Chapter 3

Using GAZE for gene finding in *Caenorhabditis elegans*

3.1 Introduction

This chapter documents my work in applying the GAZE system to the prediction of gene structures in the genome sequence of the nematode worm *Caenorhabditis elegans*. Much work in the past has focused on gene prediction in the sequences of vertebrates (particularly human), and a later chapter shows how GAZE can be applied effectively to vertebrate genomes. However, GAZE was originally envisioned as a curation tool for *C.elegans* sequence annotators, and this work on its application to the worm genome is rooted in the origins of the project.

Although gene prediction in *C.elegans* sequences is considered by most researchers to be easier than in vertebrate sequences, certain complications make it a non-trivial problem (see section 1.5). Indeed, the author of one of the most widely used and accurate gene prediction programs [21] has admitted difficulty in arriving at a set of parameters that work well on worm sequences [http://genes.mit.edu/-Limitations.html].

Below, I outline the steps involved in the definition of a configuration from first principles, starting with only candidate gene features generated from simple signal and content sensors. I then go on to successively refine this model, at each stage explaining both specific steps taken and exploring the resulting impact on prediction accuracy. In particular, I make use of the flexibility of GAZE to firstly take account of a post-transcriptional modification process that is peculiar to *C.elegans* and similar animals, namely *trans*-splicing; and secondly to improve accuracy by the incorporation of similarity information, specifically alignments of Expressed Sequence Tags.

3.2 Gene prediction materials for *C.elegans*

3.2.1 WormBase and The WormSeq dataset

To evaluate the accuracy of the various gene-structure models presented in this chapter, both in comparison with each other and with other gene prediction programs, it is necessary to construct a test-set. The principle source of data for the test-set I have constructed was WormBase[1] [107], a database containing the complete, annotated genome sequence of *C. elegans*.

WormBase provides as part of its annotation a complete set of gene structures for the *C. elegans* genome. These gene structures represent manually-inspected predictions of the *coding* regions (or *CDS*) of the genes, based on GENEFINDER predictions (see below) and other available supporting evidence for the structure. Such evidence most frequently comes in the form of *spliced alignment* of expressed sequence (cDNAs or ESTs) back to the genomic sequence by a program such as EST_GENOME [79] or BLAT [61], giving the intron-exon junctions of the structure (see chapter 1). As more cDNAs and ESTs are sequenced and deposited in nucleotide databases, they are aligned to the genome and the set of curated gene structures revised and updated to be consistent with the alignments. At any one time then, the set of curated gene structures in WormBase represents a current "best-guess" based on the available supporting evidence.

[1]Specifically WS52, September 2001

The test-set I have built represents an attempt to identify the subset of curated gene structures that have sound and complete supporting evidence for their validity. An initial set was constructed by taking the curated structures supported by the alignment of at least one "external" cDNA to the genome (i.e. those deposited in the EMBL database by a group not working directly on the *C.elegans* genome sequencing project). Since these alignments (produced by EST_GENOME, [79]) were only present (at the time of construction) for the half of the worm genome that is maintained at the Sanger Institute ("WormBase_Sanger"), the set contained only Sanger Institute genes. However, the restriction to external cDNAs provides a degree of independent verification to the structures.

The initial set of gene structures was then subjected to a set of filtering steps, removing the following entries:

1. Those for which the set of supporting cDNAs did not contain at least one entry annotated as having "complete CDS" in its EMBL entry. It is possible for a structure to be confirmed by two separate partial cDNAs aligning to different parts of the structure, but I took the conservative approach of removing such entries.

2. Those structures that overlap with at least one other curated gene structure. This removes those genes that are known to be alternatively spliced, and also those situated within the introns of other genes.

To check the consistency of the gene structures with respect to the cDNAs supporting them, I performed a Smith-Waterman [100] local alignment of each cDNA to the CDS of the corresponding gene structure, using the program DNAL (E. Birney, unpublished). Those entries where the gene CDS did not align precisely with the EMBL-annotated CDS of the cDNA were presented to the Sanger Institute *C.elegans* curation group for examination, which resulted in the editing of some of these gene structures. The final set consisted of 325 gene structures (157 situated on the forward strand of the genomic sequence and 168 on the reverse). The average number of exons per gene in the set is 6.9 (compared to 6.3 for all curated gene structures in WormBase), with 16 single-exon genes.

Traditionally, assessments of gene prediction programs have been performed against sequences that contain a single gene for which the structure has been confirmed [23] [94]. However, when gene prediction programs are used in a production environment on large, unannotated fragments of genomic DNA, they cannot know in advance how many genes, if any, the sequence contains. Modern programs are therefore capable of predicting many genes in a query sequence. To assess the multiple-gene-prediction capabilities of such programs, test sequences containing several genes are necessary. Large genomic fragments for which the entire exon-intron structure for all genes has been confirmed are extremely hard to come by however. There is also the problem of having confidence that we know about all of the genes on a test sequence, and that the regions annotated as intergenic really do contain no genes.

Having identified a set of 325 cDNA-confirmed genes, it would be ideal if they were located together in the worm genome. Unsurprisingly, this was not the case, and the genes are spread across 9 genomic contigs. In an attempt to recreate the conditions faced by gene-prediction programs in practical use, I constructed an artificial genomic contig containing the confirmed genes. Unlike the dataset made by Guigo and fellow workers [53], where the intergenic regions were generated "randomly", I took the approach of embedding the gene sequences in real genomic DNA. Specifically, the DNA underlying each confirmed gene structure was extracted, along with half of the intergenic DNA to the nearest other *curated* gene (confirmed or unconfirmed) in each upstream and downstream direction. These sequences were then concatenated in the order in which the were situated on the original genomic contigs to make a contiguous artificial genomic sequence of 2,079,582 base pairs. This sequence is referred to as *WormSeq*. The proportion of WormSeq that is protein-coding is 0.24, which is representative of estimates for the genome as a whole based upon all curated structures in WormBase.

The WormSeq sequence, and its annotated gene structure, forms the basis for much of the analysis presented in this chapter. It can be obtained from http://www.-sanger.ac.uk/Software/analysis/GAZE/wormseq.

3.2.2 A source of gene prediction data: GENEFINDER

GAZE is not an integrated gene prediction program; it requires a set of features, segments, and length penalty functions. For the analyses presented in this chapter, the GENEFINDER program (P. Green, unpublished) was the effective source for these data. Although unpublished, GENEFINDER is widely regarded as one of the most accurate gene prediction programs for the worm. The GAZE scoring function explained in chapter 2 is similar to the one used in GENEFINDER in that the score for a complete gene structure is comprised of a sum of the scores of the features that define the structure, along with scores for the regions between these features. Also, GENEFINDER includes length penalty files and frequency tables for various gene features, the details of which are described below. Permission for use of these files was kindly granted by the author [P. Green, pers.comm].

The ACeDB package [www.acedb.org] contains a module adapted from the original GENEFINDER code, GF_FEATURES, that takes as input a set of GENEFINDER frequency tables and a query DNA sequence and produces predictions of features corresponding to the given tables, in GFF. The GF_FEATURES program was used together with the tables from the 980506 distribution of GENEFINDER to produce features and segments in the manner described below for all of the analyses in this chapter, unless stated otherwise.

Signal sensors in GENEFINDER

The GENEFINDER tables are used to construct weight matrices of log-likelihood ratios of nucleotide b in position i for true sites compared to randomised DNA. Specifically, if for a feature of interest *True* and *Rand* are respectively the tables for the true sites and randomised DNA, then the log likelihood-ratio for nucleotide b in position

i is calculated as

$$llr_i(b) = ll_i^{True}(b) - ll_i^{Rand}(b)$$

where the log likelihood calculated from a table T, ll_i^T is:

$$\begin{aligned} ll_i^T(b) &= \log(1+\frac{1}{T_i(b)})T_i(b) \\ &- \log(1+\frac{1}{\sum_j T_i(j)})\sum_j T_i(j) \\ &+ \log(\frac{1+T_i(b)}{1+\sum_j T_i(j)}) \end{aligned}$$

If the width of the table for a particular type of site is n, then a score $g(S)$ for a candidate site $S = s_1 s_2 \ldots s_n$ can be calculated as:

$$g(S) = \sum_{i=1}^{n} llr_i(s_i)$$

GENEFINDER provides tables for the following gene features: translation start sites representing positions -9 through +11 (where 0 is the position of the A in the completely conserved ATG, which is enforced); both the donor and acceptor splice sites, representing 6 and 25 nucleotides of the corresponding exon and intron respectively, with enforcement of the GT-AG intron rule; and finally, translation termination sites representing 13 nucleotides of the upstream coding region and 92 nucleotides of the downstream untranslated region, with enforcement of the (TAG|TAA|TGA) rule. The table for each feature is accompanied by a table of corresponding dimensions populated by counts from "random" DNA. The GF_FEATURES programs constructs *llr* matrices from the given tables, scores windows of the query sequence using the matrices, and outputs predictions of each feature scoring above the default threshold (usually 0.0, but -2.0 for acceptor splice sites).

Content sensors in GENEFINDER

As well as predicting features using frequency tables, GF_FEATURES can also be used as a content sensor, in particular in the detection of protein-coding regions. It calculates log-likelihood ratios for each n-mer based on the frequency of occurrences in protein-coding regions compared with randomised DNA. It then scans the sequence in each of the six reading frames, at each position storing the sum of scores over all non-overlapping n-mers up to that point. The cumulative score array thus obtained for the whole query sequence for a reading frame is then used to obtain a set of maximal scoring coding segments for that frame, and those segments scoring above 1.0 are output in GFF as predicted coding regions.

The n-mer tables in GENEFINDER 980506 for the detection of coding-regions are in-frame 3-mers, i.e. codons.

Length Penalty functions in GENEFINDER

GENEFINDER 980506 includes length-penalty tables for introns, and initial, internal and terminal exons. These are defined as (length, penalty) pairs, so could be used largely as found, except where otherwise stated. Single exon genes and intergenic regions are subject to a constant, length-independent penalty.

3.3 Definition of a GAZE configuration in three steps

This section outlines the steps involved in the development of a simple GAZE model for drawing together the signal, content and length penalty information provided by GENEFINDER into predictions of complete gene structures. The rules presented here form the core for all of the models that I have subsequently developed; indeed the principal way in which I envision GAZE being used in practice is for models to be developed by taking existing models and tweaking them for specific situations. I see this simple model as forming a base for practically all future models that one might develop.

```
<?xml version="1.0" encoding="US-ASCII"?>

<gaze>
    <declarations>
        <feature id="start" st_off="0" en_off="3"/>
        <feature id="stop" st_off="3" en_off="0"/>
        <segment id="coding_seg" scoring="standard_max"/>
        <lengthfunction id="single_exon_pen"/>
    </declarations>

    <gff2gaze>
        <gffline feature="atg" source="Genefinder" strand="+">
            <feat id="start"/>
        </gffline>
        <gffline feature="stop" source="Genefinder" strand="+">
            <feat id="stop"/>
        </gffline>
        <gffline feature="coding_seg" source="Genefinder" strand="+">
            <seg id="coding_seg"/>
        </gffline>
    </gff2gaze>

    <dna2gaze>
        <dnafeat pattern="atg">
            <feat id="start"/>
        </dnafeat>
        <dnafeat pattern="taa">
            <feat id="stop"/>
        </dnafeat>
        <dnafeat pattern="tag">
            <feat id="stop"/>
        </dnafeat>
        <dnafeat pattern="tga">
            <feat id="stop"/>
        </dnafeat>
    </dna2gaze>

    <lengthfunctions>
        <lengthfunc id="sngl_ex_pen">
            <point x="0" y="4"/>
            <point x="1" y="4"/>
        </lengthfunc>
    </lengthfunctions>

    <model>
        <target id="END">
            <source id="BEGIN">
                <output feature="no genes"/>
            </source>
            <source id="stop">
                <output feature="intergenic"/>
            </source>
        </target>

        <target id="start">
            <source id="BEGIN">
                <output feature="intergenic"/>
            </source>
        </target>

        <target id="stop">
            <killfeat id="stop" target_phase="0"/>
            <useseg id="coding_seg" target_phase="0"/>

            <source id="start" mindis="6" len_fun="sngl_ex_pen" phase="0">
                <output feature="CDS" strand="+" frame="0"/>
            </source>
        </target>
    </model>

</gaze>
```

Figure 3.1: A complete GAZE-XML configuration file for the prediction of a single, single-exon gene on the forward strand of a DNA sequence

3.3.1 A single, single-exon gene

Figure 3.1 shows a GAZE configuration for the prediction of a single, single-exon gene on the forward strand. Although simple, this configuration makes use of the majority of features of GAZE.

The XML configurations file can be viewed quite simply as comprising five sections. The *declarations* sections declares the features, segments and length penalty functions that GAZE is going to work with, along with some of the core properties that are common to all elements of each type. Of particular note here is the *scoring* attribute given for the "coding_reg" segments, which dictates in this case that segments of this type should be scored according to the "maximal single" scheme, given by equation 2.4.

The *gff2gaze* section dictates how the input GFF files are used to obtain lists of features and segments. In particular here, the *gffline* tag is used, together with the *source*, *feature* and *strand* attributes to specify which GFF lines are relevant, and which features to create when lines matching those criteria are observed. The *dna2gaze* section also allows for the creation of features from simple sequence motifs observed in the input DNA sequence.

The *model* section contains the *source* $\rightarrow$ *target* rules. Those involving the "stop" target are of particular interest here because they make use of the majority of features available in GAZE.

Firstly, a segment qualifier (denoted by the *useseg* tag) that is *global* to the target is used to denote the fact that "coding_reg" segments that are in-phase with respect to the target (via the *target_phase* attribute) are considered relevant for *all* legal sources for this target.

Secondly, an interruption constraint (denoted by the *killfeat* tag) is used in a similarly global way to invalidate the region between the target and *any* legal upstream source when interrupted by a "stop" feature that is in-phase with the target.

Thirdly, the *start* $\rightarrow$ *stop* rule specifically contains minimum-distance and phase constraints, as well as denoting that the "sngl_ex_pen" length penalty function should be used. The length function itself, taken from GENEFINDER, is defined at the top of the second column. Note that this particular length penalty is actually length independent, achieved by giving consecutive distances the same penalty.

Finally, the output qualifiers define information for how the regions in the final gene structure should be presented to the user. The output format of GAZE is GFF, hence output GFF tags can be attached to each rule.

3.3.2 Extension to spliced structures

Although this simple model is satisfactory for explaining some of the features of GAZE, it does not have much worth in practice because coding portions of the majority of *C. elegans* genes are interrupted by introns. Figure 3.2 show how the simple model in the configuration above can be easily extended to model spliced gene structures.

The first point of note is the way that intron *phases* are dealt with. Introns can

```
⋮
<gff2gaze>
    <gffline feature="splice5" source="Genefinder" strand="+">
        <feat id="5ss_0"/>
        <feat id="5ss_1"/>
        <feat id="5ss_2"/>
    </gffline>

    <gffline feature="splice3" source="Genefinder" strand="+">
        <feat id="3ss_0"/>
        <feat id="3ss_1"/>
        <feat id="3ss_2"/>
    </gffline>
    ⋮
</gff2gaze>

<dna2gaze>
    ⋮
    <takedna id="5ss_1" st_off="0" en_off="1"/>
    <takedna id="3ss_1" st_off="1" en_off="−1"/>
    <takedna id="5ss_2" st_off="−1" en_off="1"/>
    <takedna id="3ss_2" st_off="1" en_off="0"/>
</dna2gaze>
⋮
<model>
    ⋮
    <target id="stop">
        <useseg id="coding_seg" target_phase="0"/>
        <killfeat id="stop" target_phase="0"/>

        <source id="start" mindis="6" len_fun="sngl_ex_pen" phase="0">
            <output feature="CDS" strand="+" frame="0"/>
        </source>

        <source id="3ss_0" mindis="3" len_fun="term_ex_pen" phase="0">
            <output feature="CDS" strand="+" frame="0"/>
        </source>

        <source id="3ss_1" mindis="3" len_fun="term_ex_pen" phase="2">
            <output feature="CDS" strand="+" frame="1"/>
        </source>

        <source id="3ss_2" mindis="3" len_fun="term_ex_pen" phase="1">
            <output feature="CDS" strand="+" frame="2"/>
        </source>
    </target>
```

```
    ⋮
    <target id="5ss_1">
        <useseg id="coding_seg" target_phase="1"/>
        <killfeat id="stop" target_phase="1"/>

        <source id="start" mindis="3"  len_fun="init_ex_pen" phase="1">
            <output feature="CDS" strand="+" frame="0"/>
        </source>

        <source id="3ss_0" mindis="20" len_fun="int_ex_pen" phase="1">
            <output feature="CDS" strand="+" frame="0"/>
        </source>

        <source id="3ss_1" mindis="20" len_fun="int_ex_pen" phase="0">
            <output feature="CDS" strand="+" frame="1"/>
        </source>

        <source id="3ss_2" mindis="20" len_fun="int_ex_pen" phase="2">
            <output feature="CDS" strand="+" frame="2"/>
        </source>
    </target>
    ⋮

    <target id="3ss_1">
        <source id="5ss_1" mindis="39" len_fun="intron_pen">
            <killdna source_dna="t" target_dna="aa"/>
            <killdna source_dna="t" target_dna="ag"/>
            <killdna source_dna="t" target_dna="ga"/>
            <output feature="intron" strand="+"/>
        </source>
    </target>
    ⋮

</model>
⋮
```

Figure 3.2: The main elements of a GAZE configuration for the prediction of single, possibly-spliced gene structures of the forward strand of a DNA sequence

interrupt the coding region of a gene either between two codons (a phase 0 intron) or in the middle of a codon, either between the first and second codon positions (phase 1 intron) or between the second and third positions (phase 2 intron). The total length of the spliced coding region must be a multiple of 3 in order for it to be successfully translated, which means that is important for programs to keep track of intron phases in order to produce sensible predictions.

The technique used here to account for intron phases is to consider the phases of the donor and acceptor splice site features that define the intron region. As shown in figure 3.2, for each given predicted splice site, three features are made, one for each of the possible codon positions that the splice site may occur at. For example, "5ss_1" denotes a donor splice site occurring between the 1st and 2nd codon position. The rules are constructed in such as way as to ensure that any partial candidate gene structure ending with a “5ss_1” ends with the first base of an incomplete codon; this is achieved firstly by the use of phase constraints, and secondly by dictating that phases must be conserved across introns; notice from the figure that a phase 1 acceptor splice site (“3ss_1”) can only be legally preceded by a donor splice site of the same phase (“5ss_1”).

In the simple model presented earlier, candidate coding exons with in-frame stop-codons were disallowed by using an interruption constraint. Splicing also introduces the possibility that stop-codons occur at the junction formed by the concatenation of two exons. The figure shows how DNA constraints are used to disallow such structures. Firstly, in the *dna2gaze* section, *takedna* directives are given for certain features, which are instructions for GAZE to keep a record of the DNA sequence occurring at the location of these these feature, specifically between the given start and end position of the features, each adjusted by the offsets defined in the directive (“st_off” and “en_off”). The rule for the “3ss_1” target contains the DNA constraints themselves (*killdna*), an example of which states that the region between a “5ss_1” and “3ss_1” is illegal if the DNA at the “5ss_1” is ’T’ and the DNA at the “3ss_1” is ’AA.’

This mechanism has a limitation: it is technically possible for a stop-codon to be formed from three successive exons, where the second exon consists of a single base-pair. Internal exons this small, even if biologically possible, would be extremely rare, and in fact are disallowed in the all of my models by use of a minimum distance constraint. As shown in the figure, the minimum I use, taken from GENEFINDER, is 20 base pairs.

3.3.3 Extending to multiple genes on both strands

Figure 3.3 gives a flavour of extensions to the model to allow for the possibility of more than one gene on the query sequence. The “start” feature can now be immediately preceded by a “stop” source marking the end of another gene, with

```
⋮
<gff2gaze>
   ⋮
   <gffline feature="atg" source="Genefinder" strand="−">
     <feat id="start_rev"/>
   </gffline>
   ⋮
   <gffline feature="splice5" source="Genefinder" strand="−">
     <feat id="5ss_0_rev"/>
     <feat id="5ss_1_rev"/>
     <feat id="5ss_2_rev"/>
   </gffline>
   ⋮
</gff2gaze>

<dna2gaze>
   ⋮
   <dnafeat pattern="cat">
     <feat id="start_rev"/>
   </dnafeat>
   ⋮
</dna2gaze>
```

```
  <lengthfunctions>
     ⋮
     <lengthfunc id="intergene_pen">
         <point x="0" y="4"/>
         <point x="1" y="4"/>
     </lengthfunc>
     ⋮
  </lengthfunctions>

  <model>
     ⋮
    <target id="start">
       <source id="BEGIN">
          <output feature="intergenic"/>
       </source>
       <source id="stop" mindis="0" len_fun="intergene_pen">
          <output feature="intergenic"/>
       </source>
       <source id="start_rev" mindis="0" len_fun="intergene_pen">
          <output feature="intergenic"/>
       </source>
    </target>
     ⋮
  </model>
</gaze>
```

Figure 3.3: A fragment of a GAZE configuration showing the elements involved in modelling multiple genes on both strands of a DNA sequence.

a new length-penalty function for the intergenic region implied by a pair of such features.

Some gene prediction programs model reverse strand genes by predicting separately on each of the given sequence and its reverse complement, and then merging the predictions back together. This can cause problems if the gene prediction signal is strong on both strands at the same location, where it is not obvious what the gene prediction should be in this region. Other programs, for example GENSCAN, incorporate a single integrated model for the prediction of genes on both strands. They do this by firstly choosing (arbitrarily) a strand as the reference strand, and secondly treating opposite strand genes as comprising of the same features but occurring in reverse order. Hence from the point of view of the reference strand, opposite-strand genes begin with a stop-codon and end with a start-codon. This approach is easily implemented in GAZE, demonstrated by the rules for the "start" target in figure 3.3, which include the "start_rev" source, marking the start of a gene on the opposite strand. The full model, which I refer to as GAZE_std, is represented pictorially in figure 3.4.

3.4 Applying the model to *C.elegans* sequences

The accuracy of the model presented above in comparison to other available programs is examined in detail in a subsequent section, where it is shown how various refinements affect the performance. This section uses the application of the model to the WormSeq test sequence to demonstrate some aspects of the functionality of GAZE.

3.4.1 Predicting genes in WormSeq

Table 3.1 shows the accuracy of GAZE_std at the whole gene level. For the purposes of specificity, a prediction is considered correct only if the complete gene structure matches precisely the structure of the corresponding cDNA-confirmed structure; likewise, for the purposes of sensitivity, a confirmed gene structure is considered to be correctly predicted only if it is matched precisely by a corresponding GAZE-predicted gene. The stringency of this measure is shown in the table; only about a third of WormSeq genes are predicted correctly by GAZE_std and only a third of GAZE_std predictions are correct. Upon visual inspection of the predictions in comparison to the correct gene structures in ACeDB, it is apparent that many of them have the correct or nearly-correct intron-exon structure, but err in the location of their start and/or end. It is widely observed that the ends of genes are more difficult to identify correctly than their internal intron-exon structure (see chapter 1), and observation of the accuracy of GAZE_std on WormSeq supports this.

3.4.2 Using feature-selection to refine the predictions

If the starts and ends of genes are the most difficult features to identify, it is interesting to ask how the accuracy of GAZE_std changes when it is told where the

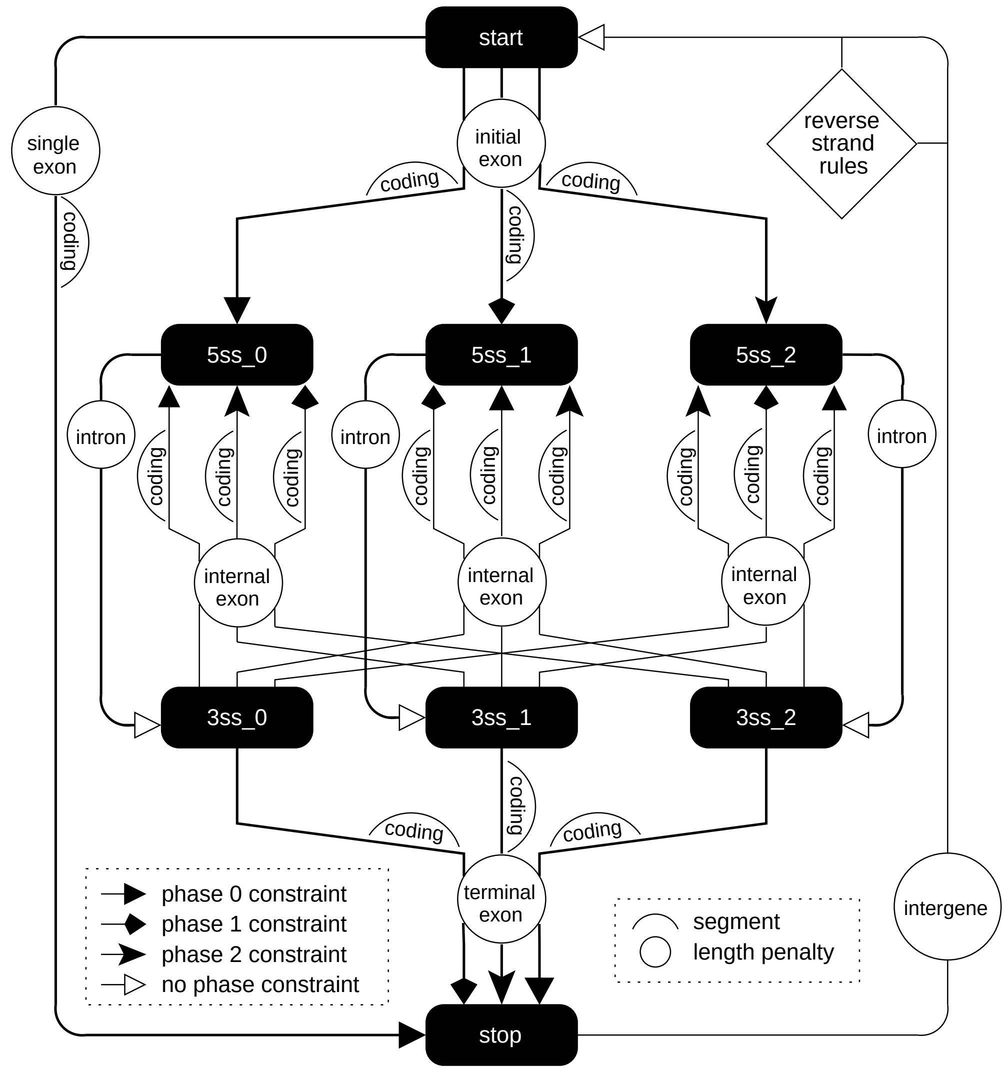


Figure 3.4: A pictorial representation of a GAZE-XML model for multiple genes on both strands. The features are represented by filled boxes, and *source* → *target* rules by different types of arrows, each corresponding to a phase constraint. The labelled circles give the name of the length-penalty function used for each rule, which are themselves defined elsewhere in the configuration file; the labelled humps indicate the segments that contribute to the score for the region implied by the rule. The rules for reverse-strand target features are not shown in their entirety for reasons of clarity, but are reverse complementations of the forward-strand rules. Also omitted are the distance, interruption and DNA constraints, as well as the BEGIN and END features, which mark the ends of the sequence being searched for genes and act as source and target (respectively) to all other features; this accounts for the possibility of a gene structures which extend beyond the end(s) of the sequence.

	Sn	Sp	Av	MG	WG	SG	JG
GAZE_std	0.41	0.24	0.33	0.003	0.368	1.14	1.03
GAZE_std+	0.67	0.36	0.52	0.000	0.387	1.13	1.00
GAZE_std + +	0.71	0.71	0.71	0.000	0.000	1.00	1.00
GAZE_std_gf	0.35	0.35	0.35	0.012	0.076	1.03	1.08
GENEFINDER	0.50	0.44	0.47	0.012	0.104	1.07	1.04

Table 3.1: Gene-level accuracy of GAZE_std plus variants on WormSeq. Sensitivity (Sn), Specificity (Sp), Average (Av), Missing genes (MG), Wrong genes (WG), Split genes (SG) and Joined genes (JG) are the measures described in section 1.4.1

starts and stops of the genes in WormSeq really are. It is straightforward to answer this question in the context of GAZE. I made a GFF file of the confirmed starts and stops of the WormSeq genes, and used the feature selection mechanism of GAZE to force the inclusion of these features in the prediction. The results, referred to as GAZE_std+, are shown in the second row of table 3.1. Several things are notable. Firstly there is a big jump in gene-level sensitivity; 26% more WormSeq genes are identified precisely correctly. Secondly, the figure of 1.00 for Joined genes indicates that no predicted gene extends over the region covered by two or more WormSeq genes. This is expected, because the feature-selection forces the correct splitting of such genes. Thirdly, there is a noticeable increase in 'wrong' genes. It is counter-intuitive that supplying the system with gene starts and ends should lead to more predictions that do not overlap any confirmed gene structure. However, figure 3.5 shows how additional wrong genes can arise from such an approach.

3.4.3 Adjusting the score to refine the predictions

For the purposes of demonstration only, I made a slight modification to the model, increasing the length-independent penalty for intergenic regions from 4.0 to 100.0. This penalty is incurred whenever a gene is introduced; in making it large, the prediction of genes that do not contain any user-selected features is effectively dis-

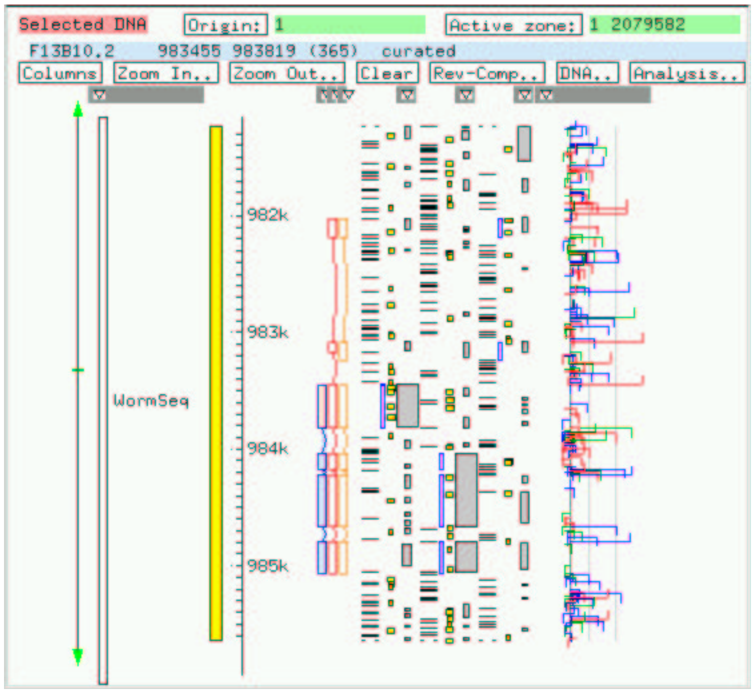


Figure 3.5: How specifying gene start/end information can lead to an increase in the number of Wrong Genes (WG). The GAZE_std prediction (in red) extends into the region 5' of the cDNA-confirmed structure (in blue), but is not classed as "wrong" because it overlaps with the correct structure. Although the GAZE_std+ model (orange) correctly identifies the structure of the gene, the pseudo-signal in the 5' region is strong enough to lead to the prediction of a separate gene which does not overlap any correct gene structure.

allowed. The results of this refinement appear as GAZE_std + + in table 3.1. The MG, WG, SG and JG lines show that the intended increase in specificity has been achieved, and exactly one gene is predicted for each cDNA-confirmed structure.

It will never be the case in practice that the starts and ends of all the gene structures will be known *a priori* for an otherwise unannotated large stretch of genomic DNA. It may be that the starts and ends for *some* of the genes will be known, but in that case, the trick of increasing the length penalty will lead to the missing of gene structures for which they are not known. For this reason, using GAZE in a way such as this is artificial and was largely for demonstration only.

However, some insights can be gleaned from the results. In particular, they show that GAZE_std is 71% accurate in identifying the complete internal intron-exon structure of the genes in this test-set. Furthermore, the feature-selection mechanism has proved and will prove useful for the manual curation of gene structures; it gives the ability to anchor parts of the structure and identify the most likely total structure that is consistent with the anchored points. In this way, it provides an elegant means for curators to make use of incomplete evidence.

3.4.4 A comparison with GENEFINDER

Since GAZE_std integrates the signal, content and length-penalty information from the GENEFINDER program, it is natural to ask how the accuracy of the two compare. The bottom row in table 3.1 shows the gene-level accuracy of GENEFINDER on WormSeq. It is immediately noticeable that GENEFINDER is significantly more accurate than GAZE_std at the precise identification of complete gene structures. It is also more specific, with only 38 wrong genes, compared with a figure of 210 for GAZE_std. The difference in Split genes, 1.07 compared to 1.14 for GAZE_std, is also striking. GAZE_std and GENEFINDER use the same gene prediction signal, content and length-penalty information, so where is this difference coming from?

Inspection of the GENEFINDER source-code reveals some of the answers. Firstly, GENEFINDER subjects candidate exons to a further penalty just before they are assembled into gene structures, and this penalty is different for internal exons ($\log(0.8)$), initial exons ($\log(0.2) + \log(0.5)$) and other non-internal exons ($\log(0.2)$). Since the penalties for non-internal exons are larger, this has the effect of discouraging the splitting of genes. Secondly, GENEFINDER removes all genes scoring less than 7.0, effectively reducing the number of wrong genes.

These subtleties prove a test of the flexibility of GAZE. It turns out to be straightforward to incorporate the additional exon penalties, by simply adding these terms to the penalties for all distances in the appropriate length-penalty functions. The removal of low scoring genes cannot be performed in GAZE itself, but is achieved with a simple Perl post-processing filter.

The resulting model, the results of which are referred to as GAZE_std_gf in table 3.1 represents an attempt to duplicate the output of the GENEFINDER program using GAZE. The table shows GAZE_std_gf to be outperformed by GENEFINDER at the whole-gene level, whereas examination of the comparative accuracy at the base-pair and exon level (see table 3.5) reveals no notable difference. This discrepancy in gene-level accuracy is due not to any differences in the signal and content data used by the two systems, nor differences in the length penalty functions, nor any hidden post-processing, but to the fact that GENEFINDER assembles its exons over a model of gene structure that is designed specifically for prediction in *C.elegans* sequences. In the next section, I show how the GAZE framework allows worm-specific model features to be quickly and effectively introduced into the mode of gene structure without any change to the GAZE source-code itself.

3.5 Towards a *C.elegans*-specific model of gene structure

The GAZE_std_gf configuration is specific to *C. elegans* in that it reads features and segments from the GENEFINDER program, which have been detected using worm-specific signal and content models[2], and also length-penalty functions that have been designed from observation of the distances between such components in real worm genes. However, the *model* of gene structure itself over which gene assembly takes place is specific only to eukaryotes in that it can predict spliced gene structures. There is nothing in the model itself that suits it to the prediction of gene structure in *C.elegans* specifically. The GENEFINDER program on the other hand takes account an unusual splicing mechanism that takes place only in the cells of nematode worms and some other primitive eukaryotes, and it is this that gives it greater accuracy.

[2]More accurately, general models that have been parameterised by observation of confirmed gene features in *C. elegans* sequences.

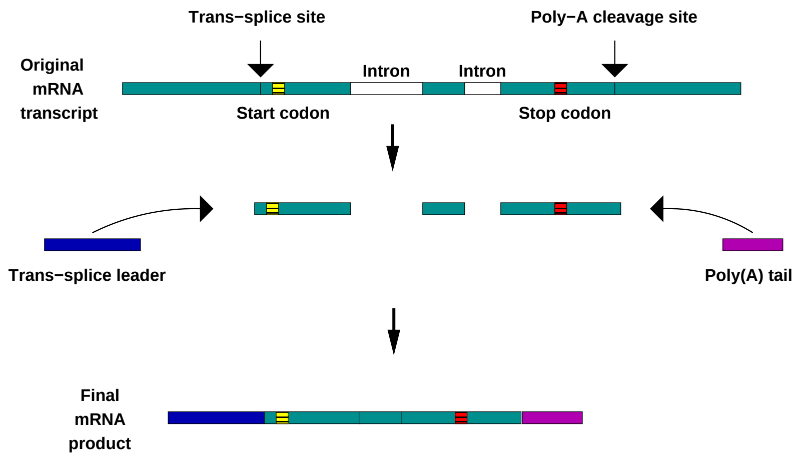


Figure 3.6: Schematic representation of *trans*-splicing in *C.elegans*

3.5.1 Splicing mechanisms in *C.elegans*

In nematode worms such as *C.elegans* and *C.briggsae*, as well as some other primitive eukaryotes such as trypanosomes, a splicing mechanism exists that is unlike the conventional intron-removal mechanism (known as *cis*-splicing). In *trans*-splicing ([66]; review in [13]), the pre-mRNA transcript is cleaved at a site upstream of the translation start site, and the resulting protein-coding fragment is appended to a 21-23 base-pair sequence called the *trans*-splice leader, which itself has been transcribed from elsewhere in the genome. The process is summarised in figure 3.6.

The biochemical process of *trans*-splicing is closely related to that of *cis*-splicing. In fact, it has been shown that the signal for *trans*-splicing to occur is simply the presence of a sequence at the 5' end of the pre-mRNA that looks like an intron but has no functional upstream donor splice site [27]. The *trans*-splice site itself forms the 3' end of this *outron* sequence, and has the same consensus as the splice acceptor involved in intron-removal.

The splice-leader RNAs themselves are always one of two distinct sequences: SL2 leaders are appended to all but the first gene in an operon, and have slight variation in their sequences. The more common SL1 leaders are appended to all other *trans*-spliced gene products, and are all identical in sequence. For the identification of gene structures in worm genomic DNA, the splice-leader sequences cannot be used as a signal for detection, because they do not appear in the genomic sequence in proximity to the gene they are spliced to.

3.5.2 *Trans*-splicing confuses gene prediction programs

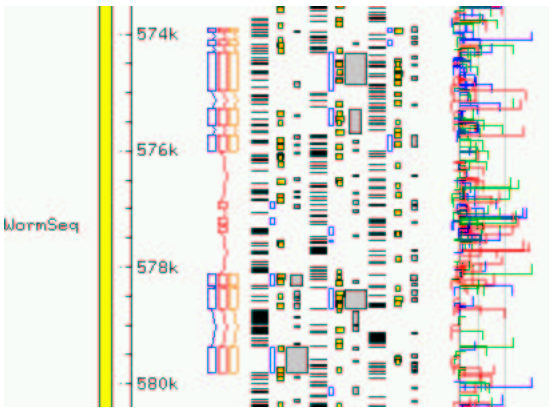
Because the recognition site for *trans*-splicing reaction has the same consensus as an acceptor splice site, gene prediction programs that have not accounted for *trans*-splicing can be confused into mistaking the initial exon of a gene for an internal exon and erroneously extending the prediction upstream. The relatively high gene-density in the worm genome, especially in operons where the genes are typically as close as a thousand bases apart, compounds this problem, often causing a program to mistake two adjacent genes for a single gene. This is apparent in the Joined Genes figure for GAZE_std_gf in table 3.1, which is relatively high compared to the *trans*-splicing aware GENEFINDER. Figure 3.7 illustrates the problem.

3.5.3 A GAZE model accounting for *trans*-splicing

In terms of the scoring function, the problem arises because high-scoring acceptor splice site predictions that are in fact *trans*-splice acceptors can only be included in the gene structure (and thus contribute towards the score) if the 5' end of the prediction is compromised in some way, e.g. by the addition of a low-scoring initial exon upstream. The idea then is to provide a way for the trans-splice acceptor to contribute towards the overall score without having to make this compromise.

Figure 3.8 shows, in spirit, the nature of the changes that are necessary to accommodate *trans*-spliced genes. The first thing to note is that it is not necessary to generate *a priori* predictions of trans-splice acceptor sites; the sequence signal is practically indistinguishable from that displayed by conventional *cis*-splice sites. It is therefore sufficient to direct GAZE to make a candidate *trans*-splice acceptor

(a)



(b)

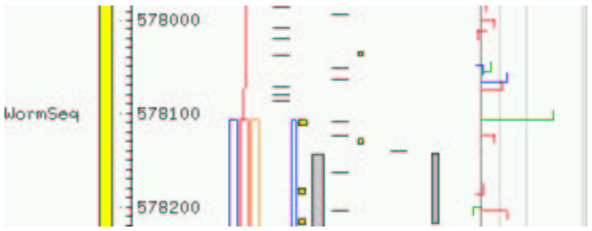
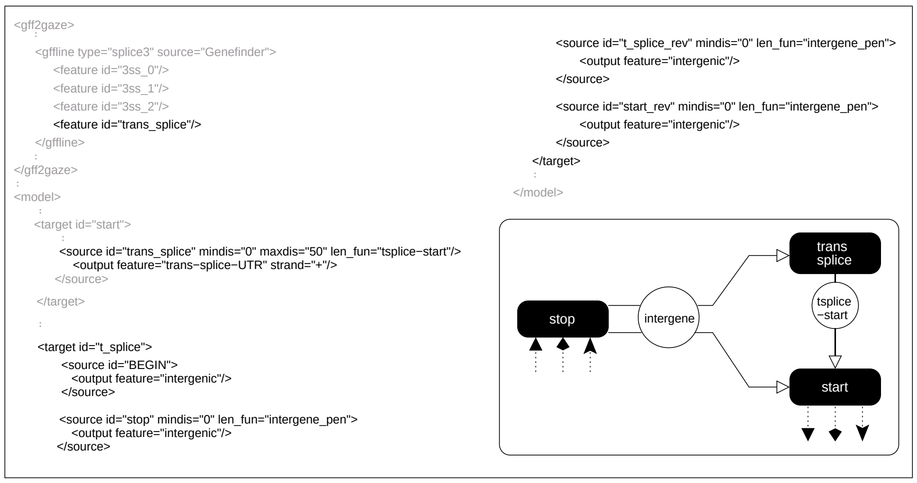


Figure 3.7: (a) Two cDNA-supported gene structures in WormSeq, with WormBase identifiers F11A5.10 and W06D12.3 (blue), and structures predicted by GAZE_std (red) and GAZE_trans (orange). The GAZE_std model, which does not account for *trans*-splicing, has been confused into mistaking the *trans*-splice site for an acceptor splice site, extending the gene-prediction 5', and in this case amalgamating it with the upstream gene; (b) an enlargement of the 5' end of the downstream gene, showing the *trans*-splice site (green hook). The *trans*-splice aware model, GAZE_trans, splits the structures correctly (orange)

feature ("trans_splice") from each predicted *cis*-acceptor encountered in the GFF file.

Secondly, minor modifications are necessary to the gene structure rules to accommodate the new feature; a "start" target feature, representing a start-codon candidate on the forward strand, can now be preceded by the stop-codon of a previous gene as before, or for a *trans*-spliced gene, the *trans*-splice acceptor itself.

The distance from the translation start site to the upstream *trans*-splice acceptor in *trans*-spliced genes is usually small. Data in [13] compiled from a sample of 83 genes experimentally confirmed to be *trans*-spliced, showed that in 43% the *trans*-



```
<gff2gaze>
  :
   <gffline type="splice3" source="Genefinder">
       <feature id="3ss_0"/>
       <feature id="3ss_1"/>
       <feature id="3ss_2"/>
       <feature id="trans_splice"/>
   </gffline>
     :
</gff2gaze>
:
<model>
   :
   <target id="start">
        :
       <source id="trans_splice" mindis="0" maxdis="50" len_fun="tsplice-start"/>
          <output feature="trans-splice-UTR" strand="+"/>
       </source>
   </target>

    :

  <target id="t_splice">
       <source id="BEGIN">
          <output feature="intergenic"/>
       </source>

       <source id="stop" mindis="0" len_fun="intergene_pen">
          <output feature="intergenic"/>
       </source>

       <source id="t_splice_rev" mindis="0" len_fun="intergene_pen">
          <output feature="intergenic"/>
       </source>

       <source id="start_rev" mindis="0" len_fun="intergene_pen">
          <output feature="intergenic"/>
       </source>
  </target>
   :
</model>
```

Figure 3.8: Distillation of the changes required to the forward-strand part of GAZE_std to allow for the possibility of *trans*-sliced genes. The inset shows pictorially how the new "trans_splice" feature fits into the model of gene structure

splice site is within 5 base-pairs of translation initiation, 65% fall within 10 base-pairs, 82% fall within 15 base-pairs, and in only 5% is the distance greater than 30 base-pairs.

The decrease in the likelihood of a *trans*-splice candidate with distance can be modelled naturally with a length penalty function. Again, I looked to GENEFINDER, but this time it was necessary to inspect the source-code for the details of the function used. GENEFINDER approaches the problem by considering all initial exons as beginning not with the translation start candidate, but 50 base-pairs upstream. It adjusts the score of a candidate initial exon beginning at base-pair i by adding a log-probability for the extra 50 base-pairs calculated in the following way:

$$Adj(i) = \max \left[\begin{array}{c} \log(1 - P_{tr}), \\ \log P_{op}P_{tr} + \max_{j=i-50}^{i-1} 3ss(j) + (i - j + 1)\log(1 - P_{op}) \end{array} \right]$$

$3ss(j)$ is the log probability-ratio of an acceptor splice site at position j, or $-\infty$ if the score under the splice acceptor model at j does not exceed the cutoff. The values P_{tr}, P_{op} can be given as parameters to GENEFINDER. P_{tr} represents the prior probability that a gene is *trans*-spliced; P_{op} and $1 - P_{op}$ can be thought of as

	Sn	Sp	Av	MG	WG	SG	JG
GAZE_trans	0.47	0.42	0.44	0.012	0.093	1.07	1.04
GAZE_std_gf	0.35	0.35	0.35	0.012	0.076	1.03	1.09
GAZE_std	0.41	0.24	0.33	0.003	0.368	1.14	1.03
GENEFINDER	0.50	0.44	0.47	0.012	0.104	1.07	1.04

Table 3.2: Comparative Gene-level accuracy of GAZE_trans on WormSeq. Accuracy measures are explained in section 1.4.1

the probability of remaining in and leaving (respectively) the *trans*-splice "state" (viewing the 50 residues as being generated by a Hidden Markov model). The default values for P_{tr} and P_{op} are 0.5 and 0.1 respectively.

It is straightforward to derive a GAZE length-penalty table for the distance between a translation start site and an upstream candidate *trans*-splice site using this function. The only caveat is that the length-penalty table for initial exons already includes the term incurred by non-trans-spliced genes ($\log(1 - P_{tr})$), so this term is *subtracted* from the $tsplice \rightarrow start$ function in order to avoid incurring it twice.

The resulting variant of GAZE_std_gf (with additions to allow for the possibility of *trans*-spliced genes on both forward and reverse strands) is referred to as GAZE_trans. Table 3.2 shows gene-level accuracy for the GAZE_trans model in comparison to the standard models.

The number of genes for which the complete structure has been identified correctly is noticeably higher for GAZE_trans compared with GAZE_std_gf, and is now on-par with that achieved by GENEFINDER. What is also noticeable is the sharp decrease in joined genes, which is exactly what was intended (see figure 3.7). However, this increase seems to have come at a cost with respect to Split genes. In introducing a model innovation that is specifically intended to split gene predictions, we run the risk of introducing erroneous splits, and this is what has happened here. In this case though, the extra splits have apparently occurred in predictions that were also incorrectly predicted by GAZE_std_gf, hence the net increase in gene-level sensitivity. A closer look reveals that 42 structures which are correctly predicted by GAZE_trans were incorrectly predicted by GAZE_std_gf, and only 7 structures correctly predicted by GAZE_std_gf are incorrectly predicted by GAZE_trans.

The GAZE_trans model is, to all intents and purposes, a GAZE implementation of the 980506 version of GENEFINDER. This begs the question of why their accuracies, although comparable, are not identical. The reason for this is the way that GAZE makes use of the GENEFINDER *coding_seg* maximal scoring segments of high protein-coding potential. Inspection of the GENEFINDER source-code reveals that it does not use segments calculated in advance for the whole query sequence (as explained in section 3.2) at all; instead, it obtains a maximal coding score for each candidate *exon*, using the same cumulative-array approach. This difference in exon scoring schemes is only observable when a GAZE "coding_seg" extends beyond either the 5' or 3' end of the candidate exon. In that case, as implied by equation 2.4, the score is scaled by the proportion of the segment that lies in the region. This incorrectly assumes that the score for segment is distributed evenly along its length. For that reason, the GENEFINDER method for computing coding scores is more accurate than the approximate method used by GAZE. However, as shown by the results so far, it seems to have little impact on the overall accuracy; in fact, at the exon level (as shown in table 3.5), GAZE_trans, although marginally less sensitive than GENEFINDER, is slightly more specific.

3.6 Integrating similarity information

As explained in chapter 1, the effective use of similarity information can improve gene prediction accuracy. In this section, I outline the changes required to the GAZE_trans configuration to make use of the similarity information in the form of EST alignments.

3.6.1 ESTs and gene prediction

An Expressed Sequence Tag (EST) is conceptually a sequence read of a cDNA copy of an expressed cellular mRNA. ESTs differ from the full-length cDNAs deposited in the nucleotide databases (such as those used to build the WormSeq dataset) in that: (1) they are often of lower quality, and (2) they represent the sequencing of only a subsequence of the original cDNA, typically around 300-500 base-pairs, read from either the 5' or 3' end of the transcript.

In much the same way as the EMBL full-length cDNAs were used to confirm the genes in WormSeq, the alignment of ESTs to genomic sequence can provide evidence for gene structures. Since ESTs represent only a draft-quality read of a subsequence of a transcript, any single EST will normally only provide evidence for *part* of a gene structure.

The utility of EST data

The alignment of ESTs back to the genomic sequence can be extremely useful for gene prediction in several ways. Firstly, they act as an aid to novel gene discovery, highlighting regions of genomic sequence where genes were not thought to exist before. Secondly, they can provide evidence for the spliced structure of the gene, at least in cases where the EST extends across an exon-exon boundary in the corresponding spliced mRNA. Thirdly, ESTs can help elucidate the structures of genes that give rise to several alternatively spliced transcripts. Finally, they are useful for identifying the extremities of genes. Ideally, the alignment of a 5' EST to the genome identifies the 5' end of a gene, and likewise with 3' ESTs. Many ESTs even exist as pairs, corresponding to 5' and 3' reads of the *same* cDNA, and a pair of such alignments ideally identifies both the 5' and 3' end of a gene, if not the complete internal intron-exon structure.

Problems with EST data

If the data were perfectly reliable, each EST would provide perfect, unquestionable evidence for either the 5' or 3' end of a gene, and in addition perhaps part of the intron-exon structure. Unfortunately, ESTs are naturally error-prone, due to their high-throughput, single-read nature. This can lead to errors when they are aligned. In addition, there are other problems associated with EST data:

Sample bias The pool of available ESTs is unavoidably biased towards genes that are highly and ubiquitously expressed. EST databases are therefore not generally a useful resource for either the discovery or the elucidation of the structures of genes expressed at at low levels or under very specific conditions.

Pseudo poly-A sites The EST sequencing process begins by the reverse transcription of the mRNA into a double-stranded complementary DNA (cDNA), primed from the poly-A tail at the 3' end of the transcript. If the transcript by chance contains a string of A residues somewhere other than the 3' end, then reverse transcription could begin from this place, resulting in a cDNA for which part of the 3' end of the gene is missing. When aligned back to the genome, the 3' EST match ends somewhere upstream of the true 3' end of the gene, sometimes in the protein-coding portion.

5' end incompleteness It is often the case that the 5' end of the cDNA is missing, either because the mRNA it was synthesised from had been partially digested at the 5' end, or because the reverse-transcription reaction did not carry through to completion. When a 5' EST is aligned back to the genome therefore, the match can begin somewhere downstream of the true 5' end of the gene, often in the protein-coding portion.

Ambiguous matching The low-fidelity of EST sequences means that mismatches must be allowed when aligning them to genomic sequence. A common implication of this is that the EST will align to multiple places in the genomic sequence, and it is a not always obvious to identify the "correct" alignment.

Annotation errors It is not uncommon for an EST sequence to be deposited in a nucleotide database with the incorrect assignment of orientation, e.g. a 5' EST being annotated as a 3' EST. This can cause confusion when inferring information from an EST alignment such as the strand to which it matches.

The source of *C.elegans* EST data

The WS52 release of WormBase contained around 100,000 *C.elegans* EST sequences. As part of the Sanger Institute worm sequence curation process, each of these ESTs is isolated to a small number of localised regions in the genome by BLASTN [2], and then accurately aligned using the spliced-local-alignment program EST_GENOME [79]. It is important to note that some ESTs are aligned to several places in the genome using this procedure, and others not at all. For those that are aligned, the result is a set of "exons" each of which can be described by a 5-tuple: (EST-identifier, EST-start, EST-end, genome-start, genome-end).

For WormSeq, these exons were re-mapped into (EST-identifier, EST-start, EST-end, WormSeq-start, WormSeq-end) 5-tuples, discarding those falling outside the regions extracted from the genome to form artificial sequence, and truncating those with partial overlap to these regions where necessary. As a result, 261 of the 325 gene loci in WormSeq have at least one EST match.

3.6.2 A GAZE model for the use of EST alignments

The natural approach is to use the EST match exons as segments providing evidence for protein-coding regions. These segments will be expected overlap with the untranslated regions of genes however, so such a method would lead to the over-prediction of the coding portions of the genes. I therefore extend the model of gene structure to include the untranslated regions at the ends of genes, incorporating features for the beginning and end of *transcription*: "transcript_start" and "transcript_stop".

With this modification, EST match segments can now be used as evidence for protein-coding regions and untranslated regions. However, such a model is not exploiting the power of EST alignments in their identification of intron-exon boundaries and gene extents. I therefore devised a general EST pre-processing strategy that would make the gene structure information encoded in the alignments more readily available to be used by GAZE. The pre-processing is performed according to the following schedule:

- From the pool of EST 'exons', construct a set of EST 'transcripts', lists of the exons from a single EST ordered by their location on the genome. Transcripts with an average match identity to the genome of less than 95% are discarded at this stage.
- For each transcript, generate:
  - "EST_match" segments for the exons;
  - "EST_intron" segments for the regions between transcript exons that are adjacent in the EST but separated in the genome sequence;
  - if EST is a 5' read, a "transcript_start" feature for the beginning of the transcript;
  - if EST is 3' read, a "transcript_stop" feature for the end of the transcript.
- For those cDNAs for which there is exactly one 5' and one 3' transcript, generate an "EST_span" segment for the region between the start of the 5' transcript and the end of the 3' transcript.

Although the matches produced by the EST_GENOME program score each exon according to its percentage identity to the genome, the model was found to be ineffective when these scores were used directly (data not shown). In addition, derived features and segments do not have an associated score. I therefore used the following scoring scheme for the EST-derived features and segments:

EST_match These were given the score $\frac{(identity-95).length}{100}$. The rationale for this is to ascribe more confidence to long, contiguous exons with high identity than to short exons with high identity. The latter are an artifact of the EST alignment process, where single base-pair deletions in the EST with respect to the genome cause two shorter high-identity exons to be created. Also, since it is expected that the number of "EST_match" segments falling in a given region will be highly variable (from 0 to hundreds), the "projected per-base" segment scoring approach (equation 2.5) is most appropriate.

EST_intron These were scored according the average identity of the flanking exons. A scaling factor of 0.05 was applied to bring the order of the scores into line with other segments being used. Since these segments are expected to match candidate intron regions precisely, their use is qualified with a match constraint[3].

EST_span These segments correspond to regions containing exactly one complete gene. However, due to the problems with EST data explained earlier it will often be the case that an "EST_span" segment covers only part of a gene. The segments are therefore interpreted as regions that should contain *no more* than one gene. The way that this is realised in GAZE is to use the segments as supporting evidence for intergenic regions, but to give them very high negative scores, in this case -10000. This penalises the prediction of intergenic regions where EST_spans lie, effectively preventing gene splitting. This is pertinent as the Split Genes figure for GAZE_trans was quite high (see table 3.2).

transcript_start and **transcript_stop** features were assigned the neutral log probability ratio of 0.

Figure 3.9 shows pictorially a GAZE configuration for incorporating these EST-derived features and segments, over a model of gene structure that now accounts for

[3]Recall from chapter 2 that match constraint stipulates that the a segment should only contribute towards the score if its extent matches the region being considered precisely.

	Sn	Sp	Av	MG	WG	SG	JG
GAZE_EST	0.58	0.53	0.56	0.009	0.088	1.02	1.03
GAZE_trans	0.47	0.42	0.44	0.012	0.093	1.07	1.04
GENEFINDER	0.50	0.44	0.47	0.012	0.104	1.07	1.04

Table 3.3: Comparative gene-level accuracy of GAZE_EST in comparison to GAZE_trans and GENEFINDER. Accuracy measures are defined in section 1.4.1.

the untranslated regions at the ends of genes (as well as *trans*-splicing). This model is referred to as GAZE_EST.

Table 3.3 shows the gene-level accuracy of the GAZE_EST model in comparison to the GAZE_trans model and GENEFINDER. GAZE_EST displays an improvement over all models presented so far, as well as GENEFINDER.

In describing the GAZE_trans model, I noted that some incorrect splitting of gene structures was inevitable but showed that the *net* gain in accuracy achieved by this innovation was significant. In introducing EST evidence, there is no obvious reason why the accuracy of some predictions might become worse. However the net gain of 37 additional correct structures identified by GAZE_EST over GAZE_trans consists of 39 that GAZE_EST correctly identifies whilst GAZE_trans did not, and 2 that GAZE_trans correctly identified whilst GAZE_EST does not. There are also 2 additional examples of a GAZE_EST prediction having fewer correct exons than the corresponding GAZE_trans predicted structure. Closer examination of these 4 examples reveals a variety of reasons for the decrease in accuracy:

Introns in UTR. In two of the cases, the EST evidence supports the presence of an intron in the 5' untranslated region of the gene. Since the GAZE_EST model only allows for introns in the coding portion of the gene, the prediction of this coding portion has been extended in the 5' to accommodate the intron.

Overlapping transcription units. In one case, the final coding exons of a gene on the forward strand overlapped with the 3' UTR of a gene on the reverse strand,

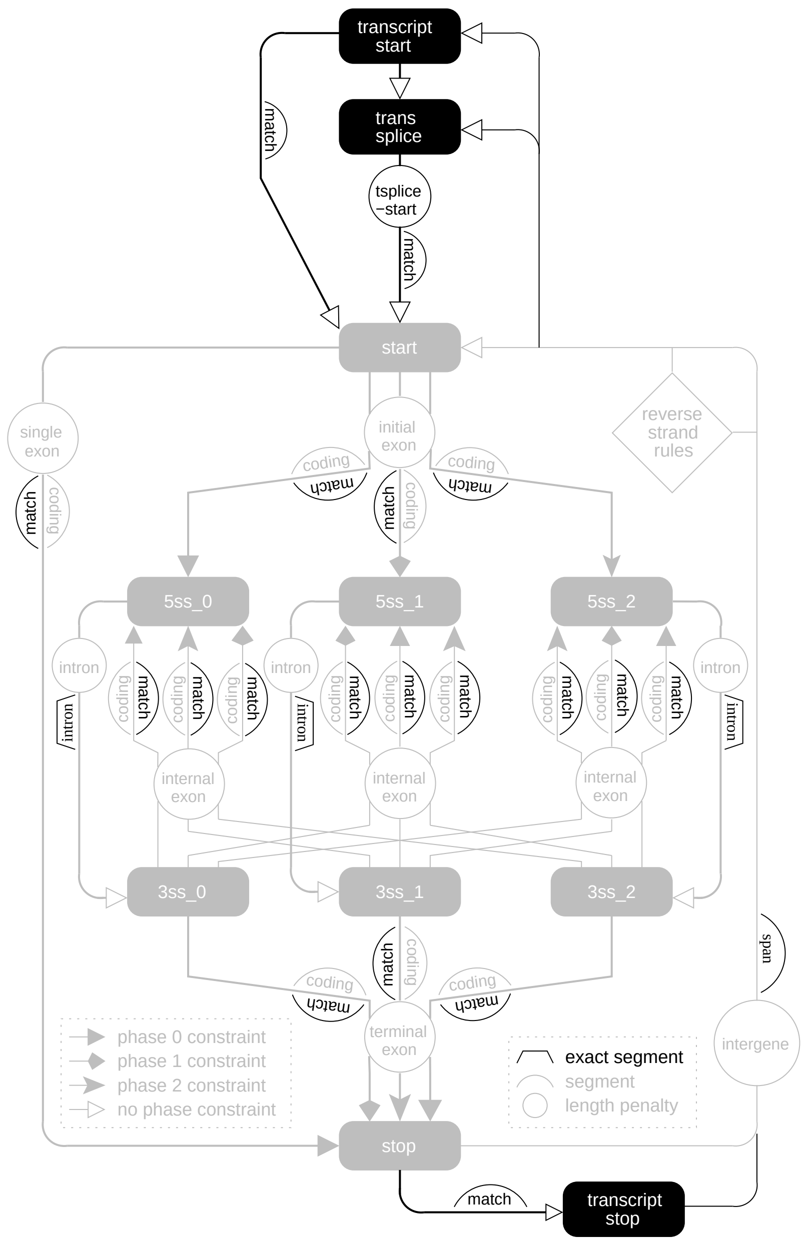


Figure 3.9: The GAZE_EST model, allowing for *trans*-spliced genes and untranslated regions. It is a simple extension of the GAZE_std model (figure 3.4), which is shown in pale-shade for reference. The "match", "intron" and "span" segments shown are the "EST_match", "EST_intron" and "EST_spans" segments referred to in the text.

as supported by an EST alignment. GAZE, at least as I have presented it so far, classifies each base in the sequence as belonging to exactly one function class. The region here was classified as a 3' UTR on the reverse strand, causing the final coding exon of the forward strand gene to be missed.

Incorrect alignment In the final case, a cluster of EST alignments in the intron of the gene caused GAZE_EST to split the structure incorrectly. This region also had a partial match to a nucleotide database cDNA that matched with a higher score elsewhere in the genome, suggesting the possibility of a pseudogene.

Such problems were not confined to these four examples; in other cases though, the prediction was unaffected. In the design of the scoring scheme I was attempting to achieving a balance between the gains to be had by treating the EST exons as strong evidence for genic regions, and the losses incurred by treating them as 'the truth.' The fact that only a small number of prediction were affected by EST-confusion supports the validity of the scheme.

3.7 A closer look at the accuracy of GAZE

This section, provided for completeness, examines various aspects of the accuracy of the GAZE models presented earlier. Thus far, results have been presented at the gene level only for purposes of illustration.

The performance of the GAZE models has so far been assessed in comparison to the GENEFINDER program which, indirectly at least, works with the same signal, content and length-penalty models. As the basis for a more objective evaluation, I therefore also include in the following results for FGENESH [96], an HMM-based gene prediction program that works in a similar manner to the more widely used GENSCAN [21]. Unlike the latter, however, FGENESH comes with a parameter file for prediction in *C.elegans* sequences specifically, as well as a "-nematode" command-line directive.

	Sn	Sp	Av	MG	WG	SG	JG
GAZE_EST	0.59	0.53	0.57	0.009	0.088	1.02	1.03
GAZE_trans	0.47	0.42	0.44	0.012	0.093	1.07	1.04
GAZE_std_gf	0.35	0.35	0.35	0.012	0.076	1.03	1.09
GAZE_std	0.41	0.24	0.33	0.003	0.368	1.14	1.03
GENEFINDER	0.50	0.44	0.47	0.012	0.104	1.07	1.04
FGENESH	0.51	0.42	0.47	0.006	0.144	1.08	1.03
FGENESH_NO-W	0.18	0.16	0.17	0.012	0.125	1.07	1.09

Table 3.4: Comparative gene-level accuracy of various programs on WormSeq. Accuracy measures are explained in section 1.4.1

3.7.1 Gene-level accuracy

Table 3.4 collates the gene-level accuracy results for all GAZE models presented earlier (with the exception of GAZE_std+ and GAZE_std + +, which were for illustrative purposes only), as well as GENEFINDER and FGENESH.

The table shows that the accuracy of FGENESH is comparable with that obtained by GENEFINDER and GAZE_trans, although the former is slightly more sensitive and less specific. Reassuringly, the GAZE model making use of EST evidence performs best of all.

The fact that the GENEFINDER and GAZE_trans take account of *trans*-splicing begs the question of whether the comparable accuracy of FGENESH is obtained by it too incorporating a worm-specific model of gene structure. Since FGENESH is known to perform well on the sequences of a variety of organisms, it is natural to assume that it is the parameter files that give it organism specificity. Examination of an FGENESH parameter file reveals elements for signal, content and length distribution models, but nothing for the model of gene structure itself. It is informative therefore to run the program against WormSeq with the *C.elegans* parameter file but without specifying the "-nematode" command-line option (referred to in table 3.4 as FGENESH_NO-W). A marked decrease in accuracy is observed, suggesting that the option

	Base accuracy			Exon accuracy				
	Sn	Sp	CC	Sn	Sp	Av	ME	WE
GAZE_EST	0.99	0.93	0.94	0.90	0.84	0.87	0.02	0.09
GAZE_trans	0.98	0.91	0.93	0.86	0.80	0.83	0.03	0.11
GAZE_std_gf	0.98	0.90	0.92	0.84	0.77	0.80	0.04	0.12
GAZE_std	0.99	0.84	0.88	0.85	0.67	0.76	0.03	0.24
GENEFINDER	0.98	0.90	0.92	0.87	0.78	0.83	0.03	0.13
FGENESH	0.98	0.91	0.92	0.88	0.80	0.84	0.03	0.13

Table 3.5: Comparative base-pair-level and exon-level accuracy of GAZE_std on WormSeq. The accuracy measures are explained in section 1.4.1

activates a *C.elegans*-specific strategy, either a model of gene structure (as here), or something else such as a different evidence weighting scheme. Without access to the source-code, it is impossible to tell what exactly this is.

3.7.2 Accuracy at base-pair and exon-level

Table 3.5 shows the accuracy of the all models at the base-pair and exon-level.

The results are largely consistent with those at the gene-level, but some interesting elements are evident. Firstly, all programs show strikingly similar accuracy at the base-pair level, making it in this context at least not very useful as an accuracy measure. Secondly, although FGENESH was more sensitive and less specific than GENEFINDER at the gene-level, at the exon level it is both slightly more sensitive and specific. This suggests that the incorrect genes predicted by FGENESH contain small numbers of exons.

3.7.3 Accuracy by exon-type

Table 3.6 shows exon-level accuracy for each of initial, internal and terminal exons, as well as single-exon genes (termed “single”).

The greater accuracy of all programs in the identification of internal exons sup-

		Sn	Sp	Av	ME	WE
Initial (309)	GAZE_EST	0.79	0.74	0.77	0.06	0.16
	GAZE_trans	0.72	0.67	0.70	0.11	0.19
	GAZE_std_gf	0.57	0.56	0.57	0.13	0.30
	GAZE_std	0.64	0.43	0.54	0.10	0.46
	GENEFINDER	0.72	0.66	0.69	0.11	0.22
	FGENESH	0.75	0.61	0.68	0.11	0.24
Internal (1620)	GAZE_EST	0.92	0.86	0.89	0.01	0.06
	GAZE_trans	0.89	0.83	0.86	0.01	0.08
	GAZE_std_gf	0.90	0.80	0.85	0.01	0.09
	GAZE_std	0.90	0.80	0.85	0.01	0.11
	GENEFINDER	0.92	0.82	0.87	0.01	0.10
	FGENESH	0.92	0.87	0.90	0.01	0.07
Terminal (309)	GAZE_EST	0.85	0.80	0.83	0.04	0.16
	GAZE_trans	0.81	0.74	0.78	0.06	0.18
	GAZE_std_gf	0.78	0.78	0.78	0.06	0.15
	GAZE_std	0.82	0.55	0.69	0.04	0.37
	GENEFINDER	0.80	0.72	0.76	0.07	0.21
	FGENESH	0.84	0.68	0.71	0.06	0.26
Single (16)	GAZE_EST	0.94	0.73	0.84	0.00	0.22
	GAZE_trans	0.94	0.59	0.77	0.00	0.26
	GAZE_std_gf	0.94	0.71	0.83	0.00	0.24
	GAZE_std	0.88	0.13	0.51	0.00	0.85
	GENEFINDER	1.00	0.63	0.81	0.00	0.29
	FGENESH	0.63	0.77	0.70	0.00	0.23

Table 3.6: Exon level accuracy on WormSeq, by exon type. The number exons of each type in WormSeq is shown in parentheses. Accuracy measures are explained in section 1.4.1

ports the observation that the ends of genes are more difficult to identify than the internal exon-intron structure. This is where EST evidence helps, and the table shows that the win in overall accuracy of GAZE_EST over all other programs is due largely to its better identification of initial and terminal exons; indeed, FGENESH has the slight edge for internal exons. The other notable aspect of these results is the difference between GENEFINDER and FGENESH in their identification of single-exon genes. GENEFINDER identifies all of them correctly at the expense of predicting many exons as single when in fact they belong as part of a multi-exon structure. FGENESH is apparently more conservative in predicting single-exon genes.

3.7.4 Genome scale accuracy

The precise identification of complete gene structures can depend on the genomic context of the genes; that is, their relationships to each other with respect to distance and orientation. It might be argued that extracting the genes from their genomic context, as was done in the construction of WormSeq, provides an artificial problem for gene prediction programs.

In construction of the WormSeq dataset, I have tried to provide as far as possible a realistic context for the cDNA-confirmed genes, by not inserting them into a randomly generated intergenic landscape as has been done by others [53], but by extracting the surrounding intergenic DNA with the genes. The technique of taking half of the region to the next curated gene in each upstream and downstream direction was an attempt to ensure that the distances between the genes was also realistic. However, it remains the case that certain aspects of gene organisation, such as operon structure, are disrupted by extracting the genes from their genomic context.

To address such concerns, I applied all of the models (as well as GENEFINDER and FGENESH) to the WormBase_Sanger DNA sequence, from which the WormSeq genes were extracted. This DNA amounted to 48,722,743 nucleotides, arranged in 9 contiguous sequences ranging in length from around 1 million to 12 million

	Base accuracy		Exon accuracy			Gene accuracy			
	Sn	Pred	Sn	Pred	ME	Sn	Pred	MG	SG
GAZE_EST	0.99	12.61M	0.90	60559	0.02	0.57	9075	0.012	1.02
GAZE_trans	0.98	12.76M	0.86	60860	0.03	0.80	9393	0.009	1.07
GAZE_std_gf	0.98	12.81M	0.85	61487	0.03	0.39	8645	0.009	1.03
GAZE_std	0.99	14.05M	0.86	71545	0.02	0.44	14685	0.003	1.14
GENEFINDER	0.98	13.00M	0.88	63560	0.03	0.51	9584	0.009	1.07
FGENESH	0.98	12.91M	0.88	62668	0.03	0.50	10707	0.004	1.07

Table 3.7: Comparative accuracy on WormBase_Sanger. Accuracy measures are defined in 1.4.1. The number of predicted residues, exons and genes (Pred) are quoted in lieu of specificity measures, which are not defined for a genome scale analysis

nucleotides. On a standard Compaq DS10 workstation, GAZE, GENEFINDER and FGENESH are happy dealing with gene-prediction data from sequences of a million base pairs or more, but for 12 million bases, all programs require memory resources that are beyond the limits of such a machine. For GAZE, the GENOME_GAZE script introduced in chapter 2 makes the analysis of such large sequences straightforward, without any requirement to split the DNA into several files. No such luxury existed for GENEFINDER and FGENESH, so it was therefore necessary to split the DNA into 1.1 Megabase chunks, with 0.1 Megabase overlap between each chunk. The predictions in the overlapping regions for these two programs were resolved by inspection. The results of applying all programs to what amounts to half of the *C.elegans* genome are depicted in table 3.7.

It is interesting that GENEFINDER seems to have a slight edge over FGENESH here, both in sensitivity and specificity. The reverse was true for WormSeq. Overall though, these results are consistent with those presented for WormSeq, suggesting that the test sequence is indeed a good approximation of a real genomic contig.

3.8 Examining the probabilistic aspects of GAZE

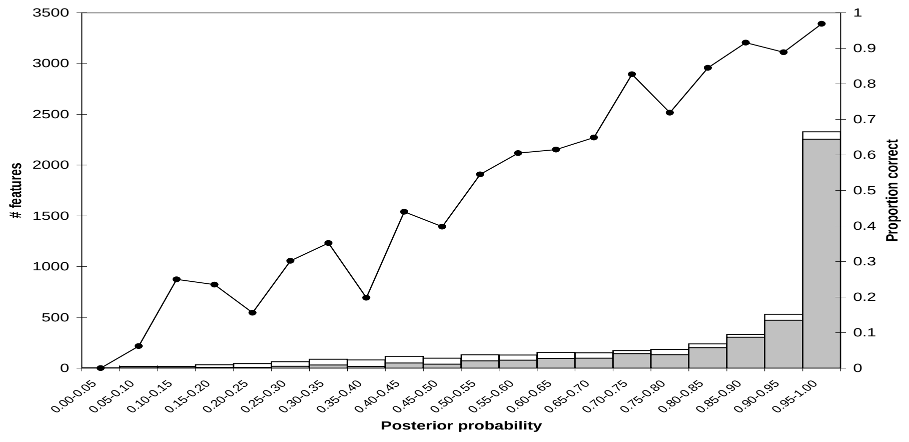
One of the novelties of GAZE with respect to other "exon assembly" based gene prediction systems is the ability to interpret the predictions in a probabilistic manner. As explained in chapter 2, the definition of a probability distribution over all possible gene structures (given a gene structure model), allows the calculation of *posterior* probabilities for individual gene components. In this section, I show how this facility can be used to reason about the reliability of predictions made by GAZE. I also discuss how posterior probabilities can act as an aid for manual curation of gene structures, in particular beginning to address the difficult problem of identifying alternatively spliced genes in *C.elegans*.

GAZE can be instructed to report posterior probabilities for (a) the *features* comprising the highest-scoring gene structure; (b) all candidate features; (c) the *regions* defined by adjacent features in the highest-scoring gene structure; and (d) all candidate regions of specified types. In the analysis presented here, I make use of the first two of these facilities, largely because predicted and candidate features can be designated as correct (with respect to the cDNA-confirmed gene structure) or incorrect. Predicted and candidate *regions* also carry the possibility of being *partially* correct, which adds an unnecessary complication to the analysis.

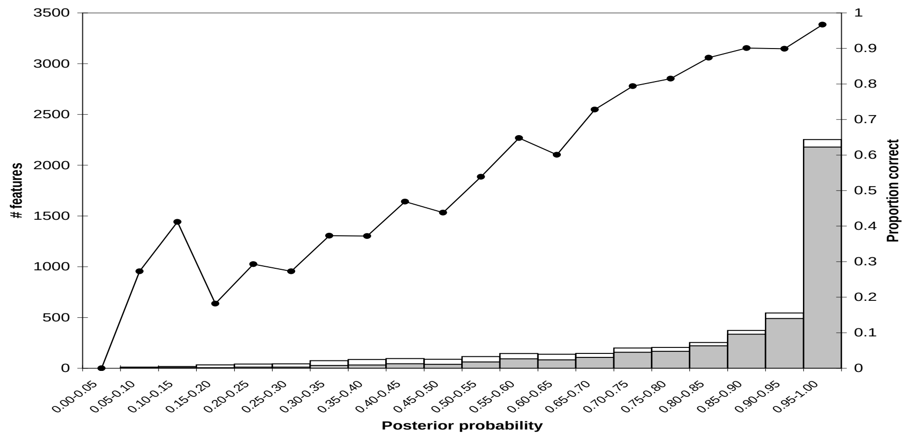
3.8.1 The reliability of GAZE predictions

GAZE reports a posterior probability for each feature that it identifies as belonging to the optimal gene structure. Since these are intended to be interpreted as degrees of belief in the correctness of the features, it is worth investigating how well they perform as indicators of reliability. For example, does a reported posterior probability of 0.5 for a feature really mean that we can be "50 percent sure" that the feature is correct? Figure 3.10 shows the posterior probabilities for the features comprising GAZE-predicted genes. Plotted for a variety of probability intervals are the number of features belonging to GAZE-predicted genes with a posterior in that interval, and the proportion of them that are correct.

(a)



(b)



(c)

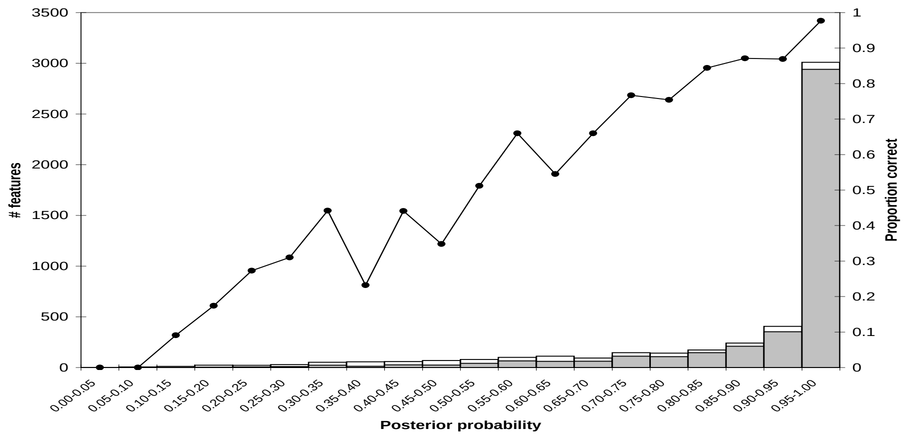


Figure 3.10: Posterior feature probabilities for GAZE models. Shown for features part of the coding-portions of predicted gene structures in each case are the number of features with a posterior probability in each interval (bars), the number of those predicted features that were actually correct (shaded portions of bars), and this number as a proportion of the predicted features in the interval (line). (a) GAZE_std_gf (4916 features); (b) GAZE_trans (4866 features); (c) the GAZE_EST (4832 features).

Two points are evident from the plots. Firstly, as the sophistication of the model increases, so does the number of features with high posterior probability. This implies that improving the model not only increases the accuracy of GAZE predictions, but also generally improves the confidence that it assigns to predictions. This particularly makes sense with the GAZE_EST model, where the EST evidence would be expected to add weight to many features belonging to gene structures predicted by the less sophisticated models. Secondly, the lines plotting the proportion of predicted features that are correct in each posterior probability range are close to the ideal, 1:1 line. This shows that the posterior probabilities are accurate indicators of reliability.

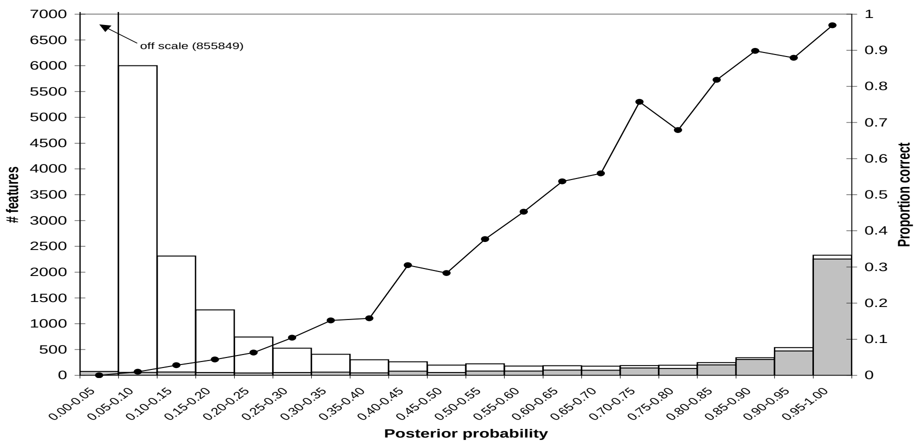
3.8.2 Feature probabilities can aid manual curation

The posterior probabilities reported by GAZE can also provide an aid for the *manual* curation of gene structures, which involves selecting from large numbers of candidate gene features, those that imply gene structures that are most consistent with the evidence. Figure 3.11 shows posterior probabilities for all of the candidate features presented to the GAZE models. Again, the proportion of features within each interval that are correct is consistent with the posterior probability computed by GAZE, suggesting that they are good indicators of reliability.

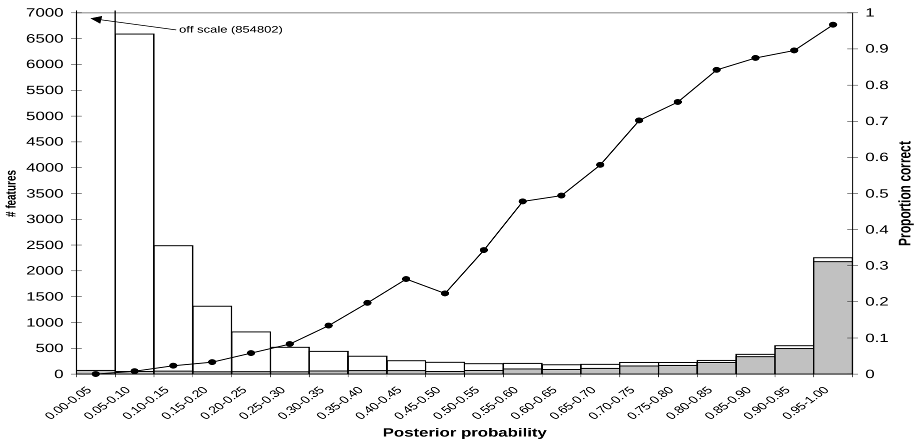
A striking feature of figure 3.11 is the number of features with low posterior probability. Because features with low or zero posterior probabilities do not fit into sensible gene structures, such features can be ignored by a human annotator, reducing the number of possible assemblies and therefore the likelihood of mistakes. Of the 872482 candidates for features that comprise the coding part of WormSeq genes (i.e. the starts, stops and splice sites), 587021 (67 percent) have zero probability (to 4 decimal places) according to GAZE_std_gf model, 576517 (66 percent) according to GAZE_trans, and 632067 (72 percent) according to GAZE_EST.

It is necessary to assess the likelihood that, by discarding features with zero probability, we discard features that are in fact correct. The only example of this

(a)



(b)



(c)

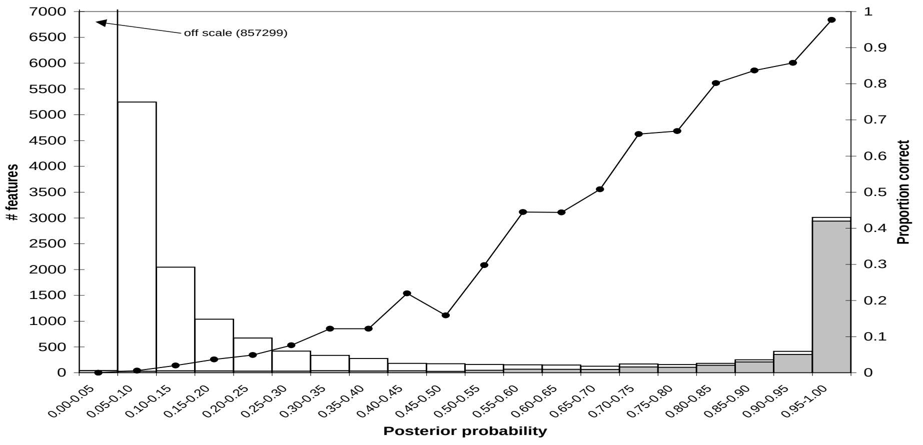


Figure 3.11: Posterior feature probabilities for the three GAZE models, for all candidate features; (a) GAZE_std_gf; (b) GAZE_trans; (c) GAZE_EST

occurring in WormSeq is the F09E8.3 gene, for which the high-scoring splice acceptor at the 5' end of the third exon is given zero probability by all three GAZE models. The problem here is that the donor splice at the 3' end of the exon is not detected by GENEFINDER at the default cutoff (it scores marginally below). In this case, it therefore becomes impossible to incorporate the acceptor splice feature into any gene structure with measurable probability. When the missing donor supplied as an external feature (a trivial task with GAZE), not only does the acceptor feature now have a very high posterior probability under all models (0.9993), but the low-scoring donor itself has a posterior probability of 1.0, underlining its vital importance in the gene structure. This particular example demonstrates well the utility to be gained from the posterior probabilities and the care that should be taken when interpreting them.

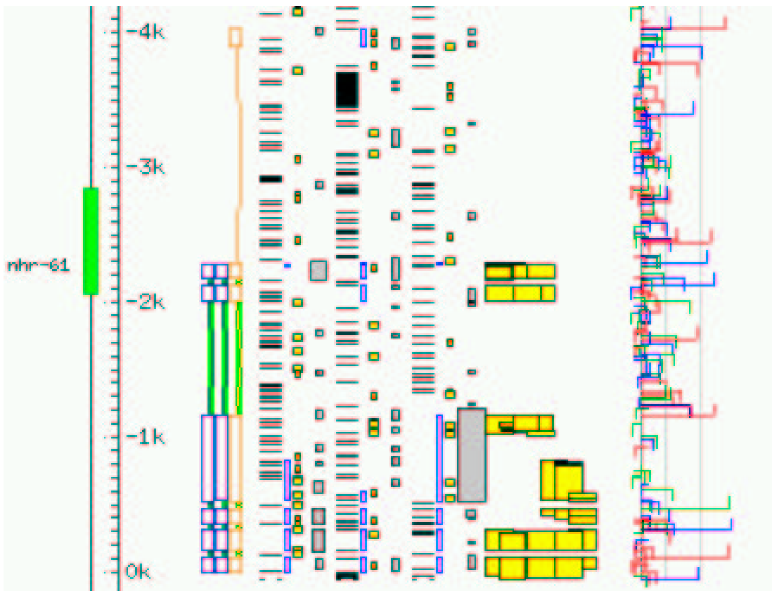
3.8.3 Feature probabilities could be used to identify alternative splicing events

Like most existing gene prediction programs, GAZE does not explicitly address the problem of identifying all of the variant gene structures for genes that are alternatively spliced, and this is still very much an open problem (see section 1.5). Burge [20] gave an example of how the sub-optimal exons reported by the GENSCAN program can sometimes be correct in alternative splice forms of the gene. GAZE offers the possibility of identifying not only sub-optimal exons, but also introns and other types of region, as well as features themselves. I show here how the *feature* posterior probabilities can begin to be used in the prediction of alternatively spliced genes.

Figure 3.12 depicts the gene structure of the nhr-61 locus in *C.elegans*, with its two alternatively spliced isoforms. The structure of this gene is has been confirmed by the alignment of full-length cDNAs for each isoform to the genome. As shown by the figure, the initial exon of neither isoform is identified precisely by any of the GAZE models presented (the one predicted by GAZE_EST is shown in the figure).

Upon examination of the posterior probabilities, we find first that of the fea-

(a)



(b)

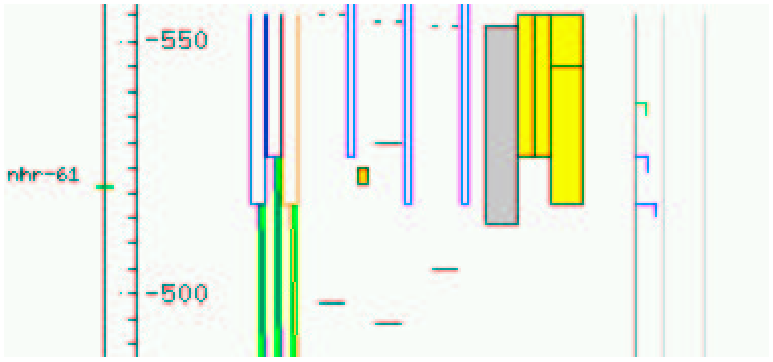


Figure 3.12: The *C.elegans* nhr-61 gene locus (WormBase WS52 identifier W01D2.2), with its two alternatively spliced isoforms (blue), and the GAZE_EST predicted structure (orange). (a) GAZE_EST fails to correctly identify the initial exon of either isoform; (b) An enlargement of the 3' end of the third exon of the correct gene structures, showing alternative splice donors, both supported by alignments of ESTs to the genomic sequence by EST_GENOME (yellow). Although only one of the two alternative donor features belongs to the correct gene structure, the posterior feature probabilities reported by GAZE provide evidence for both, as explained in the text.

tures predicted by GAZE that are not part of either correct gene structure, none of them have strikingly high probability; 0.432, 0.431 and 0.705 for the incorrect start, donor and acceptor. The probability reported for the *correct* start (not predicted by GAZE) is 0.269, not insignificant, suggesting that it is a viable alternative. Secondly, of the GAZE-predicted features part of both correct gene structures, all of them have probability of 0.999 or greater. Taken together, these two observations support what the earlier graphs showed, that the posterior probabilities are good indicators of reliability; GAZE has reported a higher degree of confidence about the parts of its prediction that turn out to be correct. Where the two isoforms differ, in their choice of donor splice site at the 3' end of the third exon, GAZE is less confident; 0.751 for the donor that is part of the GAZE prediction. Upon inspection of the posterior probabilities for all candidate features in this region, it becomes apparent that the "missing" probability is found in the alternative donor (0.249).

Although further work is required to bring together these ideas and observations into an automated system, this example (another was presented in [59]) demonstrates well how the GAZE feature posterior probabilities can point towards firstly elements of predicted gene structures that may not be correct, and secondly elements not part of predicted gene structures that might be correct, either in the single correct structure for the gene, or in one of the several possible structures of alternatively spliced genes.