Inhaltsverzeichnis

Abkürzungsverzeichnis

GFF	General Feature Format
GHMM	Generalisiertes-Hidden-Markow-Modell
GTF	Gene Transfer Format
HSMM	Hidden-Semi-Markow-Modell
NGS	Next Generation Sequencing
TAIR	The Arabidopsis Information Resource

1. Einleitung

Dank der Fortschritte in der DNA-Sequenzierung können mittlerweile selbst ganze Genome relativ schnell sequenziert werden[4]. Damit die Genvorhersage Schritt halten kann, bedarf es schnellerer und automatisierter Methoden[15], da konventionelle, überwachte Trainingstechniken die Validierung durch Experten benötigen[1]. Dies erfordert jedoch nicht nur mehr Zeit, als die Genomsequenzierung, außerdem sind durch das Treffen einer Vorauswahl eventuell Gene eines bestimmten Typs überrepräsentiert[1]. Das kann jedoch sowohl bei überwachten als auch bei unüberwachten Trainingsmethoden auftreten[1]. Der Vorteil von unüberwachten Trainingsalgorithmen liegt aber nicht nur in ihrer Geschwindigkeit, sie liefern wahrscheinlich auch genauere Ergebnisse als überwachte, wenn die verfügbare Trainingsmenge an validierten Genen für überwachtes Training nicht sehr groß ist[1].
Akkurate ab initio Algorithmen sind zudem wichtig, da sie selbst Gene aufspüren, die nicht durch verlässliche externe Quellen unterstützt werden[1]. Aber auch wenn extrinsische Hinweise oft unvollständig, meist unzuverlässig sind und alleine für gewöhnlich nicht ausreichen, um die kompletten Genstrukturen wiederherzustellen, können sie trotzdem in Kombination mit intrinsischen Hinweisen verwendet werden, um die Genvorhersage zu verbessern[13].
Mit Hilfe eines Generalisierten-Hidden-Markow-Modells (GHMM), welches Hinweise über potentiell proteinkodierende Regionen verwendet, wobei sowohl intrinsische als auch extrinsische Informationen berücksichtigt werden, ermöglicht das Programm *AUGUSTUS*[14] eine akkurate Genvorhersage[13]. Dafür benötigt es jedoch Parameteranpassungen für die vorherzusagende Spezies in Form einer Trainingsmenge von mindestens einigen hundert Genstrukturen[11].
Eine geeignete Trainingsmenge für *AUGUSTUS* kann allerdings mit Hilfe des Programms *GeneMark-ET*[1] erstellt werden. Da *GeneMark-ET* sich mit Hilfe von RNA-Seq-Daten iterativ selbst trainieren kann[1][11], sind bereits bestätigte Gene nicht notwendig[1].
Die im Verlauf dieser Arbeit entwickelt und programmierte Pipeline *BRAKER1* soll nun, vereinfacht gesagt, die Vorteile von *GeneMark-ET* des unüberwachten Selbsttrainings mit der akkuraten Genvorhersage von *AUGUSTUS* verbinden. Dafür werden eine Genomsequenz und zugehörige RNA-Seq-Daten in Form von Intron-Hinweisen an *GeneMark-ET* übergeben. Nach Abschluss des Trainings werden basierend auf den Genvorhersagen von *GeneMark-ET* die Parameter für *AUGUSTUS* optimiert, was im Idealfall in einer noch genaueren Vorhersage resultiert.
In den folgenden Kapiteln werden zunächst kurz die beiden Programme *GeneMark-ET* und *AUGUSTUS* vorgestellt und der Aufbau von *BRAKER1* erläutert. Zudem soll vermittelt werden, wie die nicht spezies-spezifischen Parameter, die *AUGUSTUS* verwendet, für möglichst alle Spezies optimal angepasst wurden.
Danach werden die für diese Arbeit in Perl implementierten Skripte näher beleuchtet. Hierfür werden zunächst die Funktionsweise des Hauptskripts *braker.pl* von *BRAKER1* geschildert, seine nötigen Voraussetzungen aufgezeigt und seine Optionen beschrieben. Für die Kompati-

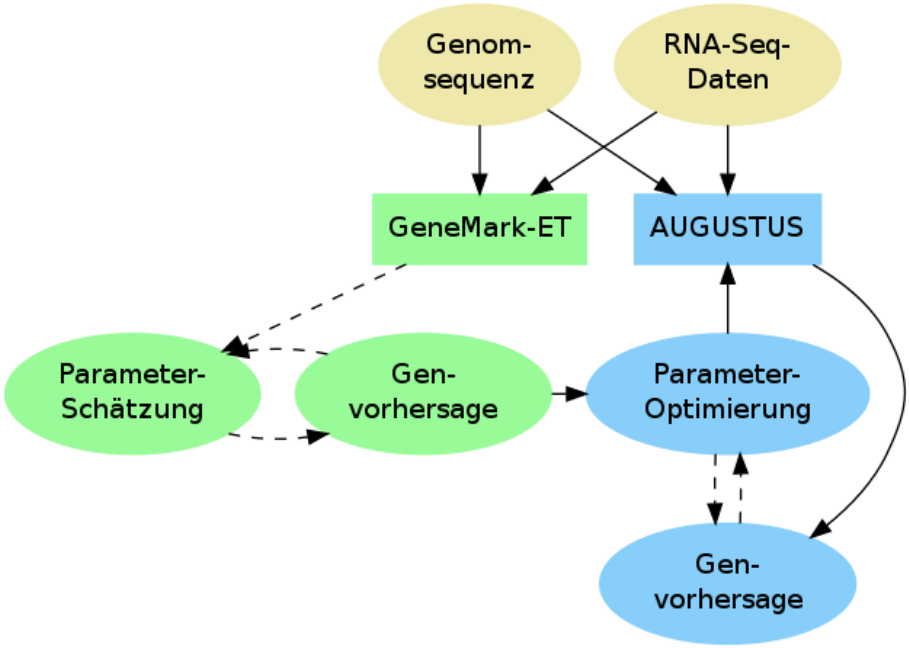

Abb. 1.1: Vereinfachter Ablauf von *BRAKER1*

bilität von *GeneMark-ET* mit *AUGUSTUS* sind noch zwei weitere Skripte zur Konvertierung nötig gewesen, deren Funktion anschließend illustriert wird.
Anhand von vier Beispiel-Spezies sollen dann die *BRAKER1*-Ergebnisse ausgewertet werden. Bei den vier Spezies handelt es sich um typische eukaryotische Modellorganismen verschiedener Reiche. Verglichen werden sollen die *BRAKER1*-Ergebnisse auch mit Ergebnissen einer ähnlichen Pipeline: *MAKER2*. Diese kann unter anderem annähernd so benutzt werden wie *BRAKER1*, verbindet aber *GeneMark-ES*, den Vorgänger von *GeneMark-ET*, mit *AUGUSTUS*. Leider ist *MAKER2* für diese Zwecke umständlich in der Bedienung und sehr zeitintensiv. Dieser Umstand regte unter anderem die Idee für *BRAKER1* an. Idealerweise soll nun *BRAKER1* bessere Ergebnisse liefern und nach Möglichkeit auch schneller und einfacher in der Handhabung sein.
Abschließend soll dann anhand der Ergebnisse überprüft werden, inwieweit *BRAKER1* die gesetzten Ziele erreichen konnte.

...

2. Methoden

BRAKER1 verbindet die beiden Programme *GeneMark-ET* und *AUGUSTUS*, um ihre jeweiligen Vorteile zu nutzen. *GeneMark-ET* der Arbeitsgruppe um Mark Borodovsky des Georgia Tech kann sich mit Hilfe von RNA-Seq-Daten iterativ selbst trainieren[1][11]. Dies ist kein einfaches Verfahren, verbessert aber die Genauigkeit der Genvorhersage[1]. Dadurch eignet es sich besonders für neue Genomprojekte, da es nicht auf bereits validierte Gene angewiesen ist[1][11].
AUGUSTUS auf der anderen Seite gehört zu den akkuratesten Genvorhersage-Programmen, benötigt jedoch Parameteranpassungen für die vorherzusagende Spezies in Form einer Trainingsmenge von mindestens einigen hundert Genstrukturen[11]. Solch eine Trainingsmenge lässt sich zum Beispiel aus der Vorhersage von *GeneMark-ET* gewinnen.
Die folgenden Abschnitte sollen nun zunächst die beiden Programme *GeneMark-ET* und *AUGUSTUS* beschreiben. Danach werden die einzelnen Schritte der Pipeline näher erläutert. Am Endes dieses Kapitels wird außerdem gezeigt, wie die von *AUGUSTUS* verwendeten, nicht spezies-spezifischen, Parameter optimiert wurden.

2.1 GeneMark-ET

Wie bereits angesprochen, liegt *GeneMark-ET*s größter Vorteil bei seiner Fähigkeit, sich selbst zu trainieren. Vereinfacht gesagt handelt es sich bei *GeneMark-ET* um eine Erweiterung von *GeneMark-ES*, dem einzigen Genvorhersage-Algorithmus für Eukaryoten, der automatisiertes Training im ab initio Modus ermöglicht[1]. *GeneMark-ET* integriert jedoch zusätzlich RNA-Seq read (gibt es in dem Zusammenhang eine anständige deutsche Übersetzung?) Alignments in den Selbst-Trainingsschritt[1]. Hierbei ist die Einführung sogenannter *Anchor Splice Sites* (*Anker Spleißstellen*?), Spleißstellen, die sowohl durch die Genvorhersage, als auch durch die RNA-Seq read Alignments unterstützt werden, der wichtigste Punkt bei der Kombinierung von ab initio Genvorhersage und RNA-Seq-Daten[1]. Somit ist *GeneMark-ET* nicht auf eine Trainingsmenge aus kompletten oder fast kompletten Genstrukturen angewiesen, sondern erstellt durch iteratives Training verlässliche Trainingsmengen aus Genelementen, Exons und Introns, die durch *Anchor Splice Sites* unterstützt werden[1]. Das sollte nicht nur den Annotationsprozess bei großen Genomen vereinfachen und beschleunigen, sondern auch die Genauigkeit der Genomidentifizierung verbessern[1].
Durch die Verwendung von mapped (kann man gemappt sagen?) RNA-Seq reads als extrinsische Hinweise, wird streng genommen aus dem unüberwachten ein semi-überwachter Trainingsalgorithmus[1]. Dieser semi-überwachte Ansatz umgeht das Problem, aus den RNA-Seq reads komplette Transkripte zu erstellen[1]. Ein komplettes Transkript, welches ohne Fehler an das Genom gemappt wurde, würde zwar mit hoher Genauigkeit ein Gen identifizieren, jedoch ist ein signifikanter Anteil der Transkript Assemblies (gibt es in dem Zusam-

menhang eine anständige deutsche Übersetzung?) fehlerbehaftet[1].
Normalerweise nimmt die Leistung von unüberwachtem Training für große eukaryotische Genome ab etwa einer Größe von mehr als 300Mb ab[1]. Genome, die eine größere durchschnittliche Intronlänge und intergenische Region haben, sowie große Repeat-Populationen können ebenfalls problematisch sein[1]. Eine starke Genom-Assembly Fragmentierung bereitet unüberwachtem Training eventuell auch Schwierigkeiten[1].
GeneMark-ET liefert jedoch konstant gute Ergebnisse, selbst wenn die Größe der Menge an mapped Introns, repeat Inhalte und Fragmentierung der Genomassembly schwankt[1]. Bei einer Größe von 100% bis 25% des Genoms bleiben die Ergebnisse von *GeneMark-ET* relativ konstant, zwischen 25% und 5% lässt die Leistung langsam nach und bei weniger als 5% des Genoms fällt sie stark ab (schriftliche Korrespondenz mit Alexandre Lomsadze[1]). Zu kleine Eingaben werden von *GeneMark-ET* mit folgender Fehlermeldung abgelehnt:

```
Must input more than one data point! at essuite/parse_ET.pl line 213.
Invalid regression data
```

2.1.1 Gliederung des GeneMark-ET-Algorithmus

Nachfolgend soll kurz der Algorithmus hinter *GeneMark-ET* betrachtet werden.

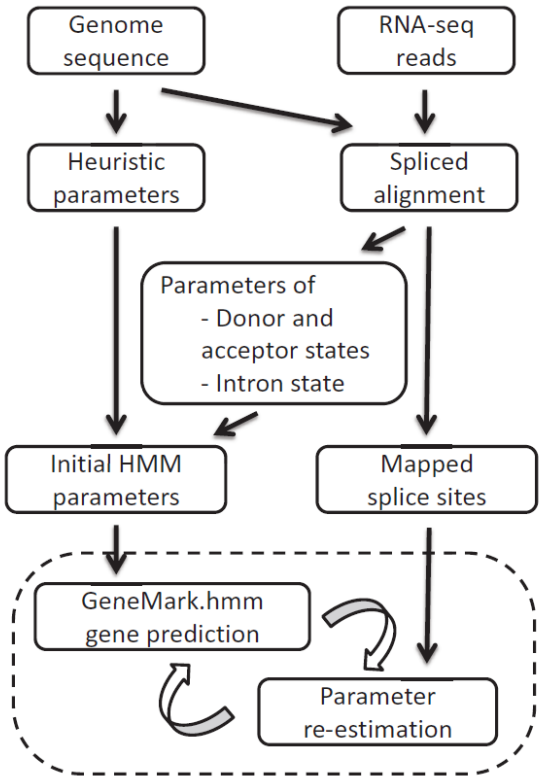

Abb. 2.1: Diagramm des iterativen semi-überwachten Trainings von *GeneMark-ET*[1]

Als Eingabe benötigt *GeneMark-ET* eine Genomsequenz und RNA-Seq reads[1][2]. Dann werden zuerst die RNA-Seq reads durch die Nutzung eines Short-Read-Alignment-Algorithmus

an die Genomsequenz aligniert[1]. Besonders relevant für *GeneMark-ET* sind die gespleißten Alignments, die Spleißstellen (splice junction = splice site?) in RNA-Seq reads und die zugehörigen Introns in der Genomsequenz identifizieren[2]. Der Algorithmus nutzt dann für die erste Vorhersage eine anfänglich durch heuristische Regeln definierte Menge an Parametern des *Hidden-Semi-Markow-Modells* (HSMM), um proteinkodierende Regionen in der Genomsequenz vorherzusagen[1][2][3]. Im nächsten Schritt wird eine Teilmenge der neu vorhergesagten Gene und nicht-kodierenden Regionen ausgewählt, die dann für die HSMM-Parameter Neuschätzung genutzt werden[1]. Die Auswahl geschieht jedoch nicht zufällig und im Gegensatz zu konventionellen Ansätzen, nutzt *GeneMark-ET* nicht für jeden versteckten Zustand dieselbe Trainingsmenge, sondern erstellt während des Trainings für jeden Zustand unabhängig eine eigene[2]. Je nach Zustand verwendet *GeneMark-ET* unterschiedliche Trainingsmethoden[2]. So werden Introns mit Hilfe von überwachtem, Exons mit semi-überwachtem und Translationsstart und -stop sowie intergenische Regionen mit unüberwachtem Training geschätzt[2].
Dafür nutzt der Algorithmus zwei Mengen an gemappten Introns, eine allgemeine Menge und eine *high confidence* (englisch okay oder sonst verlässlichere o.Ä.?) Menge, deren Introns einen *Score* $S > 0{,}1$ bzw. $S > 0{,}9$ haben. Proteinkodierende Exons werden nur dann hinzugefügt, wenn das vorhergesagte Exon mindestens eine *Anchor Splice Site* in der allgemeinen Menge hat[1][2]. Ausnahmen sind vorhergesagte Exons, die mehr als 800 Nukleotide enthalten[1]. Diese werden selbst unverankert für die Schätzung der Emissionswahrscheinlichkeit der Translationsinitiation und -terminations Stellen benutzt[1][2]. Um eventuell falsch vorhergesagte Exongrenzen zu berücksichtigen, werden die langen Exons um jeweils 60 Nukleotide an beiden Enden verkürzt[1]. Die Schätzung der Intron Längenverteilung und der Parameter für Donor- bzw. Akzeptorspleißstelle geschieht mit Hilfe der *high confidence* Menge[2].

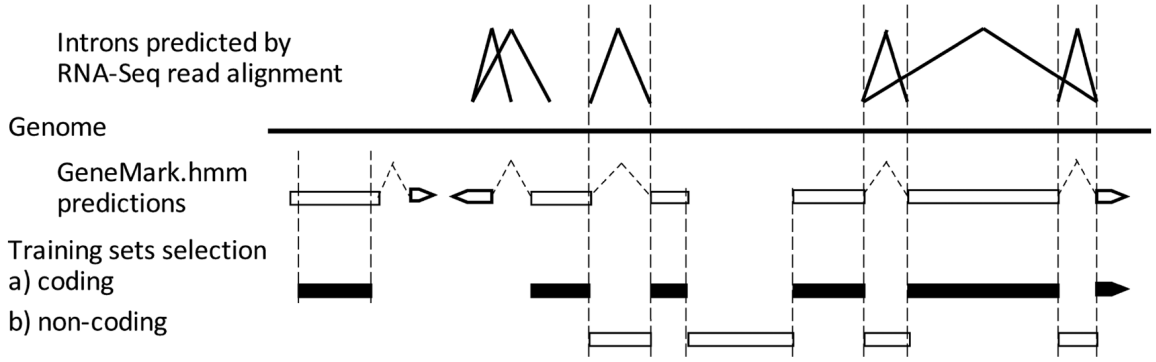

Abb. 2.2: Trainingsmenge von *GeneMark-ET* für die nächste Iteration[1]

Die beiden Schritte der Vorhersage und Neuschätzung werden solange wiederholt, bis sie konvergieren oder eine bestimmte Anzahl an Iterationen abgeschlossen wurde[1][2].
Wegen alternativer in-frame und out-of-frame ATG-Kodons hat *GeneMark-ET* jedoch Pro-

bleme, den Beginn der Translations-Initierung vorherzusagen[1]. Außerdem sind initiale Exons durchschnittlich kürzer als terminale Exons, was sie schwerer zu identifizieren macht[1].

2.2 AUGUSTUS

Im Gegensatz zu *GeneMark-ET* basiert *AUGUSTUS* auf einem Generalisierten-Hidden-Markow-Modell (GHMM), bei dem jeder Zustand der biologischen Bedeutung der Sequenz (z.B. Exon, Intron oder intergenische Region) entspricht[12][15]. Dies verwandelt eine Genstruktur in eine Folge von Zuständen, zusammen mit ihren Koordinaten, zu einem sogenannten *Parse*[15]. Um nun den bestmöglichen *Parse* mit dem höchsten *Score* zu finden, wird wie bei *GeneMark-ET*, der Viterbi-Algorithmus benutzt (FÜR GENEMARK-ET SCHON VORHER ERWÄHNEN?)[1][2][3][12][15]. In einem GHMM wird mit *Score* die Verbundwahrscheinlichkeit, dass das Modell die Zielsequenz unter Verwendung des *Parses* generiert, bezeichnet[15].
Normalerweise liefert der Viterbi-Algorithmus nur den maximalen *Parse*. Durch die Verwendung eines Sampling Algorithmus, der *n* zufällige *Parses* generiert, können die a-posteriori Wahrscheinlichkeiten von Exons, Introns, Transkripten und Genen geschätzt werden, mit deren Hilfe dann die Vorhersage von alternativen Transkripten möglich ist[12].
AUGUSTUS versucht Gene vorherzusagen, die biologisch konsistent sind, Exons enthalten also keine in-frame Stoppkodons und die Spleißstellen folgen der GT-AG-Regelung[10]. Alle kompletten Gene starten mit ATG und enden mit einem Stoppkodon, wobei ein neues Gen erst beginnt, wenn das vorherige abgeschlossen ist[10]. Außerdem dürfen die Introns und Exons eine spezies-spezifische Länge nicht unterschreiten[10]. Transkripte, deren Kodierende-Sequenz eine bestimmte Länge nicht erreichen, deren einzelne Exons oder Introns a-posteriori Wahrscheinlichkeit bzw. bei denen das geometrische Mittel aller Exons und Introns unter einer Konstanten liegen, werden verworfen[12].
AUGUSTUS kann zudem Hinweise von externen Quellen, hier die RNA-Seq-Daten, verwenden, um sie mit einer ab initio Vorhersage zu kombinieren[15]. Da aber jede Spezies für die Vorhersage ein eigenes GHMM hat, deren Parameter individuell angepasst werden müssen, benötigt AUGUSTUS etwa einige hundert Genstrukturen als Trainingsmenge[8][11][12].

2.2.1 Aufbau der verwendeten *AUGUSTUS*-Komponenten

Da *AUGUSTUS* auch Programme und Skripte enthält, die für diese Arbeit nicht verwendet wurden, wird nachfolgend keine komplette Beschreibung von *AUGUSTUS* geliefert, sondern nur eine Beschreibung, wie *BRAKER1* es nutzt.

TODO: Algorithmus-Ablauf;

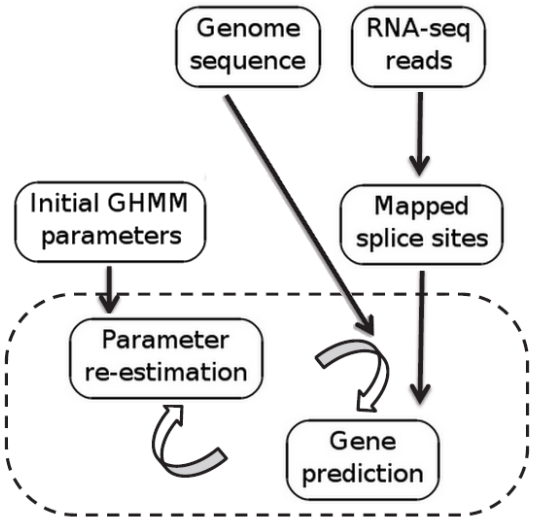

Abb. 2.3: Diagramm des iterativen unüberwachtem Training von *AUGUSTUS*

2.3 BRAKER1

Wie schon in der Einleitung erwähnt, ist *BRAKER1* eine in Perl programmierte Pipeline, die *GeneMark-ET* und *AUGUSTUS* verbindet. Als Eingabe benötigt *BRAKER1* zwei Dateien, eine Genomsequenz im FASTA-Format und RNA-Seq-Daten im BAM-Format.
Zunächst werden die RNA-Seq-Daten mit Hilfe von *bam2hints*[14] in Intron-Hinweise im *General Feature Format* (GFF) konvertiert. Die erstellte Hinweisdatei ist so aber noch nicht mit *GeneMark-ET* kompatibel. *GeneMark-ET* benötigt zusätzlich noch Informationen darüber, auf welchem Strang sich das jeweilige Intron befindet. Außerdem verwendet *GeneMark-ET* als *Score* den Wert, der bei *AUGUSTUS* unter „*mult*“ in der letzten Spalte zu finden ist. Dieser Wert gibt an, wie verlässlich der Hinweis, hier das Intron, ist. Je größer also dieser Wert ist, desto wahrscheinlicher liegt bei der angegebenen Stelle wirklich ein Intron vor. Vorliegender Konvertierungsschritt wird von *filterIntronsFindStrand.pl* (siehe auch Abschnitt 3.1.1) vorgenommen.
Basierend auf der konvertierten Hinweis- und der Genomdatei erstellt *GeneMark-ET* eine Genvorhersage im *Gene Transfer Format* (GTF). Die resultierende Datei wird dann mit *filterGenemark.pl* (siehe auch Abschnitt 3.1.2) gefiltert. In diesem Vorgang teilt das Programm die Gene in „gute“ und „schlechte“ ein. Hierbei gilt ein Gen als gut, wenn alle seine Introns als exakte Kopie in der Hinweisdatei vorliegen oder es intronlos ist. Die übrigen Gene werden als schlecht bezeichnet.
Danach erstellt *gff2gbSmallDNA.pl*[14] aus der *GeneMark-ET*-Vorhersage und der Genomdatei eine Datei im Genbank-Format, aus der *filterGenesIn_mRNAname.pl*[14] die schlechten Gene aussortiert, sodass nur noch die guten Gene verbleiben. Falls diese Datei mehr als 1000 Gene enthält, werden sie mit *randomSplit.pl* zufällig in eine Trainings- und eine Testmenge aufgeteilt, wobei die Testmenge 1000 Gene enthält und die Trainingsmenge die übrigen. Sind weniger als 1000 Gene vorhanden, wird die Datei als Trainings- und Test-

menge genutzt. Mindestens 300 Gene werden empfohlen, andernfalls gibt *BRAKER1* eine Warnung aus bzw. unterbricht die Ausführung, falls keine unterstützten Gene gefunden wurden.

new_species.pl erzeugt dann die Speziesparameterdateien, die *AUGUSTUS* später für seine Genvorhersage benötigt. Diese Parameterdateien werden mit *etraining*[14] und *optimize_augustus.pl*[14] mit Hilfe der Trainings- und Testmenge optimiert.

Nach Abschluss der Parameteroptimierung werden die Hinweis- und die Genomdatei an *AUGUSTUS* übergeben und darauf basierend eine Genvorhersage im GFF-Format erstellt.

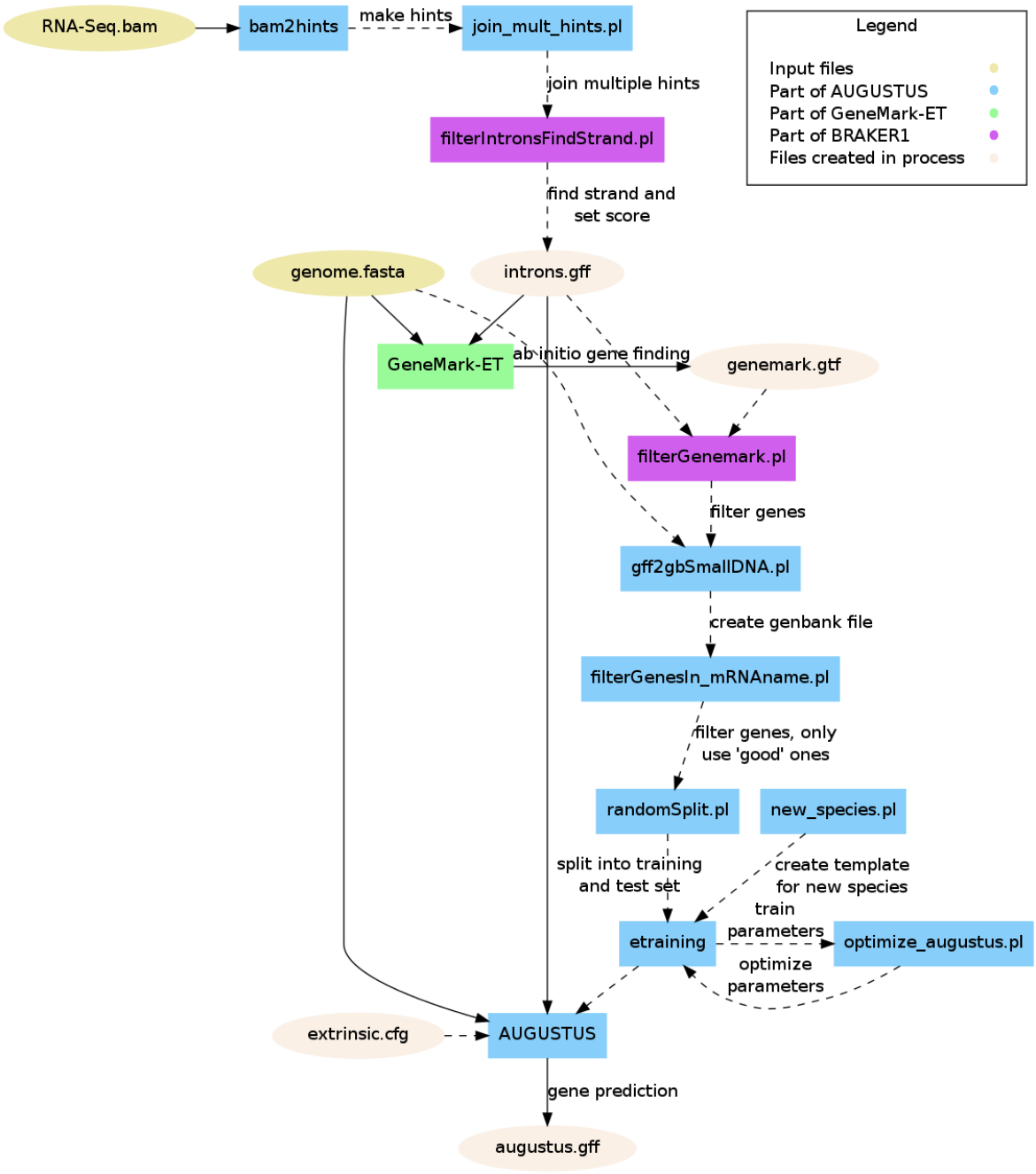

Abb. 2.4: Übersicht der einzelnen Vorgänge in BRAKER1

2.4 Parameter

AUGUSTUS benutzt für seine Genvorhersage mehrere Parameter. Einige davon werden im Verlauf von *BRAKER1* mit *etraining*[14] bzw. *optimize_augustus.pl*[14] automatisch für die jeweilige Spezies optimiert. Auch die Häufigkeiten der Stoppkodons TAG, TAA und TGA werden spezies-spezifisch anhand der ersten *etraining*-Ausgabe[14] angepasst.
Einige weitere Parameter jedoch sind nicht spezies-spezifisch. Diese befinden sich in der *extrinsic.cfg*[14] Datei und geben an, wie stark die jeweils vorhergesagte Genstruktur belohnt oder bestraft wird[16]. Dies ist der entsprechende Auszug der *extrinsic.cfg*[14] Datei:

```
[GENERAL]
      start        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
       stop        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
        tss        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
        tts        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
        ass        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
        dss        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
   exonpart        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
       exon        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
 intronpart        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
     intron     1.15      .25  M    1  1e+100  RM  1     1    E 1    1    W 1    1
    CDSpart        1  1     1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
        CDS        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
    UTRpart        1  1     1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
        UTR        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
     irpart        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
nonexonpart        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
  genicpart        1        1  M    1  1e+100  RM  1     1    E 1    1    W 1    1
```

Da *GeneMark-ET* momentan nur Intron-Hinweise verarbeiten kann, werden auch nur diese aus den RNA-Seq-Daten extrahiert. So müssen nur die Einträge für die Intron-Hinweise optimiert werden, die übrigen Einträge bleiben neutral, also weder Belohnung noch Bestrafung für die Vorhersage der jeweiligen Genstrukturen. Durch mehrere Testläufe von *BRAKER1* wurde versucht, die Intron-Hinweise für alle Spezies möglichst optimal anzupassen.
Zu Beginn wurde pro Durchlauf jeweils nur ein Wert geändert, um zu sehen, ob eine Vergrößerung bzw. Verkleinerung des Eintrags sich positiv oder negativ auf die Vorhersage auswirkt. Am Anfang wurden nur die Parameter für *Schizosaccharomyces pombe* getestet, da seine Testgenomgröße mit drei Chromosomen am kleinsten war und so in kürzerer Zeit mehr Testdurchläufe ausgewertet werden konnten. Anschließend wurden die vielversprechendsten Änderungen auch an den übrigen drei Spezies getestet und überprüft, ob sich für diese die neuen Werte ebenfalls bewähren.
Das war leider nicht der Fall, weswegen weitere Anpassungen nötig waren.
AUSWERTUNG NOCH NICHT FÜR ALLE SPEZIES ABGESCHLOSSEN

3. Implementierung

Realisiert wurde das Projekt in Perl. Zur Ausführung der Pipeline werden zusätzlich Dateien von *AUGUSTUS* und *GeneMark-ET* benötigt. Diese befinden sich auf der beigelegten DVD, können aber auch hier `http://bioinf.uni-greifswald.de/augustus/binaries` bzw. hier `http://exon.gatech.edu/Genemark/license_download.cgi` heruntergeladen werden. Bevor das Programm benutzt werden kann, müssen die Umgebungsvariablen mit

```
~$ export BAMTOOLS_PATH=/path/to/bamtools_executable
~$ export GENEMARK_PATH=/path/to/GeneMark-ET/gmes_petap.pl
~$ export AUGUSTUS_CONFIG_PATH=/path/to/augustus/config
~$ export PATH=/path/to/augustus/bin:/path/to/augustus/scripts
```

bzw. entsprechendem Pfad gesetzt werden. Diese Ordner enthalten die Konfigurations- und Parameter-Dateien bzw. die Binärdateien und die Skripte. Optional lässt sich so auch der Pfad zur ausführbaren Datei von *Samtools* setzen. Dieses Programm wird hier standardmäßig nicht benötigt, *BRAKER1*kann aber unzulässige Sequenznamen in der RNA-Seq Eingabedatei bei Bedarf automatisch korrigieren, wenn *Samtools* auf dem System installiert ist und der entsprechende Pfad angegeben wurde.
Für *GeneMark-ET* müssen außerdem weitere Perl-Module heruntergeladen werden. Diese sind: *YAML*, *Hash::Merge*, *Logger::Simple* und *Parallel::ForkManager*. Die folgenden Module müssen vor der Benutzung von *BRAKER1* gegebenenfalls auch installiert werden: *Cwd*, *File::Path*, *File::Basename*, *File::Spec::Functions*, *List::Util*, *Module::Load::Conditional*, *POSIX* und *Scalar::Util::Numeric*. Mit Hilfe des *CPAN Minus-Tools* und folgenden Befehlen lassen sich zunächst das nötige *Tool* und danach die obigen Module unter Linux leicht installieren.

```
~$ sudo apt-get install cpanminus
~$ cpanm [Module::Name]
```

Um nun die Pipeline über die Shell zu starten, lautet der Befehl:

```
~$ perl braker.pl --species=SpeziesName --genome=Genom.fasta
--bam=BAMDatei.bam
```

Hierbei handelt es sich aber nur um die nötigsten Angaben. Weitere Optionen lassen sich mit

```
~$ perl braker.pl --help
```

anzeigen. Diese sind:

```
braker.pl     annotate genomes based on RNAseq using GeneMark-ET and AUGUSTUS

SYNOPSIS
```

```
braker.pl [OPTIONS] --species=sname --genome=genome.fa
--bam=accepted_hits.bam

sname is an identifier for the species. Use augustus --species=help to see
a list.

  --genome=genome.fa          fasta file with DNA sequences
  --bam=rnase.bam             bam file with spliced alignments from RNA-Seq


OPTIONS

    --help                              Print this help message
    --alternatives-from-evidence=true   Output alternative transcripts based
                                        on explicit evidence from hints
                                        (default is true).
    --AUGUSTUS_CONFIG_PATH=/path/       Set path to AUGUSTUS (if not specified
      to/augustus                       as environment variable). Has higher
                                        priority than environment variable.
    --BAMTOOLS_PATH=/path/to/           Set path to bamtools (if not specified
      bamtools                          as environment variable). Has higher
                                        priority than the environment variable.
    --CPU                               Specifies the maximum number of CPUs
                                        that can be used during
                                        computation
    --fungus                            GeneMark-ET option: run algorithm with
                                        branch point model (most useful for
                                        fungal genomes)
    --GENEMARK_PATH=/path/to/           Set path to GeneMark-ET (if not
      gmes_petap.pl                     specified as environment variable).
                                        Has higher priority than environment
                                        variable.
    --hints=hints.gff                   Alternatively to calling braker.pl
                                        with a bam file, it is possible to
                                        call it with a file that contains
                                        introns extracted from RNA-Seq data in
                                        gff format. This flag also allows the
                                        usage of hints from additional
                                        extrinsic sources for gene prediction
                                        with AUGUSTUS. To consider such
                                        additional extrinsic information,
```

```
                                    you need to use the flag --optCfgFile
                                    to specify parameters for
                                    all sources in the hints file
                                    (including the source Ë" for intron
                                    hints from RNA-Seq).
    --optCfgFile=ppx.cfg            Optional custom config file for
                                    AUGUSTUS (see --hints).
    --overwrite                     Overwrite existing files (except for
                                    species parameter files)
    --SAMTOOLS_PATH=/path/to/       Optionally set path to samtools (if
      samtools                      not specified as environment
                                    variable) to fix BAM files
                                    automatically, if necessary. Has
                                    higher priority than environment
                                    variable.
    --skipGeneMark-ET               Skip GeneMark-ET and use provided
                                    GeneMark-ET output (e.g. from a
                                    different source)
    --skipOptimize                  Skip optimize parameter step (not
                                    recommended).
    --softmasking
    --species=sname                 Species name. Existing species will
                                    not be overwritten.
                                    Uses Sp_1 etc., if no species is
                                    assigned
    --useexisting                   Use the present config and parameter
                                    files if they exist for 'species'
    --UTR                           Predict untranslated regions. Default
                                    is off.
    --workingdir=/path/to/wd/       Set path to working directory. In the
                                    working directory results and
                                    temporary files are stored
    --version                       Print version number of braker.pl


DESCRIPTION

  Example:

    braker.pl [OPTIONS] --genome=genome.fa  --species=speciesname
    --bam=accepted_hits.bam
```

Liegen also noch weitere Hinweisdateien im GTF-Format vor, können diese mit `--hints` zugewiesen werden.
Die Verwendung der Option `--UTR` ist momentan nur für die von *AUGUSTUS* mitgelieferten Spezies möglich. Eine Liste der in *AUGUSTUS* verfügbaren Spezies, die mit der Option `--useexisting` genutzt werden können, lässt sich mit

```
~$ augustus --species=help
```

anzeigen. Die zugehörigen Dateien können ebenfalls hier heruntergeladen werden:
http://bioinf.uni-greifswald.de/augustus/binaries. Ohne diese Option werden Dateien für eine neue Art unter dem mit `--species` angegebenen Namen angelegt, falls diese noch nicht existiert, andernfalls wird der Vorgang abgebrochen. Wird kein Name angegeben, verwendet *breaker.pl Sp_Nummer* als Bezeichnung, wobei die Pipeline beendet wird, falls bereits 10000000 solcher Ordner unter dem aktuellen Verzeichnis existieren. Mit `--workingdir` kann bei Bedarf ein anderes Verzeichnis als Arbeitsverzeichnis angegeben werden.
Die Option `--fungus` ist nur für *GeneMark-ET* verfügbar und sollte nur verwendet werden, wenn das Genom zu einem Pilz gehört.
Da *GeneMark-ET* und auch *AUGUSTUS* die Verwendung mehrerer Prozessoren unterstützen, kann mit `--CPU` angegeben werden, wie viele Prozessoren parallel benutzt werden dürfen. Falls der Nutzer mehr CPUs zuweisen möchte, als das System besitzt, gibt *BRAKER1* eine entsprechende Warnung aus, dass das System nicht über die angegebene Anzahl an Prozessoren verfügt und dass stattdessen alle vorhandenen CPUs genutzt werden.
Beide Programme unterstützen auch die Verwendung der `--softmasking` Option.

TODO: Infos zur softmasking Option

Standardmäßig gibt *AUGUSTUS* alternative Transkripte, basierend auf den expliziten Hinweisen der RNA-Seq-Daten, mit aus. Diese Ausgabe kann unterbunden werden, wenn `--alternatives-from-evidence` auf `false` gesetzt wird. Weitere Änderungen der Standardeinstellungen können durch die Übergabe einer optionalen Konfigurationsdatei mit `--optCfgFile` an *AUGUSTUS* vorgenommen werden. Diese Datei hat eine höhere Priorität als die Datei, die sonst *new_species.pl*[14] erstellt.
Falls gewünscht, können mit `--skipGeneMark-ET` und `--skipOptimize` die Ausführung von *GeneMark-ET* bzw. die Optimierung der Spezies Parameterübersprungen werden. Diese Optionen dienten jedoch mehr zur Entwicklung von *BRAKER1*, als die damalige *GeneMark-ET*-Version noch nicht auf jedem System reproduzierbare Ergebnisse lieferte und die zeitintensive Parameteroptimierung noch nicht bei jedem Durchlauf notwendig war. Für den normalen Aufruf von *BRAKER1* wird ihre Verwendung nicht empfohlen. Wird *BRAKER1*, aus welchen Gründen auch immer, an einer bestimmten Stelle unterbrochen, wird bei einem erneuten Aufruf beim letzten Arbeitsschritt wieder angesetzt, da die *uptodate*-Funktion des *helpMod.pm*[14] Perl-Moduls von *AUGUSTUS* bei jedem Schritt die Erstelldaten der jeweiligen Dateien vergleicht und diesen nur ausführt, wenn die Ausgabedatei noch nicht vorhanden oder älter als die Eingabedatei ist. Mit der Option `--overwrite` kann jedoch die Überschreibung der bereits vorhandenen Ausgabedateien erzwungen werden. Um aber die Speziesparameter von *AUGUSTUS* zu schützen, hat diese Option keine Auswirkung auf bereits bestehende Dateien unter *AUGUSTUS_Config_PATH*.
Mit `--version` wird zusätzlich die aktuelle Versionsnummer von *BRAKER1* ausgegeben.
Im aktuellen Arbeitsverzeichnis erstellt *BRAKER1* folgenden Verzeichnisbaum:

- braker
  - sname
    - augustus.gff ..... Genvorhersage von *AUGUSTUS* im GFF-Format
    - bam_header.map ..... enthält alte und neue (ohne "|") Sequenznamen der BAM-Datei
    - braker.log . enthält die verwendeten Shell-Befehle und den Zeitpunkt, zu dem sie gestartet wurden
    - corrected.bam .... korrigierte BAM-Datei, falls Eingabedatei "|"enthielt. Nur vorhanden, falls *samtools* installiert ist
    - errors
      - ProgrammName.stderr..... enthält Standardfehlerausgabe der einzelnen Programme
    - filterGenemark.stdout ..... Standardausgabe von *filterGenemark.pl*
    - firstetraining.stdout Standardausgabe des ersten *etraining* Durchgangs
    - firsttest.stdout ..... Standardausgabe des ersten *AUGUSTUS* Tests
    - genbank.gb .. enthält Informationen zu allen von *GeneMark-ET* vorhergesagten Genen im Genbank-Format
    - genbank.good.gb..... enthält Informationen zu allen von *GeneMark-ET* vorhergesagten guten Genen
    - genbank.good.gb.test..... nur, falls genbank.good.gb mehr als 1000 Gene enthält
    - genbank.good.gb.train..... nur, falls genbank.good.gb mehr als 1000 Gene enthält
    - GeneMark-ET
      - genemark.average_gene_length.out ..... durchschnittliche Länge der kompletten Gene
      - genemark.c.gtf . korrigierte *GeneMark-ET* Ausgabe (für Version 4.21 mit genemark.gtf identisch)
      - genemark.f.bad.gtf .....gefilterte *GeneMark-ET* Ausgabe, nur schlechte Gene
      - genemark.f.good.gtf gefilterte *GeneMark-ET* Ausgabe, nur gute Gene
      - genemark.gtf.....Genvorhersage von *GeneMark-ET* im GTF-Format
      - weitere, für *BRAKER1* nicht relevante Dateien und Ordner
    - GeneMark-ET.stdout .....Standardausgabe von *GeneMark-ET*
    - genome.fa ..... Genom im fasta Format
    - genome.split.NR.fa . Teil der Genom-Datei, erstellt mit *splitMfasta.pl* [14]
    - header.map .. enthält alte und neue (ohne "|"oder Leerzeichen) Sequenznamen der Genom-Datei
    - hintsfile.gff ..... enthält Intron-Hinweise, die mit *bam2hints* [14] aus der BAM-Datei gewonnen und mit *filterIntronsFindStrand.pl* für *GeneMark-ET* kompatibel gemacht wurden
    - optimize_augustus.stdout ... Standardausgabe von *optimize_augustus.pl*
    - secondetraining.stdout ..... siehe firstetraining.stdout
    - secondtest.stdout.....siehe firsttest.stdout
    - species
      - sname.....Ordner enthält die Parameter-Dateien
        - extrinsic.sname.cfg
        - sname_exon_probs.pbl

```
├── sname_igenic_probs.pbl
├── sname_intron_probs.pbl
├── sname_metapars.cfg
├── sname_metapars.cgp.cfg
├── sname_metapars.utr.cfg
├── sname_parameters.cfg
├── sname_parameters.cfg.orig1
└── sname_weightmatrix.txt
```

3.1 Perl-Skripte

Im folgenden Abschnitt sollen die übrigen Perl-Skripte beschrieben werden, die *BRAKER1* verwendet und die nicht Teil von *GeneMark-ET* oder *AUGUSTUS* sind.

3.1.1 filterIntronsFindStrand.pl

Um die RNA-Seq-Daten für *GeneMark-ET* verwendbar zu machen, muss das Format der Hinweise von *bam2hints*[14] bzw. *join_mult_hints.pl* [14] angepasst werden. Dies geschieht mit dem *filterIntronsFindStrand.pl* Skript, welches mit Hilfe der Genom-Eingabedatei für jedes Intron den zugehörigen Strang ermittelt. Für die Verwendung durch *GeneMark-ET* ist es außerdem erforderlich, dass die *mult*-Werte der Hinweisdatei auch in der sechsten Spalte als *Score* eingetragen werden, was durch die Hinzunahme des zusätzlichen Parameters `--score` geschieht. Da der *Score* der Hinweisdatei, die durch *AUGUSTUS* erzeugt wurde standardmäßig 0 ist, können diese Einträge bedenkenlos überschrieben werden. Die Ausgabedatei von *filterIntronsFindStrand.pl* ist, wie die Eingabedatei, im GFF-Format. Mit

```
~$ perl filterIntronsFindStrand.pl --help
```

lässt sich eine nähere Beschreibung des Skripts ausgeben.

```
filterIntronsFindStrand.pl     find corresponding strand for introns from two
input files genome.fa and introns.gff

SYNOPSIS

filterIntronsFindStrand.pl genome.fa introns.gff [OPTIONS] > introns.s.f.gff

  genome.fa           DNA file in fasta format
  introns.gff         corresponding introns file in gff format


OPTIONS

    --help                          Print this help message
    --allowed=gtag,gcag,atac        Allowed acceptor and donor splice site
```

```
                                types
    --score                     Set score to 'mult' entry or '1', if the
                                last column does not contain a 'mult'
                                entry
    --genome=genome.fa          see above
    --introns=introns.gff       see above


DESCRIPTION

 Example:

  filterIntronsFindStrand.pl genome.fa introns.gff [OPTIONS] > introns.s.f.gff
```

Zusätzlich kann mit der Option `--allowed=erlaubteSpleißtypen` die erlaubten Spleißtypen angegeben werden, Standardeinstellung sind die drei häufigsten Muster „GT-AG“, „GC-AG“ und „AT-AC“[7].

3.1.2 filterGenemark.pl

Die GTF-Ausgabedatei von *GeneMark-ET genemark.gtf* kann mit Hilfe des Perl-Skripts *filterGenemark.pl* gefiltert werden, wobei überprüft wird, welche Gene als gut und welche als schlecht zu bewerten sind. Da die Gen- und Transkript-Identifikatoren in *genemark.gtf* bis Version 4.21 keine Anführungszeichen (wie sonst üblich[9]) enthalten, wird dies bei Bedarf ebenfalls angepasst, weswegen *filterGenemark.pl* drei Ausgabedateien *genemark.c.gtf*, *genemark.f.good.gtf* und *genemark.f.bad.gtf* erstellt, wobei *genemark.c.gtf* alle Gene, *genemark.f.good.gtf* die guten und *genemark.f.bad.gtf* die schlechten enthält.
Auch hier lassen sich mit

```
~$ perl filterGenemark.pl --help
```

weitere Informationen zum Skript ausgeben.

```
filterGenemark.pl     filter GeneMark-ET files and search for "good" genes

SYNOPSIS

filterGenemark.pl [OPTIONS] genemark.gtf introns.gff

  genemark.gtf          file in gtf format
  introns.gff           corresponding introns file in gff format
```

```
OPTIONS

    --help                          Print this help message
    --introns=introns.gff           Corresponding intron file in gff format
    --genemark=genemark.gtf         File in gtf format
    --output=newfile.gtf            Specifies output file name. Default is
                                    'genemark-input_file_name.c.gtf'
                                    and 'genemark-input_file_name.f.good.gtf'
                                    and 'genemark-input_file_name.f.bad.gtf'
                                    for filtered genes included and not
                                    included in intron file respectively

Format:
  seqname <TAB> source <TAB> feature <TAB> start <TAB> end <TAB> score <TAB>
  strand <TAB> frame <TAB> gene_id value <TAB> transcript_id value


DESCRIPTION

  Example:

    filterGenemark.pl [OPTIONS] --genemark=genemark.gtf --introns=introns.gff
```

4. Beispiele

Die Pipeline *BRAKER1* wurde an vier unterschiedlichen Spezies getestet. Diese waren der Kreuzblütler *Arabidopsis thaliana*, der Fadenwurm *Caenorhabditis elegans*, die Taufliege *Drosophila melanogaster* und der Hefepilz *Schizosaccharomyces pombe*.
Die dafür verwendeten Genom-Dateien und RNA-Seq-Daten stammen von Alexandre Lomsadze[1]. So sollte sichergestellt werden, dass alle an dem Projekt beteiligten Personen mit den gleichen Daten arbeiteten.
Die von *BRAKER1* erzeugten *genemark.gtf* und *augutus.gff* bzw. die zum Vergleich in GFT-Format konvertierte *augutus.gtf* Dateien wurden anschließend mit den Ergebnissen von *AUGUSTUS*, basierend auf den mitgelieferten Standardparametern der einzelnen Spezies und vergleichbaren *MAKER2*-Ausgaben, anhand einer Referenzannotation mit Hilfe des *Eval*-Software-Pakets[6] ausgewertet. Die *Eval*-Ergebnisse für *MAKER2* wurden von Katharina Hoff[14] zur Verfügung gestellt, da sie auch die *MAKER2*-Durchläufe ausgeführt hat.
Die verwendeten Referenzannotationen liegen im GTF-Format vor. Sie stammen von unterschiedlichen Genom-Datenbanken, die in den jeweiligen Abschnitten näher erläutert werden.
Erstellt wurden die Ergebnisse auf einem 64 bit Rechner unter Ubuntu 14.04 mit vier Prozessoren. Verwendet wurden außerdem die *Perl 5* Version 18, Subversion 2 (v5.18.2), *AUGUSTUS* Version 3.0.3, *GeneMark-ET* Version 4.21, die *Bamtools* bzw. *Samtools* Versionen 2.3.0 bzw. 0.1.19 und die *Eval* Version 2.2.8.
Der folgende Abschnitt zeigt exemplarisch anhand von *Arabidopsis thaliana*, wie die Ausgabedateien und die Evaluationsergebnisse gewonnen wurden.
In den darauffolgenden Abschnitten werden für die übrigen drei Spezies nur noch die Ergebnisse beschrieben.

4.1 Arabidopsis thaliana

Für *Arabidopsis thaliana* wurde *BRAKER1* an den ersten fünf verfügbaren Chromosomen getestet. Diese stammen von *The Arabidopsis Information Resource* (*„TAIR“*) und liegen im FASTA-Format vor. Verwendet wurden für die DNA-Sequenzen und die Referenzannotation jeweils die Version *TAIR* 10 von 2010. Sie können unter www.arabidopsis.org heruntergeladen werden.

4.1.1 Durchführung

Um nun *BRAKER1* für eine neue Spezies *A.thaliana* zu starten, kann folgender Aufruf benutzt werden:

```
~$ perl braker.pl --species=A.thaliana --bam=accepted-hits.bam
```

```
--genome=genome.fa
```

So werden auch alternative Transkripte mit ausgegeben. Diese Funktion ist aber nur für die Vorhersage mit *AUGUSTUS* verfügbar und nicht für *GeneMark-ET*. Die für die Auswertung verwendeten *GeneMark-ET*-Vorhersagen stammen ebenfalls aus dem *BRAKER1*-Durchlauf. Dies ist ein kurzer Auszug aus der *GeneMark-ET* Ausgabedatei *genemark.gtf*:

HIER AUSZUG EINFÜGEN

Die zum Vergleich verwendeten *AUGUSTUS*-Vorhersagen mit den mitgelieferten Standardparametern für *Arabidopsis thaliana* lassen sich folgendermaßen erzeugen:

```
~$ augustus --species=A.thaliana --alternatives-from-evidence=true
--genome=genome.fa > augustus_abinitio.gff
```

So sehen die von *AUGUSTUS* erzeugten Ausgaben standardmäßig aus:

HIER AUSZUG EINFÜGEN

Um die erzeugten Ausgabedateien mit der Referenzannotation vergleichbar zu machen, sind weitere Konvertierungsschritte nötig. Zunächst werden die *AUGUSTUS* Ausgaben *augustus.gff* und *augustus_abinitio.gff* auf das GTF Format gebracht. Dies geschieht mit Hilfe des *gtf2gff.pl*[14] Skripts und folgenden Aufrufen:

```
~$ cat augustus.gff | perl -ne 'if(m/\tAUGUSTUS\t/){print $_;}' |
perl /path/to/augustus/scripts/gtf2gff.pl --printExon
--out=augustus.gtf

~$ cat augustus_abinitio.gff | perl -ne 'if(m/\tAUGUSTUS\t/)
{print $_;}' | perl /path/to/augustus/scripts/gtf2gff.pl
--printExon --out=augustus_abinitio.gtf
```

Da die Referenzannotationen keine Informationen zu Start- und Stoppkodons enthalten, müssen in den GTF-Dateien die Stoppkodons entfernt werden, damit *Eval*[6] die Dateien richtig vergleichen kann. Somit werden alle Gene als unvollständig betrachtet. Andernfalls wären die Prozentzahlen auf dem Genlevel fast Null, da *Eval*[6] komplette und nicht komplette Gene nicht als gleich erkennt, selbst wenn sie in ihren Exons übereinstimmen. Die nicht benötigten Einträge können folgendermaßen entfernt werden:

```
~$ cat augustus.gtf | perl -ne '@t = split(/\t/); if(($t[2] eq
“CDS”) or ($t[2] eq “exon”) or ($t[2] eq “start_codon”) or ($t[2]
eq “UTR”)){print $_;}' >augustus.f.gtf

~$ cat augustus_abinitio.gtf | perl -ne '@t = split(/\t/); if((
```

```
$t[2] eq‘‘CDS’’) or ($t[2] eq ‘‘exon’’) or ($t[2] eq ‘‘start_codon’’)
 or ($t[2]eq ‘‘UTR’’)){print $_;}’ >augustus_abinitio.f.gtf

~$ cat genemark.gtf | perl -ne ’@t = split(/\t/); if(($t[2] eq
‘‘CDS’’) or ($t[2] eq ‘‘exon’’) or ($t[2] eq ‘‘start_codon’’) or ($t[2]
eq ‘‘UTR’’)){print $_;}’ >genemark.f.gtf
```

Nach der Entfernung der überschüssigen Einträge, können die ersten Zeilen der Dateien zum Beispiel so aussehen:
HIER AUSZUG EINFÜGEN

Danach werden die gekürzten GTF-Dateien mit dem *GTF-Validator*[6] des *Eval*-Software-Pakets kontrolliert. Dieser verifiziert, dass das Format korrekt ist und die Gene die Regeln der Genstruktur einhalten[5]. Der *GTF-Validator*[6] wird wie folgt aufgerufen:

```
~$ perl /path/to/eval/validate_gtf.pl -c -f augustus.f.gtf
~$ perl /path/to/eval/validate_gtf.pl -c -f augustus_abinitio.f.gtf
~$ perl /path/to/eval/validate_gtf.pl -c -f genemark.f.gtf
```

Die Optionen `-c` bzw. `-f` unterdrücken Fehlermeldung über fehlende Start- und Stoppkodons bzw. erstellen eine neue Datei ‘‘`.fixed.gtf`’’ ohne GTF-Format-Fehler[5]. Die Nutzung des *GTF-Validators* wird empfohlen, wenn die Eingabedateien nicht von denselben Entwicklern stammen und deswegen eventuell Unterschiede aufweisen könnten[5]. Ein Problem, auf das zum Beispiel in der *Eval*-Dokumentation hingewiesen wird, sind in-frame-Stoppkodons[5]. Dieses trat hier nicht auf, jedoch enthielten die Transkripte der *GeneMark-ET*-Ausgabe für *Arabidopsis thaliana* inkonsistente Phasenwerte. Die neu formatierten Dateien sehen beispielsweise so aus:
HIER AUSZUG EINFÜGEN

Anschließend können die validierten GTF-Dateien mit *evaluate_gtf.pl*[6] ausgewertet werden.

```
~$ perl /path/to/eval/evaluate_gtf.pl TAIR10-anno.fff.gtf
augustus.f.fixed.gtf augustus_abinitio.f.fixed.gtf
genemark.c.f.fixed.gtf >A.thaliana.eval.out
```

Hierbei muss die erste Datei die Referenzannotation sein[5]. Nachfolgend können beliebig viele Vorhersagen im GTF-Format angegeben werden[5]. Jedoch ist zu beachten, dass bei der Angabe von mehreren Dateien das System eventuell überlastet wird. Auf dem verwendeten Rechner führte beispielsweise die Übergabe von zu vielen Vorhersagedateien dazu, dass der Prozess vorzeitig beendet wurde, da nicht ausreichend Arbeitsspeicher verfügbar war.

4.1.2 Auswertung

ERGEBNISSE MIT NEUEN PARAMETERN NOCH NICHT FERTIG

4.2 Caenorhabditis elegans

Das Training und die Vorhersage für *Caenorhabditis elegans* wurden für die Chromosomen I, II, III, IV, V und X durchgeführt. Die Genomsequenz kommt von www.wormbase.org und liegt ebenfalls im FASTA-Format vor. Für die DNA-Sequenzen und die Referenzannotation wurden jeweils die Version WS240 von 2014 benutzt.

4.2.1 Auswertung

ERGEBNISSE MIT NEUEN PARAMETERN NOCH NICHT FERTIG

4.3 Drosophila melanogaster

Für *Drosophila melanogaster* stammen die Daten von flybase.org. Die DNA-Sequenz ist erneut im FASTA-Format. Verwendet wurden für die Genomsequenzen und die Referenzannotation die Version R5.55 von 2014. Zum Trainieren und für die Vorhersage wurden nur die euchromatinen Regionen 2L, 2R, 3L, 3R, X und 4 genutzt.

4.3.1 Auswertung

ERGEBNISSE MIT NEUEN PARAMETERN NOCH NICHT FERTIG

4.4 Schizosaccharomyces pombe

Bei *Schizosaccharomyces pombe* wurde das Training und die Vorhersage für die Chromosomen I, II und III durchgeführt. Die dafür nötige Genomsequenz und Referenzannotation können hier heruntergeladen werden: ensemblgenomes.org. Benutzt wurden die Versionen ASM294v2.23 von 2014.

4.4.1 Auswertung

ERGEBNISSE MIT NEUEN PARAMETERN NOCH NICHT FERTIG
ungefähres vorläufiges Beispiel:

```
**Summary Stats** s.pombe
Annotation:    all.fixed.gtf
Predictions:           AUGUSTUS ab initio    GeneMark-ET    BRAKER1
Gene Sensitivity               78.02%        79.99%         81.73%
Gene Specificity               84.59%        84.93%         85.78%
Transcript Sensitivity         78.01%        79.98%         81.71%
Transcript Specificity         84.59%        84.93%         85.78%
Exon Sensitivity               85.19%        85.18%         88.21%
```

```
Exon Specificity                89.09%          89.03%          89.86%
Nucleotide Sensitivity          98.26%          98.88%          99.01%
Nucleotide Specificity          99.57%          99.45%          99.58%
**General Stats**
Predictions:
             all.fixed.gtf   AUGUSTUS ab initio   GeneMark-ET   BRAKER1
Gene
    All
        Count        5123.00       4725.00         4825.00         ~4800
```

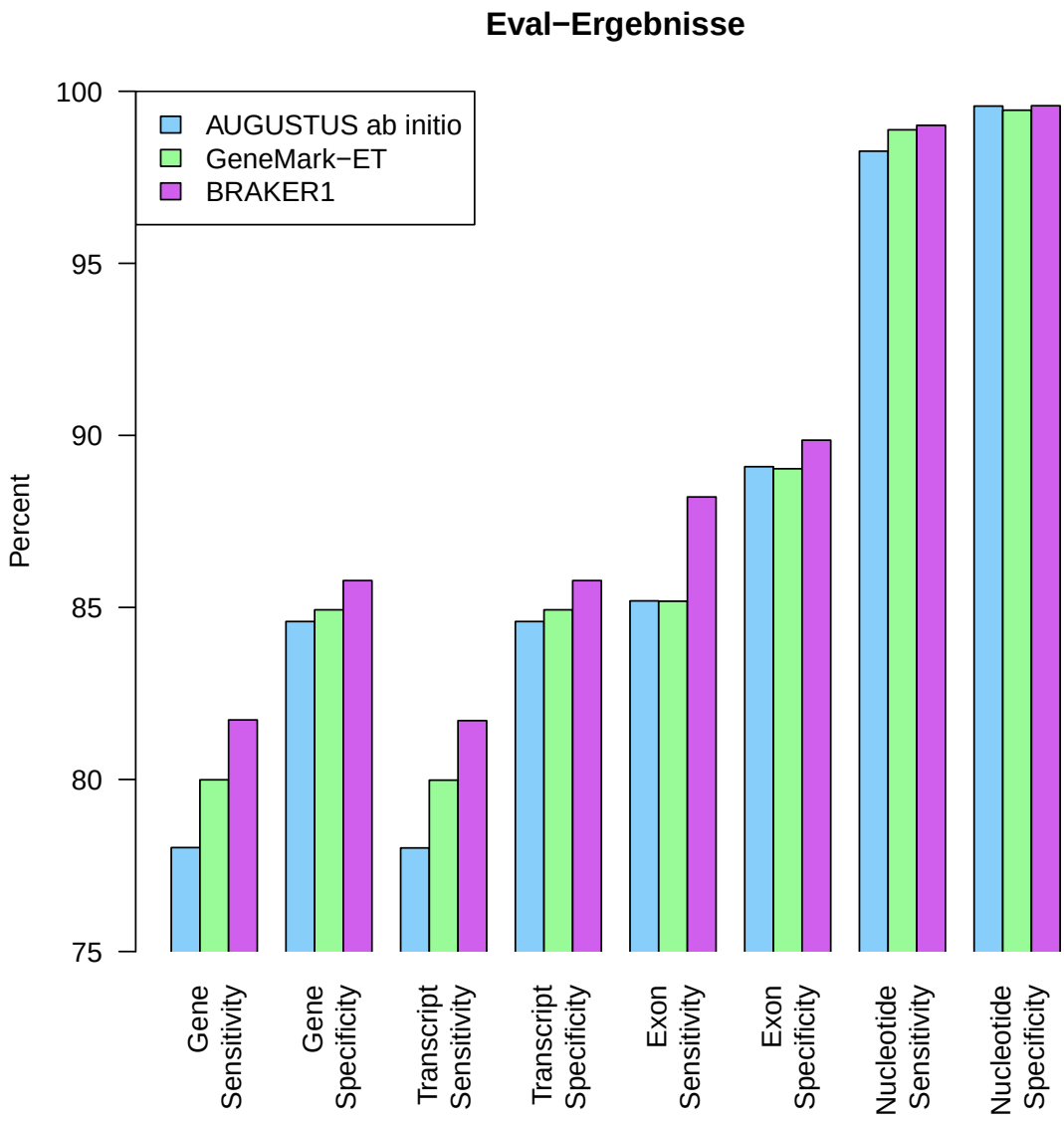

Abb. 4.1: Eval-Ergebnisse für *Schizosaccharomyces pombe*

5. Zusammenfassung

A DVD-Verzeichnis

- augustus-3.0.3.tar.gz
- BRAKER1.tar.gz
  - braker.pl ..... Pipeline, die *GeneMark-ET* und *AUGUSTUS* ausführt
  - filterGenemark.pl . filtert *GeneMark-ET* Ausgabe in gute und schlechte Gene
  - filterIntronsFindStrand.pl . findet für Intron-Hinweise jeweiligen Strang und setzt den Score für *GeneMark-ET*
  - licence.txt
  - README.braker
- examples
  - Arabidopsis thaliana
    - augustus.gff ..... Genvorhersage von *AUGUSTUS* im GFF-Format
    - braker.log . enthält die verwendeten Shell-Befehle und den Zeitpunkt, zu dem sie gestartet wurden
    - genbank.gb .. enthält Informationen zu allen von *GeneMark-ET* vorhergesagten Genen im Genbank-Format
    - genbank.good.gb..... enthält Informationen zu allen von *GeneMark-ET* vorhergesagten guten Genen
    - genbank.good.gb.test..... nur, falls genbank.good.gb mehr als 1000 Gene enthält
    - genbank.good.gb.train..... nur, falls genbank.good.gb mehr als 1000 Gene enthält
    - GeneMark-ET
      - genemark.average_gene_length.out ..... durchschnittliche Länge der kompletten Gene
      - genemark.f.bad.gtf .....gefilterte *GeneMark-ET* Ausgabe, nur schlechte Gene
      - genemark.f.good.gtf gefilterte *GeneMark-ET* Ausgabe, nur gute Gene
      - genemark.gtf.....Genvorhersage von *GeneMark-ET* im GTF-Format
    - GeneMark-ET.stdout .....Standardausgabe von *GeneMark-ET*
    - genome.fa ..... Genom im fasta Format
    - hintsfile.gff ..... enthält Intron-Hinweise, die mit *bam2hints* [14] aus der BAM-Datei gewonnen und mit *filterIntronsFindStrand.pl* für *GeneMark-ET* kompatibel gemacht wurden
    - species
      - A.thaliana..... Ordner enthält die Parameter-Dateien
  - Caenorhabditis elegans ..... analog zu Arabidopsis thaliana
  - Drosophila melanogaster ..... analog zu Arabidopsis thaliana
  - Schizosaccharomyces pombe ..... analog zu Arabidopsis thaliana
- gm_et_linux_64.tar.gz .....*GeneMark-ET* Version 4.21 für 64bit Systeme
- gm_key_32.tar.gz ..... *GeneMark-ET* Schlüssel für 64bit Systeme
- RNA-Seq-basierte_strukturelle_Genomannotation_basierend_auf_unüberwachtem_Training.pdf ..... PDF-Version dieser Arbeit

Literaturverzeichnis

[1] Alexandre Lomsadze, Paul D. Burns, und Mark Borodovsky. Integration of mapped RNA-Seq reads into automatic training of eukaryotic gene finding algorithm. *Nucleic Acids Research*, 42(15):e119, 2014. URL http://nar.oxfordjournals.org/content/42/15/e119.abstract.

[2] Alexandre Lomsadze, Paul D. Burns, und Mark Borodovsky. Integration of RNA-Seq Data into Eukaryotic Gene Finding Algorithm with Semi-Supervised Training, 2014.

[3] Alexandre Lomsadze, Vardges Ter-Hovhannisyan, Yury O. Chernoff, und Mark Borodovsky. Gene identification in novel eukaryotic genomes by self-training algorithm. *Nucleic Acids Research*, 33(20):6494–6506, 2005. URL http://nar.oxfordjournals.org/content/33/20/6494.abstract.

[4] Erik Pettersson, Joakim Lundeberg, und Afshin Ahmadian. Generations of sequencing technologies. *Genomics*, 93(2):105 – 111, 2009. URL http://www.sciencedirect.com/science/article/pii/S0888754308002498.

[5] Evan Keibler. Eval: A Gene Set Comparison System. URL http://mblab.wustl.edu/media/software/eval-documentation.pdf.

[6] Evan Keibler und Michael R. Brent. Eval: A software package for analysis of genome annotations. *BMC Bioinformatics*, 4(1):50, 2003. URL http://www.biomedcentral.com/1471-2105/4/50.

[7] Josep F. Abril, Robert Castelo, und Roderic Guigó. Comparison of splice sites in mammals and chicken. *Genome Research*, 15(1):111–119, 2005. URL http://genome.cshlp.org/content/15/1/111.abstract.

[8] Katharina Hoff und Mario Stanke. Current Methods for Automatic Structural Genome Annotation, 2014.

[9] W. James Kent. Genome browser at UCSC. URL http://genome.ucsc.edu.

[10] Mario Stanke und Burkhard Morgenstern. AUGUSTUS: a web server for gene prediction in eukaryotes that allows user-defined constraints. *Nucleic Acids Research*, 33(suppl 2):W465–W467, 2005. URL http://nar.oxfordjournals.org/content/33/suppl_2/W465.abstract.

[11] Mario Stanke und Katharina Hoff. Unsupervised and Protein-Family-Based Training of Augustus. *Master Thesis Outline*, 2014.

[12] Mario Stanke, Oliver Keller, Irfan Gunduz, Alec Hayes, Stephan Waack, und Burkhard Morgenstern. AUGUSTUS: ab initio prediction of alternative transcripts. *Nucleic Acids*

Research, 34(suppl 2):W435–W439, 2006. URL http://nar.oxfordjournals.org/content/34/suppl_2/W435.abstract.

[13] Mario Stanke, Oliver Schoffmann, Burkhard Morgenstern, und Stephan Waack. Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. *BMC Bioinformatics*, 7(1):62, 2006. URL http://www.biomedcentral.com/1471-2105/7/62.

[14] Mario Stanke und Stephan Waack. Gene Prediction with a Hidden-Markov Model and a new Intron Submodel. *Bioinformatics*, 19(suppl 2):ii215–ii225, 2003. URL http://bioinformatics.oxfordjournals.org/content/19/suppl_2/ii215.abstract.

[15] Mario Stanke, Stephan Waack, Oliver Keller, und Martin Kollmar. A novel hybrid gene prediction method employing protein multiple sequence alignments. *Bioinformatics*, 2011. URL http://bioinformatics.oxfordjournals.org/content/early/2011/01/06/bioinformatics.btr010.abstract.

[16] Mario Stanke. Incorporating RNA-Seq into AUGUSTUS. URL http://augustus.gobics.de/binaries/readme.rnaseq.html.