BERTective: ein Open-Source-Author-Profiling-Tool

Konrad Brüggemann, Friedrich Kierbaum
unter Aufsicht von Brielen Madureira, M.Sc.

BND Summer of Code 2023
Abschlussbericht

Die Programmierphase des BND Summer of Code 2023 ist vorbei und hinter uns liegen drei Monate unterhaltsamer Trainingsdaten, verzweifelter Debugging-Sessions, spannender neuer Erkenntnisse in den Bereichen Software Development, Datenbeschaffung sowie -aufbereitung, Modelltraining und Text-Analyse, aber auch lehrreicher Erfahrungen in Projektstrukturierung und Teamkommunikation. Wir möchten in diesem Bericht rückblickend auf unsere ursprüngliche Vision eingehen, inwiefern wir dieser gerecht werden konnten, wie zufriedenstellend das entstandene Produkt geworden und wie benutzerfreundlich unser Code dabei ist. Dafür geben wir auf den folgenden Seiten einen umfangreichen Einblick in den gesamten Ablauf des Projekts, verschiedene Evaluationsergebnisse und technische Hürden auf dem Weg.

Die Idee

Im Rahmen des Summer of Code haben wir es uns zur Aufgabe gemacht, ein robustes Programm zu entwickeln, mit dem sich verlässliche Vorhersagen über demographische Eigenschaften des Urhebers eines beliebigen deutschen Textes treffen lassen. Die Eigenschaften, die von diesem Tool vorhergesagt werden können, sind das Geschlecht und das Alter des Autors sowie dessen Herkunft anhand seines Regiolekts und sein Bildungsgrad. Um diese Vorhersagen zu treffen, sammeln wir Texte aus unterschiedlichen Quellen und implementieren komplexe computerlinguistische Methoden für das Feature Engineering. Entschieden haben wir uns für vier Features, welche die Qualität des Textes im Hinblick auf Rechtschreibung und Orthographie ermitteln, die Verwendung von Fachjargons aus verschiedenen Bereichen darstellen, mithilfe des ZDL-Regionalkorpus[1] den Regiolekt des Autors eruieren und durch verschiedene mathematisch berechenbare Maße die lexikalische Diversität und andere linguistische Eigenschaften repräsentieren. Mithilfe der entstandenen Features werden durch Deep Learning im Anschluss Modelle trainiert, die mit hoher Genauigkeit die festgelegten Autor-Eigenschaften vorhersagen können. Der Anwender des Tools hat nun die Möglichkeit, bereits von uns trainierte Modelle herunterzuladen und diese für die Analyse eigener Texte zu verwenden, aber durch den Open-Source-Rahmen auch weiterhin die komplette Freiheit über Skript-Anpassung und Training eigener, neuer Modelle. Es können ganz neue Dimensionen signifikanter Sprachfeatures entdeckt und integriert, die existierenden Herangehensweisen erweitert oder etwa Textart-spezifisch einzelne Methoden ersetzt werden.

Das Programm

Die Implementierung des Programms wurde inkrementell effizienter, benutzerfreundlicher und modularer. Während es zum Zeitpunkt des Zwischenberichts nur ein Dutzend verschiedener, experimenteller Skripte gab, existiert nun eine zentrale Datei, mit welcher der Nutzer jede Funktion des Programms – vom Download der Datensätze über Training und Evaluierung verschiedenster Modelle bis hin zur Inference – bequem über die Command-Line ansteuern kann. Der zeitaufwändigste und somit letztendlich umständlichste Schritt ist hierbei der Download der Datensätze. Denn obwohl alle Daten, die wir für das Training und das Feature Engineering verwenden, frei zugänglich sind, können wir sie nicht einfach redistributieren. Das bedeutet, dass jeder Nutzer zuallererst sowohl die benötigten Blog-Beiträge als auch das Gutenberg-Korpus und die Reddit-Posts mithilfe unserer Scraping-Tools herunterladen muss. Je nach Internetverbindung und Rechenleistung des Computers kann dieser Schritt durchaus mehrere Stunden in Anspruch nehmen. Sobald alle Datensätze heruntergeladen sind, können diese auch fast sofort für das

[1] https://www.dwds.de/d/korpora/regional

Training verwendet werden. Zuerst müssen die Texte jedoch in numerische, für eine Maschine interpretierbare Features beziehungsweise Vektoren umgewandelt werden. Für diesen Schritt kann sich der Nutzer entweder für eines unserer vier Features entscheiden oder ihre Synergie ausnutzen und in einem etwas rechenaufwändigeren Schritt alle Features zusammenwirken lassen. So können die besten Ergebnisse erzielt werden. Nach dem Training können die Modelle evaluiert werden. Denn nur 80 % der verfügbaren Daten werden für das Training verwendet, mithilfe der restlichen 20 % wird die jeweilige Performance der Modelle getestet. Diese Herangehensweise ist im Bereich des Maschinellen Lernens typisch und wichtig, denn ein Modell, dessen Qualität nicht bekannt ist, ist praktisch unbrauchbar.
Mithilfe verschiedener Einheiten, auf welche im Abschnitt *Evaluierung des Modells* genauer eingegangen wird, kann die Qualität der Vorhersagen des Modells also auf diesen Testdaten berechnet werden. Nach erfolgreicher Evaluierung kann das Modell anschließend für Inference, also das Vorhersagen von Autor-Eigenschaften auf einzelnen ungesehenen Texten, genutzt werden. So kann ein Text in Form einer txt-Datei an das Modell übergeben werden, und es wird ein komplettes Autorprofil berechnet. Ab dem Abschnitt *Regiolektvorhersage mit dem ZDL-Regionalkorpus* werden die Features und Evaluierungsmethoden genauer erläutert.

Technische Hürden

Selbstverständlich lief bei unserer Arbeit an diesem Projekt nicht alles nach Plan. So stießen wir schon in der ersten Woche der Programmierphase auf das erste (und insgesamt weiterhin größte) Problem: die Knappheit an verwendbaren hochqualitativen Daten. Um ein maschinelles Modell zu trainieren, benötigt man Daten – und zwar möglichst viele und möglichst hochwertige. In unserem Fall sind das Texte, über deren Autoren möglichst viel bekannt ist, idealerweise ihr Alter zum Zeitpunkt des Verfassens des Textes sowie Geschlecht, Bildungsgrad und Herkunft. Es gibt keinen fertigen deutschen Datensatz, der diesen Anforderungen entspricht, also mussten wir selbst auf die Suche nach geeigneten Daten gehen. Gefunden haben wir schließlich drei Hauptquellen: Reddit[2], den deutschen Blog *Die Achse des Guten*[3] und das Gutenberg-Korpus[4]. Während man sich bei den Gutenberg- und den Achgut-Daten sehr sicher sein kann, dass die Informationen über die Autoren korrekt sind, ist dies bei Reddit etwas schwieriger. Posts, in denen der Autor freiwillig Alter und Geschlecht angibt, gibt es viele – zum Beispiel in *r/DatingDE* oder *r/Beziehungen* – diese haben wir mithilfe von Regex-Patterns annotiert und die Angaben anschließend aus den Texten entfernt. Posts, in denen der Autor angibt, woher er kommt, gibt es jedoch wenige, und auf Grund der hohen linguistischen Varianz, die Herkunftsangaben haben können, lassen sich diese schwer mit Regex-Patterns erkennen. Deshalb haben wir eine gewagte Entscheidung getroffen: Wir gehen davon aus, dass in Subreddits, die speziell für deutsche Städte erstellt wurden (z. B. *r/Heidelberg*), hauptsächlich Bewohner dieser Städte miteinander kommunizieren. So haben wir eine große Liste an Städte-Subreddits erstellt, Posts aus diesen Subreddits heruntergeladen und anschließend dem Label des Areals, in dem sich die jeweiligen Städte befinden, zugewiesen. Es ist unklar, ob dies die Genauigkeit der Vorhersagen erhöht oder gesenkt hat, da wir kaum Referenzdaten haben. Nur von achgut.com erhalten wir weitere Texte, bei denen die Herkunft des Autors problemlos annotiert werden kann. Doch eine signifikante Menge an Texten erhalten wir von dieser Quelle nicht, insgesamt nur etwa 4000, was für ein robustes Modell leider nicht ausreicht. Die Vorhersage des Regiolekts ist demzufolge für das Modell am schwierigsten und die Ergebnisse nur mittelmäßig.

Wie im Zwischenbericht bereits angedeutet, war auch die manuelle Annotation der vorhandenen Daten nicht trivial. Eine kostenlose Lizenz für Prodigy, das Annotationstool unserer Wahl, haben wir zügig und unbürokratisch erhalten, was uns sehr begeistert. Prodigy liefert direkt diverse sogenannte Recipes (fertige Konfigurationen für unterschiedliche Annotationstasks) und ist stark anpassbar, lässt sich aber auch mit Kenntnissen von Python, auf dem es basiert, nicht im Handumdrehen komplett durchschauen. Der Plan war, ein Start-Batch an Annotationen an das große Sprachmodell BERT zu liefern, das dann entsprechend lernen und die Annotation fortführen sollte, damit auf zahlreichen annotierten Textdaten das Detective-Modell trainiert werden kann. Für so ein Vorhaben braucht man wasserdichte signifikante Labels, einen guten, diversen Grunddatensatz und genug Zeit für eine ausreichende Menge an Annotationen. Doch da das Gesamtkonzept der manuellen linguistischen Annotation erst nach und nach erarbeitet werden musste, lief hier nicht alles wie in unserer Projektskizze gewünscht. Es wurden gemeinsam Labels festgelegt und eines der bereits fertigen Recipes für den Annotationsprozess ausgewählt. Auf dieser Grundlage haben wir einen internen Annotations-Guide implementiert, um sicherzustellen,

[2]https://www.reddit.com/
[3]https://www.achgut.com/
[4]https://www.projekt-gutenberg.org/info/texte/allworka.html

dass die Arbeit konsistent vonstattengeht. Im Verlauf des Projekts konnten wir so bis dato tausend Sätze mit den Labels *Interpunktionsfehler, Rechtschreibfehler, Umgangssprache, Anglizismus, fehlendes Wort, Jargon, gehobene Sprache, emotionale Verstärkung, ungewöhnliche Wortreihenfolge, redundantes Wort* annotieren. Der Fortschritt erwies sich für unseren Anwendungsfall schon Ende August als alles andere als ausreichend für eine zuverlässige Abgabe des Tasks an BERT, als hier zwar langsam besser werdende, aber viel zu unsichere Ergebnisse erzielt wurden, weshalb die bisher erreichten Annotationen momentan nicht in die letztendliche Pipeline eingebunden sind. Wir gehen stark davon aus, dass BERT von noch mehr manuell annotierten Sätzen profitieren würde, jedoch erwiesen sich manche der festgelegten Labels als relativ subjektiv und schwer zu definieren, was bei menschlicher Datenannotation ein bekanntes Risiko und natürlich unvorteilhaft für Modelltraining ist, und nicht zuletzt die Zeit als knapp. Außerdem schließen wir inzwischen nicht aus, dass einige Labels nur insignifikant zur Performance beitragen, dies kann allerdings, wie an anderer Stelle bereits erwähnt, nur intuitiv eingeschätzt und kaum kontrolliert werden. In Zukunft können wir durch das Gelernte unsere Ressourcen effizienter nutzen und durch Auswahl besserer Labels die Qualität der Annotationsarbeit steigern. Wir planen, die manuelle Annotation in Variationen fortzusetzen, und sind guter Dinge, dass BERT früher oder später hilfreiche Text-Labels auf Worte und Phrasen anwenden kann.
Eine weitere schwierige Aufgabe war es, dem Modell beizubringen, Entscheidungen nicht auf Grund semantischer Differenzen, sondern tiefgründigerer, linguistisch komplexerer Eigenschaften, zu treffen. Die Vektoren von Sprachmodellen wie BERT enthalten hauptsächlich semantische Informationen, sie sind darauf ausgelegt, sich zu ähneln, also im Vektorraum näher beieinander zu liegen, wenn die repräsentierten Wörter ähnliche Bedeutungen haben. So neigt ein Modell schnell dazu, Vorhersagen basierend auf der Identifizierung eines Themas und nicht einer linguistischen Analyse zu treffen. Wir brauchen also zum einen Vektoren, die neben reinen semantischen Informationen auch sprachstilistische Informationen enthalten, und zum anderen einen Datensatz, der reich an verschiedenen Themen, Textarten und Autoren ist. Denn die Wörter „Schrippe“ und „Semmel“ haben quasi dieselbe Bedeutung und somit fast identische BERT-Vektoren, doch könnten ihre Verwendungsgebiete kaum unterschiedlicher sein. Um dieses Problem zu lösen, haben wir uns für die Verwendung des ZDL-Regionalkorpus entschieden, auf welche wir im Folgenden detailliert eingehen.

Regiolektvorhersage mit dem ZDL-Regionalkorpus

Zu den Features, von denen wir uns am meisten erhofft haben, gehörten von Anfang an die Regiolekt-Vektoren, die wir mithilfe des ZDL-Regionalkorpus generieren. Das Korpus ermöglicht es, für ein beliebiges Wort die Verwendungshäufigkeit in sechs Teil-Arealen Deutschlands einzusehen. Diese sechs Werte speichern wir in einem sechs-dimensionalen Vektor. Für jeden Text generieren wir also so viele Vektoren, wie es Wörter in dem Text gibt. Anschließend werden diese in einem Array gespeichert. So wird jeder Text repräsentiert durch ein Array aus Vektoren, von denen wir uns erhofft haben, dass sie präzise den Regiolekt des Autoren wiedergeben würden. Jedoch scheint leider die Aussagekraft dieses Features begrenzt zu sein und nur in circa 50 % der Fälle zu einer korrekten Vorhersage zu führen.
Währenddessen liefern andere Features, welche nicht speziell für die Regiolektberechnung implementiert wurden, teils bessere Ergebnisse. Wir können nicht genau sagen, woran dies liegt, jedoch vermuten wir, dass gerade im Internet die verwendete Sprache immer einheitlicher wird und die Menge an Worten, die einen starken Bezug zu einer bestimmten Region haben, im Sprachgebrauch abnimmt. Die Kraft der regionalen Unterscheidung von Dialekten fällt womöglich hinter die der Soziolekte und Register zurück. Des Weiteren enthält nicht jeder Text von einem nordostdeutschen Autor das Wort „Schrippe“, die große Mehrheit der verwendeten Wörter sind regional ungebunden und werden von Autoren aus allen Arealen Deutschlands gleichermaßen verwendet. Die naive Lösung für dieses Problem wäre eine klassische Stoppwortbereinigung, also das Entfernen aller Wörter, die keine eigene lexikalische Bedeutung zum Text beitragen, im Deutschen zum Beispiel Artikel und Konjunktionen. Jedoch sind nicht nur Stoppwörter regional unabhängig, sondern auch der Großteil aller bedeutungstragenden Wörter. Substantive wie Haus, Fahrrad oder Schule müssten dann ebenfalls entfernt werden. Zum einen ist es schwierig, eine solche Wortliste zuverlässig und neutral zu kreieren, zum anderen würden in den meisten Texten dann nur wenige oder gar keine Wörter mehr übrig bleiben, was eine präzise Vektorrepräsentation stark erschwert. Dieser Ansatz ist somit nicht zielführend. Wir haben uns also für eine etwas komplexere Methode entschieden, welche akzeptiert, dass einzelne Wörter keine große regionale Gebundenheit haben, aber davon ausgeht, dass die Reihenfolge, in welcher diese Wörter auftreten beziehungsweise die Muster, die diese bilden, etwas aussagekräftiger sind. Deshalb haben wir uns dazu entschieden, keine Wörter aus den Texten zu entfernen, bevor diese in ihre ZDL-Vektorrepräsentation umgewandelt werden, und stattdessen ein geordnetes Array zurückzugeben, welches die Reihenfolge des originalen Textes beibehält. Mit diesen Vektoren

wird anschließend ein Recurrent Neural Network (RNN) trainiert, welches in seiner Architektur an erster Stelle hinter dem Input-Layer einen Long-Short-Term-Memory-Layer (LSTM-Layer) enthält. Diese Art von Neural Network eignet sich für die Verarbeitung von sequenziellen Daten, wie Aktienkursen oder eben natürlichsprachlichen Texten, da diese neben dem tatsächlichen Wert des jeweiligen Datenpunkts auch dessen Position in der Sequenz in der Berechnung der Vorhersage beachten. Mit dieser Methode konnten die erzielten F1-Werte im Vergleich zu denen, die ein klassisches Convolutional Neural Network (CNN) mit von Stoppwörtern bereinigten Vektoren erzielen konnte, um knapp 20 Prozentpunkte übertroffen werden.

Textqualität mit Orthographie- und Fehleranalyse

Ein Feature, welches sich als sehr aussagekräftig erwies, sind die Analyse der verwendeten Orthographie-Version und eine oberflächliche Fehlersuche. Dem zugrunde liegt die Hypothese, dass genau diese Faktoren bereits verhältnismäßig sauber nach Alter (Version der Rechtschreibregeln) und – vereinfacht – Bildungsgrad (verbreitete Fehlschreibweisen) filtern. Für beides haben wir Ressourcen der Website *korrekturen*[5] verwendet. Die Website enthält unter diesem Link eine Liste sogenannter beliebter Fehler sowie ihrer entsprechenden korrekten Schreibweise. Wir haben diese Standardversionen und deren häufige Abweichungen gecrawlt und in Python-Listen gespeichert. Dasselbe gilt für die Wortliste zur neuen Rechtschreibung mit entsprechenden Verweisen auf veraltete Rechtschreibungen, welche unter diesem Link abrufbar ist.
Wir erhalten also insgesamt fünf Wortlisten, die je circa tausend kurze deutsche Phrasen enthalten und bestimmte grammatische Auffälligkeiten widerspiegeln. Auf den Texten wird anschließend ein String-Matching mit diesen Listen durchgeführt; das bedeutet, dass überprüft wird, ob die N-Gramme aus den Listen in den jeweiligen Texten auftreten. Ist dies der Fall, wird das jeweilige N-Gramm mithilfe der tok2vec-Methode von spaCy in einen 96-dimensionalen Vektor umgewandelt, welcher es akkurat repräsentiert. Treten in einem Text mehrere Matches mit einer Liste auf, wird die Summe der Vektoren gebildet und für den weiteren Verlauf verwendet. Ein Beispiel ist hier zu sehen:

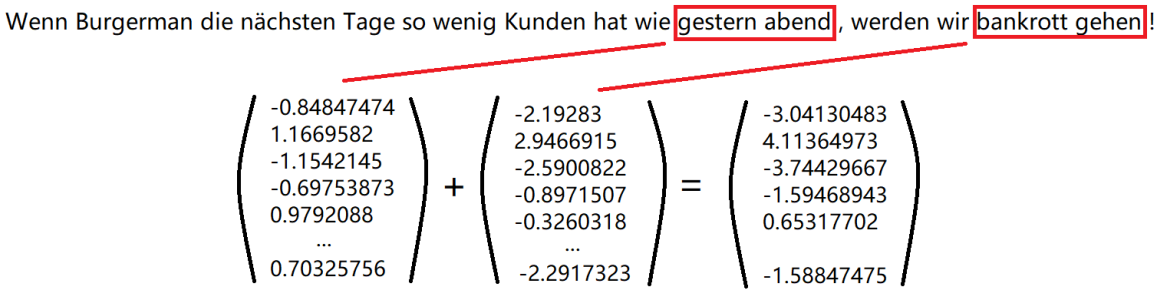


Abbildung 1: Orthographievektorenberechnung eines Beispielsatzes

So grenzen sich Texte mit vielen Fehlern in der Vektorrepräsentation klar von fehlerfreien Texten ab. Die Vektoren, die basierend auf den fünf Listen generiert werden, werden anschließend in einem Array gespeichert, welches schließlich fünf 96-dimensionale Vektoren enthält. Es kann also als ein Tensor der Größe 5×96 interpretiert werden. So kann jeder Text durch einen solchen Tensor repräsentiert und ein CNN-Modell bequem mit wiederum diesen trainiert werden. Ursprünglich haben wir dieses Feature hauptsächlich dafür implementiert, Vorhersagen über den Bildungsgrad eines Autors zu treffen. Jedoch haben wir beim Experimentieren festgestellt, dass es sich auch für die Vorhersage anderer Features sehr gut eignet und somit generell eine hohe Aussagekraft hat.

[5] `https://www.korrekturen.de/`

Erkennung von Fach- und Fremdsprach-Einflüssen mit Wiktionary-Listen

Mithilfe ausgewählter Wortlisten des Online-Wörterbuchs Wiktionary[6] versuchen wir, verschiedene Fachjargons und allgemeine sprachliche Besonderheiten in den Texten zu quantifizieren und durch ein Array zu repräsentieren. Insgesamt haben wir 27 Wortlisten gecrawlt und führen mit diesen ebenfalls String-Matching auf den Texten durch. Anhand von Ausmaß, Sauberkeit und linguistischer Verarbeitbarkeit der jeweiligen Liste beziehungsweise ihrer Einträge haben wir die folgenden Seiten für nützlich befunden: *Abkürzungen im Internet*, *Anglizismen*, *Arabismen*, *Aufwertung*, *Binomiale*, *Fachwortliste*, *Chinesische Wörter in der deutschen Sprache*, *DDR-Sprache*, *Gallizismen*, *Geowissenschaften*, *Geschichte*, *Hispanismen*, *Japanische Wörter in der deutschen Sprache*, *Jägersprache*, *Netzjargon*, *Pharmazie*, *Philosophie*, *Politik*, *Rechtswissenschaften*, *Religion*, *Romanismen*, *Slawismen*, *Weinbau*, *Wirtschaft*, *Informationstechnik*, *Philatelie* und *Sprichwörter*. Wir fangen also unterschiedliche Fremd- und Lehnwörter, diverse Fachtermini, berufsspezifische Register und ähnliche Varietäten ab, alles auf Basis der allgemeinen Hypothese, dass jede auf diese Weise erreichte Gruppierung der Texte im Modell die für unsere Vorhersagen relevanten Gemeinsamkeiten betont. Gibt es in den Trainingsdaten beispielsweise eine Korrelation zwischen Regiolekt und Nutzung „DDR-sprachlicher“ Begriffe, zwischen Bildungsgrad und exzessiver Verwendung von Gallizismen und Sinismen oder zwischen Alter und besonderer Häufung von Anglizismen, so wird dies als Quasi-Kausalität erlernt. (Hier sei sicherheitshalber ergänzt, dass Korrelation zwar in der Regel ausdrücklich nicht gleich Kausalität bedeutet, die beiden sich jedoch mit zunehmender Stichprobengröße ähnlicher werden, was deshalb beim Training von Vorhersagemodellen ein zugrunde liegendes Prinzip ist.) Für jede Liste wird die Anzahl an Matches im jeweiligen Text in einer Array-Zelle gespeichert. Ein Text wird also durch ein Array der Länge 27 repräsentiert. Diese Methode ist nicht nur zeitlich effizient, sondern liefert auch gute Ergebnisse. Gerade für die Vorhersage von Alter und Bildungsgrad eignet sich dieses Feature sehr und erreicht F1-Werte von bis zu 60 %. Doch trotz aller Erfolge bei der Vorhersage gibt es auch hier Luft nach oben, dies soll hier kurz reflektiert werden. Beim Schreiben dieses Berichts fiel auf, dass „Abkürzungen im Internet“ inzwischen sowohl durch Matching mit einer Wiktionary-Wortliste als auch durch eine Python-Methode im Rahmen der *simpleren Statistiken* gesammelt wurden. Dieses Beispiel illustriert jedoch gut die zeitweise starke Diskrepanz zwischen der primären Auslegung der Wiktionary-Listen für das Nachschlagen von Lemmata (zum Beispiel im Fall der Internet-Abkürzungen – Stand 30.09.2023 – nur ein Eintrag für „lol“, immerhin mit der Alternativschreibung „LOL“) und unserem Python-Skript, das „lol“, „lul“, „lel“ und „lül“ inklusive aller Groß-Klein-Schreibungs-Varianten findet. Am Beispiel „tldr“, das ähnlich häufig als „tl,dr“, „tl;dr“ oder „tl:dr“ auftreten kann, wird außerdem klar, wie viel weniger gepflegt diese Wiktionary-Listen sind als die stereotypisch gründliche deutsche Wikipedia. Das deutsche Wiktionary führt dieses Lemma unter keiner gebräuchlichen Form auf. Was Fachterminologie angeht, so sollte beim ersten Blick auf einige der entsprechenden Listen klar sein, dass sie bei Weitem nicht das gesamte Sondervokabular ihrer Domänen abbilden. Einerseits ist unter der Überschrift einer Liste von *Fach*-Termini (anders als bei einer zusammenfassenden Sammlung von bildungssprachlichen Begriffen) eine Begrenzung auf die *typischsten* Ausdrücke angemessen, um die semantische Abgrenzung zu Nachbarfächern zu wahren. Andererseits ist wohl die Wahrscheinlichkeit, dass ein bestimmter Begriff überhaupt in einer solchen Liste steht, etwa proportional zu seiner tatsächlichen Häufigkeit. Das bedeutet, dass wir bei Zuhilfenahme solcher Listen, die tendenziell die reale Nutzung der Begriffe abbilden, nur in den wenigen Fällen besonders tiefer fachlicher Spezialisierung Einbußen bei der richtigen Klassifizierung erleiden, auch wenn exhaustive Listen für reines String-Matching natürlich genau deshalb optimal wären.

Simple statistische Maße

BERTective soll es dem Anwender bei der Erstellung eines Autorenprofils ersparen, wertvolle Zeit mit mühsamer Handarbeit zu verschwenden. Bereits anhand einfacher statistischer Merkmale eines Textes können erste Einordnungen vorgenommen und die grundsätzliche Authentizität beurteilt werden. Darum wollten wir ursprünglich bei Eingabe eines zu analysierenden Dokuments einige statistische Features zum manuellen reflektierten Vergleich, beispielsweise mit Durchschnittswerten aus eigenen Korpora, ausgeben. Nach automatischer Annotation unseres Gesamtkorpus mit diesen Metadaten erwiesen sie sich jedoch als ausgezeichnete Trainingsfeatures, da sie mit einer Genauigkeit von 82 % das Geschlecht eines Autors vorhersagen konnten und auch bei der Vorhersage anderer Eigenschaften gute Ergebnisse lieferten. Wir legten uns darauf fest, sie lieber zum Training als nur zur zusätzlichen Information über neue Texte zu benutzen – beides gleichzeitig erschien uns gewissermaßen unehrlich, da ihr Effekt so möglicherweise doppelt gewirkt hätte (innerhalb der Modellvorhersage und bei der menschlichen Interpretation).

[6] `https://www.wiktionary.org/`

Mithilfe der Python-Bibliothek spaCy[7] sowie öffentlich zugänglicher Online-Listen werden verschiedene Rechnungen und Zählungen durchgeführt: Ein Klassiker der Literatur-Analyse ist die Type-Token-Ratio, ein Maß, das Schlüsse auf die lexikalische Diversität eines Dokuments ermöglicht. Dieses muss hier jedoch mit Vorsicht genossen werden, da das Ergebnis stark von der Textlänge abhängig ist. Ergänzt wird es durch die jeweilige durchschnittliche Länge von Wörtern und Sätzen im Dokument. Als spezifischere Komplexitätsmaße kommen die Durchschnittslängen von Substantiven, Verben, Adjektiven und Adverbien, der vier großen lexikalisch relevanten Wortarten, und deren jeweilige Häufigkeit pro Satz sowie Verhältnisse wie Artikel pro Substantiv hinzu. Einige besondere Gewohnheiten werden durch Maße wie Vokal-Konsonanten-Verhältnis und Großschreibungsrate abgefangen. Einfache Erfassung verwendeter Emoji[8], Emoticons und Chat-Akronyme ermöglichen Schlüsse auf mögliche Register, können zusammengenommen aber auch Vorhersagen übers Alter verbessern.

Evaluierung des Modells

Im Bereich des maschinellen Lernens gibt es einige Einheiten, die sich für die Evaluierung aufgrund ihrer domänenunabhängigen Aussagekraft mehr durchgesetzt haben als andere. So sind vor allem Precision und Recall sowie der sich aus diesen ergebende F1-Wert etablierte Maße für die Auswertung der Qualität eines Modells. Um diese Werte zu berechnen, wird ein Test-Datensatz benötigt, welcher die erwarteten beziehungsweise korrekten Labels für die enthaltenen Datenpunkte (in diesem Fall Texte und Autor-Eigenschaften) enthält. Anschließend sagt das Modell für jeden Text im Test-Datensatz ebenfalls die Eigenschaften des Autors vorher. So ergeben sich zwei Listen, von denen die erste die korrekten Labels und die zweite die vorhergesagten Labels enthält. Diese zwei Listen können nun in einer Konfusionsmatrix der Größe n^2 mit $n = \textit{Anzahl verschiedener Labels}$ dargestellt werden. Der Bildungsgrad enthält beispielsweise vier Labels und die Konfusionsmatrix, die mit den Vorhersagen des Modells entsteht, sieht so aus:

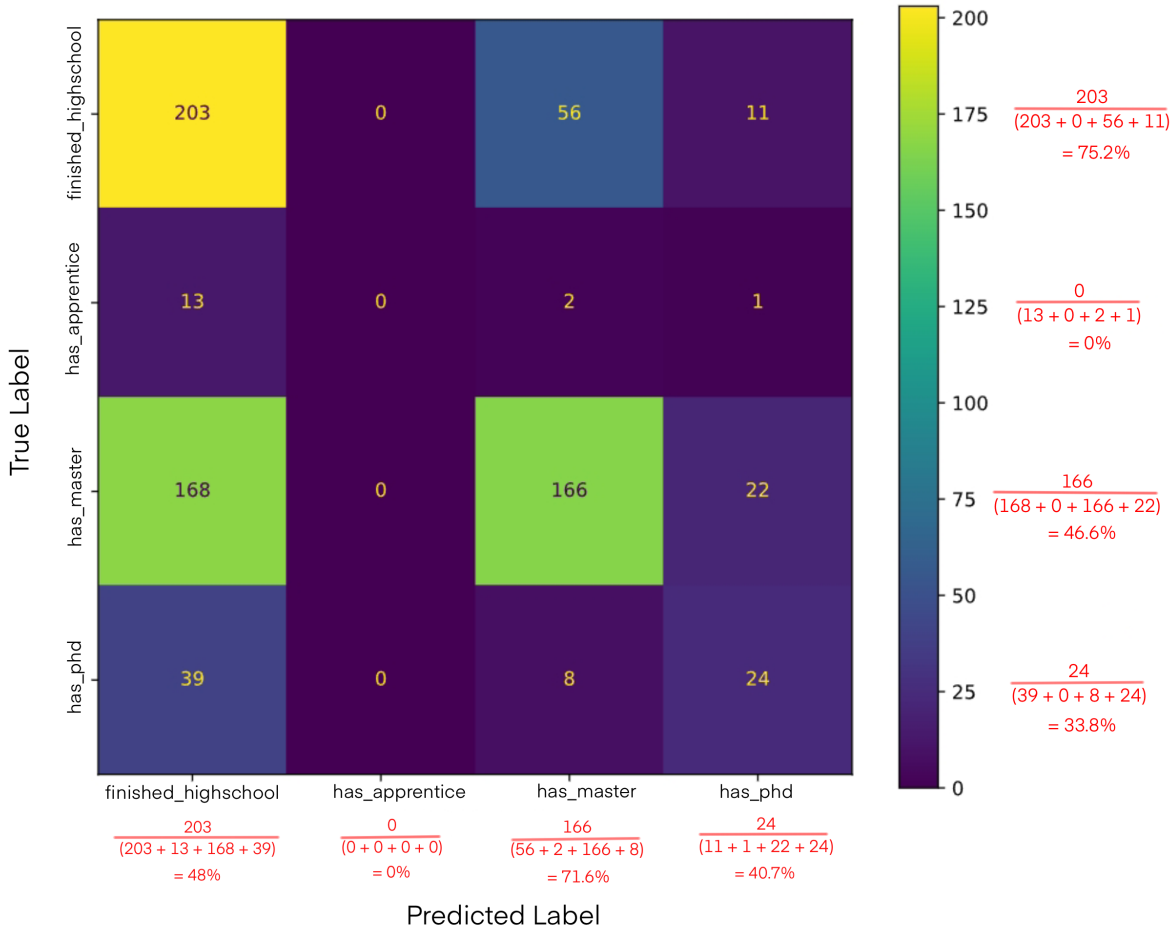


Abbildung 2: Konfusionsmatrix der Bildungsgradvorhersagen

[7] https://spacy.io/
[8] https://github.com/carpedm20/emoji/

Es fällt direkt auf, dass die Klasse *has_apprentice*[9] tatsächlich nie korrekt vorhergesagt wurde, was allerdings daran liegt, dass es kaum Trainingsdaten und demzufolge noch weniger Testdaten gibt. Die wenigen Daten, die wir im Datensatz haben, stammen von *r/Azubis*. Ebenfalls erkennbar ist, dass die beiden Klassen, die mehr Trainingsdaten haben – *finished_highschool*[10] und *has_master*[11] –, häufig verwechselt werden. Dies deutet darauf hin, dass die Aussagekraft der berechneten Features nicht so hoch ist wie erhofft. Am rechten Rand ist die Berechnung der Precision-Werte der einzelnen Klassen dargestellt und unter der Matrix die der Recall-Werte. Für beide Maße kann der Durchschnitt der Werte der einzelnen Klassen gebildet werden, um Macro-average-Precision beziehungsweise Macro-average-Recall zu erhalten. Ihr harmonisches Mittel ist der F1-Wert, welcher in diesem Fall 40 % beträgt.
Einschub: Während einer gemeinsamen Konsultation am Tag der geplanten Abgabe des Projekts ist uns ein Fehler in der Implementation des Orthographie-Features aufgefallen: Die Vektoren hätten summiert werden müssen, nicht deren Durchschnitt gebildet. Nach Behebung dieses Fehlers konnte ein F1-Wert von 61 % erreicht werden. Die Konfusionsmatrix ist also nicht aktuell – die wahren Ergebnisse sind besser.
Leider konnten wir mit unserem Modell trotzdem bei den anderen Vorhersagen nicht die Ergebnisse erreichen, die wir uns in der Projektskizze zum Ziel gesetzt hatten. So erreichten wir bei der Altersvorhersage lediglich einen F1-Wert von 67 %, obwohl wir großzügige Altersgruppen gewählt hatten[12]. Ein Random Classifier, also ein Modell, welches einfach zufällig eine der möglichen Altersgruppen vorhersagt, liegt bereits in 25 % der Fälle richtig, wurde also outperformt, aber zufriedenstellend ist dieses Ergebnis nicht.
Bei der Vorhersage des Geschlechts schnitt das Modell deutlich besser ab, hier konnte ein F1-Wert von bis zu 87 % erreicht werden. 61 % der Daten im Testset haben das Label *male*, also könnte ein Majority Classifier, welcher immer *male* vorhersagt, einen F1-Wert von 61 % erreichen. Wir sind mit der Performance zufrieden.
Enttäuscht wurden wir bei der Vorhersage des Regiolekts. Selbst mit Recurrent Neural Networks konnten wir F1-Werte von 50 % beim Testen nicht übertreffen. Gründe hierfür sind wahrscheinlich zum einen die geringe Qualität unserer Trainingsdaten, zum anderen die vermutlich für diese Art der Anwendung nicht optimalen Werte aus dem ZDL-Regionalkorpus. So entwickelten unsere Modelle stets einen Bias für die Vorhersage von *DE-NORTH-EAST* – trotz zumindest in der Hinsicht ausgeglichener Trainingsdaten. Nach etwas Nachforschung haben wir festgestellt, dass dieser Bias bereits im ZDL-Regionalkorpus existiert und *DE-NORTH-EAST* dort bei vielen eigentlich allgemein gebräuchlichen Wörtern deutlich höhere PPM-Werte hat als andere Areale. BERTective ist für die präzise Vorhersage des Regiolekts beziehungsweise der Herkunft eines Autors also leider nicht zu gebrauchen. Wahrscheinlich sollte dies, wie viele andere Teile dieses Projekts auch, ein eigenständiges Forschungsprojekt und erst dann Teil eines großen Modells wie BERTective werden. Für Geschlechts- und Altersvorhersagen kann es jedoch eine brauchbare Grundlage bieten. So haben wir das Modell auf eigenen und auf von Kommilitonen stammenden Texten getestet und stets korrekte Vorhersagen erhalten. Ein Beispiel ist auf dem Poster zu sehen.

Ideen für die Zukunft

Eine Idee, die wir gegen Ende der Programmierphase hatten und vor Abgabe leider nicht mehr rechtzeitig implementieren konnten, ist das Anpassen der Konfidenz-Werte in Abhängigkeit von der Label-Verteilung im Datensatz. Ein Deep-Learning-Modell, welches für eine Multiclass-Classification-Aufgabe verwendet wird, gibt typischerweise mithilfe einer Softmax-Funktion für jedes mögliche Label (zum Beispiel die verschiedenen Altersgruppen oder Regiolekte) eine Wahrscheinlichkeit zurück, welche man auch als Konfidenz interpretieren kann. Die Summe der Konfidenz-Werte ergibt 100 % und das Label mit dem höchsten Wert wird zurückgegeben. Wenn die Label-Verteilung in einem Datensatz jedoch nicht ausgeglichen ist, hat das Label mit der größten Anzahl an Trainingsdaten einen Bias und wird häufig zu Unrecht hohe Konfidenz-Werte bekommen. Durch Implementierung einer relativen Konfidenz-Schwelle oder anderen mathematischen Bedingungen würde es sich ermöglichen lassen, auch unterrepräsentierte Labels mit besserer Genauigkeit vorherzusagen. Des Weiteren halten wir es, wie im Zwischenbericht bereits erwähnt, für sinnvoll, zu erkunden, inwiefern fremdsprachige Trainingsdaten die Genauigkeit des Modells auf deutschen Texten verbessern könnten. Dazu sind wir bisher leider noch nicht gekommen.

[9] Autoren mit abgeschlossener Ausbildung
[10] Autoren mit Abitur
[11] Autoren mit abgeschlossenem Studium
[12] Spannen: 10–25, 26–40, 41–60, 61-90

Kommentar unserer Mentorin

Konrad Brüggemann and Friedrich Kierbaum führten das Projekt selbständig durch, mit viel Motivation und Eigeninitiative, um nach Ideen und Lösungen zu suchen, wann immer es nötig war. Während des Projekts wurden auf Wunsch der Studenten Treffen vereinbart, um offene Fragen zu besprechen. Die Themen betrafen vor allem Daten, Dokumentation, technische Aspekte der Implementierung, Designentscheidungen und Evaluation. Wir erörterten auch ethische Überlegungen und was sie tun könnten, um das Risiko einer so genannten "dual use" des Tools zu verringern. Ich gab aufgrund meiner Erfahrung Ratschläge, ließ ihnen aber im Allgemeinen den Spielraum, den Weg zu gehen, den sie für ihre Zwecke als geeigneter erachteten. Wie bei einem forschungsorientierten Projekt nicht anders zu erwarten, traten einige unerwartete Ereignisse ein, die Anpassungen an der ursprünglichen Idee erforderten (z. B. die Neudefinition der vorhergesagten Klassen angesichts der Verfügbarkeit von Daten). Ich glaube, dass der Summer of Code eine großartige Gelegenheit für Studierenden ist, mehr praktische Erfahrung mit dem Programmieren zu sammeln, indem sie ein voll funktionsfähiges Stück Software implementieren, und auch das Bewusstsein für Freie Software und Open Source zu erhöhen. Ich bin froh, dass ich als Mentorin dazu beitragen konnte.

Brielen Madureira (madureiralasota@uni-potsdam.de)

Übersetzt mit Unterstützung von www.DeepL.com/Translator (kostenlose Version)

Vielen Dank für Ihre Unterstützung und Ihr Vertrauen in unsere Idee. Wir fanden die letzten drei Monate sehr lehrreich und hatten viel Spaß an der gemeinsamen Arbeit.
`https://github.com/kobrue02/BERTective/`