Angular: Platform and Framework for Web Application Development

Jesús Moreno Durán
Fulda University of Applied Science
jesus.moreno-duran@informatik.hs-fulda.de

Contents

ABSTRACT

In this write-up, we will be discussing Angular, a framework and platform that enables the development of client-side single-page applications (SPA) through the use of TypeScript and HTML. Additionally, we will provide a hands-on demonstration of how to create a web application using Angular, specifically one that allows for car sharing and ride-sharing. The platform enables users to publish travel routes and fares, while others can join in. However, it's important to mention that the application's implementation is based on dummy data and can only be accessed through the frontend.

1. INTRODUCTION

Sharing cars has gained immense popularity in recent times due to its cost-effective and travel-efficient benefits. However, managing coordination between users can often be challenging. To tackle this issue, an Angular-based web application has been created that enables users to share rides and travel together seamlessly, thereby reducing traffic congestion and also promoting environmental sustainability by reducing the number of cars on the road.

1.1 What is Angular?

Angular is a powerful platform and open-source framework used for developing single-page applications (SPA). It was created and is maintained by Google, and provides a complete solution for building dynamic web applications using HTML, CSS, and TypeScript.

The framework is designed to simplify the development process by providing a structured approach for building complex client-side applications. Angular utilizes declarative templates and dependency injection to separate application logic from presentation, allowing developers to focus on the core business logic of their applications.

One of the key features of Angular is its ability to handle complex data models and provide a scalable architecture for building large applications. It also includes a robust set of built-in tools and libraries that simplify common tasks such as routing, forms handling, and testing.

Angular's popularity is due in part to its flexibility and ability to integrate with other tools and technologies, making it an ideal choice for a wide range of web development projects.

Angular was initially developed as a private project by Google, with the first version being released internally in 2009. However, it wasn't until September 2016 that the framework was officially made available to the public, with the release of Angular 2. This move to open-source made

Angular accessible to a wider community of developers, leading to its rapid growth and adoption as a leading framework for building single-page applications. Despite its transition to an open-source project, Angular continues to be supported and maintained by Google, ensuring that it remains a cutting-edge technology for years to come.

1.2 Features of Angular

- TypeScript: Angular is built using TypeScript, a statically typed superset of JavaScript that provides improved error checking and code maintainability.
- Components: Angular provides a powerful component based architecture that allows develop complex applications by composing simple components. This makes it easy to reuse and share code, and simplifies the development of large-scale applications.
- Dependency Injection: Angular uses dependency injection to manage object creation and simplify the development of complex applications.
- Declarative Templates: Angular utilizes declarative templates to separate application logic from presentation, making it easier to maintain and scale applications.
- Routing: Angular includes a powerful routing system that enables developers to build applications with multiple views and navigate between them.
- Progressive Web Apps (PWA): Angular provides support for building Progressive Web Apps, which can be installed on users' devices and function like native apps. PWAs built with Angular are fast, responsive, and can work offline, making them a powerful tool for engaging users and improving user experience.

1.3 How to create a project

In order to create a project in Angular you need to follow these steps:

1. Install a code editor. To edit the project, you need to have a code editor installed on your system. Visual Studio Code is a good option as it has extensions to facilitate development in Angular.
2. Install Node.js and npm. Angular uses Node.js and npm as package managers. Hence, you need to have them installed on your system.
3. Install the Angular client. To install the Angular client, use the command line tool and run this command: `npm install -g @angular/cli`. It will install the Angular client globally on your system.
4. Create a new project. To create a new project, execute: `npm new [project-name]`. This command creates a new folder with the name of your project and generates the basic structure of the project in it.
5. Run the project. Navigate to the project folder using the command `cd [project-name]` and then execute the following command: `ng serve -o`. This command compiles the project and starts it in your default browser, allowing you to see the changes in real-time as you make them.

2. STATE OF THE ART

2.1 Angular vs React

Angular and React are both popular JavaScript frameworks used for building web applications. Differences:

- Flexibility: React is known for its flexibility, giving developers more freedom to structure their projects and choose their preferred syntax. It also allows the use of different complementary technologies. Angular, on the other hand, offers a stricter project structure and syntax that can be beneficial for novice developers or large teams needing to follow a defined standard.
- Origin: Google develops and maintains Angular, while Facebook is responsible for React.
- Typing: Angular uses Typescript, a typed programming language that provides greater security and ease of maintenance for large projects. React, on the other hand, only requires JavaScript and JSX for component creation.
- Templates: Angular uses HTML templates for component creation, which can be beneficial for web designers and make it easier to create reusable components. In contrast, React uses JSX, a JavaScript extension that allows for component creation with a syntax similar to HTML.
- Data binding: Angular allows for bidirectional data binding, where changes to the data model are automatically reflected in the view. React, however, uses unidirectional data binding, requiring changes in the data model to be explicitly passed to the view using props.

2.2 Angular information sources

Learn Angular is not difficult at all. There are several sources where you can learn.

YouTube is a great resource for finding channels that offer tutorials for beginners. For example, 'Amigos Code' provides straightforward tutorials that are easy to follow. 'FreeCodeCamp.org' and 'Traversy Media' are also excellent channels to check out. Additionally, using the search bar can help you find exactly what you need to learn.

Always is a good option take courses in platforms like Udemy, Coursera or others similar.

There are several resources available to developers looking for help with Angular. The official Angular documentation is a recommended source, as it provides comprehensive information about the framework. In addition, there

are several forums such as 'Stack Overflow', 'Reddit' and 'Angular Discussions', where developers can ask specific questions and get answers from the community. These resources can be useful for troubleshooting, gathering information and learning best practices related to Angular development.

3. METHODOLOGY

An example of how Angular is applied in web application development is demonstrated through a carpooling application. This practical application enables users to share rides and travel together, offering a platform for posting routes and prices for others to join. The application is based on dummy data and is only implemented on the frontend.

In the subsequent paragraphs, we will explore various sections of the Angular code written for the development of this carpooling application.

app.module.ts

```
const appRoutes: Routes=[
  {path:'', component: IndexPageComponent},
  {path:'login', component: LoginPageComponent},
  {path:'register', component: RegisterPageComponent},
  {path:'newtrip', component: NewTripPageComponent},
  {path:'trips', component: TripsPageComponent},
  {path:'mytrips', component: MyTripsPageComponent}
]

@NgModule({
  declarations: [
    AppComponent,
    IndexPageComponent,
    NavarComponent,
    FooterComponent,
    LoginPageComponent,
    RegisterPageComponent,
    NewTripPageComponent,
    TripsPageComponent,
    TripCardComponent,
    MyTripsPageComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    RouterModule.forRoot(appRoutes),
    HttpClientModule,
    GoogleMapsModule
  ],
  providers: [UsersServiceService,
  TripServiceService],
  bootstrap: [AppComponent]
})
```

Figure 1. Declaration of appRoutes and NgModule with components and their corresponding routes.

app.module.ts is used to define and configure the application's components, services, and dependencies. It is the root module of an Angular application, and it sets up the dependency injection system, which is used to provide instances of services and other dependencies to components and other parts of the application.

In the Figure 1, the appRoutes array defines the routes for the application and maps each path to a corresponding component. When a user navigates to a path in the application, Angular uses the defined routes to determine which component to render. For example, the IndexPageComponent is associated with the root path (''), while the LoginPageComponent is associated with the 'login' path.

The @NgModule decorator is used to define the metadata for the root module of the application. The declarations property is an array of components that belong to the module. The imports property is an array of other modules used in the application. The providers property is an array of services that will be at the application.

App Component

```
<app-navar></app-navar>
<br>
<router-outlet></router-outlet>
<br>
<app-footer></app-footer>
```

Figure 2. app.component.html file of Carida project.

In the Figure 2 we see template of the root component. It has a navbar and a footer and between that two it exist a component called 'router-outlet', it shows the specific component depending on the route defined on app.module.ts in the Figure 1

Models

```
import { Trip } from "./trip.module";

export class User{

  name?:string;
  email?:string;
  password?:string;
  trips?:Trip[];

  constructor(email?:string, password?:string,
    name?:string, trips?:Trip[]){
    this.email = email;
    this.password = password;
    this.name = name;
    this.trips = trips;
  }
}
```

Figure 3. User class created in Angular.

In Angular we use models to define objects and we use this models in the aplication. In the Figure 3 we can see how we define a user in Carida. The class has a constructor method tho initialize the properties of the objects.

In Carida we also define the model Trip which has the properties: id, start, finish, price, date, driver, freeSeats, occupiedSeats, startLatitude, startLongitude, finishLatitude,

and finishLongitude. The definition of this model is pretty similar to the model user.

Services

```
@Injectable({
  providedIn: 'root'
})
export class TripServiceService {

  //Dummy data trips
  trips: Trip[] = [...];
  //Id of the data (dummy)
  singleID: number = 5;

constructor() { }

addTrip(trip:Trip){
  trip.id = this.singleID;
  this.singleID++;
  this.trips.push(trip);
}

findTrip(trip:Trip): Trip | undefined{
  return this.trips.find(TripToFind => TripToFind.start == trip.start
    && TripToFind.finish == trip.finish);
}

getTrips(): Trip[]{
  return this.trips;
}

}
```

Figure 4. Trip service with the methods that are implemented.

In the Figure 4 is shown the service that is used to manage the information related to the created trips.

The annotation '@Injectable(providedIn: 'root')' indicates that only one instance of this service will be created and shared by the whole application.

The dummy data we have in the application is created and stored in the trips list. As there are 4 trips created at the beginning, the ID control starts at 5.

In the service we have also defined 3 methods that have been necessary to manage the information:

- 'addTrip(trip:Trip)': this method creates a trip by assigning it an id and adding it to the list. In a real situation this method would be in charge of sending the Trip to the backend and nothing else.
- 'findTrip(trip:Trip): Trip — undefined': this method searches for a specific trip based on the departure and arrival points and returns the object or undefined if it does not exist.
- 'getTrips(): Trip[]': all trips are returned.

In the project we use two more services, one dedicated to manage the users and the other to manage the map shown in the application.

Components

In the Figure 5 we see the code of the page where we can see all the trips avaliable to book, the page resulting from this code can be seen in the Figure 7.

```
<div class="container bg-palid-green rounded shadow">
  <p class="text-center fs-1 p-4 fw-bold">Trips</p>
</div>

<div class="container" id="main-container">

  <ng-container *ngFor="let trip of trips">
    <app-tripCard [trip]="trip" [booked]="false"></app-tripCard>
  </ng-container>

</div>
```

Figure 5. HTML code of the trips page component.

The first section of the code corresponds to the title of the page. Further down, we utilize the Angular syntax to implement a foreach loop that iterates over each trip in the trips list.

The trips list, which contains all the available trips for booking, is created in the tripsPage.component.ts file and is populated by the 'getTrips()' method of the service, as seen in Figure 4.

The foreach loop generates cards for each trip, as shown in Figure 7. We pass the trip data to the 'tripCard' component and set the 'booked' variable to false. If the variable were set to true, the button 'Book a seat' would appear in red with the text 'Cancel Reservation'.

4. RESULTS

Although Carida is not a functional application at the moment, the result of this project is a simple and intuitive website that makes car sharing easy for anyone.

Every user, trip or booking created during a sesion is deleted once the page is reloaded.

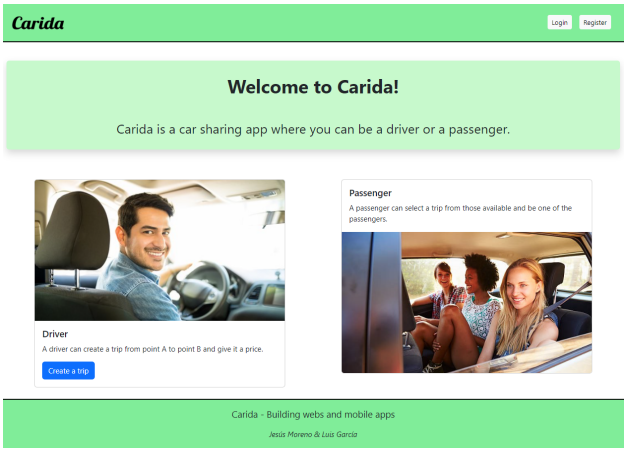

Figure 6. Carida's main page.

Figure 6 displays the main page, which features a navbar that offers the options to Login or Register. A brief description of the page is also provided through a short text, along with two cards that showcase the various roles available to users.

The login and register sections have no special features.

Once a user is logged the buttons in the navbar change

and allows you to go to avaliable trips sections, to create a trip and let you see your trips.

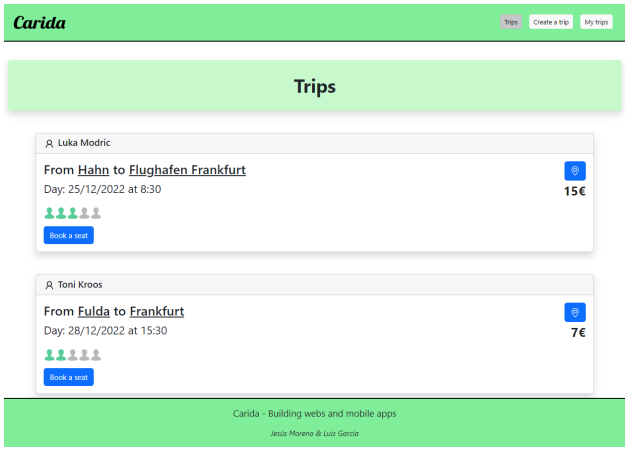

Figure 7. Page of Carida where you can see all the trips avaliable to book.

Figure 7 displays all available trips that can be booked by the user. These trips are the ones that have not been fully booked or booked by the user already. The available trip card provides information about the route, date, price, time, and available and booked seats. Furthermore, if the user clicks on the location button, a map will open as hte one on the Figure 8, allowing them to view the exact point of the route.

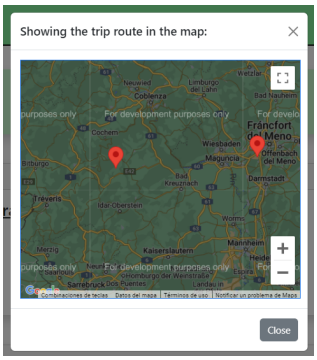

Figure 8. Screen opened where you can see the route of a trip.

The 'Create a trip' section is a form that requires the user to input all necessary details about the trip they wish to create. Additionally, the starting and ending points can be manually specified on the map. The information given is validated using Angular validators.

The map that is used in Carida is providad by Google and let the user navigate throght it.

the 'My Trips' sections let the user check all the Trips he has booked and let him cancel them if necessary. The page is so similar to tha page seen in Figure 7 but the button 'Book a seat' is red with the text 'Cancel reservation'.

Carida offers users a convenient and efficient way to discover and join trips, as well as cancel their reservation if necessary. Integration with Google Maps enables users to preview the trip route and evaluate if it meets their requirements before reserving a seat. This functionalities improves the overall user experience and expands the accessibility of car sharing.

5. DISCUSSION

5.1 How to create a project

This triggered that NLP has been traditionally a complex problem to solve [1]. However, Recurrent Neural Networks and two significant advances - one in 2017 and another in 2019 - brought substantial improvements to NLP. In 2017, a new form of deep learning model called Transformer [2] made it possible to parallelize ML training more efficiently, resulting in vastly improved accuracies. In 2019, Google introduced Bidirectional Encoder Representations from Transformers (BERT) [3], which improves the above Transformer architecture. Straightaway BERT helped achieve state-of-the-art performance [4] on several NLP tasks such as reading comprehension, text extraction, sentiment analysis, etc. These two advancements meant that NLP could easily outdo average humans in many tasks and in some cases, even exceed the performance of subject matter experts.

5.2 Our question

But does recurrent neural network propagation of information works similar as our brain does? We propose to look at brain activity of subjects reading naturalistic text as a source of additional information for interpreting neural networks. And to do this we use brain imaging recordings of subjects reading complex natural text to interpret word and sequence embeddings from 4 recent NLP models - ELMo, USE, BERT and Transformer-XL, and study how their representations differ across layer depth, context length, and attention type.

6. BLA

Most work investigating language in the brain has been done in a controlled experiment setup where two conditions are contrasted [5]. These conditions typically vary in complexity (simple vs. complex sentences), in the presence or absence of a linguistic property (sentences vs. lists of words) or in the presence or absence of incongruities (e.g. semantic surprisal) [5]. A few researchers instead use naturalistic stimulus such as stories [6]. Some use predictive models of brain activity as a function of multi-dimensional features spaces describing the different properties of the stimulus [7].

A few previous works have used neural network representations as a source of feature spaces to model brain activity. Wehbe et al. [7] aligned the MEG brain activity we use here with a Recurrent Neural Network (RNN), trained on an online archive of Harry Potter Fan Fiction. The authors aligned brain activity with the context vector and the word embedding, allowing them to trace sentence comprehension at a word-by-word level. Jain and Huth [8] aligned layers from a Long Short-Term Memory (LSTM) model to

fMRI recordings of subjects listening to stories to differentiate between the amount of context maintained by each brain region. Other approaches rely on computing surprisal or cognitive load metrics using neural networks to identify processing effort in the brain, instead of aligning entire representations [9].

There is also a little work that evaluates or improves NLP models through brain recordings. Søgaard [10] proposes to evaluate whether a word embedding contains cognition-relevant semantics by measuring how well they predict eye tracking data and fMRI recordings. Fyshe et al. [11] build a non-negative sparse embedding for individual words by constraining the embedding to also predict brain activity well and show that the new embeddings better align with behavioral measures of semantics.

7. ADSFASDF

We investigate how the representations of all four networks (ELMO, BERT, USE and T-XL) change as we provide varying lengths of context. We compute the representations x in each available intermediate layer ([1, 2] for ELMo; [1, 12] for BERT; the layer is the output embedding for USE; and [1, 19] for T-XL). We compute x for word w by passing the most recent k words through the network.

7.1 fMRI and MEG data

We use fMRI and MEG data which complement each other very well. fMRI is sensitive to the change in oxygen level in the blood which is a consequence to neural activity, it has high spatial resolution (2-3 mm) and low temporal resolution (multiple seconds). MEG measures the change in the magnetic field outside the skull due to neural activity, it has low spatial resolution (multiple cm) and high temporal resolution (up to 1KHz). We use fMRI data published by Wehbe et al. [7]: 8 subjects read chapter 9 of Harry Potter and the Sorcerer's stone [12] which was presented one word at a time for a fixed duration of 0.5 seconds each, and 45 minutes of data were recorded. The fMRI sampling rate (TR) was 2 seconds. The same chapter was shown by Wehbe et al. [13] to 3 subjects in MEG with the same rate of 0.5 seconds per word. Details about the data and preprocessing can be found in the supplementary materials.

7.2 Encoding models

For each type of network-derived representation x, we estimate an encoding model that takes x as input and predicts the brain recording associated with reading the same k words that were used to derive x. We estimate a function f, such that f(x) = y, where y is the brain activity recorded with either MEG or fMRI. We follow previous work [7] and model f as a linear function, regularized by the ridge penalty. The model is trained via four-fold cross-validation and the regularization parameter is chosen via nested cross-validation.

7.3 Evaluation of predictions

We evaluate the predictions from each encoding model by using them in a classification task on held-out data, in the four-fold cross-validation setting. The classification task is to predict which of two sets of words was being read based on the respective feature representations of these words [7, 13]. This task is performed between sets of 20 consecutive TRs in fMRI (accounting for the slowness of the hemodynamic response), and sets of 20 randomly sampled words in MEG. The classification is repeated a large number of times and an average classification accuracy is obtained for each voxel in fMRI and for each sensor/-timepoint in MEG. We refer to this accuracy of matching the predictions of an encoding model to the correct brain recordings as "prediction accuracy". The final fMRI results are reported on the MNI template, and we use pycortex to visualize them [14].

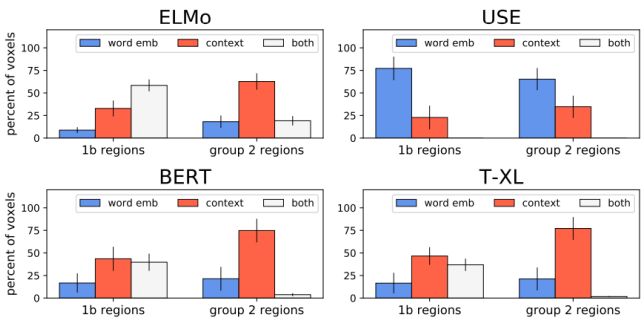

Figure 9. Amount of group 1b regions and group 2 regions predicted well by each network-derived representation: a 10-word representation corresponding to the 10 most recent words shown to the participant (Red) and a word-embedding corresponding to the last word (Blue). White indicates that both representations predict the specified amount of the regions well (about 0.7 threshold).

7.4 Proof of concept

Since MEG signals are faster than the rate of word presentation, they are more appropriate to study the components of word embeddings than the slow fMRI signals that cannot be attributed to individual words. We know that a word embedding learned from a text corpus is likely to contain information related to the number of letters and part of speech of a word. We show in section 4 of the supplementary materials that the number of letters of a word and its ELMo embedding predict a shared portion of brain activity early on (starting 100ms after word onset) in the back of the MEG helmet, over the visual cortex. Indeed, this region and latency are when we expect the visual information related to a word to be processed (Sudre et al., 2012). Further, a word's part of speech and its ELMo embedding predict a shared portion of brain activity around 200ms after word onset in the left front of the MEG sensor. Indeed, we know from electrophysiology studies that part of speech violations incur a response around 200ms after word onset in the frontal lobe (Frank et al., 2015). We conclude from these experiments that the ELMo embedding contains information about the number of letters and the part of speech of a word. Since we knew this from the onset, this experiment serves as a proof of concept for

String value	Numeric value
Hej SMC!	2023

Table 1. Table captions should be placed below the table, like this.

using our approach to interpret information in network representations.

8. HEADINGS

First level headings are in Times 12 pt bold, centered with 1 line of space above the section head, and 1/2 space below it. For a section header immediately followed by a subsection header, the space should be merged.

8.1 Second Level Headings

Second level headings are in Times 10 pt bold, flush left, with 1 line of space above the section head, and 1/2 space below it. The first letter of each significant word is capitalized.

8.1.1 Third Level Headings

Third level headings are in Times 10 pt italic, flush left, with 1/2 line of space above the section head, and 1/2 space below it. The first letter of significant words is capitalized.

Using more than three levels of headings is strongly discouraged.

9. FLOATS AND EQUATIONS

9.1 Equations

Equations should be placed on separated lines and numbered. The number should be on the right side, in parentheses.

$$r = \sqrt[13]{3} \tag{1}$$

Always refer to equations like this: "Equation (1) is of particular interest because..."

9.2 Figures, Tables and Captions

All artwork must be centered, neat, clean and legible. Figures should be centered, neat, clean and completely legible. All lines should be thick and dark enough for purposes of reproduction. Artwork should not be hand-drawn. The proceedings will be distributed in electronic form only, therefore color figures are allowed. However, you may want to check that your figures are understandable even if they are printed in black-and-white.

Numbers and captions of figures and tables always appear below the figure/table. Leave 1 line space between the figure or table and the caption. Figure and tables are numbered consecutively. Captions should be Times 10pt. Place tables/figures in the text as close to the reference as possible, and preferably at the top of the page.

Always refer to tables and figures in the main text, for example: "see Fig. 10 and Table 1". Figures and tables

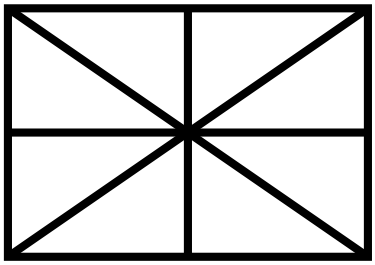

Figure 10. Figure captions should be placed below the figure, exactly like this.

may extend across both columns to a maximum width of 17.2cm.

Vectorial figures are preferred, e.g., eps. When using `Matlab`, export using either (encapsulated) Postscript or PDF format. In order to optimize readability, the font size of text within a figure should be no smaller than that of footnotes (8 pt font-size). If you use bitmap figures, make sure that the resolution is high enough for print quality.

9.3 Footnotes

You can indicate footnotes with a number in the text [1], but try to work the content into the main text.Use 8 pt font-size for footnotes. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a 0.5 pt horizontal rule.

10. CITATIONS

All bibliographical references should be listed at the end, inside a section named "REFERENCES". References must be numbered in order of appearance. You should avoid listing references that do not appear in the text.

Reference numbers in the text should appear within square brackets, such as in [15] or [15–17]. The reference format is the standard IEEE one. We highly recommend you use BibTeX to generate the reference list.

11. CONCLUSIONS

Please, submit full-length papers. Submission is fully electronic and automated through the Conference Web Submission System. Do not send papers directly by e-mail.

Acknowledgments

At the end of the Conclusions, acknowledgements to people, projects, funding agencies, etc. can be included after the second-level heading "Acknowledgments" (with no numbering).

12. REFERENCES

[1] K. S. Jones, "Natural language processing: an overview," *International encyclopedia of linguistics (ed W. Bright)*, pp. 3–59, 1992.

[1] This is a footnote example.

[2] Z. Wang, Y. Ma, Z. Liu, and J. Tang, "R-transformer: Recurrent neural network enhanced transformer," *arXiv preprint arXiv:1907.05572*, 2019.

[3] U. Kamath, K. L. Graham, and W. Emara, "Bidirectional encoder representations from transformers (bert)," *Transformers for Machine Learning; Chapman and Hall/CRC: New York, NY, USA*, pp. 43–70, 2022.

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[5] A. D. Friederici, "The brain basis of language processing: from structure to function," *Physiological reviews*, vol. 91, no. 4, pp. 1357–1392, 2011.

[6] J. Brennan, Y. Nir, U. Hasson, R. Malach, D. J. Heeger, and L. Pylkkänen, "Syntactic structure building in the anterior temporal lobe during natural story listening," *Brain and language*, vol. 120, no. 2, pp. 163–173, 2012.

[7] L. Wehbe, B. Murphy, P. Talukdar, A. Fyshe, A. Ramdas, and T. Mitchell, "Simultaneously uncovering the patterns of brain regions involved in different story reading subprocesses," *PloS one*, vol. 9, no. 11, p. e112575, 2014.

[8] S. Jain and A. Huth, "Incorporating context into language encoding models for fmri," *Advances in neural information processing systems*, vol. 31, 2018.

[9] S. L. Frank, L. J. Otten, G. Galli, and G. Vigliocco, "The erp response to the amount of information conveyed by words in sentences," *Brain and language*, vol. 140, pp. 1–11, 2015.

[10] A. Søgaard, "Evaluating word embeddings with fmri and eye-tracking," in *Proceedings of the 1st workshop on evaluating vector-space representations for NLP*, 2016, pp. 116–121.

[11] A. Fyshe, P. P. Talukdar, B. Murphy, and T. M. Mitchell, "Interpretable semantic vectors from a joint model of brain-and text-based meaning," in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, vol. 2014. NIH Public Access, 2014, p. 489.

[12] J. Rowling, *HARRY POTTER and The Sorceres Stone*. Bloomsbury Publishing Plc, 2012.

[13] L. Wehbe, A. Vaswani, K. Knight, and T. Mitchell, "Aligning context-based statistical models of language with brain activity during reading," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 233–243.

[14] J. S. Gao, A. G. Huth, M. D. Lescroart, and J. L. Gallant, "Pycortex: an interactive surface visualizer for fmri," *Frontiers in neuroinformatics*, p. 23, 2015.

[15] A. Someone, B. Someone, and C. Someone, "The title of the conf. paper," in *Proc. Int. Conf. Sound and Music Computing*, Porto, 2009, pp. 213–218.

[16] X. Someone and Y. Someone, *The Title of the Book*. Springer-Verlag, 2010.

[17] A. Someone, B. Someone, and C. Someone, "The title of the journal paper," in *J. New Music Research*, 2008, pp. 111–222.