syntactic features, both PoS unigram and GR unigram manage to obtain higher correlation and lower RMSE values that the set of standard features on the grd00/eval1 pair, whereas in the case of grd02/eval3 just PoS unigrams manage to obtain better scores. Nevertheless, these results show that Grammatical Relations tags can achieve comparable or near comparable performance in comparsion to the more widely used PoS tags. We notice that in the the first pair, the combiantion of F1 and Grammatical Relation unigrams has a positive effenct on the overall performance, whereas in the other pair it results in an almost double RMSE value. None of the above pairs achieves better results when combining both PoS tags and GR tags with the standard F1 feature set.

8.3 Structural Parse Tree Features

In this section we are present our findings regarding the feasability of using structural features such as Skeletons , Annotated Skeletons or „1-layer deep" Annotated Skeletons. We start by investigating the performance of using structural parse tree features as sole inputs to our Gaussian Grader.For all the experiments involving strcutural aprse tree, we use the Grammatical Centered Tree as the default tree from which we derive the skeletons. The following table summarizes our findings:

Skeleton type	25		75		150		300		500	
	Corr.	RMSE	Corr.	RMSE	Corr.	RMSE	Corr.	RMSE	Corr.	RMSE
Skeletons	58.31%	5.966	62.34%	5.775	65.61%	5.655	65.43%	5.655	66.20%	5.677
Annotated Skeletons	68.27%	5.516	66.76%	5.649	68.64%	5.503	69.75	5.516	69.32%	5.513
„1-layer deep" Annotated Skeleton	68.30	5.466	68.55	5.554	66.78	5.635	68.38	5.532	70.87	5.456

Table 8: Gaussian Process grader results depending using structural features

While in the process of identiying and counting all the possible skeletons present in our data we have arrived at a number of approximately 14000 different skeletons. Hence, we investigate how performance varies as we change the number of top occuring skeletons in the pair BLXXXgrd00/BLXXXeval1.

By looking at the results for each different type of structural parse tree architecture, we notice a steady increase in the correlation with the expert grades as we increase the number of skeletons which we include in the feature space. This is due to the fact that skeletons which are at the top in terms of appearences,such as the ones in top 25, tend to encode fairly simple syntactical and semantical information since they appear very often. As we progressively include more less occuring skeletons in the feature space, we thus manage to encode syntactic information of higher complexity which would be a sign of spoken language proficiency.

Another immediate observation which we can draw from the above table is the general increase in correlation with the expert grades as we increase the complexity of the information encoded in aut structural parse tree features, respectively transitioning from „blank" parse trees such as the Skeletons to ones which encode just the Grammatical Relation tag of the root node and finally transitioning to ones which also encode the Grammatical Relation tag of the children of the root node. Perhaps surprising is the relative small difference in performance between Annotated Trees and „1-layer deep" Annotated Trees.

We are to also investigate the results of using structural features with audio and fluency features, as they should complement each other. We use the top-500 skeletons in each case.

Features	Correlation (%)	RMSE
F1 + Skeletons	77.30	5.284
F1 + Annotated Skeletons	78.36	5.184
F1 + „1-layer deep" Annotated Skeletons	80.43	5.066

Table 9: Gaussian Process grader using standard feature set and structural features onBLXXXgrd00/BLXXXeval1 using the HC3 system

None of the structural features have been able to outperform the standard feature set baseline.

8.4 Deep Gaussian Process Regression

In the original paper showcasing the applications of Deep Gaussian Process using Expectation Propagation to real-life datasets [Bui et al., 2016] they investigate different DGP arhitectures with one hidden layer with a number of hidden units ranging from 1 to 3. Their experiments involve several large scale datasets from the UCI repository, such as the „year" dataset which comprises of 50,000 samples with a dimensionality of 90. The best results that they have obtained were using a Deep Gaussian Process with one hidden layer of size 3. Therefore, we are to start our experiments with similar configurations of the toolkit „deepGP_approxEP" which is used in this project,by using

ADAM with the default learning rate of 0.001,iterating for 4000 times and using Stochastic Expectation Propagation. However, we start our prelimiary experiments by using a slighly larger arhitecture with 6 hidden units.

8.5 Sparse Gaussian Process Regression using „1-layer" DGP-SEP

We start by obtaining baseline scores for our model by using it with one GP layer and no hidden layers, effectively acting as a sparse Gaussian Process. The following results were obtained:

Number inducing points	Correlation(%)	RMSE	AUC_{σ}	AUC_{MAX}	$\frac{AUC_{\sigma}}{AUC_{MAX}}$
50	83.13	4.867	0.396	0.725	0.546
100	81.05	4.671	0.461	0.752	0.612
200	79.36	5.095	0.484	0.743	0.650

Table 10: Sparse DGP-SEP using 1 layer results trained and evaluated on BLXXXgrd00/BLXXXeval1 using the F1 features set on the HC3 decoder data.

Number inducing points	Correlation(%)	RMSE	AUC_{σ}	AUC_{MAX}	$\frac{AUC_{\sigma}}{AUC_{MAX}}$
50	78.07	3.864	0.589	0.812	0.725
100	78.42	3.744	0.6136	0.8161	0.7519
200	78.93	3.792	0.563	0.811	0.693

Table 11: Sparse DGP-SEP using 1 layer results trained and evaluated on BLXXXgrd02/BLXXXeval3 using the F1 features set on the HC3 decoder data.

Looking at the table 10, we notice that using the DGP-SEP model with 50 inducing points as a simple Sparse Gaussian Process results in Pearson correlation scores which are above the 82.67% obtained in the case of using a full Gaussian Process. In terms of RMSE scores,our baseline model from the full Gaussian Process model has a RMSE value of 5.047,which is surpassed by our DGP-SEP models with 50 and 100 inducing points by quite a considerable proportion. It is surprising that better results can be achieved with such high sparsification.

By looking at table 11, we notice that our sparse DGP-SEP model has not managed to obtain the same performance as the full Gaussian Process applied on BLXXXgrd02/BLXXXeval3, in all three cases underperforming both in terms of correlation and RMSE since our baseline has a correlation of 82.37% and a RMSE of 3.564

In terms of pure predictive power, our sparse DGP-SEP model has peformed well on the BLXXXgrd00/BLXXXeval1 pair, we notice an extremely poor performance in terms of the rejection scheme based on predictive variance(Van Dalen et al., 2015).On the other hand, our model exhibits a reverse behaviour on the pair BLXXXgrd02/BLXXXeval3. The following plots are taken for the case

of 50 inducing points for each of the trening/testing pairs and are illustrative of the other cases as well:

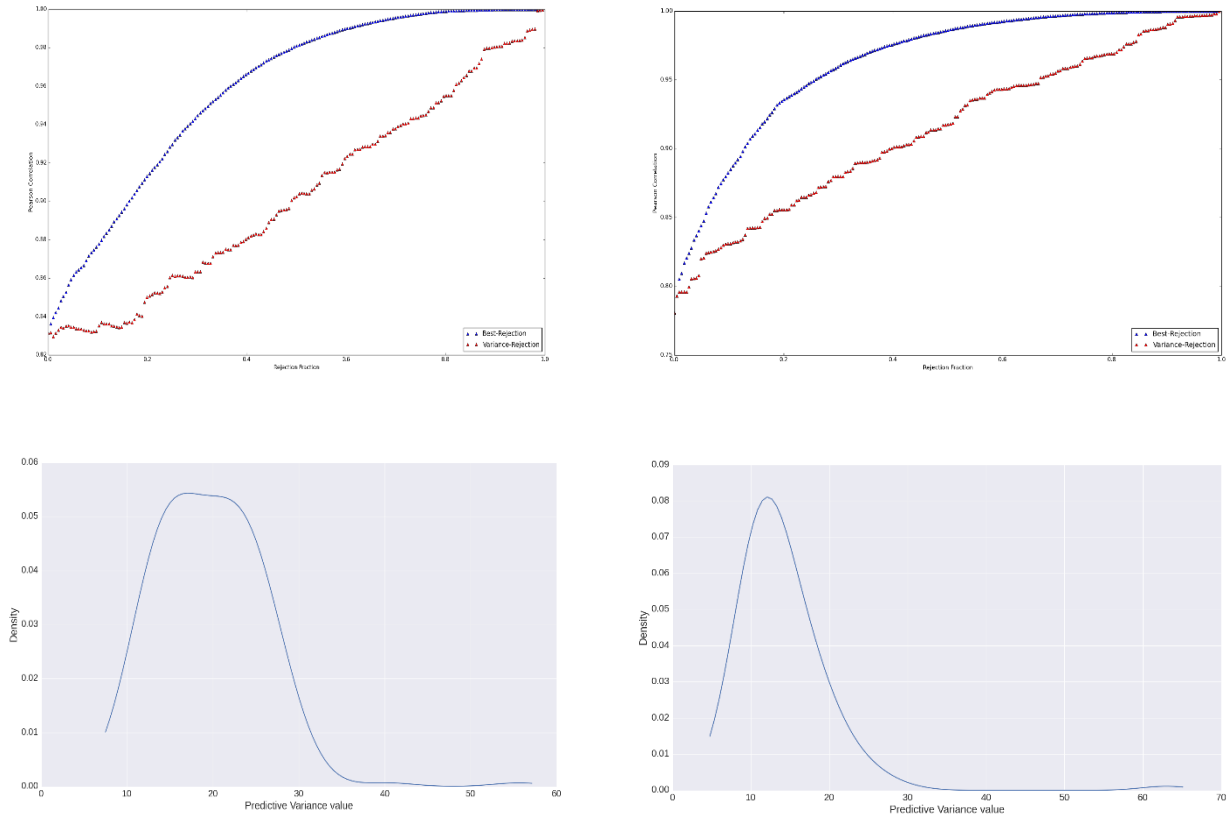

Figure 7: Left – AUC Rejection plot and variance density for BLXXXgrd00/BLXXXeval1;

Right – AUC rejection plot and variance density for BLXXXgrd02/BLXXXeval3;

From the above figure which depict cases representative also for 100 and 200 inducing points, we notice that the predictive variance in the case of BLXXXgrd00/BLXXXeval1 is far greater that the one for BLXXXgrd02/BLXXXeval3, resulting in erroneous behaviour of the rejection scheme which achieves score lower than 0.5 in all cases.

We are now to examine what are the effects of extending our DGP-SEP model with one hidden layer with 6 hidden units. The results are summarized in the following table:

Number inducing points	Correlation(%)	RMSE	AUC_{σ}	AUC_{MAX}	$\frac{AUC_{\sigma}}{AUC_{MAX}}$
50	83.37	5.132	0.364	0.719	0.507
100	82.97	4.914	0.417	0.727	0.574
200	83.07	5.143	0.358	0.7123	0.502

Table 12: Sparse DGP-SEP using 1 hidden layer with 6 units results trained and evaluated on BLXXXgrd00/BLXXXeval1 using the F1 features set on the HC3 decoder data

Number inducing points	Correlation(%)	RMSE	AUC_{σ}	AUC_{MAX}	$\frac{AUC_{\sigma}}{AUC_{MAX}}$
50	78.71	3.7226	0.5897	0.8072	0.7306
100	79.43	3.8329	0.5652	0.8003	0.7062
200	79.02	3.902	0.5594	0.8033	0.6964

Table 13: Sparse DGP-SEP using 1 hidden layer with 6 units results trained and evaluated on BLXXXgrd02/BLXXXeval3 using the F1 features set on the HC3 decoder data

For the BLXXXgrd00/BLXXXeval1 pair, we notice an unilateral increase in correlation compared to the previous case without any hidden layers. However, we did not observe an improvement in terms of RMSE scores or in the ration of areas under the curve. This comes as a surprise as the hidden layer should have added more flexibility in modelling and more well-calibrated uncertainty.

For the other pair, the results more or less stay the same which motivated us to increase the number of hidden units to 25. The results are summarized in the following two tables:

Number inducing points	Correlation(%)	RMSE	AUC_{σ}	AUC_{MAX}	$\frac{AUC_{\sigma}}{AUC_{MAX}}$
50	83.33	5.125	0.3213	0.7176	0.4477
100	82.31	4.853	0.4322	0.7378	0.5858
200	82.60	5.154	0.3399	0.7079	0.4801

Table 14: Sparse DGP-SEP using 1 hidden layer with 25 units results trained and evaluated on BLXXXgrd00/BLXXXeval1 using the F1 features set on the HC3 decoder data

Number inducing points	Correlation(%)	RMSE	AUC_{σ}	AUC_{MAX}	$\frac{AUC_{\sigma}}{AUC_{MAX}}$
50	79.67	3.730	0.5700	0.8038	0.7091
100	80.58	3.609	0.5681	0.8035	0.7070
200	80.19	3.740	0.5600	0.7968	0.7033

Table 15: Sparse DGP-SEP using 1 hidden layer with 25 units results trained and evaluated on BLXXXgrd02/BLXXXeval3 using the F1 features set on the HC3 decoder data

Even upon expanding the hidden layer dimensionality to 25, in the case of BLXXXgrd00/BLXXXeval1 we still do not observe a noticeable enhancement of the variance associated to its predictions, whereas in the other case we do not manage to improve the results in neither correlation, nor RMSE value. Further experiments were conducted on the BLXXXgrd00/BLXXXeval1 pair with up to 3 hidden layers but no significant improvement was noticed. One possibility was that given the small scale dataset and the low dimensionality fo just 29, the model reached its limit even from the case with no hidden layers. This plausible explanation motivated us to try the DGP-SEP model on data sets with larger dimensionality.

We now move on to apply the DGP-SEP model on the F1 features plus the PoS unigrams data sets. All of the results that are shown are produced with 50 inducing points. The following table highlights our findings:

Data Pair	Hidden Layers	Hidden Units	Corr.	RMSE
BLXXXgrd00 - BLXXXeval1	0	-	82.83%	4.7990
	1	6	88.56%	4.5193
	1	25	88.75%	4.4465
	1	50	88.66%	4.4650
BLXXXgrd02 - BLXXXeval3	0	-	85.12%	3.3205
	1	6	84.87%	3.3921
	1	25	85.00%	3.3520
	1	50	85.08%	3.4102

Table 16: DGP-SEP results applied on the F1 + PoS unigram data sets

In the case of pair BLXXXgrd02/BLXXXeval3, the DGP-SEP model outperforms the standard GP regression baseline of 84.43% and RMSE score of 3.437 even without a hidden layer. Adding a hidden layer does not help in improving significantly the performance.

In terms of the pair BLXXXgrd00/BLXXXeval1, the DGP-SEP model without any hidden layers does not manage to reproduce the scores obtained by the standard GP model, but as we increase to one hidden layer of a relatively small dimensionality, respectively 6, we already obtain a correlation on 88.56%, which represents an almost 3% increase.

9 Conclusions

We can organise the contributions of this project in three different parts.

Firstly, by choosing the Smoothed Partial Tree Kernel we have added flexibility to the process of detecting similairty between ASR output and manual transcriptions of the same speech recordings. As we have seen in the difference in scores between the „as-is" and „cleaned-up" sets of transcriptions, Smoothed Partial Tree Kernels are able to discriminate better when the syntactic and semantic elements of sentences subly change. On top of this, due to the fact that it is not operating in the sub-tree domain, but in the partial tree domain, meaning that exact matches of all children of a node are not imposed, it becomes more suitable for ASR transcriptions with high Word Error Rate. Besides these, by introducing the Grammatical Centered Tree and the Lexical Centered Tree alongisde the fact that they obtain scores of over 75% in similairty with manual transcriptions, we have proven empirically the feasability of suing PoS or GR tags in a grader system. In addition, the good similarity scores obtained by the Compositional Grammatical and Lexical Centered Trees, have also proven empirically the feasability of using n-grams based on PoS or GR tags. Nevertheless, further in-depth study must be done on exploring at what depth in the parse trees the similarity is more pronounced for different parse tree structures. This could be done by varying the horizontal and vertical decay factors in the theoretical construction of the Smoothed Partial Tree Kernel.

Secondly, in our work we have proven that Grammatical Relation tags work when introduced even as sole inputs in a standard Gaussian Process grader.However, they do not provide the same accuracy as PoS tags. A promising research direction for future work is the development of the structural parse trees and extending them to incorporate more information. Feasbile designs might include mixing PoS tags and GR tags in a mixed Annotated Skeleton.

Lastly, in our work we have implemented Deep Gaussian Process graders which increased the accuray by a significant percentage, such as in the case of the F1 feature set combined with the Part of Speech unigram feature set, increasing from approximately 85% to almost 89%. However, in some cases we have not been able to train the Deep Gaussian Process grader to give satisfactory results. Future projects with Deep Gaussian Process graders should attempt an exhaustive experimentation of more complex and deep architectures for more input features. In addition, another interesting research direction might be to incorporate in any Gaussian Process based grade, an uncertainty in the inputs, which might better reflect the reality of features stemming from statistical parse trees.

References:

Basili, R., Annesi, P., Castellucci, G. and Croce, D., A Compositional Perspective in Convolution Kernels.

Bui, T.D., Hernández-Lobato, D., Li, Y., Hernández-Lobato, J.M. and Turner, R.E., 2016. Deep Gaussian Processes for Regression using Approximate Expectation Propagation. *arXiv preprint arXiv:1602.04133*.

Collins, M. and Duffy, N., 2001. Convolution kernels for natural language. In *Advances in neural information processing systems* (pp. 625-632).

Croce, D., Moschitti, A. and Basili, R., 2011, July. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (pp. 1034-1046). Association for Computational Linguistics.

Damianou, A.C. and Lawrence, N.D., 2013, August. Deep Gaussian Processes. In *AISTATS* (pp. 207-215).

Durrande, N., Ginsbourger, D. and Roustant, O., 2012. Additive covariance kernels for high-dimensional Gaussian process modeling. In *Annales de la Faculté de Sciences de Toulouse* (Vol. 21, No. 3, pp. p-481).

Franco, H., Neumeyer, L., Digalakis, V. and Ronen, O., 2000. Combination of machine scores for automatic grading of pronunciation quality. *Speech Communication*, *30*(2), pp.121-130.

Gelman, A., Vehtari, A., Jylänki, P., Robert, C., Chopin, N. and Cunningham, J.P., 2014. Expectation propagation as a way of life. *arXiv preprint arXiv:1412.4869*.

Ghahramani, Z. and Snelson, E., 2006. Sparse Gaussian process using pseudoinputs. *Advances in Neural Info. Processing Sys., Cambridge, MA*.

Ghahramani, Z., 2010, A tutorial on Gaussian Processes, Eurandom, 2010.

Hensman, J. and Lawrence, N.D., 2014. Nested variational compression in deep Gaussian processes. *arXiv preprint arXiv:1412.1370*.

Jiang, J. and Zhai, C., 2007, June. Instance weighting for domain adaptation in NLP. In *ACL* (Vol. 7, pp. 264-271).

Massung, S., Zhai, C. and Hockenmaier, J., 2013, September. Structural parse tree features for text representation. In *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on* (pp. 9-16). IEEE.

Moschitti, A., 2006, April. Making Tree Kernels Practical for Natural Language Learning. In *EACL* (Vol. 113, No. 120, p. 24).

Moschitti, A., 2006, September. Efficient convolution kernels for dependency and constituent syntactic trees. In *European Conference on Machine Learning* (pp. 318-329). Springer Berlin Heidelberg.

van Dalen, R.C., Knill, K.M. and Gales, M.J., 2015, August. Automatically grading learners' English using a Gaussian process. ISCA.

Verhelst, N., Van Avermaet, P., Takala, S., Figueras, N. and North, B., 2009. *Common European Framework of Reference for Languages: learning, teaching, assessment*. Cambridge University Press.

Wilson, A.G., Knowles, D.A. and Ghahramani, Z., 2011. Gaussian process regression networks. *arXiv preprint arXiv:1110.4411*.
Titsias, M.K., 2009, January. Variational Learning of Inducing Variables in Sparse Gaussian Processes. In *AISTATS* (Vol. 12, pp. 567-574).

Yu, Z., Ramanarayanan, V., Suendermann-Oeft, D., Wang, X., Zechner, K., Chen, L., Tao, J., Ivanou, A. and Qian, Y., 2015, December. Using bidirectional lstm recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 338-345). IEEE.

Zechner, K. and Bejar, I.I., 2006, June. Towards automatic scoring of non-native spontaneous speech. In *Proceedings of the main conference on human language technology conference of the North American chapter of the association of computational linguistics* (pp. 216-223). Association for Computational Linguistics.